Six-dimensional spinors in NLO amplitudes and a new approach to the complex universe

GRAY, WENDY,ELIZABETH

Six-dimensional spinors in NLO amplitudes

and

a new approach to the complex universe

Wendy Gray

A thesis presented for the degree of
Doctor of Philosophy

Institute of Particle Physics Phenomenology
Department of Physics
University of Durham
England

4 November 2025

Declaration

The work in this thesis is based on research carried out at the Institute of Particle Physics Phenomenology (IPPP), the Department of Physics, University of Durham, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

I would like to express my sincere appreciation and gratitude to my supervisor, Daniel Maitre, for his consistent and knowledgable support and friendship throughout my time at the IPPP, and for his willingness to allow me time to explore the project reported in Part II of this thesis. In that regard, I would also like to thank the many people who read parts of that work and provided valuable feedback and difficult questions: in particular, Daniel, the late Richard Bower, and Rameez.

I am also immensely grateful to those individuals who encouraged me in the journey to undertake a part-time PhD, particularly Paul Heslop and Frank Krauss. I thank everyone at IPPP and, indeed, in the Physics Department, for welcoming me, someone with a very different background from the usual, so warmly.

The University also played a part - when I needed to begin with the MSc in Particles, Strings and Cosmology, a strictly full-time course, they allowed me to register to do it part-time over two years. My employer at the time, Mazars LLP, similarly allowed me to flex my working profile to enable me to attend two terms of lectures almost entirely full-time.

I also owe a big thank you to my long-suffering family who, for years, have cheerfully put up with me squeezing physics into every available moment.

Last but not least, I wish to express my gratitude to the late Daisaku Ikeda, President of the lay Buddhist organisation Soka Gakkai International, without whose lifelong, dedicated efforts to propagate a philosophy of profound respect for the dignity of life I would never even have begun this work.

Contents

Chapter 1

Preamble and abstracts

At the Institute of Particle Physics Phenomenology (IPPP) at Durham University, where I have had the good fortune to study, it is normal for projects involving complex calculations and the development of numerical applications to be undertaken in relatively short timescales. Active and fruitful collaborations are routine. In order to accommodate a part-time PhD student, the first in the IPPP's history, it has been necessary (a) to choose a project for which part-time working results in an acceptable timescale; and (b) to avoid introducing delays into active collaborations. It was these considerations that led to the choice of the project reported in Part I of this thesis, in which I explore the possibility that a six-dimensional spinor helicity approach might offer a scalable tool which can be used to calculate NLO amplitudes for Higgs boson production in conjunction with increasing numbers of jets.

Working within the IPPP environment and thus with access to wider academic resources and the work of researchers in diverse fields, I have been allowed the freedom also to explore a very different research question, alongside my work on NLO amplitude calculations. I had been puzzled for some time about the relationship between the big unanswered questions in cosmology, including the nature of dark matter and dark energy, and the simplifications employed in the standard model of cosmology. Part II of this thesis reports on the outcome of my research so far, and presents a proposed new theoretical framework for cosmology which treats the universe as an evolving *complex* system.

I hope that readers find both interesting.

Abstract: part I

Part I of this thesis explores a numerical, six-dimensional spinor helicity approach to next-to-leading-order (NLO) virtual amplitude calculations for the gluon-fusion production of a Higgs boson, scalable for high-multiplicity jets.

Abstract: part II

The standard cosmological model has achieved the status of a concordance model, but remains incomplete and subject to increasing challenge. Part II of this thesis is an exposition of an alternative approach, that acknowledges the universe as a complex system. The new theoretical framework I suggest puts complexity centre-stage by incorporating the self-organised criticality (SOC) paradigm first proposed by Bak, Tang and Wiesenfeld [1] and releasing GR from the FLRW constraints into which it has historically been squeezed. This SOCGR framework predicts observed phenomena without the necessity of unknown matter or energy and supports further scientific investigation.

Part I

Six-dimensional spinor helicity as a numerical tool for NLO $gg \to h$+jets

Chapter 2

Introduction

Part I of this thesis explores a numerical, six-dimensional spinor helicity approach to next-to-leading-order (NLO) virtual amplitude calculations. It is framed in the context of gluon-fusion (ggF) production of a Higgs boson with high-multiplicity jets, but the approach could equally be applied to other processes.

The 2012 discovery of the Higgs boson during run I (2009-2013) of the Large Hadron Collider (LHC) [2,3] was a significant achievement for the scientific community. Research into the properties of the Higgs boson remains an important field, and precision measurement of the coupling of the Higgs boson to Standard Model particles is a key objective of the LHC, HL-LHC and future high-energy collider projects. In that context, it is crucial that high-precision theoretical predictions keep pace with the availability of experiemental data for relevant cross-sections.

Scattering amplitudes are the basic building blocks of the cross-section calculations by which the Standard Model is challenged by experimental data from collider experiments. High precision calculations are of fundamental importance, therefore, in high energy physics. Monte-Carlo tools like Sherpa excel at calculating tree-level amplitudes and related quantities such as infrared subtraction terms, but to generate full cross-sections need the virtual matrix elements to be supplied by external packages. In the construction of high-precision amplitudes, bottlenecks can arise in the calculation of these virtual matrix elements.

There also continues to be considerable scope for improvement in the preci-

sion of theoretical predictions. In QCD in general, high precision requires beyond leading order (LO), i.e. one-loop, corrections: only at NLO it is possible to address the LO problem of dependence on renormalization and factorization scales arising only via the running coupling and the evolving parton distribution functions. In fact, at least next-to-next-to-leading order (NNLO) calculations are desirable, as is clear from the bi-annual Les Houches report on the precision wishlist [4]. The current state of the art reported there for ggF production of the Higgs boson is N^3LO in the high top mass limit, without jets. For the ggF Higgs boson plus three jets only NLO, also in the high top mass limit, is known (ibid).

Increasing the multiplicity of jets included in available amplitude calculations is thus also important. It is this objective that is the subject of this thesis, which is work done to extend the boundaries of availability of theoretical predictions by calculating virtual matrix elements with multiple jets.

Specifically, the work reported here is in relation to the calculation of the NLO virtual amplitude contributions to cross-sections for $gg \to h + jj...$ The reasons for selecting this process are:

- Quantitative measurement of the Higgs sector potential is important in the context of challenging the Standard Model.
- At the 13TeV LHC the $gg \to h$ production mode is dominant, representing about 85 per cent of the total inclusive Higgs production cross section [5].
- The main source of backgrounds comes from multi-jet final states, but multi-jet virtual matrix elements are difficult to calculate and are available only for one-, two- or three-jet states. Traditional Feynman diagram approaches to calculations suffer from very rapid growth in the number of diagrams as external legs are added (at tree level, $gg \to 8g$ requires more than one million diagrams [6]).
- It is desirable to calculate for no fewer than three jets in order to differentiate from Higgs production via weak boson fusion $qq \to hjj$ (also known as vector boson fusion, VBF). Although ggF is the dominant production mode for Higgs at the LHC, VBF is also interesting because it offers an important test for unitarity [1]. Higgs production from VBF at LO has

[1] A study of the ggF for Higgs + ≤ 3 jets in the high top mass limit as background to VBF is reported in [7].

two (forward) jets in the final state with no radiation in the middle. Thus calculations for ggF production which involve a three- or more jet signature can be clearly differentiated from VBF.

- Increasing the number of jets beyond three improves differentiation and precision still further and, ideally, we would like to calculate higher multiplicities of parton jets. However, higher multiplicity brings with it increasing complexity in calculation. In particular, there is a substantial step up in 'bookkeeping' requirements between calculations involving a single quark anti-quark pair, which can arise in a three-jet final state, and calculations involving two such pairs which can arise in the four- and five-jet cases.

- There is a gap in availability of calculations for the $gg \to h$+jets process. Many NLO tools cannot be readily applied to it because it is in fact a two-loop process at NLO. The high top mass limit effective theory approach used to reduce it to a one-loop process at NLO is a simple one, but nevertheless invalidates or complicates some approaches typically used at one loop.

This thesis is not the first to attempt a solution to this problem. Armstrong's 2017 thesis [8] set out to calculate the same virtual amplitudes, for any number of jets, but met with both practical and theoretical obstacles. That work has been useful in signposting difficulties and has informed the choice of alternative approach in this project.

The question I seek to answer in this work is, then, 'can six-dimensional spinor helicity methods be used to set up a numerical framework to calculate the NLO virtual contribution to $gg \to h$ that can be scaled for an arbitrary number of additional jets?' The new work is (a) a numerical implementation of selected 6D spinor helicity methods for multi-leg tree amplitudes and (b) assembly of a framework for incorporating that implementation in a calculation of NLO Higgs plus many jets virtual amplitudes. Despite the interesting question of managing quark pairs I have worked with purely gluon jets for this project. However, the answer to this question for gluon jets potentially opens a similar pathway for quarks, subject to additional 'bookkeeping'.

It is possible, perhaps likely, that there is in fact some MHV-like structure

to be uncovered in the amplitudes for this process, despite the Higgs boson's status as a scalar. An added incentive for the numerical solution which is the eventual goal towards which this project is progressing is that ready availability of numerical calculations may enable a simple structure of the amplitude to be uncovered. This may be by using machine methods to search for analytical expressions for the numerical solutions, for example as employed in [9], or by trying out analytical ansätze which can be tested by comparison with this numerical method.

The ultimate aim is to support development of a package that can be added to the existing BlackHat [10, 11] software suite for NLO amplitudes.

The structure of this report is as follows:

- Chapter 3 outlines the nature of the problem and describes the choice of approach to seeking a solution.
- Chapter 4 sets out the six-dimensional spinor helicity formalism that supports the core generator for the necessary tree amplitudes that feed the one-loop calculation.
- Chapter 5 describes the implementation of the numerical six-dimensional spinor helicity calculations, which are coded in Python. In fact it describes two implementations. The first case enables up to the calculation of six-dimensional four point amplitudes and has working six-dimensional BCFW and three-point amplitudes. However, it is not a stable framework on which to build. The second implementation, which is presented in an addendum to Chapter 5, uses the experience of the first and is significantly more robust. The code in the addendum has been written recently by Daniel Maitre and is immensely valuable.
- In Chapter 6 I present work that fits neatly at the end of this report but was actually carried out very early in the project. Before we can begin to code an effective numerical solution, we need to have a complete theoretical basis for the entire calculation. This chapter provides a framework for the entire one-loop calculation within the context of the chosen approach, making clear which parts are already available to connect with the new implementation in this project.
- In Chapter 7 the work and its implications are discussed and conclusions are drawn.

Chapter 3

Choosing an approach

3.1 Context

Recent decades have been an exciting time for those involved in calculating amplitudes for theoretical predictions of cross-sections for processes arising in collider experiments. Onerous Feynman diagram approaches have largely given way to on-shell techniques, spinor helicity approaches have hugely simplified amplitudes, and unitarity considerations have transformed calculations at loop- as well as tree-level [12, 13]. Structure has become evident in some amplitudes, notably Parke Taylor amplitudes for n-gluon scattering [14]. In the decade following the Les Houches 2005 Workshop at TeV Colliders [15], which saw NLO $2 \to 4$ processes as the cutting edge, a 'NLO revolution' took place [16]. This saw the development of automated tools for NLO corrections for 4- [17] or even 5-jet processes [18] in some cases.

Against this background, the calculation of NLO $gg \to h + jj...$ amplitudes remains stubbornly difficult for high jet multiplicity. The process at NLO is actually a two-loop process. Reducing it to a single loop by applying the top-mass limit $m_t \to \infty$ (HTL) rules out the four-dimensional loop-level recursion ('bootstrap') approach to evaluating the rational terms that arise in the loop amplitude as a result of dimensional regularisation. Constrained to use D-dimensional unitarity approaches instead, where $D > 4$, not only are four-dimensional spinors inadequate but also all known MHV-type simplification is stripped away. Furthermore, in whatever $D > 4$ we choose to work, the amplitude for external

momenta ultimately must be stated in four dimensions if it is to contribute to a physically-relevant result. As the number of external legs increases, i.e. as more jets are added, this requirement to bring loop amplitudes back into four dimensions becomes a non-trivial problem.

This thesis is a successor to an earlier project [8] in which NLO $gg \to h + n$ jets was addressed. In that project, four-dimensional, cut-constructible parts of the NLO amplitude were successfully calculated but the full framework of a solution remained incomplete. In particular, the objective of using MHV-related simplification to carry out efficient amplitude calculations for a tower of any number of final-state jets proved to be unworkable because of the need to calculate rational parts of the amplitude in more than four dimensions. That work has been useful in demonstrating the practical problems that can arise in the calculation and has provided solid information about what is required, what might work well, and what does not. The outcome of that earlier project is summarised in Section 3.4.1.

The current project is necessarily directed towards a slightly different objective: rather than a single calculation for any number of final-state hadrons, the search is for a framework that can be scaled for additional jets in a reasonably tame way. The selection of theories, methods and techniques has of course been informed by the earlier attempt. We also benefit from the fact that it is no longer necessary to focus on computationally efficient methods: since it is now increasingly possible to derive analytic expressions from numerical results (see, for example, [9]), we need only to find an implementation that works, it does not need to minimise steps in calculation. The code implemented in BlackHat could then be the relevant analytic expressions .

Clearly, there is more than one way to approach the calculation of NLO $gg \to h + jj...$ virtual amplitudes. In this chapter I first address the nature of the problem, including why it is a difficult one. What, exactly, is the process? Why does the existing BlackHat approach to the virtual matrix elements not work in this case? I then summarise the approaches that are available, including what work has already been done and what can we learn from them. The aim is to conclude on a practical, numerical approach to constructing the necessary one-loop amplitudes.

The chosen way forward is set out at the end of the chapter, in Section 3.5.

To check that all the components necessary to compute the full, NLO virtual amplitude are available, a 'road map' for the entire calculation is presented in Chapter 6.

3.2 The $gg \to h + jj...$ process

In the Standard Model, the Higgs boson coupling to two gluons is mediated at LO by a heavy-quark loop. The Higgs boson coupling is proportional to mass, hence the strongest interaction in QCD is with the top quark: contributions from other quarks are suppressed by at least a factor of $\mathcal{O}(m_b^2/m_t^2)$ where m_b is the mass of the bottom quark and m_t is the mass of the top.

Figure 3.2.1 shows that LO $gg \to h$ is already a one-loop process: therefore, NLO ggF production of higgs plus jets is a two-loop process. Presently, true NLO (i.e. two-loop) calculations are not defined beyond Higgs plus one jet. However, it is conventional to treat the top quark as approaching infinite mass, i.e. HTL, and to approximate other quarks as massless. That this approximation is a reasonable one whenever the mass of the Higgs and the transverse momentum of the jets are not larger than the mass of the top quark has been confirmed via LO computations of Higgs plus two and three jets with full top-mass dependence [19]. Harlander et al [20] have also demonstrated that the HTL approximation is valid at the $2-3\%$ level as long as the transverse momentum of the Higgs remains below about 150 GeV.

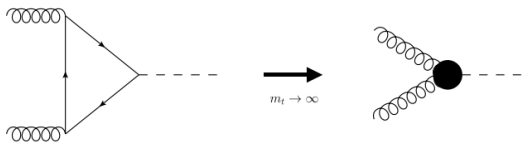


Figure 3.2.1: Leading order $gg \to h$ and the HTL effective vertex

In the HTL approximation the degrees of freedom of the loop quark can be integrated out to give a new effective vertex between two gluons and a Higgs boson as shown in Figure 3.2.1.

The effective operator is

$$\mathcal{L}_H^{\text{int}} = \frac{C}{2} H \text{tr} G_{\mu\nu} G^{\mu\nu} \tag{3.2.1}$$

where C is the dimensionful coupling constant which can be calculated at order α_s, H is the Higgs boson field and $G_{\mu\nu}$ is the gluon field strength tensor. The NLO computation is then brought down to one loop, but at the expense of introducing an additional power of the loop momentum in the numerator of the amplitude. As a result, the formalism used so effectively in the BlackHat package for one-loop W and Z amplitudes cannot be applied to this problem.

The NLO process then, shown here with gluon jets, is represented in Figure 3.2.2.

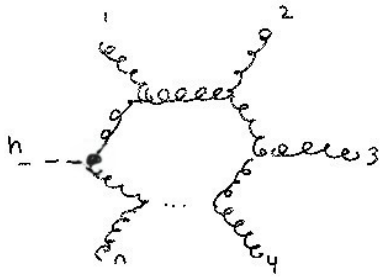


Figure 3.2.2: NLO $gg \to h + jj...$

3.3 On-shell, unitarity methods

In the traditional Feynman diagram approach to amplitude calculations one algorithmically draws and computes all the possible diagrams and sums them. For a simple process this is very straightforward but, as the number of external legs increases, there is rapid growth in complication. At *tree* level in QCD $g + g \to 8g$ requires more than one million diagrams [6]. Fortunately, since the pioneering work by Bern, Dixon and others [21–24] in the mid-1990s more efficient methods have been developed. In particular, on-shell methods for computing amplitudes focus on the key analytic properties that any physical amplitude must satisfy and so avoid gauge non-invariant information intermediate states. They are therefore much more efficient than Feynman diagram techniques.

The basic toolkit for such calculations is now well understood, and there are numerous sources of detailed descriptions of the approach, for example [12]. The procedure, which is outlined below, begins with separating out colour permutations.

3.3.1 Colour decomposition

QCD is a Yang-Mills theory which uses the $SU(3)$ gauge group to describe the strong force. The first step in the on-shell technology is to simplify the calculations by separating $SU(3)$ colour information from kinematics. This step is well understood, clearly defined and obviously helpful. I therefore take it as read that this will form part of the final approach and leave the description of the method to Section 6.2 of the 'roadmap' in Chapter 6, where the components of the chosen approach are set out in order.

3.3.2 Unitarity at tree level: factorisation and BCFW

Since the pioneering work of Britto, Cachazo, Feng and Witten (BCFW) [25] it has been known that on-shell tree amplitudes can be factorised into two lower-multiplicity tree amplitudes, as shown in Figure 3.3.1.

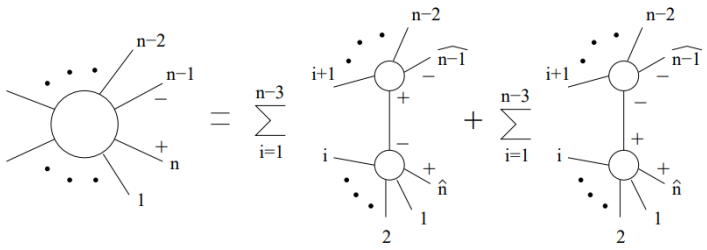


Figure 3.3.1: Pictorial representation of the BCFW recursion relation in 4D. The difference between the terms in the two sums is just the helicity assignment of the internal line. Source: [26]

BCFW recursion is based on the idea that tree-level amplitudes are continuously deformable, analytic functions of the scattering momenta, which are retained on-shell. It is therefore possible to construct amplitudes for generic kinematics from their behaviour in singular, limiting kinematics. In other words, we use the behaviour of an amplitude under a complex deformation of the particle momenta

to calculate the amplitude. In practice, we impose a complex shift of the helicity spinors λ [1] for two neighbouring legs of the momenta, say legs $n-1$ and n:

$$\begin{aligned} \lambda_{n-1} \to \hat{\lambda}_{n-1}(z) &= \lambda_{n-1} - z\lambda_n \\ \tilde{\lambda}_n \to \hat{\tilde{\lambda}}_n(z) &= \tilde{\lambda}_n + z\tilde{\lambda}_{n-1} \end{aligned} \tag{3.3.1}$$

where z is a complex parameter and the deformation preserves momentum conservation and on-shell conditions. Let us call the deformed partial amplitude $\mathscr{A}(z)$. We need to understand how

$$\mathscr{A}(z) = \delta^{(4)}\left(\sum_{i=1}^{n} p_i\right) A_n(z) \tag{3.3.2}$$

relies on how the amplitude $A_n(z)$ behaves in the z complex plane. It is found that [27]

1. With $P_i \equiv p_1 + p_2 + ... + p_{i-1}$, the amplitude $A_n(z)$ has simple poles in z at positions

$$z_{p_i} = \frac{P_i^2}{\lambda_{n,\alpha} P_i^{\alpha\dot{\alpha}} \tilde{\lambda}_{n-1,\dot{\alpha}}} \tag{3.3.3}$$

2. Near the pole z_{p_i} the amplitude $\lim_{z \to z_{P_i}} A_n(z)$ factorises into a product of lower-point, 'left' and 'right' amplitudes

$$\begin{aligned} \lim_{z \to z_{p_i}} A_n(z) = \frac{1}{z - z_{p_i}} \frac{-1}{\langle n|P_i|n-1]} \sum_s A^L\big(\hat{1}(z_{p_i}), 2, ..., i-1, -\hat{P}^h(z_{p_i})\big) \\ \times A^R\big(\hat{P}^{\hat{h}}(z_{p_i}), i, ..., n-1, \hat{n}(z_{p_i})(z_{p_i})\big) \end{aligned} \tag{3.3.4}$$

3. Complex analysis and the residue theorem can then be used to construct $A_n(z=0)$ from what is known about the poles of $A_n(z)$. If $A_n(z) \to 0$ as $z \to \infty$ for a suitable shift and C is the circle at infinity then

$$\begin{aligned} 0 &= \frac{1}{2\pi i} \oint_C dz \frac{A_n(z)}{z} \\ &= A_n(0) + \sum_{i=2}^{n-1} Res\left[\frac{A_n(z)}{z}\right]|_{z=z_{p_i}} \end{aligned} \tag{3.3.5}$$

[1] These four-dimensional helicity spinors are defined in the next chapter, Section 4.1.

So

$$A(0) = -\sum_{i=2}^{n-1} Res\left[\frac{A_n(z)}{z}\right]|_{z=z_{p_i}} \tag{3.3.6}$$

With the expression for A_n in equation 3.3.4 and the pole at z given by 3.3.3 we can solve for $A(0)$ for the BCFW recursion relation:

$$A_n = A(0) = \sum_{i=2}^{n-1}\sum_{h} A_L^h(z_{P_i})\frac{1}{P_i^2}A_R^{\bar{h}}(z_{P_i}) \tag{3.3.7}$$

Since knowledge of three-point amplitudes then allows the construction of all higher point amplitudes, generation of multi-leg tree amplitudes is greatly simplified by this method. It will be crucial in our approach.

3.3.3 Loop-level methods

On-shell and unitarity methods are now commonly used in one-loop computations in the high parton multiplicity of LHC events [13]. The unitarity method developed by Bern, Dixon, Dunbar and Kosower in the 1990s [22, 23] uses the branch-cut structure of loop integrals to find the coefficients of the integrals in terms of products of on-shell tree amplitudes, see Figure 3.3.2. In contrast to Feynman diagram approaches, computational algorithms exhibit only polynomial complexity.

A review of these on-shell and unitarity methods in the context of numerical implementations is given in Ita 2011 [28] and for a thorough description of the conceptual and technical aspects see Ellis et al 2012 [29]. A nice summary of the development of these approaches is given in [30], including Ossola, Papadopoulos and Pittau's (OPP) parametrisation of loop momenta to compute higher order terms [31] further developed by Forde [32] using complex analysis to isolate the coefficients of the scalar integrals. The methods have been applied for some processes in numerical calculations at loop level in BlackHat [10, 33–35].

In brief, the unitarity constraint allows factorisation of an amplitude into cuts and rational parts, such that the cut amplitude can be written as a tree amplitude on one side of the cut multiplied by a tree amplitude on the other side, with the loop integral replaced by an integral over the phase space of the particles crossing the cut [23].

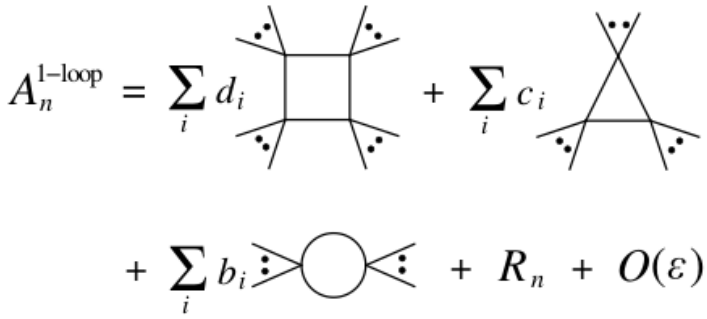


Figure 3.3.2: Decomposition of 1-loop amplitude into basis scalar box, triangle and bubble integrals multiplied by kinematical coefficients, with a rational part R_n which cannot be detected using cuts with four-dimensional cut loop momenta. Source: [13]

At loop level, then, divergences result in rational pieces of the amplitude that are not captured by the four-dimensional unitarity cuts. There are essentially two ways of dealing with the rational parts, which I outline below.

Option 1: Loop-level recursion, a.k.a. 'bootstrapping'

In this method, which exploits multi-particle factorisation properties [36–38], the rational pieces are constructed by on-shell recursion relations in four dimensions [10, 38–40]. It has been used to derive analytic expressions for many helicity amplitudes up to eight final state gluons [40] as well as all-mulitplicity expressions for one-loop MHV amplitudes [39, 41].'

The method applies equally well to amplitudes with massive external particles, including gluons coupling to a Higgs boson [42–44].

In BlackHat, these recursion relations are combined with Forde's coefficient extraction [32].

Option 2: Generalised D-dimensional unitarity

D-dimensional cutting techniques also completely determine the loop amplitude [24, 36, 45]. Some rational coefficients are known (Badger 2009), but we wish to build a scalable solution and so need to be able to calculate unknown

rational coefficients.

The identified options are:

- A numerical D-dimensional unitarity procedure for any one-loop QCD amplitude was developed by Giele, Kunszt and Melkikov [46] using the OPP on-shell procedure of reducing integrals [31]. Ellis, Giele, Kunszt and Melnikov [47] extended the generalised D-dimensional unitarity method for numerical evaluation of one-loop amplitudes by incorporating massive particles.

- A related approach uses the relationship between states that are massive in 4D and states in extra dimensions to reduce the integrals obtained from unitarity cuts [24, 30]. These D-dimensional approaches were extended by Badger [30] to use Forde's complex analysis techniques for cut-constructible as well as rational parts. The additional mass-dependent integral coefficients can be extracted from the large mass limit, analytically or numerically, associating the $D-4$-dimensional pieces with masses to give rational terms. The rational terms in $D = 4 - 2\epsilon$ dimensions can be determined from quadruple, triple and double cuts without need for independent pentagon contributions, using a massive integral basis. A numerical implementation (BlackHat, see Section 3.4.2 [33, 48]) has been developed and applied for some processes.

- The 6D spinor helicity formalism created by Cheung & O'Connell produces cut-constructible and rational parts for all helicities. Davies [49] uses D-dimensional unitarity and Cheung and O'Connell's six-dimensional spinor helicity formalism together to calculate QCD amplitudes, capturing cut-constructible and rational pieces together. His work is analytical rather than numerical.

In his one-loop use of the Cheung and O'Connell six-dimensional spinor helicity formalism, Davies [49] used the Four Dimensional Helicity Scheme (FDH) [21, 24, 50] for regularisation. At the end of the calculation the loop momenta are analytically continued to $D = 4 - 2\epsilon$ dimensions and a state-sum reduction is performed to reduce the spin states of the six dimensions to match the FDH scheme. Other schemes are available, for example the 'tHooft-Veltman Scheme (HV) [51], the Conventional Dimensional Regularisation Scheme (CDR) [52], the Dimensional Reduction Scheme (DRED) [53]. The schemes share the dimensional

regularisation of momentum integrals but differ in the ways in which they handle observed states and spin degrees of freedom. FDH is common and well understood. Although Kilgore showed that FDH is not unitary at higher-orders [54], it is effective at one loop and suitable for our purpose.

3.3.4 Spinor helicity tools

On-shell and unitarity methods rely upon tree amplitudes. The methods work particularly well when combined with the spinor helicity formalism for amplitudes which, in four dimensions, is very well established and is summarised at the beginning of Chapter 4. We have seen in Section 3.3.3, however, that four dimensions are not sufficient to obtain the rational parts of the amplitude we are calculating: the existing approach in BlackHat, using an on-shell recursion to reconstruct rational pieces, requires knowledge of the pole structure of the cut terms which we could only gather in this case by resorting to the Feynman diagram techniques we are seeking to avoid. Instead, we must turn to D-dimensional generalised unitarity. So, if we wish to use spinor helicity methods, we must do so in more than four dimensions. As our approach is numerical, the number of dimensions must be an integer.

The earlier project addressing the NLO $gg \to h + jj..$ problem [8] included a six-dimensional spinor helicity approach but it remained incomplete and did not reach the stage of calculating any six-dimensional amplitudes. I have already mentioned that an alternative six-dimensional spinor helicity formalism has been pioneered at tree level by Cheung and O'Connell [55], which does specify tree amplitudes and does include an analytic approach to the derivation of a four-point amplitude using a definition of six-dimensional BCFW. This formalism is described in Chapter 4.

Cheung and O'Connell's formalism has been combined with the generalised unitarity method by Bern, Carrasco, Dennen, Huang and Ita [45] to construct analytic loop-level scattering amplitudes, using the example of QCD one-loop four-point amplitudes. In the same year Davies [49] applied the method to an analytic approach to one-loop QCD processes with a Higgs boson and three partons. Badger, Brønnum-Hansen, Buciuni and O'Connell [56] have used the six-dimensional helicity formalism to perform generalised unitarity cuts in d dimensions and applied the method to $gg \to t\bar{t}$, with a numerical implementation

of two of the leading colour primitive amplitudes required for this process. None of these approaches has included a public, working, six-dimensional implementation of three-point amplitudes or the BCFW framework. These will be required if we are to be able to build tree amplitudes with increasing numbers of partons.

The Cheung and O'Connell six dimensional formalism is immediately attractive, as it offers the extra-dimensionality required to calculate both the cut and the rational part of the virtual amplitude together and it is amenable to coding for numerical approaches. However, it adds complication. Whereas in the four-dimensional formalism many sub-amplitudes vanish as a result of simplification possible using MHV rules, in the six-dimensional extension of the spinor helicity method that simplification is lost. In 4D spinor helicity formalism terms, each external jet must be represented by either a chiral[2] or an antichiral spinor, but not both. In 6D, however, each particle is labelled by a pair of little group indices. In six dimensions the little group SU(2) x SU(2) connects all helicities together and a single formula describes the amplitude. Specific helicity cases must then be calculated by choice of little group indices and the solution we use must be capable of reducing the additional chirality states in the six-dimensional spinors back to the four-dimensional physical process.

3.3.5 Treatment of the Higgs boson

The Higgs is a scalar boson: the only fundamental scalar particle so far found to exist outside bound states. Some very nice work has been done treating the Higgs boson as the real part of a complex scalar in amplitude calculations, with $H = \phi + \phi^\dagger$ [57, 58]. This appealing approach enables the application of MHV rules to eradicate up-front a number of vanishing tree amplitude components and to establish towers of amplitudes up to n gluons in some cases. In six dimensions, however, it is not the case that all required tree amplitudes are eradicated: see Figure 3.3.3. On this figure, all combinations of gluon helicity that are either blank or have an open circle may be non-vanishing and require some element of calculation. It is for this reason that we are not in a position to generate a solution that can be applied to an arbitrary number of jets, even in the all-gluon case.

[2] *Helicity* is the projection of its spin onto its momentum direction. *Chirality* is the left- or right-handedness of a particle's spin projection along its direction of motion, and does not change. For a massless particle, helicity and chirality are the same. In this project, all external particles are either a Higgs boson, which is a scalar, or massless gluons, for which chirality and helicity are the same.

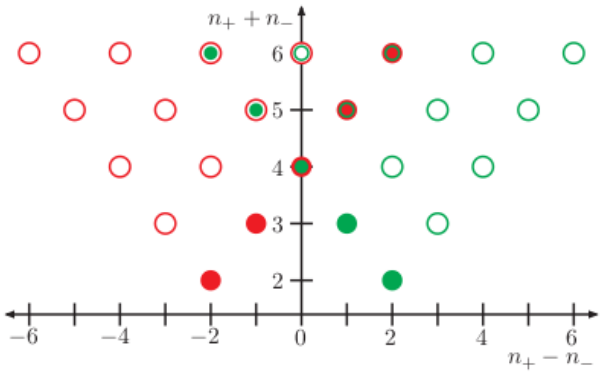


Figure 3.3.3: The combined MHV tower for $\phi + n$ gluons and anti-MHV tower for $\phi^\dagger + n$ gluon amplitudes. The vertical axis labels the total number of gluons, $n_+ + n_-$. The horizonal axis measures the degree of 'helicity violation', labelling the difference $n_+ - n_-$. Solid dots represent full 'ϕ-MHV' vertices, whilst open circles represent amplitudes which are composites built from ϕ-MHV vertices and pure-gauge-theory MHV vertices. MHV is red, anti-MHV is green. Source: [57] figure 3.

Therefore, we must select an approach that enables us readily to calculate the tree amplitudes that are required and, in order to obtain the coefficients of the rational part of the loop amplitude, to calculate them in $D > 4$. We have seen above that the strongest candidate for these calculations precludes MHV-type simplifications. With the major advantage of the complex scalar approach thus inaccessible I therefore choose to avoid having the additional step of combining calculations for ϕ and $\phi^\dagger$ and simply treat Higgs as a real scalar.

The Higgs is also, of course, massive. As it appears only as an external leg, I will incorporate its mass in calculations via momentum conservation.

3.4 Previous work

3.4.1 Work on ggF Higgs boson production

A useful overview of the status of calculations using on-shell techniques, and to computation of rational terms, is given in the *Les Houches 2015 Standard*

Model Working Group Report [59], see pp 77 onwards. The more recent 2023 Les Houces report [4] provides a summary of the current status of amplitude calculations.

The landscape of existing approaches to QCD calculations for Higgs plus jet production from ggF is summarised below.

Table 3.4.1: Previous work on $gg \to h$+jets.

Order	**Jets**	**Summary**
LO	2j	In the 1990s work was done in HTL, see Dawson and Kauffman [60], Kauffman, Desai and Risal [61].
LO	2j	By 2001 top mass dependence had been included: real emission corrections to $gg \to H + 2$ jets were calculated at order α_s^4 by Del Duca, Kilgore, Oleari, Schmidt and Zeppenfeld [62].
LO	2j	Calculation of $gg \to H + 2$ jets scattering amplitudes as induced by top-quark triangle-, box- and pentagon-loop diagrams. The diagrams are evaluated analytically for arbitrary top mass. Del Duca, Kilgore, Oleari, Schmidt and Zeppenfeld [63].
LO	3j	Tree-level amplitudes for Higgs with five partons of fixed helicities had been made available by Del Duca, Frizzo and Maltoni [64] by 2004.
LO	MHV	Also from 2004, the work done by Dixon, Badger, Glover and Khoze to establish MHV-type rules for a complex scalar ϕ provides analytical computations of tree amplitudes for Higgs with multiple gluons [57] and with multiple partons [65] in HTL. They are available numerically through the matrix-element Monte Carlo generators ALPGEN [66] and

Continued on next page

Table 3.4.1 – continued from previous page

Order	Jets	Summary
		MadEvent [67].
NLO	1j	Schmidt computed three-parton helicity amplitudes for Higgs boson production from ggF in HTL in 1997 [68].
NLO	1j	In 2011 Davies [49] applied the six-dimensional helicity formalism combined with D-dimensional generalised unitarity to compute one-loop amplitudes in dimensionally regularized QCD. Davies [49] applies approach to NLO QCD correction to Higgs processes: compute NLO correction to Higgs plus three positive-helicity gluons amplitude in HTL.
NLO	2j	With the work outlined in LO above, the tree amplitudes and MHV rules necessary for the one-loop, two-jet calculation had been completed by 2005.
NLO	2j	A semi-numerical calculation of NLO results for $H + 4p$ is given in Zanderighi [69] All amplitudes for a Higgs with four partons are now known analytically at one-loop:
NLO	2j	Analytic results for one-loop amplitudes for a Higgs boson plus four partons in the case of two quark-antiquark pairs were reported in 2005 by Ellis, Giele and Zanderighi [70].
NLO	2j	All vanishing tree amplitudes for complex ϕ were available by 2007, Berger et al [42].
NLO	2j	One-loop phi-MHV amplitudes with two negative and

Continued on next page

Table 3.4.1 – continued from previous page

Order	Jets	Summary
		any number of positive helicity gluons were published in 2008 by Glover, Mastrolia and Williams [44]. Their work included explicit calculation for the four-gluon case.
NLO	2j	The calculation of the NMHV case of quark-antiquark pair plus negative helicity gluon pair is reported in Badger et al [71].
NLO	2j	For the one-loop amplitude for $H\bar{q}q\bar{Q}Q$ and $H\bar{q}qgg$ cases, the latter restricted to opposite-helicity gluons, see [72].
NLO	2j	With the one positive and any number of negative helicity gluon case, the full analytic results for one-loop higgs plus four gluons of any helicity was published by Badger, Glover, Mastrolia and Williams in 2010 [58]. All of the NLO 2j calculations above use the HTL approximation.
NLO	2j	Analytic formulae for the one-loop amplitude for Higgs plus 4 partons (i.e. 2 jets) with full mass dependence were reported by Budge, Ellis and others in 2022 [73].
NLO	3j	Automated calculations in relation to NLO QCD corrections for Higgs production from ggF in the HTL approximation have been computed by Cullen et al 2013 [74], Greiner et al 2015 [75] and Luisioni et al in 2016 [76]. All of these calculations have been performed numerically using GoSam and Sherpa, based on an algebraic generation of d-dimensional

Continued on next page

Table 3.4.1 – continued from previous page

Order	Jets	Summary
		integrands using a Feynman diagram approach.
NLO	ϕ MHV	The work by Berger et al 2007 [42] and Badger et al [43] referenced above use the complex *phi* approach to solve on-shell recursion relations for arbitrary numbers of gluons in MHV cases.
NNLO	low	Work is proceeding at NNLO, but not yet at high jet multiplicity. For example, in 2015 Chen et al [77] computed the entire cross section and differential distributions for the production of a Higgs boson with one hadronic jet at NNLO.
N^3LO	0j	Mistelberger [78] published an exact formula for the ggF Higgs boson in the HTL approximation, using an analytic computation involving elliptic integrals.

From the nature of the previous work that is summarised in Table 3.4.1 it is clear that the substantial progress that has been achieved has been incremental. It is surely desirable to find, if possible, a mechanism for moving forward, at least with increasing jet multiplicity, in a relatively simple way. The earlier PhD thesis [8] was an attempt to do just that, seeking to find generic results for one-loop amplitudes for n partons with arbitrary helicities, reducing the number of tree calculations as much as possible by identifying and excluding MHV cases and invalid combinations of helicities in loop cuts. However, no generic means of obtaining the rational part of the one-loop amplitude was found, and the following substantial difficulties were encountered:

- a method for calculating the six-dimensional tree amplitudes that were required remained elusive;
- defining the necessary conditions for identifying amplitudes that would vanish when reduced to four dimensions proved complex and, due to time constraints, remained incomplete; and

- no algorithm amenable to automated processing was found to handle the final conversion to four-dimensional expressions and so the author concluded this would have to be done manually (ibid, pg.101).

3.4.2 BlackHat library for NLO without Higgs

The BlackHat Library for One-Loop Amplitudes [79, 80] offers calculations for amplitudes with quarks, gluons and $W^{\pm}$ [33] or $Z, \gamma^* + 3$-Jet Distributions at the Tevatron [35] and other processes but not yet the Higgs boson.

The four-dimensional 'bootstrap' loop recursion relations approach used in the existing BlackHat suite is unsuitable for the NLO $gg \to h + jj...$ process because the HTL vertex used so effectively to reduce the NLO ggF Higgs production process to one loop introduces an additonal propagator into the numerator of the amplitude. The calculation of the rational part cannot be constrained in just four dimensions.

3.5 Chosen approach

In summary, I will work with HTL to reduce NLO to one loop for the $gg \to h+jj...$ process. I will treat the Higgs boson as a real scalar.

New developments have resulted in a capacity to use numerical solutions to extract analytical amplitudes [9, 60]. As a result, computational efficiency is no longer a principal driver for the choice of method for the numerical application, since any approach that provides reliable results can be used to obtain analytical expressions to be coded into BlackHat or other frameworks, rather than the full numerical calculation. I therefore choose D-dimensional unitarity, using Cheung and O'Connell's six-dimensional formalism both to calculate multi-leg tree aplitudes and to capture the cut and rational pieces of the one-loop virtual amplitude, with the FDH scheme for regularisation.

To check that all the components necessary to compute the full, NLO virtual amplitude are available, a 'road-map' for the full calculation has been constructed and is presented in Chapter 6. The road map indicates a complete implementation is possible, as long as the six-dimensional spinor helicity framework and its implementation can provide many-leg, six-dimensional tree

amplitudes. This comes down to a question of whether three-point amplitudes and six-dimensional BCFW can be reliably constructed. The main purpose of this part of this thesis, then, is to implement the six-dimensional spinor helicity formalism in order to address those questions.

Chapter 4

Spinor helicity in theory

The six-dimensional spinor helicity formalism was developed by Cheung and O'Connell [55]. I begin, as did they, with a brief recap of the well-understood four-dimensional spinor helicity technology and then move on to describe the six-dimensional formalism.

This chapter serves to define the basic spinor notation choices. The function names for our Python implementation of the formalism described in Chapter 5 are given in boldface alongside the definitions where appropriate.

Throughout, the convention that all momenta are outgoing is used.

4.1 Four-dimensional spinor helicity

For 4D objects indices are:

SL(2,C) fundamental labels	$\alpha, \beta, ... = 1, 2$ and $\dot{\alpha}, \dot{\beta}, ... = 1, 2$
SO(3,1) vector labels	$\mu, \nu, ... = 0, 1, 2, 3$

Following [55] we use the mainly minus Minkowski metric:

$$\eta_{\mu\nu} = \text{diag}(+, -, -, ..., -) \tag{4.1.1}$$

Pauli matrices

For the Pauli matrices we use the normal convention

$$\textbf{sig4i:} \quad \sigma^0 = \mathbf{1},\ \sigma^1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ \sigma^2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ \sigma^3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

With

- $(\tilde{\sigma}^\mu)^{\alpha\dot{\alpha}} = (\mathbf{1}, -\sigma)$
- $(\sigma^\mu)_{\alpha\dot{\alpha}} = \epsilon_{\alpha\beta}\epsilon_{\dot{\alpha}\dot{\beta}}(\tilde{\sigma}^\mu)^{\beta\dot{\beta}} = (\mathbf{1}, \sigma)$
- $(\tilde{\sigma}_\mu)^{\alpha\dot{\alpha}} = (\mathbf{1}, \sigma)$
- $(\sigma_\mu)_{\alpha\dot{\alpha}} = (\mathbf{1}, -\sigma)$

using the totally anti-symmetric Levi-Civita tensors

$$\textbf{lev2u} \quad \epsilon^{\alpha\beta} \equiv \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\textbf{lev2d} \quad \epsilon_{\alpha\beta} \equiv \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Note that we later also use higher-dimensional anti-symmetric Levi-Civita tensors

$$\textbf{lev3} \quad \epsilon^{\alpha\beta\gamma}$$
$$\textbf{lev4} \quad \epsilon^{\alpha\beta\gamma\delta}$$

In these cases the index up and down tensors have the same matrix representation.

4-momentum

For a momentum four-vector $p^\mu = (p^0, p^i) = (E, p^i)$ and hence $p_\mu = \eta_{\mu\nu}p^\nu = (p_0, -p_1, -p_2, -p_3)$, contraction with Pauli matrices gives

$$\begin{aligned} p^{\alpha\dot{\alpha}} \equiv \underset{\textbf{psig4u}}{p^\mu \tilde{\sigma}_\mu^{\alpha\dot{\alpha}}} &= \begin{bmatrix} p^0 + p^3 & p^1 - ip^2 \\ p^1 + ip^2 & p^0 - p^3 \end{bmatrix} \equiv \begin{bmatrix} p_+ & p_-^\perp \\ p_+^\perp & p_- \end{bmatrix} \\ p_{\alpha\dot{\alpha}} \equiv \underset{\textbf{psig4d}}{p_\mu \sigma^\mu_{\alpha\dot{\alpha}}} &= \begin{bmatrix} p^0 - p^3 & -(p^1 - ip^2) \\ -(p^1 + ip^2) & p^0 + p^3 \end{bmatrix} \equiv \begin{bmatrix} p_- & -p_-^\perp \\ -p_+^\perp & p_+ \end{bmatrix} \end{aligned} \tag{4.1.2}$$

The determinant of the momentum, $|p^{\alpha\dot{\alpha}}| = p^2$.

Weyl spinors

For massless momenta, the rank of the $p_{\alpha\dot{\alpha}}$-matrix is one, so it can be written as an outer product of two, two-vector Weyl (helicity) spinors:

$$p_{\dot{\alpha}\alpha} = \lambda_\alpha \tilde{\lambda}_{\dot{\alpha}} \tag{4.1.3}$$

where an explicit case is

$$\begin{array}{lllll}
\textbf{la} & \lambda^\alpha & \equiv \frac{1}{\sqrt{p^0+p^3}} \begin{bmatrix} p^0+p^3 \\ p^1+ip^2 \end{bmatrix} & = \frac{1}{\sqrt{p_+}} \begin{bmatrix} p_+ \\ p_+^\perp \end{bmatrix} \\
\textbf{lat} & \tilde{\lambda}^{\dot{\alpha}} & \equiv \frac{1}{\sqrt{p^0+p^3}} [\, p^0+p^3, p^1-ip^2)] & = \frac{1}{\sqrt{p_+}} [\, p_+,\ p_-^\perp] \\
\textbf{cla} & \lambda_\alpha & \equiv \frac{1}{\sqrt{p^0+p^3}} [-(p^1+ip^2),\ p^0+p^3] & = \frac{1}{\sqrt{p_+}} [-p_+^\perp, p_+] \\
\textbf{clat} & \tilde{\lambda}_{\dot{\alpha}} & \equiv \frac{1}{\sqrt{p^0+p^3}} \begin{bmatrix} -(p^1-ip^2), \\ p^0+p^3 \end{bmatrix} & = \frac{1}{\sqrt{p_+}} \begin{bmatrix} -p_-^\perp \\ p_+ \end{bmatrix}
\end{array}$$

As usual the Levi-Civita tensors raise and lower the spinor indices:

$$\lambda^\alpha = \epsilon^{\alpha\beta}\lambda_\beta \qquad \lambda_\alpha = \epsilon_{\alpha\beta}\lambda^\beta \qquad \tilde{\lambda}^{\dot{\alpha}} = \epsilon^{\dot{\alpha}\dot{\beta}}\tilde{\lambda}_{\dot{\beta}} \qquad \tilde{\lambda}_{\dot{\alpha}} = \epsilon_{\dot{\alpha}\dot{\beta}}\tilde{\lambda}^{\dot{\beta}} \tag{4.1.4}$$

Spinors λ_i and $\tilde{\lambda}_i$ are independent when extended into the complex plane.

For Weyl spinors we define the short-hand notation

$$\underset{\textbf{la}}{|i\rangle \equiv \lambda^\alpha(p_i)} \quad \underset{\textbf{cla}}{\langle i| \equiv \lambda_\alpha(p_i)} \quad \underset{\textbf{lat}}{[i| \equiv \tilde{\lambda}^{\dot{\alpha}}(p_i)} \quad \underset{\textbf{clat}}{|i] \equiv \tilde{\lambda}_{\dot{\alpha}}(p_i)}$$

4D spinor contractions and products

Lorenz-invariant quantities are constructed using the contraction of angle and square bracket spinors:

$$\begin{array}{ll}
\textbf{sp22} & \langle ij\rangle \equiv \lambda^\alpha(p_i)\lambda_\alpha(p_j) \\
& [ij] \equiv \tilde{\lambda}_{\dot{\alpha}}(p_i)\tilde{\lambda}^{\dot{\alpha}}(p_j)
\end{array}$$

Contracted with the Pauli matrices Weyl spinors may also form a vector product

$$\langle i|\sigma^\mu|j]$$

The vector product has the property that it allows the original 4-vector to be recovered:

$$p^\mu = \frac{\langle p|\sigma^\mu|p]}{2} \tag{4.1.5}$$

As usual, we define the Mandelstam variable:

$$\textbf{sp4ij} \quad s_{ij} = (p_i + p_j)^2$$

In the case where $p_i^2 = p_j^2 = 0$ we have

$$s_{ij} = 2p_i.p_j = p_i^{\alpha\dot{\alpha}} p_{j\alpha\dot{\alpha}}$$

Properties of products:

- $\langle ij\rangle = -\langle ji\rangle$
- $[ij] = -[ji]$
- $\langle ii\rangle = [ii] = 0$
- $s_{ij} = \langle ij\rangle[ji]$
- Schouten identity:
  $|i\rangle\langle jk\rangle = |j\rangle\langle ik\rangle + |k\rangle\langle ji\rangle$
  $|i][jk] = |j][ik] + |k][ji]$
- Fierz identity:
  $\langle i|\sigma^\mu|j]\langle k|\sigma_\mu|l] = 2\langle ik\rangle[lj]$

Evaluating spinor strings [12]: multiplying a spinor string by $1 = [i_4 i_1]\langle i_i i_4\rangle / s_{i_1 i_4}$, etc, enables any spinor string to be broken up into strings of length

- two, as above, $\langle ij\rangle[ji] = s_{ij}$
- and four, which can be evaluated using the Dirac trace:

$$\begin{aligned}\langle ij\rangle[jl]\langle lm\rangle[mi] &= \mathrm{tr}\Big(\tfrac{1}{2}(1-\gamma_5)\ \not{p}_i\ \not{p}_j\ \not{p}_l\ \not{p}_m\Big) \\ &= \tfrac{1}{2}\Big[s_{ij}s_{lm} - s_{il}s_{jm} + s_{im}s_{jl} - 4i\epsilon(i,j,l,m)\Big]\end{aligned} \tag{4.1.6}$$

  where $\epsilon(i,j,l,m) = \epsilon_{\mu\nu\sigma\rho} p_i^\mu p_j^\nu p_l^\sigma p_m^\rho$

4D Polarisation vectors

Take a massless reference vector q^μ. Then the polarization vectors for massless particles with momentum p are:

$$\varepsilon_+^\mu(p,q) = \frac{\langle q|\sigma^\mu|p]}{\sqrt{2}\langle qp\rangle} \quad \text{and} \quad \varepsilon_-^\mu(p,q) = \frac{\langle p|\sigma^\mu|q]}{\sqrt{2}[pq]}, \tag{4.1.7}$$

which obey the relations

$$p.\varepsilon_\pm(p,q) = 0, \qquad \varepsilon_\pm(p,q).\varepsilon_\pm(p,q) = 0, \qquad \varepsilon_\pm(p,q).\varepsilon_\mp(p,q) = -1 \tag{4.1.8}$$

and completeness relation

$$\varepsilon_+^\mu(p,q)\varepsilon_-^\nu(p,q) + \varepsilon_-^\mu(p,q)\varepsilon_+^\nu(p,q) = -g^{\mu\nu} + \frac{p^\mu q^\nu + q^\mu p^\nu}{p.q}. \tag{4.1.9}$$

Helicity states $\pm$ are produced by $\varepsilon_\pm$.

Recall we have chosen the convention that all momenta are considered to be outgoing. Helicity is labelled accordingly.

Solutions of 4D Dirac equation

Weyl (helicity) spinors λ_α and $\tilde{\lambda}^{\dot{\alpha}}$ are the building blocks for the positive an negative energy solutions of the massless, 4D Dirac equation

$$\gamma^\mu p_\mu \psi = 0. \tag{4.1.10}$$

Following Davies [49] we use the chiral representation of the Dirac matrices:

$$\textbf{gam4i}: \quad \gamma^0 = \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & 0 \end{bmatrix}$$
$$\gamma^i = \begin{bmatrix} 0 & \sigma_i \\ -\sigma_i & 0 \end{bmatrix}, i = 1,2,3$$
$$\gamma^5 = \begin{bmatrix} -\mathbf{1} & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$

Solutions of the 4D Dirac equation in this basis are:

$$\begin{aligned}
\mathbf{up} &= \begin{bmatrix}\mathrm{la}\\ 0\end{bmatrix}: & u_+ = v_- &= \begin{bmatrix}\lambda_\alpha\\ 0\end{bmatrix} & \equiv\ |p\rangle\\
\mathbf{um} &= \begin{bmatrix}0\\ \mathrm{clat}\end{bmatrix}: & u_- = v_+ &= \begin{bmatrix}0\\ \tilde{\lambda}^{\dot\alpha}\end{bmatrix} & \equiv\ |p]\\
\mathbf{ubp} &= \begin{bmatrix}0 & \mathrm{lat}\end{bmatrix}: & \overline{u_+} = \overline{v_-} &= \begin{bmatrix}0 & \tilde{\lambda}_{\dot\alpha}\end{bmatrix} & \equiv\ [p|\\
\mathbf{ubm} &= \begin{bmatrix}\mathrm{cla} & 0\end{bmatrix}: & \overline{u_-} = \overline{v_+} &= \begin{bmatrix}\lambda^\alpha & 0\end{bmatrix} & \equiv\ \langle p|
\end{aligned}$$

Satisfying $\not{p}|p\rangle = 0 = \not{p}|p]$. (Note that here $|p\rangle$ and $|p]$ are not Weyl spinors.)

4.2 Six-dimensional basics and notation

The 6D spinor helicity formalism used here follows that established by Cheung and O'Connell [55] and explicated by Bern et al [45] and Davies [49].

It is helpful to begin this section with a summary of the indices used in this formalism in the context of six-dimensional spinors. To begin with, the spinor representation of SO(5,1)[1] is used to express a vector p_{AB} in six dimensions, where A, B are fundamental representation indices of the covering group, SU*(4). SO(6) is the special orthogonal group with 6×6 matrices of determinant 1; SO(5,1) specifies a quadratic form having 5 positive eigenvalues and 1 negative. This corresponds to 5 spatial dimensions and one timelike. SU(4) is the complex Lie group of 4x4 unitary matrices with determinant 1 isomorphic to SO(6), and SU*(4) is the compact real form of SU(4), i.e it is a real Lie group that has the same complexification as SU(4), and is also isomorphic to SO(6). Since this group has a simpler structure than SO(6) it is common to use it in applications.

The indices used in this section, then, are:

SU*(4) fundamental labels	$A, B, ... = 1, 2, 3, 4$
SO(5,1) vector labels	$\mu, \nu, ... = 0, 1, 2, 3, 4, 5$
SU(2) x SU(2) helicity labels	$a, b, ... = 1, 2$ and $\dot{a}, \dot{b}, ... = 1, 2$

Note that the SU*(4) fundamental labels $A, B = 1, 2, 3, 4$ cannot be raised and lowered.

[1] SO(6) is the group of rotations in 6 dimensions, 6x6 orthogonal matrices with determinant 1.

For any object with little group indices, the convention is that the first index labels the rows and the second the columns.

It is possible to identify 6D objects by the presence of little group indices $a, \dot{a}$ but, to aid clarity in notation, for 6D objects we also replace 4D notation equivalents with upper case:

$$p^\mu \to P^\mu \qquad s_{ij} \to S_{ij} \qquad \sigma^\mu \to \Sigma^\mu \qquad \lambda^\alpha \to \Lambda^A \qquad \tilde{\lambda}_{\dot{\alpha}} \to \tilde{\Lambda_A} \qquad \varepsilon^\mu_\pm \to \mathcal{E}^\mu_{a\dot{a}}$$

4.2.1 6D Pauli matrices

To satisfy the Clifford Algebra

$$\Sigma^\mu \Sigma^\nu + \Sigma^\nu \Sigma^\mu = 2\eta^{\mu\nu}$$

The chosen 6D equivalent of the 4D Pauli matrices are:

$$\begin{aligned} \textbf{sig6i} \quad & \Sigma^0 \equiv i\sigma_1 \otimes \sigma_2 \\ & \Sigma^1 \equiv i\sigma_2 \otimes \sigma_3 \\ & \Sigma^2 \equiv -\sigma_2 \otimes \sigma_0 \\ & \Sigma^3 \equiv -i\sigma_2 \otimes \sigma_1 \\ & \Sigma^4 \equiv -\sigma_3 \otimes \sigma_2 \\ & \Sigma^5 \equiv i\sigma_0 \otimes \sigma_2 \end{aligned}$$

So:

$$\Sigma^0_{AB} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \quad \Sigma^1_{AB} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \Sigma^2_{AB} = \begin{bmatrix} 0 & 0 & i & 0 \\ 0 & 0 & 0 & i \\ -i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{bmatrix}$$

$$\Sigma^3_{AB} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \Sigma^4_{AB} = \begin{bmatrix} 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix} \quad \Sigma^5_{AB} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

and

sigt6i
$$\begin{aligned}\tilde{\Sigma}^0 &= -\Sigma^0\\ \tilde{\Sigma}^1 &= \Sigma^1\\ \tilde{\Sigma}^2 &= -\Sigma^2\\ \tilde{\Sigma}^3 &= \Sigma^3\\ \tilde{\Sigma}^4 &= -\Sigma^4\\ \tilde{\Sigma}^5 &= \Sigma^5.\end{aligned}$$

4.2.2 6-momentum

Contracting 6D Pauli matrices with 6-momentum P^μ gives

$$P_{AB} = P_\mu \Sigma^\mu_{AB} = \begin{bmatrix} 0 & -(ip_4+p_5) & -(p_1+ip_2) & p_0+p_3 \\ ip_4+p_5 & 0 & -(p_0-p_3) & p_1-ip_2 \\ p_1+ip_2 & p_0-p_3 & 0 & ip_4-p_5 \\ -(p_0+p_3) & -(p_1-ip_2) & -(ip_4-p_5) & 0 \end{bmatrix} \tag{4.2.1}$$

$$\mathbf{slash(p)} = \begin{bmatrix} 0 & \mathbf{-pmasst} & \mathbf{-pperp} & \mathbf{pplus} \\ \mathbf{pmasst} & 0 & \mathbf{-pminus} & \mathbf{pperm} \\ \mathbf{pperp} & \mathbf{pminus} & 0 & \mathbf{-pmass} \\ \mathbf{-pplus} & \mathbf{-pperm} & \mathbf{pmass} & 0 \end{bmatrix}$$

Likewise we have

$$P^{AB} = P_\mu \tilde{\Sigma}^{\mu AB} = \begin{bmatrix} 0 & ip_4-p_5 & -(p_1-ip_2) & -(p_0-p_3) \\ -(ip_4-p_5) & 0 & p_0+p_3 & p_1+ip_2 \\ p_1-ip_2 & -(p_0+p_3) & 0 & -(ip_4+p_5) \\ p_0-p_3 & -(p_1+ip_2) & ip_4+p_5 & 0 \end{bmatrix}$$

$$\mathbf{slasht(p)} = \begin{bmatrix} 0 & \mathbf{-pmass} & \mathbf{-pperm} & \mathbf{-pminus} \\ \mathbf{pmass} & 0 & \mathbf{pplus} & \mathbf{pperp} \\ \mathbf{pperm} & \mathbf{-pplus} & 0 & \mathbf{-pmasst} \\ \mathbf{pminus} & \mathbf{-pperp} & \mathbf{pmasst} & 0 \end{bmatrix} \tag{4.2.2}$$

Our approach will shortly require us to express spinors for 6-momenta which have $p^4, p^5 \neq 0$ in terms of spinors for pairs of 4-momenta which each have $p^4, p^5 = 0$, using the relationship

$$^{(4)}p_{\alpha\dot{\alpha}} = \lambda_\alpha \tilde{\lambda}_{\dot{\alpha}} + \rho\mu_\alpha \tilde{\mu}_{\dot{\alpha}} \tag{4.2.3}$$

where [2]

λ is the spinor for the null 4-vector $p^\flat$ defined in equation 4.2.4 below
μ is the spinor for the null 4-vector q, arbitrary except that $q.p \neq 0$
$\rho = \kappa\tilde{\kappa} = \kappa'\tilde{\kappa}' = \frac{m^2}{2\ ^{(4)}p.q}$, with:

ka	$\kappa \equiv \frac{m}{\langle\mu\lambda\rangle}$
kat	$\tilde{\kappa} \equiv \frac{\tilde{m}}{[\lambda\mu]}$
kap	$\kappa' \equiv \frac{\tilde{m}}{\langle\mu\lambda\rangle}$
kapt	$\tilde{\kappa}' \equiv \frac{m}{[\lambda\mu]}$
pmass	$m \equiv p_5 - ip_4$
pmasst	$\tilde{m} \equiv p_5 + ip_4$

We therefore treat the massless six-vector $P = (P^0P^1P^2P^3P^4P^5)$ as if it is a massive 4-vector $^{(4)}p = (P^0P^1P^2P^3)$, such that $(^{(4)}p)^2 = (P^4)^2 + (P^5)^2 = m^2$. Using the derivation in Appendix E we can then generate the 'flattened' 4-vector $p^\flat$, such that $(p^\flat)^2 = 0$, as follows:

$$\textbf{pflat} \quad p^\flat \equiv\ ^{(4)}p - \frac{m^2}{2\ ^{(4)}p.q}q \tag{4.2.4}$$

4.2.3 The 6D Dirac equation and constructing spinors

We use holomorphic and anti-holomorphic spinors Λ^A and $\tilde{\Lambda}_A$ to fulfil relations with the 6-momentum and 6D Pauli matrices similar to those in 4D.

The six-dimensional Λ matrices are the basis for solution of the six-dimensional Dirac equation:

$$P_\mu \Sigma^\mu_{AB} \Lambda^B = 0, \qquad P_\mu \tilde{\Sigma}^{\mu AB} \tilde{\Lambda}_B = 0 \tag{4.2.5}$$

However, as $P.\tilde{\Sigma}^{AB}$ has rank two and not one as in the 4D case, the spinors must carry an extra index. This index is the little group index $a, \dot{a} = 1, 2$. The 6D spinors are therefore 4×2 matrices. We have

[2] The convention here has a sign reversal compared to [49].

$$\begin{aligned}\epsilon^{ba}\Lambda_a^A\Lambda_b^B &= \Lambda^{Aa}\Lambda_a^B = P_\mu\tilde{\Sigma}^{\mu AB}\\ \tilde{\Lambda}_{A\dot{a}}\epsilon^{\dot{a}\dot{b}}\tilde{\Lambda}_{B\dot{b}} &= \tilde{\Lambda}_{A\dot{a}}\tilde{\Lambda}_B^{\dot{b}} = P_\mu\Sigma^\mu_{AB}\end{aligned} \tag{4.2.6}$$

where

A, B	SU*(4) indices label the rows
$a, \dot{a}$	little group indices with values 1 and 2 label the columns

Using the relationship in equation 4.2.3 we can write the 6D spinors Λ_a^A and $\Lambda_{A\dot{a}}$ in terms of $\lambda(p^\flat)$ and $\mu(q)$. The solutions to equation 4.2.5 then are:

sp6d $$\Lambda_a^A(p) = \begin{bmatrix}-\kappa\mu_\alpha & \lambda_\alpha\\ \tilde{\lambda}^{\dot{\alpha}} & \tilde{\kappa}\tilde{\mu}^{\dot{\alpha}}\end{bmatrix}_{aA} = \begin{bmatrix}\mathbf{-ka\ la(q)} & \mathbf{cla(pflat)}\\ \mathbf{lat(pflat)} & \mathbf{kat\ lat(q)}\end{bmatrix}$$

sp6td $$\tilde{\Lambda}_{A\dot{a}}(p) = \begin{bmatrix}\kappa'\mu^\alpha & \lambda^\alpha\\ -\tilde{\lambda}_{\dot{\alpha}} & \tilde{\kappa}'\tilde{\mu}_{\dot{\alpha}}\end{bmatrix}_{A\dot{a}} = \begin{bmatrix}\mathbf{kap\ la(q)} & \mathbf{la(pflat)}\\ \mathbf{-clat(pflat)} & \mathbf{kapt\ clat(q)}\end{bmatrix}$$

The little group indices are raised and lowered as usual by ϵ^{ab}, so

sp6u $$\Lambda^{Aa}(p) = \epsilon^{ab}\Lambda_b^A$$
sp6tu $$\tilde{\Lambda}_A^{\dot{a}}(p) = \epsilon^{\dot{a}\dot{b}}\tilde{\Lambda}_{A\dot{b}}$$

In this project external particles are in four dimensions, with the κ components all zero. Hence, for external particles the six-dimensional Λ spinors reduce to the conventional four-dimensional λ form:

$$\begin{bmatrix}0 & \lambda_\alpha(p)\\ \tilde{\lambda}^{\dot{\alpha}}(p) & 0\end{bmatrix} \tag{4.2.7}$$

4.2.4 6D spinor contractions and products

The 6D equivalent of the Mandelstam variable s_{ij} is:

sp6sij $$S_{ij} = (P_i + P_j)^2 = 2P_i.P_j$$

Lorentz-invariant inner products of 6D spinors products are defined by contraction of the SU*(4) indices as

sp62
$$\langle i^a|j_{\dot{b}}] \equiv \Lambda_i^{Aa}\tilde{\Lambda}_{jA\dot{b}} = [j_{\dot{b}}|i^a\rangle$$

$$\langle i_a|j^{\dot{b}}] \equiv \Lambda_{ia}^{A}\tilde{\Lambda}_{jA}^{\dot{b}} = [j^{\dot{b}}|i_a\rangle$$

$$\langle i_a|j_{\dot{b}}] \equiv \Lambda_{ia}^{A}\tilde{\Lambda}_{jA\dot{b}} = [j_{\dot{b}}|i_a\rangle$$

$$\langle i^a|j^{\dot{b}}] \equiv \Lambda_i^{Aa}\tilde{\Lambda}_{jA}^{\dot{b}} = [j^{\dot{b}}|i^a\rangle.$$

The following spinor contractions with the SU*(4)-invariant Levi-Civita tensor will be necessary for amplitude calculations:

sp64
$$\langle i^a j^b k^c l^d\rangle \equiv \epsilon_{ABCD}\Lambda_i^{Aa}\Lambda_j^{Bb}\Lambda_k^{Cc}\Lambda_l^{Dd}$$

$$\langle i_a j_b k_c l_d\rangle \equiv \epsilon_{ABCD}\Lambda_{ia}^{A}\Lambda_{jb}^{B}\Lambda_{kc}^{C}\Lambda_{ld}^{D}$$

$$[i^{\dot{a}} j^{\dot{b}} k^{\dot{c}} l^{\dot{d}}] \equiv \epsilon^{ABCD}\tilde{\Lambda}_{iA}^{\dot{a}}\tilde{\Lambda}_{jB}^{\dot{b}}\tilde{\Lambda}_{kC}^{\dot{c}}\tilde{\Lambda}_{lD}^{\dot{d}}$$

$$[i_{\dot{a}} j_{\dot{b}} k_{\dot{c}} l_{\dot{d}}] \equiv \epsilon^{ABCD}\tilde{\Lambda}_{iA\dot{a}}\tilde{\Lambda}_{jB\dot{b}}\tilde{\Lambda}_{kC\dot{c}}\tilde{\Lambda}_{lD\dot{d}}.$$

And for calculations involving scalars there will also be contractions of little-group indices:

$$\langle i_a j_b k_c k^c\rangle \equiv \epsilon_{ABCD}\Lambda_{ia}^{A}\Lambda_{jb}^{B}\Lambda_{ke}^{C}\Lambda_{k}^{De}$$

$$[i_{\dot{a}} j_{\dot{b}} k_{\dot{e}} k^{\dot{e}}] \equiv \epsilon^{ABCD}\tilde{\Lambda}_{iA\dot{a}}\tilde{\Lambda}_{jB\dot{b}}\tilde{\Lambda}_{kC\dot{e}}\tilde{\Lambda}_{kD}^{\dot{e}}.$$

Spinor products have the following properties:

- $\langle i^a|i_{\dot{a}}] = 0$
- $\det(\langle i_a|j_{\dot{b}}]) = S_{ij}$, where the determinant is taken over the little group indices
- $\langle ijkl\rangle\langle m| + \langle jklm\rangle\langle i| + \langle klmi\rangle\langle j| + \langle lmij\rangle\langle k| + \langle mijk\rangle\langle l| = 0$ for the chiral spinors, which is the 6D generalisation of the Schouten identity. The equivalent for anti-chiral spinors also stands.

Amplitude calculations also require spinor strings, which are defined in general as:

$$\begin{aligned}\langle i^a|\ \not{p}_1\ \not{p}_2 ...\ \not{p}_{2n+1}|j^b\rangle &= (\Lambda_i)^{A_1 a}(p_1)_{A_1 A_2}(p_2)^{A_2 A_3}...(p_{2n+1})_{A_{2n+1}A_{2n+2}}(\Lambda_j)^{A_{2n+2}b}\\ \langle i^a|\ \not{p}_1\ \not{p}_2 ...\ \not{p}_{2n}|j_{\dot{b}}] &= (\Lambda_i)^{A_1 a}(p_1)_{A_1 A_2}(p_2)^{A_2 A_3}...(p_{2n})^{A_{2n}A_{2n+1}}(\tilde{\Lambda}_j)_{A_{2n+1}\dot{b}}\end{aligned} \tag{4.2.8}$$

In practice for the tree amplitudes in the amplitude catalogue in Appendix A we require only strings up to a $\not{p}_4$ term. Hence we write:

$$\langle i^a|\ \not{p}_1|j^b\rangle = (\Lambda_i)^{A_1 a}(p_1)_{A_1 A_2}(\Lambda_j)^{A_2 b} \tag{4.2.9}$$

$$\mathbf{sp6s1} \quad = \mathbf{sp6}_i\ \mathbf{psig}_1\ \mathbf{sp6}_j$$

$$\langle i^a|\ \not{p}_1\ \not{p}_2|j^b] = (\Lambda_i)^{A_1 a}(p_1)_{A_1 A_2}(p_2)^{A_2 A_3}(\tilde{\Lambda}_j)_{A_3 b} \tag{4.2.10}$$

$$\mathbf{sp6s2} \quad = \mathbf{sp6}_i\ \mathbf{psig}_1\ \mathbf{psig}_2\ \mathbf{sp6}_j$$

$$\langle i^a|\ \not{p}_1\ \not{p}_2\ \not{p}_3|j^b\rangle = (\Lambda_i)^{A_1 a}(p_1)_{A_1 A_2}(p_2)^{A_2 A_3}(p_2)_{A_3 A_4}(\Lambda_j)^{A_4 b} \tag{4.2.11}$$

$$\mathbf{sp6s3} \quad = \mathbf{sp6}_i\ \mathbf{psig}_1\ \mathbf{psig}_2\ \mathbf{psig}_3\ \mathbf{sp6}_j$$

$$\langle i^a|\ \not{p}_1\ \not{p}_2\ \not{p}_3\ \not{p}_4|j_{\dot{b}}] = (\Lambda_i)^{A_1 a}(p_1)_{A_1 A_2}(p_2)^{A_2 A_3}(p_3)_{A_3 A_4}(p_4)^{A_4 A_5}(\tilde{\Lambda}_j)_{A_5 \dot{b}} \tag{4.2.12}$$

$$\mathbf{sp6s4} \quad = \mathbf{sp6}_i\ \mathbf{psig}_1\ \mathbf{psig}_2\ \mathbf{psig}_3\ \mathbf{psig}_4\ \mathbf{sp6}_j$$

with the raised or lowered indices defined as required.

Vector products can also be defined

$$\langle i_a \Sigma^\mu j_b\rangle, \qquad [i_{\dot{a}} \tilde{\Sigma}^\mu j_{\dot{b}}], \tag{4.2.13}$$

which have a Lorentz-index and two little group indices and hence can be written as 6-vectors of 2 x 2 matrices. They have the properties

$$P^\mu = -\frac{1}{4}\langle P^a \Sigma^\mu P_a\rangle = -\frac{1}{4}[P_{\dot{a}} \tilde{\Sigma}^\mu P^{\dot{a}}] \tag{4.2.14}$$

$$\langle i_a \Sigma^\mu j_b \rangle P_{i\mu} = \langle i_a \Sigma^\mu j_b \rangle P_{j\mu} = [i_{\dot{a}} \tilde{\Sigma}^\mu j_{\dot{b}}] P_{i\mu} = [i_a \tilde{\Sigma}^\mu j_b] P_{j\mu} = 0 \tag{4.2.15}$$

$$\langle i_a \Sigma^\mu j_b \rangle [k_{\dot{c}} \tilde{\Sigma}_\mu l_{\dot{d}}] = 2\Big(\langle i_a l_{\dot{x}}] \langle j_b k_{\dot{c}}] - \langle i_a k_{\dot{c}}] \langle j_b l_{\dot{d}}] \Big) \tag{4.2.16}$$

4.2.5 6D polarisation vectors

6D polarisation vectors are written in terms of the vector products

$$\textbf{pol} \quad \mathcal{E}^\mu_{a\dot{a}}(P,Q) = \frac{-1}{\sqrt{2}} \frac{\langle P_a \Sigma^\mu Q_b \rangle}{\langle Q_b P^{\dot{a}}]} = \frac{1}{\sqrt{2}} \frac{[Q_{\dot{b}} \tilde{\Sigma}^\mu P_{\dot{a}}]}{\langle P^a Q_{\dot{b}}]} \tag{4.2.17}$$

where Q is the axial reference vector, and a redefinition of Q will hence correspond to a gauge transformation.

Note that the polarisation states cannot be labelled as $+$ or $-$ because in six dimensions gluons have four polarisation states, labelled by $a, \dot{a}$. The six-dimensional polarisation states are related to four-dimensional states: when the reference vector is four-dimensional, states with $(a, \dot{a})$ being either $(1, \dot{1})$ or $(2, \dot{2})$ correspond to positive and negative helicity states, respectively, whilst labels $(1, \dot{2})$, $(2, \dot{1})$ correspond in four dimensions to scalars. Of course, the specific map between four-dimensional helicities and six-dimensional quantum numbers depends on the particular embedding of four-dimensional spinors in the six-dimensional space [45].

We have the following completeness relationship:

$$\mathcal{E}^\mu_{11}\mathcal{E}^\nu_{22} + \mathcal{E}^\mu_{22}\mathcal{E}^\nu_{11} - \mathcal{E}^\mu_{12}\mathcal{E}^\nu_{21} - \mathcal{E}^\mu_{21}\mathcal{E}^\nu_{12} = -g^{\mu\nu} + \frac{P^\mu Q^\nu + Q^\mu P^\nu}{P.Q}. \tag{4.2.18}$$

4.3 Building tree amplitudes in six-dimensions

4.3.1 6D BCFW recursion relation

Using the BCFW principles outlined in Section 3.3.2 a six-dimensional BCFW relation is given in [55] as

$$x^a\tilde{x}^{\dot{a}}A_{a\dot{a}b\dot{b}\ldots}(p_1,p_2,...) = \sum_{L,R}\sum_{c\dot{c}}(-\frac{i}{k^2})x^a\tilde{x}^{\dot{a}}A_{a\dot{a}c\dot{c}}(\hat{p}_1(z^*),...,\hat{k}) \times A^{c\dot{c}}_{b\dot{b}}(\hat{p}_2(z^*),...,-\hat{k}) \tag{4.3.1}$$

where:

$x^a\tilde{x}^{\dot{a}} = X^{a\dot{a}}$	labels the deformation of the shift, i.e.
	$\hat{p}_1 = p_1 + zX^{a\dot{a}}\epsilon_{1a\dot{a}}$
	$\hat{p}_2 = p_2 - zX^{a\dot{a}}\epsilon_{1a\dot{a}}$
z	is the complex shift paramter
A	are tree amplitudes
L, R	sums over partitions of the external legs into two groups
$c\dot{c}$	is the polarization of the intermediate leg
k	is the physical momentum of the intermediate leg
$\hat{k}$	is shifted momentum of the intermediate leg
...	denotes the other external momenta

4.3.2 Four-point tree amplitude

Using BCFW and a derived expression for the six-dimensional three-point amplitude, Cheung and O'Connell find that the colour-ordered Yang Mills, six-dimensional four-point amplitude is simply

$$A_4(1_{a\dot{a}}, 2_{b\dot{b}}, 3_{c\dot{c}}, 4_{d\dot{d}}) = -\frac{i}{st}\langle 1_a 2_b 3_c 4_d\rangle[1_{\dot{a}}2_{\dot{b}}3_{\dot{c}}4_{\dot{d}}] \tag{4.3.2}$$

Whilst calculation of this four-point amplitude is straightforward, numerical calculation of the three-point necessary to obain higher-point amplitudes is not. Therefore I leave further discussion about six dimensional tree amplitudes to the implementation Chapter 5.

Chapter 5

Implementing 6D spinor helicity in practice

5.1 Introduction

In the Chapter 6 roadmap we see that much of the functionality needed to carry out the multi-leg NLO amplitude calculations we seek is already available. The roadmap sets out a full procedure by which both the cut and rational parts of the loop amplitude can be obtained. The crucial functionality that is missing in order to implement the solution is the capacity to calculate multi-leg six-dimensional tree amplitudes. For this we need an effective, six-dimensional BCFW recursion.

In the previous chapter we outlined the spinor helicity formalism as it stands in theory. Working on the implementation, it took a while before it became apparent that the principal challenge lies with the calculation of three-point amplitudes. These are essential components of the BCFW strategy, but their calculation is not straightforward. Cheung and O'Connell [55] themselves offered a four-dimensional, special case of the approach in their paper. Bern et al [45] did not go beyond this, and for the analytic loop-amplitude approach constructed by Davies [49] only four-point amplitudes were used. Badger et al [81] presented numerical code consistent with the four-dimensional, simplified construction of a three-point amplitude in Cheung and O'Connell's paper, but did not use it in their four-point calculation. In order to meet the aims of the current project, it is necessary to compute fully six-dimensional, three-point amplitudes.

In this chapter we address the six-dimensional three-point amplitude and BCFW construction for all helicities. We set out the practical implementation of a numerical approach to the construction of six-dimensional tree amplitudes and a six-dimensional BCFW method. Together, these create the missing functionality.

In fact, we present two approaches to the code: the initial development, described in Sections 5.2 to 5.4, includes full functionality but retains consistency problems relating to multiples of -1 and contraction of indices within the multi-layer, nested manipulation of the six-dimensional spinors and the 2x1 objects derived from spinor products. Although after much work many of the problems have been eradicated, it is clear that this implementation is not a sound foundation for further development. Using the lessons learned during development of the first implementation a second approach has been tried successfully and is presented as an addendum in Section 5.5. This code, which uses a bespoke package for tensor definitions **tensors.py**, has been written by Daniel Maitre.

The original application and that in the addendum are both coded in Python. The sections in this chapter are related to the modules in the Python code. The code itself is presented in Jupyter notebook form in Appendix F.

5.2 Basic building blocks

module: utils

Utils is used by all other modules in the package. Then necessary four- and six-dimensional sigma and gamma matrices, levi civita tensors, Minkowski products, slashed momenta and Mandelstam variable functions are gathered together in this module. It also contains the $P_\flat$ function that is necessary for the construction of the six-dimensional spinors.

module: randrot

In order to test our amplitude calculations we need to be able to generate the six-dimensional spinors from rotated momenta so that they retain Lorentz invariance and can be used to compare against the results of calculations using

the original four-dimensional momenta values.

The **randrot** module handles rotation of four-dimensional momenta into six dimensions and generates random momenta and random phase space points in both dimensions. It also constructs the required rotated spinors. The getSpinors function which does this is derived from a separate module, **lieAlgebra**.

module: spinors

The straightforward basic elements of the formalism described in the previous chapter are included in module **spinors**, including calculation of the κ components of the spinor solutions Λ_a^A and $\tilde{\Lambda}_{A\dot{a}}$ to equation 4.2.5, and the six-dimensional spinor products that are necessary for amplitude calculations.

The convention that all momenta are outgoing is used throughout, and the function **momSpinors** handles negative energy momenta by applying $\Lambda_{-p} = i\Lambda_p$.

5.3 Trees and BCFW in six dimensions

Module: tree_with_spinors

5.3.1 Four-point tree amplitudes

Cheung and O'Connell provided the very nice analytic expression for the four-point, all gluon amplitude in equation 4.3.2, which they derived from six-dimensional BCFW using analytic manipulation of the expression for three-point amplitudes.

In the **tree_with_spinors** module we begin by calculating the four-point amplitude using equation 4.3.2 with spinors and spinor products defined in the **spinors** module. For phase space points of external particles the result agrees with output from the S@M package.

5.3.2 Implementing a 6d BCFW approach

In brief, the route to calculating a BCFW amplitude from two factorised amplitudes with a shift by a complex parameter z between P_1 and P_2 as shown in

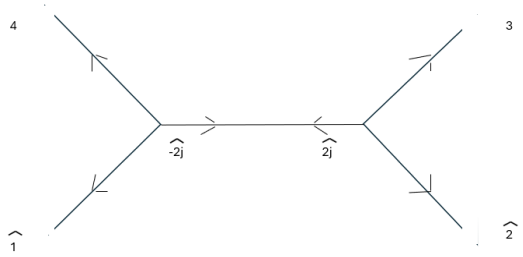


Figure 5.3.1: Momenta for BCFW split between P1 and P2

Figure 5.3.1 is as follows:

- Create an auxiliary matrix $X^{a\dot{a}} = x^a \tilde{x}^{\dot{a}}$ where $x^a, \tilde{x}^{\dot{a}}$ are arbitrary.
- Define polarisation vector for P_1 with respect to P_2.
- Calculate a null vector r which will satisfy $P_1.r = P_2.r = 0$.
- Calculate sum of momenta, $P_{2j} = P_2 + ... + P_j$.
- Calculate z_{2j}, the location of the pole at $P_{2j}(z)^2 = 0 : \quad z_{2j} = \frac{-P_{2_j}^2}{2r.P_{2j}}$.
- Calculate shifted spinors for $\hat{P}_1 = P_1 + zr, \hat{P}_2 = P_2 - zr$ using C&O'C eqns 5.7-10.
- Use spinors to calculate unshifted amplitude as $X^{a\dot{a}} A_n^{tree}(0) = \sum_{j=3}^{n-1} \sum_h X^{a\dot{a}} L(\hat{P}_2, ..., P_j, -\hat{P}_{2_j}^{-h}) \times \frac{1}{P_{2j}^2} A_R(\hat{P}_{2j}^h, P_{j+1}, ..., \hat{P}_1)|_{z=z_{2j}}$.

Each of these steps is addressed in turn in Sections 5.3.3 to 5.3.9. The most difficult part of this process is the construction of three-point amplitudes, which will be necessary for all higher-point amplitudes in this numerical application, is addressed separately in Section 5.4.

5.3.3 Auxiliary matrix

$$\begin{aligned} \textbf{xuu} \quad & X^{a\dot{a}} = x^a \tilde{x}^{\dot{a}} \\ \textbf{xdd} \quad & X_{a\dot{a}} = x_a \tilde{x}_{\dot{a}} \\ \textbf{xud} \quad & X_{\dot{a}}^{a} = x^a \tilde{x}_{\dot{a}} \\ \textbf{xdu} \quad & X_{a}^{\dot{a}} = x_a \tilde{x}^{\dot{a}} \end{aligned}$$

where x^a is arbitrary and we have used

$$x^a = \tilde{x}^{\dot{a}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

so that

$$X^{a\dot{a}} = \begin{bmatrix} 1 & 0 \\ 0 & 0. \end{bmatrix} \tag{5.3.1}$$

5.3.4 Polarisation vector

With a null reference vector Q, such that $P.Q \neq 0$, there are associated spinors $P = |q^a\rangle\langle q^b|\epsilon_{ab}$ and $Q = |q_{\dot{a}}][q_{\dot{b}}|\epsilon^{\dot{a}\dot{b}}$. The polarisation vectors are then defined to be

$$\begin{aligned} \textbf{pol} \quad \mathcal{E}^{\mu}_{a\dot{a}} &= \frac{1}{\sqrt{2}}\langle p_a|\Sigma^{\mu}|q_b\rangle(\langle q_b|p^{\dot{a}}])^{-1} \\ &= \frac{1}{\sqrt{2}}(\langle p^a|q_{\dot{b}}])^{-1}[q_{\dot{b}}|\tilde{\Sigma}^{\mu}|p_{\dot{a}}] \end{aligned} \tag{5.3.2}$$

where the object

$$(\langle p^a|q_{\dot{b}}])^{-1} = -\frac{\langle p_a|q^{\dot{b}}]}{2P.Q}. \tag{5.3.3}$$

5.3.5 Shift vector

The vector for the shift is a null vector with the properties of a polarisation vector. It is obtained from

$$\textbf{rv} \quad r^{AB} = \frac{1}{\sqrt{2}}X^{a\dot{a}}(\mathcal{E}_1^{AB})_{a\dot{a}} \tag{5.3.4}$$

For the purposes of our BCFW shifted momenta we will use the polarisation of momentum P_1 and choose the reference spinor for the polarisation vector to be Λ_2. Note that $P_1.r = r.P_2 = 0$.

5.3.6 Calculate z and the location of the pole

Calculate location of pole for the shift between legs one and two:

$$\textbf{zPole} \quad z_{2j} = \frac{-P^2_{2_j}}{2r.P_{2j}} \tag{5.3.5}$$

where $P_{2j} = P_2 + ... + P_j$.

5.3.7 Shifted momenta

The vector for the shift is a null vector with the properties of a polarisation vector: $r^{AB} = \frac{1}{\sqrt{2}} X^{a\dot{a}} (\mathcal{E}_1^{AB})_{a\dot{a}}$. For the purposes of our BCFW shifted momenta we will use the polarisation of momentum P_1 and choose the reference spinor for the polarisation vector to be Λ_2.

The shifted momenta are in practice obtained by using the **pFromS6D** function which encodes equation 4.2.14 to extract them from their **ZSpinors**, which are described in the following section. Alternatively, with a shift between P_i and P_j the shifted momenta are given by

$$\begin{aligned} \textbf{pHat1} \quad & \hat{p}_i = p_i + z X^{a\dot{a}} \mathcal{E}_{ia\dot{a}} \\ \textbf{pHat2} \quad & \hat{p}_j = p_j - z X^{a\dot{a}} \mathcal{E}_{ia\dot{a}} \end{aligned} \tag{5.3.6}$$

where z is the shift, X is the auxiliary vector $X^{a\dot{a}} = x^a \tilde{x}^{\dot{a}}$ and $\mathcal{E}$ is the polarisation vector.

The third momentum in each left-, right-hand set of momenta is that of the internal propagator, **pHat2j** $\pm \hat{P}_{2j}$. The spinor for this momentum is found by summing the other two momenta and then using the normal **momSpinors** construction in module **spinors**.

5.3.8 Spinors for shifted momenta

Class: ZSpinors

The spinor construction of the deformed external legs $\hat{p}_i$ cannot be derived from dropping the shifted momenta into the usual code momSpinors in the spinors module. If we simply apply the normal prescription to generate the six-dimensional spinors from these transformed momenta the two columns of the resulting spinors emerge with a non-trivial distortion: the two columns are not independently Lorentz invariant. The correct approach is to construct the shifted spinors directly from spinors and z. The expressions for these shifted spinors that are given in Cheung and O'Connell, Bern et al and Davies are not completely consistent. The form of the spinors that has been found to work in practice is:

$$
\begin{aligned}
\textbf{zspu1} \quad & \Lambda_{\hat{1}}^{Aa} = \Lambda_1^{Aa} - \frac{z}{s_{12}} X_{\dot{a}}^a [1^{\dot{a}}|2^b\rangle \Lambda_{2b}^A \\
\textbf{zspu2} \quad & \Lambda_{\hat{2}}^{Ab} = \Lambda_2^{Ab} - \frac{z}{s_{12}} X_{\dot{a}}^a \Lambda_{1a}^A [1^{\dot{a}}|2^b\rangle \\
\textbf{zsptd1} \quad & \tilde{\Lambda}_{\hat{1}A\dot{a}} = \tilde{\Lambda}_{1A\dot{a}} + \frac{z}{s_{12}} X_{\dot{a}}^a \langle 1_a|2_{\dot{b}}] \tilde{\Lambda}_{2A}^{\dot{b}} \\
\textbf{zsptd2} \quad & \tilde{\Lambda}_{\hat{2}A\dot{b}} = \tilde{\Lambda}_{2A\dot{b}} + \frac{z}{s_{12}} X_{\dot{a}}^a \tilde{\Lambda}_{1A}^{\dot{a}} \langle 1_{\dot{a}}|2_{\dot{b}}]
\end{aligned}
\tag{5.3.7}
$$

where, as usual, a hatted $\hat{P}$ indicates a shifted momentum.

5.3.9 Calculate the BCFW 4-point from 3-point amplitudes

The general expression for six-dimensional BCFW is

$$
\begin{aligned}
x^a \tilde{x}^{\dot{a}} A_n^{tree}(0) = \sum_{j=3}^{n-1} \sum_h & x^a \tilde{x}^{\dot{a}} A_L(\hat{P}_2, ..., P_j, -\hat{P}_{2j}^{-h}) \\
& \times \frac{i}{P_{2j}^2} A_R(\hat{P}_{2j}^{(h)}, P_{j+1}, ..., \hat{P}_1)|_{z=z_{2j}}
\end{aligned}
\tag{5.3.8}
$$

In the specific implementation of a four-point amplitude from two three-points, for which there is only one diagram, we have:

$$
x^a \tilde{x}^{\dot{a}} A_{4;a\dot{a}b\dot{b}c\dot{c}d\dot{d}} = \frac{i}{t} x^a \tilde{x}^{\dot{a}} A_{L;a\dot{a}e\dot{e}d\dot{d}} A_{R;b\dot{b}c\dot{c}}{}^{e\dot{e}} \tag{5.3.9}
$$

Where there is a sum over the $e, \dot{e}$ little group index which defines the helicity states of the intermediate propagator. The equation also demonstrates that a contraction over the auxiliary matrix is necessary on both sides.

All external particles can be only little group 1,1 or 2,2 (or scalar, as in the case of Higgs). Mixed little group indices, however, exist in the shifted momenta but are constrained by the requirement that the shifted momentum contribution to the left-hand amplitude must have the opposite helicity to that in the right-hand amplitude. The intermediate propagator can also have mixed helicity states but these are summed over.

The actual calculation of the left- and right-hand three-point amplitudes A_L and A_R is described below.

5.4 Three-point tree amplitudes

module: tree_with_spinors.ipynb

5.4.1 Structure of the amplitudes

The component Yang Mills, six-dimensional 3-point amplitude in equation 5.3.9 is [55]:

$$\textbf{g3Amp} \quad A_3(1_{a\dot{a}}, 2_{b\dot{b}}, 3_{c\dot{c}}) = i\Gamma_{abc}\tilde{\Gamma}_{\dot{a}\dot{b}\dot{c}} \tag{5.4.1}$$

where

$$\begin{aligned} \textbf{gam} \quad & \Gamma_{abc} = u_{1a}u_{2b}w_{3c} + u_{1a}w_{2b}u_{3c} + w_{1a}u_{2b}w_{3c} \\ \textbf{gamt} \quad & \tilde{\Gamma}_{\dot{a}\dot{b}\dot{c}} = \tilde{u}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{w}_{3\dot{c}} + \tilde{u}_{1\dot{a}}\tilde{w}_{2\dot{b}}\tilde{u}_{3\dot{c}} + \tilde{w}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{u}_{3\dot{c}} \end{aligned} \tag{5.4.2}$$

and the $u, \tilde{u}$ are 2×1 objects that are described in Section 5.4.2 below. For the right hand side when we use BCFW we need alternate helicity for the shift vector, which for illustration purposes is labelled here $3^{c\dot{c}}$:

$$A_3(1_{a\dot{a}}, 2_{b\dot{b}}, 3^{c\dot{c}}) = i\Gamma_{ab}{}^{c}\tilde{\Gamma}_{\dot{a}\dot{b}}{}^{\dot{c}} \tag{5.4.3}$$

At least two of the momenta will be complex momenta shifted into six dimensions, and all helicity combinations - including those which are not utilised when the result is contracted with the auxiliary matrix - are calculated simultaneously. The $u, \tilde{u}$ components must be normalised to cyclical spinor products and, in order to ensure that the BCFW relation works, the inverse $w, \tilde{w}$ components for the three-point amplitude on each side of the factorisation must be consistent with each other and must also comply with momentum conservation. The practicality of all this is particularly tricky in six dimensions, which no doubt has played a part in the fact that no numerical implementation of the calculation, apart from the work reported here, so far exists.

In order to derive the $u, \tilde{u}, w, \tilde{w}$ we must first establish the left- and right-hand momenta groups, Section 5.3.2.

5.4.2 Find and normalise $u, \tilde{u}, w, \tilde{w}$

Function: uutilde

This function takes a set of three six-dimensional Λ and $\tilde{\Lambda}$ spinors as arguments and calculates cyclical products. Since the determinant of the matrix $|\langle i_a|j_{\dot{b}}]| = 0$, it is rank one. The aim is to express this rank one matrix as the product of two 2x1 matrices such that

$$u_{ia}\tilde{u}_{j\dot{b}} = \langle i_a|j_{\dot{b}}] \tag{5.4.4}$$

In their paper, Cheung and O'Connell [55] use a four-dimensional example for clarity and state a simple construction $u_i = [0, N_i]$ and $\tilde{u} = [0, \tilde{N}_i]$, where:

$$\begin{aligned} N_2 &= \frac{\langle 23\rangle}{\langle 31\rangle}N_1, \quad & N_3 &= \frac{\langle 23\rangle}{\langle 12\rangle}N_1, \\ \tilde{N}_1 &= \frac{\langle 12\rangle\langle 31\rangle}{\langle 23\rangle}\frac{1}{N_1}, \quad & \tilde{N}_2 &= \frac{\langle 23\rangle}{\langle 31\rangle}N_1, \quad \tilde{N}_3 = \frac{\langle 12\rangle}{N_1}, \end{aligned} \tag{5.4.5}$$

where the $\langle ij\rangle$ are four-dimensional spinors. is that of a ZSpinor. In practice, at least one of the three-point momenta is always a six-dimensional shifted leg and a second leg is formed by the internal six-dimensional propagator. Finally, when loop cuts become involved, the third momentum may also be similarly six-dimensional. None of this prevents us from obtaining the $u, \tilde{u}$ construction required by the three-point amplitude: the spinor products are all rank one matrices with zero determinants, so it is straightforward to express each of them as the product of two, two-by-one matrices normalised by choosing one of the N values and calculating each in rotation. However, a naive approach here has unpleasant repercussions when we move on to calculate the pseudoinverse objects $w, \tilde{w}$.

Equation 5.4.2 requires the construction of $w, \tilde{w}$, with the inverse condition

$$\begin{aligned} u_{ia}w_{j\dot{b}} - u_{i\dot{b}}w_{ia} &= \epsilon_{ab} \\ \tilde{u}_{i\dot{a}}\tilde{w}_{j\dot{b}} - \tilde{u}_{i\dot{b}}\tilde{w}_{i\dot{a}} &= \epsilon_{\dot{a}\dot{b}} \end{aligned} \tag{5.4.6}$$

For any object u an inverse can be found quite easily, though such an inverse is not unique. However, the inverses must be constructed such that:

1. momentum is conserved; and
2. the left-hand w for the internal momentum is consistent with the right-hand u spinor product values and vice versa.

The first requirement can be stated in the form

$$|w_1.1\rangle + |w_2.2\rangle + |w_3.3\rangle = 0 \tag{5.4.7}$$

Where the notation indicates contraction over the little group index, i.e.

$$w_{1a}|1^a\rangle + w_{2b}|2^b\rangle + w_{3c}|3^c\rangle = 0$$

With $u_i = [0, N_i]$ and $\tilde{u}_i = [0, \tilde{N}_i]$ Cheung and O'Connell construct a simple inverse

$$w_{ia} = \begin{bmatrix} \frac{1}{N_i} \\ B_i N_i \end{bmatrix}, \quad \tilde{w}_{ia} = \begin{bmatrix} \frac{1}{\tilde{N}_i} \\ B_i \tilde{N}_i \end{bmatrix} \tag{5.4.8}$$

where B_i is a constant, chosen to meet the momentum conservation condition equation equation 5.4.7.

An expression that meets the inverse condition for all including six-dimensional cases is found to be:

$$w_i = \frac{B_i}{u_i[0]} \begin{bmatrix} u_i[0] \\ u_i[1] \end{bmatrix} - \begin{bmatrix} 0 \\ 1/u_i[0] \end{bmatrix} \tag{5.4.9}$$

We can solve for B_i by substituting equation 5.4.9 into equation 5.4.7:

$$\sum_i \Lambda_i \left(\frac{B_i}{u_i[0]} \begin{bmatrix} u_i[0] \\ u_i[1] \end{bmatrix} - \begin{bmatrix} 0 \\ 1/u_i[0] \end{bmatrix} \right) = 0 \tag{5.4.10}$$

However, this approach does not necessarily lead to a satisfactory compliance with the second of our consistency requirements, i.e. that the left-hand w for the internal momentum is consistent with the right-hand u spinor product values and vice versa. Without the relation

$$w_{-\hat{P}_{2j}}.w_{\hat{P}_{2j}} = \frac{1}{u_{-\hat{P}_{2j}}.u_{\hat{P}_{2j}}} \tag{5.4.11}$$

the combination of the left- and right-hand amplitudes in BCFW does not produce the correct result.

It has been found that it is essential first to diagonalise the spinor product matrices, and this is the approach taken in **uutilde**. The construction and normalisation then become straightforward. In **uutilde**, the spinor product $\langle 1_a|2_{\dot{b}}]$ is used as the basis for both.

5.4.3 Calculate three-point amplitude

Function: spFor3g

We first obtain the spinors for each of the left- and right-hand amplitudes shown in Figure 5.3.1 .

Function **spFor3g** takes as input a four-dimensional (i.e. real) phase space point P_1, P_2, P_3, P_4 extended to a six-dimensional format, i.e. $P_4 = P_5 = 0$. For such a real phase space point all external particles are in four dimensions and the six-dimensional spinor construction reduces to the nice four-dimensional case in equation 4.2.7. The split is set to be between P_1 and P_2 (the order of the input momenta is, of course, adjustable).

With the convention that all momenta are outgoing, the cyclical order of the left-hand set of momenta is then determined to be $P_4, \hat{P}_1, -\hat{P}_{2j}$ and that of the right-hand set of momenta is $\hat{P}_2, P_3, \hat{P}_{2j}$.

Spinors for the unshifted external momenta P_3, P_4 are obtained simply from **momSpinors** in the **spinors** module, as are the spinors for the internal propagator $\hat{P}_{2j}$, as described in section 5.3.7. Those for the deformed external legs $\hat{P}_1$ and $\hat{P}_2$ are **ZSpinors**, Section 5.3.8.

Function: g3Gamma

Taking the family of $u, \tilde{u}, w, \tilde{w}$ from **uutilde** for either the left- or the right-hand side momenta as input, equation 5.4.2 is implemented.

Function: g3Amp

The amplitude calculation takes a real phase space point as input, obtains spinors from **spFor3g** and then separates these into left- and right-hand sets. It calls **uutilde** twice, using each of the spinor sets, and then uses the $u, \tilde{u}, w, \tilde{w}$ output from that function as input to **g3Gamma** in the two cases, to obtain both left- and right-hand results.

The function implements equation 5.4.1 and returns the left- and right-hand amplitudes, **g3L, g3R**.

5.5 Addendum: alternative code

The state of play with the six-dimensional BCFW four-point amplitude calculation in Section 5.3.9 is that some results are consistent with expectations but others are not. Test results for this code have been left in the **tree_with_spinors** Jupyter notebook in Appendix F. Repeated review and correction has led to steady improvement, almost all related to management of indices, contractions and factors of -1. What has become clear is that consistent manipulation of spinor and other objects in nested calculations requires a different approach to the basic operations if it is to be sufficiently robust to support further development.

A likely contender for such an alternative has been written by Daniel Maitre, and I include it in Appendix F because it provides important evidence that the six-dimensional spinor helicity approach is viable. There is still a residual factor of -1 that has to be adjusted in order for consistent results to be obtained, but undoubtedly the cause of that can be determined and corrected given more time.

Chapter 6

A road map for the full 1-loop calculation

6.1 Overview

In Chapter 3 we carried out a high-level review of options for a choice of methods to calculate a NLO virtual amplitude for the process $gg \to h + jj...$, which can be used by Monte-Carlo tools like Sherpa to calculate cross-sections. In the subsequent two chapters the focus has been on the six-dimensional spinor helicity approach to obtaining the tree amplitudes that are the building blocks of the calculation. The full family of these tree amplitudes was not previously available. The purpose of this chapter is to step back again from that focus and to ensure that the other parts that will be necessary to complete the framework are accessible. In fact this work was carried out before that reported in Chapters 4 and 5.

In this chapter I will set out all the steps required for the calculation, clearly identifying those which are already available from existing sources, and demonstrating how the six-dimensional spinor helicity formalism selected in the previous chapter is sewn into the total framework.

We begin with the generic form of a loop amplitude, which is

$$A_n^{L-loop} = \sum_j \int \Big(\prod_{l=1}^{L} \frac{d^D l_1}{(2\pi)^D} \Big) \frac{1}{S_j} \frac{n_j c_j}{\prod_{\alpha_j} p_{\alpha_j}^2} \tag{6.1.1}$$

where

L	is number of loops
j	is all possible Feynman diagrams
l_1	are the L loop-momenta
α_j	are the propagators
S_j	is the symmetry factor associated with the diagram
n_j	are polynomials of Lorentz-invariant contractions of external- and loop-momenta and polarisation vectors
c_j	are constants which depend on couplings and gauge group factors

The expression for the one-loop amplitude we wish to calculate then is

$$A_n^1 = \sum_j \int \Big(\frac{d^D l_1}{(2\pi)^D} \Big) \frac{1}{S_j} \frac{n_j c_j}{\prod_{\alpha_j} p_{\alpha_j}^2} \tag{6.1.2}$$

6.2 Colour-dressed, virtual amplitude

In QCD the gauge fields are matrices and hence do not commute: there is self-interaction in gluon fields. The underlying group is $SU(N)$, where in the QCD case $N = 3$.

The first step in the on-shell technology is to separate colour permutations from kinematics in order to simplify calculations. The Higgs boson of course does not carry colour, and has no effect on the colour ordering. When constructing the full amplitude it must simply be included in all possible positions within the colour permutation.

The tree-level n-gluon amplitude colour decomposition is as follows [64]:

$$A_n^0(1, ..., n) = 2^{(n-2)/2} g^{n-2} \sum_{\sigma \in S_n/Z_n} \mathrm{tr}(T^{a_{\sigma_1}} ... T^{a_{\sigma_n}}) A_n(\sigma_1, ..., \sigma_n) \tag{6.2.1}$$

At one-loop the n-gluon colour decomposition is [12]:

$$A_n^1(\{p_i,h_i,a_i\}) = g^n\Bigg(\sum_{\sigma\in S_n/Z_n} \mathrm{N}_c\mathrm{Tr}(T^{a_{\sigma_1}}...T^{a_{\sigma_n}})A_{n;1}(\sigma_1,...,\sigma_n) + \sum_{c=2}^{[n/2]+1}\sum_{\sigma\in S_n/Z_n} \mathrm{Tr}(T^{a_{\sigma_1}}...T^{a_{\sigma_{(c-1)}}})\mathrm{Tr}(T^{a_{\sigma_c}}...T^{a_{\sigma_n}}) \times A_{n;c}(\sigma_1,...,\sigma_n)\Bigg) \tag{6.2.2}$$

where

p_i and h_i	gluon momenta and helicities
a_i	adjoint colour index, $a = 1, 2, ..., \mathrm{N}_c^2 - 1$
g	gauge coupling $\frac{g^2}{4\pi} = \alpha_s$
S_n	set of all permutations of n objects
Z_n	subset of cyclic permutations, which preserves the trace
$A_{n;c}$	colour-ordered, primitive amplitudes calculated in our chosen six-dimensional spinor helicity formalism.

6.3 One-loop, colour-ordered amplitude in basis of master integrals

The colour-ordered amplitudes $A_{n;c}$ of equation 6.2.2 include both cut and rational parts. In the six-dimensional spinor helicity formalism we have chosen to use they are combined and calculated together [49], using:

$$A_n^{(1)} = \frac{\mu^{2\epsilon}}{(4\pi)^{2-\epsilon}}\Bigg(\sum_{K_4} C_{4;K_4}^{[0]} I_{4;K_4}^{4-2\epsilon} + \sum_{K_4} C_{4;K_4}^{[4]} I_{4;K_4}^{4-2\epsilon}[\mu^4] + \sum_{K_3} C_{3;K_3}^{[0]} I_{3;K_3}^{4-2\epsilon} + \sum_{K_3} C_{3;K_3}^{[2]} I_{3;K_3}^{4-2\epsilon}[\mu^2] + \sum_{K_2} C_{2;K_2}^{[0]} I_{2;K_2}^{4-2\epsilon} + \sum_{K_2} C_{2;K_2}^{[2]} I_{2;K_2}^{4-2\epsilon}[\mu^2]\Bigg) + O(\epsilon) \tag{6.3.1}$$

where

K_r	refers to the set of all ordered partitions of the external momenta into r distinct groups.
I	are known integrals defined below in equation 6.4.1.
$C^{[0]}$	are coefficients of cut-constructible pieces, calculated as defined in Section 6.5.
$C^{[i]}, i = 2, 4$	are coefficients of rational pieces, calculated as also defined in Section 6.5.

6.4 Master integrals

The master integrals in equation 6.3.1 are given by

$$I_n^{4-2\epsilon}[f(\mu^2)] = i(-1)^{n+1}(4\pi)^{2-\epsilon}\int \frac{d^{4-2\epsilon}l}{(2\pi)^{4-2\epsilon}} \times \frac{f(\mu^2)}{l^2(l-P_1)^2(l-P_1-P_2)^2...(l+P_n)^2} \tag{6.4.1}$$

where P_i is the momentum of the ith external leg.

All the master integrals for this calculation are already known (see, in particular, Ellis and Zanderighi [82, 83]) and can be retrieved from *qcdloop* [83] rather than be calculated from scratch. For completeness, we state them below.

6.4.1 Non-scalar

Where the renormalisation scale $f(\mu^2) \neq 1$.

Sources: [49, 58]

$$I_4^{4-2\epsilon}[\mu^4] \xrightarrow{\epsilon\to 0} -\tfrac{1}{6} \tag{6.4.2}$$

$$I_3^{4-2\epsilon}[\mu^2] \xrightarrow{\epsilon\to 0} -\tfrac{1}{2} \tag{6.4.3}$$

$$I_2^{4-2\epsilon}[\mu^2] \xrightarrow{\epsilon\to 0} -\tfrac{1}{6}(s - 3(m_1^2 + m_2^2)) \tag{6.4.4}$$

6.4.2 Scalar

$f(\mu^2) = 1$

Source: [23, 82]

Box integral

In the zero mass case:

$$I_4^{0m} = r_\tau \frac{1}{st}\Big\{\frac{2}{\epsilon^2}\big[(-s)^{-\epsilon} + (-t)^{-\epsilon} - \ln^2\Big(\frac{-s}{-t}\Big) - \pi^2\Big\} \tag{6.4.5}$$

where s and t are the Mandelstam variables

$$s = (P_1 + P_2)^2 \tag{6.4.6}$$

$$t = (P_2 + P_3)^2 \tag{6.4.7}$$

There are also given the expressions for boxes with one or more external massive legs.

Triangle integral

A single off-shell external leg depends only on the momentum invariant of the massive leg, $t_{i+2}^{[n-2]} = t_i^{[2]}$

$$I_{3;i}^{1m} = \frac{r_\tau}{\epsilon^2}(-t_i^{[2]})^{-1-\epsilon} \tag{6.4.8}$$

Bubble integral

$$\begin{aligned} I_2 : r; i \equiv I_2[1] &= \frac{r_\tau}{\epsilon(1-2\epsilon}(-t_i^{[r]})^{-\epsilon} \\ &= r_\tau\Big(\frac{1}{\epsilon} - \ln(-t_i^{[r]}) + 2\Big) + \mathcal{O}(\epsilon) \end{aligned} \tag{6.4.9}$$

,

$$I_2[p^\mu] = \frac{P^\mu}{2} I_{2:r;i} \tag{6.4.10}$$

where

$t_i^{[r]}$	$\equiv (P_i + P_{i+1} + ... + P_{i+r-1})^2$
P	$= \Sigma_{l=i}^{i+r-1} P_l$ is the total momentum flowing out of one side
r	is the number of external legs clustered on one side of the bubble starting at leg i
r_τ	is the dimensional regularization parameter, $\epsilon = (4-D)/2$
r_τ	$= \frac{\Gamma(1+\epsilon)\Gamma^2(1-\epsilon)}{\Gamma(1-2\epsilon)}$

6.5 6D Coefficients

The coefficients of the master integrals in equation 6.3.1 given here are all as stated by Davies [49].

Box

$$C_4^{[0]} = \tfrac{i}{2} \sum_\sigma A_1 A_2 A_3 A_4 (\tilde{l}_1^\sigma)|_{\mu^2 \to 0} \tag{6.5.1}$$

$$C_4^{[4]} = \tfrac{i}{2} \sum_\sigma [\mathrm{Inf}_{\mu^2} A_1 A_2 A_3 A_4](\mu^2)|_{\mu^4} \tag{6.5.2}$$

Where

- $[\mathrm{Inf}_{\mu^2} A_1 A_2 A_3 A_4](\mu^2) = \sum_{i=0}^{2} c_i \mu^{2i}$ then $C_4^{[4]}$ is restricted to be the coefficient of the μ^4 term. Note that for a numerical application, for the Inf functions we would use discrete Fourier transforms.
- A_n is a tree amplitude.

The sum is over the two solutions to the quadruple cut and the product $A_1 A_2 A_3 A_4$ must be computed for each. In this regard, Davies [49] comments that, 'The usual procedure is to sew four three-point amplitudes together, but working with three-point amplitudes in six dimensions can be complicated. It is, in fact, easier to multiply simpler four-point tree amplitudes by inverse propagators making them equivalent to the sum of products of two three-point amplitudes, given the cut conditions.' (pg. 9) See Figure 6.5.1.
For the purposes of our implementation, described in Chapter 5, this approach is inadequate because the aim is to be able to construct tree amplitudes with many legs. Hence a three-point amplitude construction for BCFW is essential.

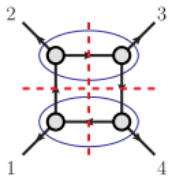


Figure 6.5.1: The four-point quadruple cut. Two pairs of three-point amplitudes are grouped together (indicated by blue ovals) to form two easier-to-use four-point tree amplitudes. The cut propagators of the four-point tree amplitudes are cancelled by multiplying by inverse propagators prior to imposing the cut conditions. (Davies fig. 5)

Triangle

$$C_3^{[0]} = -\frac{1}{2n_\gamma}\sum_\sigma [\mathrm{Inf}_t A_1 A_2 A_3(\tilde{l}_1^\sigma)](t)|_{\mu^2\to 0, t\to 0} \tag{6.5.3}$$

$$C_3^{[2]} = -\frac{1}{2n_\gamma}\sum_\sigma [\mathrm{Inf}_{\mu^2}[\mathrm{Inf}_t A_1 A_2 A_3(\tilde{l}_1^\sigma)](t)](\mu^2)|_{\mu^2, t\to 0} \tag{6.5.4}$$

where, once again, there is a polynomial expansion (in t) but only the t^0 term is retained.

The sum is over the solutions, including the conjugate-momentum solution, to the cut conditions.

Bubble

$$\begin{aligned} C_2^{[0]} = &-i[\mathrm{Inf}_t[\mathrm{Inf}_y A_1 A_2](y)](t)|_{\mu^2\to 0, t\to 0, y^m\to Y_m} \\ &-\frac{1}{2}\sum_{C_m}\sum_{\sigma_y}[\mathrm{Inf}_t A_1 A_2 A_3](t)|_{\mu^2\to 0, t^j\to T_j} \end{aligned} \tag{6.5.5}$$

$$\begin{aligned} C_2^{[2]} = &-i[\mathrm{Inf}_{\mu^2}[\mathrm{Inf}_t[\mathrm{Inf}_y A_1 A_2](y)](t)](\mu^2)|_{\mu^2, t\to 0, y^m\to Y_m} \\ &-\frac{1}{2}\sum_{C_m}\sum_{\sigma_y}[\mathrm{Inf}_{\mu^2}[\mathrm{Inf}_t A_1 A_2 A_3](t)](\mu^2)|_{\mu^2, t^j\to T_j} \end{aligned} \tag{6.5.6}$$

where

- $Y_0 = 1$
- $Y_1 = \frac{1}{2}$

- $Y_2 = \frac{1}{3}\left(1 - \frac{\mu^2}{S_1}\right)$
- $Y_3 = \frac{1}{4}\left(1 - 2\frac{\mu^2}{S_1}\right)$
- $Y_4 = \frac{1}{5}\left(1 - 3\frac{\mu^2}{S_1} + \frac{\mu^4}{S_1^2}\right)$
- $T_0 = 0$
- $T_1 = -\frac{S_1\langle\chi^-|\not{P}_3|P_1^{\flat -}\rangle}{2\tilde{\gamma}\Delta}$
- $T_2 = -\frac{3S_1\langle\chi^-|\not{P}_3|P_1^{\flat -}\rangle^2}{8\tilde{\gamma}^2\Delta^2}(S_1S_3 + P_1.P_3S_1)$
- $T_3 = \frac{-\langle\chi^-|\not{P}_3|P_1^{\flat -}\rangle^3}{48\tilde{\gamma}^3\Delta^3}(15S_1^2S_3^2 + 30P_1.P_3S_1^3S_3 + 11(P_1.P_3)^2S_1^3 + 4S_1^4S_3 + 16\mu^2S_1^2\Delta)$
- $\Delta = (P_1.P_3)^2 - S_1S_3$

State sum reduction of coefficients to regularisation scheme

In 6D, gluons have additional, non-physical polarisation states. To reduce the coefficients to those of the four-dimensional helicity (FDH) scheme, we subtract twice the contribution of cuts where all internal gluons have been replaced by scalars [49]. (See also [81] pg 16.) Expressions for amplitudes with scalars are included in Appendix A.

6.6 6D Colour-ordered tree amplitudes

Definition of the required cuts for tree amplitudes is described in Section 6.8 below.

The relevant loop momenta can be calculated using the definitions in Section 6.7 below.

Some analytic formulae for the necessary 6D tree amplitudes are already published in the 6D helicity scheme and a catalogue of these is in Appendix A. Those which are not yet available will be built using the colour-ordered Feynman rules in Appendix C where necessary and 6D BCFW relations described above. The analytic formulae that exist will be useful in testing the BCFW results.

6.7 Loop momentum solutions

To solve the cut conditions we view the massless 6D loop momentum as massive 4D momentum. To solve for the cut conditions we consider a particle with a uniform mass around the loop.

6.7.1 Box

For quadruple cut in $4-2\epsilon$ dimensions and loop momentum l, on-shell cut conditions are:

$$l_1^2 = l_2^2 = l_3^2 = l_4^2 = 0 \tag{6.7.1}$$

i.e.

$$\tilde{l}_1^2 = \tilde{l}_2^2 = \tilde{l}_3^2 = \tilde{l}_4^2 = \mu^2 \tag{6.7.2}$$

where

- $\tilde{l}_i$ represent momenta l truncated to four dimensions
- μ represents the (-2ϵ)-dimensional components, and μ^2 can be regarded as a mass term

With two external momenta P_1, P_4, both outgoing in our convention, we use Forde's formalism for parametrisation of the loop momentum [30,32] by choosing

$$\tilde{l}_1 = aP_4^\flat + bP_1^\flat + c|P_4^\flat\rangle[P_1^\flat| + d|P_1^\flat\rangle[P_4^\flat| \tag{6.7.3}$$

where

- $P_{1,4}^{b\mu}$ is the massless projection of one of the external legs in the direction of the other masslessly projected leg:
  $P_1^{b\mu} = \frac{\gamma_{14}(\gamma_{14}P_1^\mu - P_1^2 P_4^\mu)}{\gamma_{14}^2 - P_1^2 P_4^2}$
  $P_4^{b\mu} = \frac{\gamma_{14}(\gamma_{14}P_4^\mu - S_4 P_1^\mu)}{\gamma_{14}^2 - P_1^2 P_4^2}$
- $\gamma_{14} = P_1.P_4 \pm \sqrt{(P_1.P_4)^2 - P_1^2 P_4^2}$
- $S_i = P_i^2$

The coefficients are then fixed by the on-shell conditions, so:

$$\tilde{l}_1 = aP_4^\flat + bP_1^\flat + c|P_4^\flat\rangle[P_1^\flat| + \frac{\gamma_{14}ab - \mu^2}{c\gamma_{14}}|P_1^\flat\rangle[P_4^\flat| \tag{6.7.4}$$

where now

- $a = \frac{P_1^2(P_4^2+\gamma_{14})}{\gamma_{14}^2 - P_1^2P_4^2}$
- $b = -\frac{P_4^2(P_1^2+\gamma_{14})}{\gamma_{14}^2 - P_1^2P_4^2}$
- $c\pm = \frac{-c_1 \pm \sqrt{c_1^2 - 4c_0c_2}}{2c_2}$
- $c_0 = \left(ab - \frac{\mu^2}{\gamma_{14}}\right)\langle P_1^{b-}| \not{P}_2 |P_4^{b-}\rangle$
- $c_1 = a\langle P_4^{b-}| \not{P}_2 |P_4^{b-}\rangle + b\langle P_1^{b-}| \not{P}_2 |P_1^{b-}\rangle - P_2^2 - 2P_1.P_2$
- $c_2 = \langle P_4^{b-}| \not{P}_2 |P_1^{b-}\rangle$

There are four solutions but only two are independent.

6.7.2 Triangle

The triangle loop momenta are given by

$$\tilde{l}_1^\mu = aP_3^{\flat\mu} + bP_1^{\flat\mu} + \frac{t}{2}\langle P_3^{\flat-}|\gamma^\mu|P_1^{\flat-}\rangle + \frac{\gamma_{13}ab - \mu^2}{2t\gamma_{13}}\langle P_1^{\flat-}|\gamma^\mu|P_3^{\flat-}\rangle \tag{6.7.5}$$

where t is a complex, free parameter. The calculation is averaged with conjugate solutions.

6.7.3 Bubble

$$\tilde{l}_1^\mu = yP_1^{b\mu} + \frac{P_1^2(1-y)}{\tilde{\gamma}} + \frac{t}{2}\langle P_1^{\flat-}|\gamma^\mu|\chi^-\rangle + \frac{y(1-y)P_1^2 - \mu^2}{2t\tilde{\gamma}}\langle\chi^-|\gamma^\mu|P_1^{\flat-}\rangle \tag{6.7.6}$$

where

- t is a free parameter
- y is a free parameter, which for the triangle contribution to the bubble coefficient is fixed to put another propagator on-shell: $y^\pm = \frac{B_1 \pm \sqrt{B_1^2 + 4B_0B_2}}{2B_2}$ (see Davies (4.18) for B_0, B_1, B_2)
- χ is an arbitrary massless vector
- $P_1^{\flat\mu} = P_1^\mu - \frac{P_1^2}{\tilde{\gamma}}\chi^\mu$
- $\tilde{\gamma} = 2(P_1.\chi)$

6.8 Cut construction

For Higgs plus 5 gluons (i.e. a three-jet process), for example, we require up to 7-point 6D tree amplitudes in cuts, in the following categories:

- Gluon
- Gluon plus Higgs
- Both with 2 internal gluons replaced by scalars (for state sum reduction)

A sketch of the cuts for the first of the above categories, the all-gluon loop, is presented in Appendix B. Collection of the full family of cuts from existing sources is available in Armstrong [8] with code LoopCuts.

Chapter 7

Discussion and conclusion

This work has had, from the beginning, a different objective from that in the previous work on NLO $gg \to h$ with 1, 2 or 3 jets reported in Table 3.4.1. Rather than calculate a specific process, the intention has been to explore the creation of a numerical framework for the calculation of NLO $gg \to h$ with any number of jets.

Specifically, at the beginning of this work we set out to answer the question 'Can six-dimensional spinor helicity methods be used to set up a numerical framework to calculate the NLO virtual contribution to $gg \to h$ that can be scaled for an arbitrary number of additional jets?'. The stated aim was to support the development of a package to be added to the existing BlackHat software suite for NLO amplitudes, and the hope was that the construction of such a tool might ultimately facilitate a breakthrough in understanding hidden structures in the amplitudes.

The predecessor project with a similar aim [8] which, at the time, was bound by the imperative of calculation efficiency, set out to identify and exclude all vanishing amplitudes in order to simplify the calculation as much as possible. This approach was highly successful for calculating the four-dimensional cut parts of the amplitude but, in the $D > 4$ dimensions required to extract the rational parts of the loop amplitude, it remained incomplete. In part this was due to time constraints, as no six-dimensional amplitudes were calculated and spinor products were causing incorrect results. More significantly, it was concluded that the conversion of six-dimensions to four would have to be done manually as

it was impossible to declare a rule ([8] Chapter 6 pg. 101).

This previous work was highly influential in two ways: (a) it informed the choice of tools for the present thesis; and (b) it highlighted the importance of ensuring from the outset, as much as possible, that there is a theoretical route through the entire calculation. It is also very relevant that the imperative of calculation efficiency has relaxed in recent years, as new tools have become available which are expected to enable the determination, by machine methods, of analytic expressions from the numerical results. It can therefore be the analytic expressions that are implemented in BlackHat.

The current project also remains incomplete, in that there is not yet a package that produces numerical results for the NLO amplitudes we set out to calculate. In the implementation Chapter 5 the reasons for this are described. The last sticking point has been the reliable numerical calculation of six-dimensional three-point, all-gluon amplitudes in the Cheung and O'Connell formalism. Whilst this obstacle has now been overcome, by changing the entire structure of the code for six-dimensional spinors, that breakthrough has come too late for it to be carried through into the next steps in the calculation.

What, then, is the real value of the work that has been done so far? And what is the answer to the question posed at the beginning of the thesis?

This project has used the lessons learned from its predecessor. Its main contributions, then, lie in:

- Identifying a theoretical formalism that offers the potential to overcome the block encountered in the predecessor project. The Cheung and O'Connell six-dimensional spinor helicity formalism leads naturally to the calculation of cut and rational parts together for all helicity commbinations.
- Drawing together, from multiple sources of analytical work in this formalism and others, a complete 'road map' of the one-loop calculation, so that the requirements for the entire calculation are clear. Happily, the components of much of the remaining calculation are already available.
- Recognising that the key component for a scalable solution is the availability of six-dimensional three-point amplitude calculations. These are essential

in order to be able to use the BCFW approach to generating the tree amplitudes that feed the loop calculation, at increasing multiplicity.

- Building and reporting a deep understanding of the practical challenges that are faced in implementing the six-dimensional formalism in practice. I found that, whilst it is straightforward to construct four-point amplitudes directly from the analytic expression, the same is not true for three-point amplitudes. This has led to an appreciation of the need to approach the underlying spinor constructions in a way that goes beyond simply coding the analytic expressions in the theory.

- Demonstrating, with the aid of the new spinor code developed by Maitre 5.5 as a solution to the difficulties encountered in the original implementation, that the formalism can correctly calculate usable three-point amplitudes.

In conclusion, the answer to the original question is that there is every indication that it is indeed possible to use the Cheung and O'Connell six-dimensional spinor helicity formalism to build a framework for NLO $gg \to h$ that can be scaled for additional jets. It is also clear that to do so is not a trivial task, and would likely benefit from team effort.

Part II

Self-organised criticality and general relativity - a new theoretical framework for the complex universe

Chapter 8

Introduction

We believe that we are actually at the beginning of a new scientific era.
We are observing the birth of a science that is no longer limited to idealized
and simplified situations but reflects the complexity of the real world.
Ilya Prigogine (1996) [84], pg. 7

When Einstein first published the field equations of general relativity (GR) more than a century ago there was a sense that solutions to those equations needed to be found, even though Einstein himself didn't know if that would be possible. To everyone's surprise the first exact solution, by Karl Schwarzschild working whilst in the army in the awful conditions of the First World War, took only a matter of weeks. The Schwarzschild solution describes the vacuum spacetime in the region surrounding a perfectly spherical, isolated matter source. In a sense, it represents the essence of a deep problem. If the focus is on solving the equations, extreme simplification is necessary. If the focus is on understanding the universe, extreme simplification may be ineffectual. Which is it to be?

In the decades following the emergence of GR as a theory there has continued to be a focus on solution. In part this was inevitable - the only tools available in the early- and mid-20th century were those of equilibrium thermodynamics. The assumptions required to justify the use of those tools - the Friedmann-Lemaitre-Robertson-Walker (FLRW) recipe of homogeneity, isotropy, matter and radiation behaving like non-interacting, frictionless dust in the vast expanse of the universe - all seemed reasonable based on empirical observation

at the time. Solving the field equations using simplifying assumptions was an obvious thing to do. However, the FLRW framework forces some extreme conclusions, which perhaps have become accepted only because there has seemed to be no realistic alternative. Big Bang, inflation, dark energy, dark matter - these concepts have become bread and butter to cosmologists but they are far from being understood and even further from being proven.

There is the added problem that the statistical framework so effective for the analysis of perfect fluids and equilibrium conditions is actively misleading when the simplifying assumptions are unjustified. Meanwhile, new concepts and tools for understanding and working with complex systems have been developing in physics in recent decades. These neither require nor expect symmetry, regularity or equilibrium. Perhaps there is a new way of looking at GR and the evolution of the universe?

This thesis is an exposition of such an alternative approach, that is not based on solving the field equations but instead acknowledges the universe as a complex system. The new theoretical framework I suggest puts complexity centre-stage by incorporating the self-organised criticality (SOC) paradigm first proposed by Bak, Tang and Wiesenfeld [1] and releasing GR from the FLRW constraints into which it has historically been squeezed. This SOCGR framework predicts observed phenomena without the necessity of unknown matter or energy and supports further scientific investigation.

The components of this work have been drawn from a number of disciplines including complexity science, non-equilibrium statistical physics and mathematical general relativity as well as cosmology. My aim is to present sufficient information about these diverse topics, citing reputable sources, to enable a reader who is not expert in all of these disciplines nevertheless to follow the argument of the SOCGR framework. Much of the information, therefore, is not original. Even the argument that the existing approach to modelling the evolution of the cosmos is flawed is not a new one - others have also reported evidence in support of aspects of that challenge. The originality of this thesis lies in highlighting a coherent set of inadequacies in the concordance model and demonstrating that there is a viable alternative paradigm. The alternative is consistent with the theory of GR yet allows the universe to be treated as what it is - a vast web of complex interaction featuring omnipresent, multi-scale, self-organised structures.

The new framework resolves tensions in the standard cosmological model, opens doors to fresh questions and offers exciting new directions for research.

In some cases, words mean different things in different disciplines. For example, *dissipation* may be used to refer to a gradual disappearance or loss of energy. However, the technical use of *dissipative system* in the context of complex systems is defined as 'an open system that relies on external energy flows to maintain its organization and carry out self-organizing processes, dissipating energy gradients in the process'. Likewise, *scale invariance* is commonly used in cosmology in a way that is quite different from its definition in statistical physics. In cosmology the primordial spectrum is decribed as scale-invariant because the amplitude of its fluctuations does not change relative to the only scale in the expanding universe model, the horizon scale. In statistical physics, scale invariance has nothing to do with the amplitude of fluctuations but is instead associated with a particular range of power-law behaviours in the correlation function, i.e. with invariance under scale transformations (see, for example, [85] pg. 171). Throughout this work I have provided definitions where a risk of misunderstanding is evident.

This thesis is organised as follows:

- Chapter 9, the case for a new theoretical framework: key features of the standard cosmological model and challenges to it.
- Chapter 10, the nature of complexity and its impact on physical quantities and behaviour, such as mass-energy, transitions and evolution. Why it matters that the universe is a complex system, and new methods in statistical physics which are applicable to complex systems.
- Chapter 11, introducing the SOC paradigm and its theoretical implications. Can we say that the universe is - or that it isn't - a SOC system?
- Chapter 12, exploring the signficance of textbook GR, without the FLRW constraints.
- Chapter 13 presents a quantitative simulation and a qualitative discussion in the context of a five-body model galaxy. The aim is to support the argument that models which simplify by ignoring interaction or by using Newtonian gravity as a proxy for GR, or both, cannot provide a good

approximation of the behaviour or the mass-energy of a real (complex) galaxy.

- Chapter 14 applies both SOC and GR to the open questions of cosmology in the universe as a whole.
- Chapter 15 suggests directions for future work and concludes the thesis.

Chapter 9

The case for a new theoretical framework

The ΛCDM model is so widely accepted that it is essentially a concordance model. It is hugely successful, so why do we need to look for alternatives? To answer that question, in this chapter I highlight key features of the standard model and refer to some of the significant challenges to it. I begin with a brief introduction to the Friedmann-Lemaitre-Robertson-Walker (FLRW) framework, which is the foundation of ΛCDM.

9.1 The basis of FLRW cosmologies

The derivation of FLRW models of cosmology is available in many textbooks, for example [86, 87]. Since there are numerous sources, in this section I present the main features of such models largely without further citation.

Einstein's original publication of GR in 1915 presented field equations in the form[1]

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu} \tag{9.1.1}$$

[1] Various sign conventions are used in the literature and the form of the equation here is the Landau-Liftschitz Spacelike Convention (LLSC), in which the spacetime metric is $(-+++)$ and both the Riemann tensor and $\frac{8\pi G}{c^4}T_{\mu\nu}$ are positive. I use this convention throughout this thesis. A very useful summary of other sign conventions is inside the front cover of Misner, Thorne and Wheeler [86]. Einstein himself used a different convention.

where the left-hand side is spacetime geometry and the right-hand side is the source of gravity. Specifically, $R_{\mu\nu}$ is the Ricci tensor, R the Ricci scalar, $T_{\mu\nu}$ a stress energy tensor, all of which are described more fully in Chapter 12, G is the Einstein constant, and $g_{\mu\nu}$ is a generic spacetime metric. In the early 20th century, Einstein would have believed that the entire universe had a density similar to that of the Milky Way: the now-familiar hierarchy of structures including galaxy clusters, walls, filaments and voids was then unknown. The spacetime metric, therefore, would have only to apply to the relativly smooth density function of a galaxy to describe, approximately, the whole universe [88].

By 1917 Einstein had added a cosmological constant term Λ to the left hand side of the field equations:

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu} \tag{9.1.2}$$

Einstein's motivation for the new $\Lambda g_{\mu\nu}$ term was to include a mathematically viable additional component that could potentially allow the universe to exist in a steady state, a feature he thought essential.

The usual approach is to assume that the density field in the early universe can be characterised by a homogeneous and isotropic matter distribution on which tiny, Gaussian perturbations are superimposed. These perturbations are considered to be the seeds of structure formation. It is argued that at the present day there is still a 'sufficiently large' scale at which the universe is both homogeneous and isotropic. It is thus 'natural' to describe spacetime by the spatially symmetric Robertson-Walker metric tensor

$$g_{\mu\nu} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & R(t)^2 & 0 & 0 \\ 0 & 0 & R(t)^2 r^2 & 0 \\ 0 & 0 & 0 & R(t)^2 r^2 \sin^2\theta \end{bmatrix} \tag{9.1.3}$$

where $R(t)$ is the scale factor describing the expansion of the universe. From the relationship $ds^2 = g_{\mu\nu}dx^\mu dx^\nu$ the squared line element

$$ds^2 = c^2dt^2 - R^2(t)\Big[\frac{dr^2}{1-k^2r^2} + r^2(d\theta^2 + \sin^2\theta\ d\phi^2)\Big] \tag{9.1.4}$$

is derived, stated here in co-moving polar coordinates $\bar{r}, \theta, \phi$, with k being a curvature parameter which can have values ± 1, or 0.

Given a sufficiently simple stress-energy tensor, this metric allows an exact solution of the GR field equations. Any model based on such a solution assumes that the field equations can indeed be applied to the universe as a whole. In effect, that all matter and radiation is within a four-dimensional spacetime that can be described by the metric $g_{\mu\nu}$. We should note that, whilst it is straightforward to apply this argument to an early-time fluid of particles, it is less clear that it is applicable once structures have formed [88]. The fall-back argument that the universe continues to be homogeneous and isotropic at sufficiently large scales is not a comfortable one: not only is the physical reality of this assertion still unproven, there is a substantive concern arising from the question of whether *statistical* homogeneity is a sufficient condition in relation to the nonlinear field equations. Labini and others [85] have coined the term *superhomogeneity* to describe the required state, commenting that the inferred real-space correlation properties of such a universe are equivalent to those of glass [89].

The cosmological constant term Λ in equation 9.1.2 is now used to model a universe with accelerated expansion, which is somewhat ironic given that it was originally intended to ensure a steady state solution was possible. To accommodate the additional cosmological constant term it is conventional to assume that the stress-energy tensor $T_{\mu\nu}$ of the universe is adjusted by a contribution from dark energy. The stress-energy tensor is assumed to be that of frictionless dust, i.e. an ideal fluid, locally at rest with respect to a comoving observer. Of course, the real universe does not manifest a dust-like density function everywhere and at all times. The assertion that, nevertheless, on average the universe can be modelled in this way gives rise to an issue referred to as the *averaging problem*. For a clear description of the impact of scale on the averaging problem see Wiltshire [88]).

With density ρ and pressure p the conventional, frictionless dust stress-energy tensor takes the form

$$T^{\mu\nu} = \begin{bmatrix} \rho c^2 & 0 & 0 & 0 \\ 0 & p & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & p \end{bmatrix} \tag{9.1.5}$$

More precisely, there are considered to be three components to the ideal fluid:

radiation, matter and dark energy. The homogeneous cosmic density at time t is therefore

$$\rho(t) = \rho_m(t) + \rho_r(t) + \rho_\Lambda \tag{9.1.6}$$

The density of dark energy in the standard model is constant but, as the universe expands (or contracts), the densities of matter and radiation change, with the general relationships

$$\rho_m \propto \frac{1}{R^3}; \quad \rho_r \propto \frac{1}{R^4} \tag{9.1.7}$$

In an expanding universe this leads to a conclusion that there are three epochs in cosmic time: a period of radiation dominance, followed by matter dominance then, at the present time, dark energy dominance.

In practice the pressure of matter is usually ignored, in which case it is described as dust. If it is included, it is described by an equation of state

$$p_m = \omega\rho c^2 \tag{9.1.8}$$

where ω is a constant that is 0 in the case of dust.

As a result of the high degree of symmetry in the Roberston-Walker metric tensor 9.1.3 and of the many zeros in that metric tensor and in the FLRW stress-energy tensor 9.1.5, this all leads to a very simple solution to the field equations. The solution is just two independent equations, known as the Friedmann equations:

$$\begin{aligned} \Big(H^2(t) \equiv \Big) \left[\frac{1}{R}\frac{dR}{dt}\right]^2 &= \frac{8\pi G}{3}\rho - \frac{kc^2}{R^2} \\ &\text{and} \\ \frac{1}{R}\frac{d^2R}{dt^2} &= -\frac{4\pi G}{3}\Big(\rho + \frac{3p}{c^2}\Big) \end{aligned} \tag{9.1.9}$$

H is the Hubble factor, defined as shown. The calculation requires an assumed composition of the universe. The chosen densities of matter, radiation and dark energy are usually expressed as proportions of the critical density[2] ρ_c, where

[2]This use of the term *critical* in 'critical density' is quite different from that which we will encounter in the context of self-organised criticality in Chapter 11. In statistical physics, critical has a technical meaning related to states associated with phase transitions. A definition is given in Section 11.1.1.

$$\rho_c(t) = \frac{3H^2(t)}{8\pi G} \tag{9.1.10}$$

The fractional density parameters are then

$$\Omega_m(t) = \frac{\rho_m(t)}{\rho_c(t)}, \quad \Omega_r(t) = \frac{\rho_r(t)}{\rho_c(t)}, \quad \Omega_\Lambda(t) = \frac{\rho_\Lambda(t)}{\rho_c(t)} \tag{9.1.11}$$

A cosmological model based on the Robertson-Walker metric and Friedmann equation scale factor is commonly described as a FLRW model. The standard ΛCDM model is a specific example of a FLRW model.

9.2 Implications of FLRW models

The assumptions of the FLRW framework have profound consequences for the intellectual and academic framework of standard cosmology:

- In a homogenous and isotropic, spatially expanding universe one is led inexorably to the conclusion that there was an infintely dense beginning. Time evolves linearly, and the universe has an age.
- The beginning was a mathematical singularity, since the field equations break down in conditions of infinite density. In practice, quantum effects are expected to be dominant at this scale but, since FLRW can shed no light on the nature of the beginning that it advocates, within that framework it remains a singularity.
- It is valid to analyse cosmological data using statistical methods which are predicated on the existence of a well-defined mean density.
- Further, since all matter, radiation and dark energy are ideal fluids, the statistical tools of equilibrium thermodynamics are applicable.
- The field equations of GR are largely irrelevant, except for the calculation of the scale factor via Friedmann's equations 9.1.9.

.

I have already pointed out that the model requires superhomogeneity, which is not the physical reality. We will see in Section 9.3 that there are signs that even the weaker conditions of statistical homogeneity and isotropy may not apply in

the real universe.

It is also a problem that the FLRW framework has become so familiar to generations of cosmologists that it is now widely described as *the* rather than *a* GR solution. Reputable authors say, for example, 'The necessity for [dark energy] arises from using the Friedmann equation to describe the evolution of the cosmic expansion; if this equation is incorrect, it would require the replacement of Einstein's relativistic theory of gravity with some alternative.' (in Section 22.4.7 of [90]). In fact, the Friedmann equations are the solution of an extraordinarily confined and simplified statement of Einstein's relativistic theory of gravity. Before we throw out that theory we should certainly reconsider the assumptions and approximations that have been used to obtain the particular FLRW family of solutions. In fact, we see in Chapter 12 that FLRW models diminish GR to such an extent that its fundamental features are effectively erased.

Finally, the FLRW assumptions support an expectation that, in a weak-field, sub-relativistic system, gravity described by Newtonian theory serves well as a proxy for GR. This assumption drives almost all conventional cosmological analysis and modelling[3]. However, whilst it may be substantially valid for a solar system, at least over timescales that are short relative to cosmic time, without *a priori* acceptance of the FLRW assumptions there is no evidence that it is so for larger scale structures or over cosmic timescales.

9.3 ΛCDM: evidence and challenges

Lambda-Cold-Dark-Matter, ΛCDM, is a standard abbreviation for a FLRW model with cold[4] dark matter, a cosmological constant, inflationary initial conditions, standard radiation and neutrino content, and a flat universe with $\Omega_{\rm tot} = 1$. This concordance model enbodies a parameter fit that the community generally agrees provides 'a good description of a wide range of astrophysical and cosmological data' [92].

[3] For the final infall phase of inspiralling binary black holes or neutron stars it is acknowledged that Newtonian theory is inadequate. Template banks of gravitational waveforms are generated for use in experiments to detect gravitational waves, using calculations that have evolved over the last two decades from purely the Post-Newtonian approximation (see Appendix H) to numerical relativity [91]. These calculations, limited to two-bodies and constructed on a case-by-case basis, represent the state of the art for GR modelling.

[4] 'Cold' as in very sub-relativistic.

In this section I briefly review and comment on the evidence for the model and some of the challenges to it.

9.3.1 Big Bang nucleosynthesis (BBN)

Big Bang nucleosynthesis (BBN) is the theory predicting the abundances of the light element isotopes D, ^{3}He, ^{4}He and ^{7}Li and is a fundamental component of the standard cosmological model. In the 1960s it was hugely influential in causing the growing emphasis on the family of theories that would come to include ΛCDM. A Big Bang origin of the universe became widely accepted largely as a result of the observation in 1965 of the cosmic microwave background radiation (CMB) [93, 94], predicted in the late 1940s by those theories of primordial nucleosynthesis which assumed an evolving rather than a steady state universe [95–98]. I discuss the CMB in Section 9.3.3, but first I summarise where its prediction came from, the influence of those theories on the standard model of cosmology, and the extent to which current research continues to support the BBN model.

We have seen that the FLRW framework leads to a mathematical conclusion that the universe originated as an infinitely dense singularity from which all that exists burst and continues to expand. Nevertheless, there was initially a strong resistance to this view, in favour of an alternative 'steady state' universe scenario.

In 1946 Gamow [95] pointed out that the existing attempts to explain the abundance curve of elements, which were based on an equilibrium state determined by nuclear binding energies at some high temperature and density, could not hold. He proposed instead that the decrease of relative abundance along the natural sequence of elements must be understood as being caused by processes of radiative capture over a longer time, leaving the present high abundance of hydrogen as a result of competition between the β-decay of original neutrons and the processes by which the neutrons were incorporated into heavier nuclei. His student Alpher and their collaborator Herman [95, 96] developed a model which included the prediction of the existence of a relic background radiation of the order of a few kelvin [97]. Whilst this work was unaccountably omitted [99] from citation in the 1965 discovery papers by Penzias and Wilson [93] and by Dicke, Peebles, Roll and Wilkinson [94] it is now recognised to

have laid the foundation for Fred Hoyle's term 'Big Bang', originally intended to be rather derogatory, to become the accepted view of the origin of the universe.

For the current status of this theory, I summarise the review article [90] and refer the reader to source citations there. Briefly, nucleosynthesis occurs at a temperature scale of order 1 MeV, corresponding to an age $t \sim 1$ s in the standard cosmological model. In standard BBN theory, i.e. that based on the Standard Model of particle physics, the abundances of the light elements are predicted based on only one key parameter, the baryon-to-photon ratio, $\eta \equiv n_b/n_\gamma$. The value of n_γ is usually fixed by the present CMB temperature. When η is calculated using the *Planck* [100] result for baryon density, predictions for light element abundances are:

- for D/H: 'in excellent agreement' with the deuterium abundance observed in quasar aborption systems and 'in reasonable agreement' with the helium abundance observed in extragalactic HII regions, once systematic uncertainties are accounted for; and
- for Li: systematically higher than the Li abundance observed in the atmospheres of halo dwarf stars.

The predictions of the standard cosmological model BBN are thus reasonably consistent with observations of light element abundances, apart from the observed Li deficit. The latter is commonly known as *the lithium problem*.

It is important to note that the model is highly sensitive to physical conditions in the early universe and requires rapid inflation and cooling in the first second after the Big Bang for the necessary temperature scale to be reached. This early inflation is not yet understood, as discussed in Section 9.3.2.

As further comment, the origins of BBN theory lie in the assumption of a Big Bang beginning and, in the mid 20th century, this was one of only two options that were perceived: a steady state universe, or a spatially symmetrical evolving universe which began with a Big Bang[5]. If the FLRW framework is put to one side, theories of complexity make clear that there are other options. The

[5] An interesting modern variation on this is Penrose's idea of a cyclical universe based on mathematical extension beyond the intial singularity. He proposed his somewhat controversial theory of conformal cyclic cosmology (CCC) in [101].

universe can be perpetual without being steady state. In this case, the nature of the questions we ask about the abundances of elements have scope to change.

9.3.2 Inflation

The Big Bang origin of the universe results in a model that has problems of fine-tuning: it is extremely sensitive to initial conditions. In this sense the model may be considered incomplete [90], requiring some additional component(s) in order to resolve:

- The *horizon problem* and the *isotropy problem*. Since light signals can only propagate a finite distance in the time since the Big Bang, and the standard model has a period of radiation dominance in the early universe, there is a particle horizon. At the time of the formation of the CMB at $z \cong 1100$, the horizon is calculated to be of order $100Mpc$ in size subtending an angle of 1^o [90]. Thus, assuming one can integrate along the light cone back to $t = 0$, the sky is filled with regions that should be causally disconnected yet apparently manifest a nearly isotropic and homogeneous state.
- The *flatness problem*. A universe of non-zero curvature at the present time requires that the total density parameter $\Omega(t)$ must be very precisely equal to unity as the initial time tends to zero. In addition to this being a problem of fine-tuning, an $\Omega = 1$ universe is unstable [90]

To address these problems, the inflation hypothesis [102] has been proposed. In an adiabatically expanding universe the relation between the scale factor R and the temperature T is $RT \sim$ constant. The concept of inflation is based on the premise that one or more phase transitions could have occured in the early universe, during which that relation need not hold. If the value of RT changed by a factor of $\mathcal{O}(10^{29})$ in the very early universe, causing the universe to increase in spatial extent by a factor of 10^{23} during 10^{-34} seconds, the problems outlined above could be resolved. The accelerated expansion would (a) expand the causal horizon beyond the present Hubble length; and (b) drive any curvature in a Robertson-Walker spacetime towards spatial flatness [90].

Inflation is a seemingly elegant solution to the gaps in the cosmological model and many theoretical explanations for it have been proposed. A useful modern review is the 2021 Snowmass paper [103]. There are three main classes of observational signatures of inflation: primordial gravitational waves; primordial

non-Gaussianity; and deviations from the minimal power-law power spectrum of primordial fluctuations which, it is suggested, would indicate that new energy scales play an important role in inflationary dynamics. Inflation theories are therefore a very active field of research.

All of these signatures are currently explored in the statistical environment of the standard cosmologiccal model, i.e. assuming that the tools of equilibrium statistical physics are appropriate. Therefore they do not offer a test of that model. No definitive theory has yet been identified.

9.3.3 Cosmic microwave background (CMB)

In the ΛCDM framework the CMB radiation, first observed by Penzias and Wilson in 1965 [93], originated at the era of last scattering[6] ($z \approxeq 1100$). The spectrum of the CMB is described as a blackbody function with temperature estimated by WMAP together with the FIRAS[7] experiment to be $T = (2.72548 \pm 0.00057)$K [104]. The spectral form of the CMB is generally considered to be compelling, if indirect, evidence of the validity of ΛCDM. The CMB is usually analysed, however, based on the ΛCDM framework. For example, *Planck* 2018 [105] uses ΛCDM parameters and the statistical methods employed are valid only in a homogeneous population. These analyses therefore cannot be said to offer a *test* of the model's assumptions. I expand on this point in the next chapter, Section 10.4.

The main CMB observables are the angular variation in temperature and polarisation correlations. Studies of the latter are in relative infancy, so I focus here on the temperature mapping of the sky. With the final *Planck* data release in 2018 [100, 106] mapping is considered to have reached high precision.

In conventional analysis, the majority of the cosmological information is considered to be contained in the temperature variance as a function only of angular separation, i.e. the 2-point function. The argument given for this is that only weak phase correlations are seen and no preferred direction is inferred (see review paper [107]). The anisotropies are usually expressed and studied using a spherical harmonic expansion of the sky[8]

[6] Often described as the time of 'recombination'.
[7] FIRAS: Far-InfraRed Absolute Spectrophotometer
[8] For derivation see Appendix G

$$T(\theta, \phi) = \sum_{lm} a_{lm} Y_{lm}(\theta\phi) \tag{9.3.1}$$

Higher-order multipoles - variations at $l \geqslant 2$ - are interpreted as the result of perturbations in the density of the early universe. There are proposed experiments which will make it feasible to probe spectral distortion mechanisms arising from expected damping and dissipation of relatively small primordial perturbations [107]. This new data could provide valuable additional input in time.

Gaussian uniformity is not immediately obvious in the raw data used in CMB mapping and arises only when the data is both cleaned to remove foreground components and adjusted for, in particular, the dipole anisotropy in the $l = 1$ first spherical harmonic. Usually interpreted as being caused by the motion of the Solar System relative to the blackbody field, it is sometimes called the kinematic dipole isotropy. It has amplitude 3.3621 ± 0.0010mK [100] and leads to an implied velocity for the Galaxy and Local Group of galaxies relative to the CMB.

It is thus important to be aware that the familiar map of the CMB temperature, shown in Figure 9.3.1, does not represent raw data. Planck measures data across 9 fequencies and the part of the CMB that is observable, i.e. which is not obscured by the foreground of the solar system, the Milky Way and other known factors, varies depending on the frequency: usually in the region of 70% of the celestial sphere [108]. For the properties of the CMB behind the galaxy mask and other foreground features a model must be used. Usually a Gaussian distribution is assumed. The mean temperature monopole is, of course, removed and the data is also adjusted for the kinematic dipole.

The association of the dipole solely with local kinematics is subject to growing challenge. Using data for 1.36 million quasars observed by the Wide-field Infrared Survey Explorer (WISE), Secrest et al [109] reported that the amplitude of the dipole in the quasar sky is over twice as large as expected in an exclusively kinematic interpretation, with 4.9σ significance. Performing a bayesian analysis of the same sample, Dam et al [110] found that the conventional kinematic explanation of the CMB dipole is rejected with 5.7σ significance.

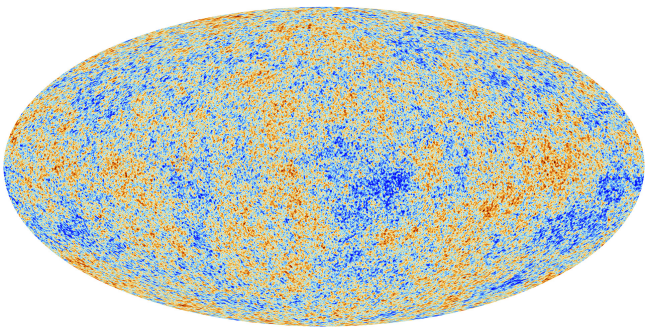


Figure 9.3.1: Planck: CMB temperature map after adjustment for foreground noise, monopole mean temperature and dipole
Image credit ESA and the Planck Collabroation

It is pertinent to ask, then, is the CMB indeed isotropic? There are features of the CMB that are regarded as unexpected, including a hemispherical power asymmetry and a large cold spot in the Southern hemisphere, which seem to violate statistical isotropy and scale invariance of inflationary perturbations. Schwarz et al [111] have carried out a critical analysis of the current understanding of these and other features, concluding that the physics of the CMB anomalies remains a puzzle which may be resolved by further observations.

More recently, Pranav and Buchert [112] used homology analysis of the CMB temperature maps to test for statistical isotropy. They found that the results of their analysis differed depending on whether the mean and variance were calculated locally from the non-masked patch or from the full masked sky, prima facie indicating a breakdown of statistical isotropy in the CMB maps. They concluded that more work is required to determine the source of the anomaly but pointed out that there may be significant consequences for parameter estimation.

If statistical isotropy of the CMB temperature maps remains unproven, what about Gaussianity? In most analyses, Gaussianity is simply assumed. In a rare attempt to carry out model-independent analysis, Buchert et al 2017 [108] used Minkowski functionals to carry out analyses on non-Gaussianity in Planck CMB maps. They reported a weak level of non-Gaussianity at $1-2\sigma$ of the foreground corrected masked *Planck 2015* maps. More work on this topic would be valuable.

Study of the CMB is a vast and varied field. I note particularly that (a) theories predict the expectation value of the CMB power spectrum whereas the data represents just one realisation, so there is an inherent source of uncertainty; (b) discarding large anisotropies from analysis can result in the truncation of what would be the tails of a power law Pareto-Levy function, leaving a distribution that is trivially Gaussian [113]; (c) the analyses is founded in an assumption that a spherical expansion is appropriate; and (d) ΛCDM parameters are routinely used in analyses. Whilst the results can then be said to be consistent with ΛCDM, we must beware of circular reasoning. It is not valid to conclude that the ΛCDM model is tested or validated by comparison with the outcome of the analyses.

9.3.4 Dark energy

In 1929 Edwin Hubble's observations caused him to conclude that the Universe is expanding [114]. Seventy years later, data from supernova surveys was interpreted as evidence that the expansion is accelerating [115, 116]. The term *dark energy* is commonly used to describe any source of this cosmic acceleration.

In the concordance model dark energy is simply the cosmological constant, Λ, in the GR field equations 9.1.2. The energy density of dark energy is approximately $(2.2 \text{ meV})^4$ when parameters $\omega = -1$, $\Omega_v = 0.68$, $h = 0.67$ are used [90], i.e. roughly $10^{-123} M_P^4$, where M_P is the Planck mass. On the one hand, this number is so small it is considered 'unnatural', since there should be contributions to dark energy from quantum effects (a classic review including a number of suggested responses to the problem is presented in Weinberg [117]). On the other hand, within the context of ΛCDM the existence of dark energy is said to be 'well established by multiple lines of independent evidence from a tight web of precise cosmological measurements' [90]. As with CMB measurements, however, the precision is an outcome of analysis which has been carried out based on the accepted model and its parameters.

Within the context of the conventional analysis of observations, which conclude that there is accelerated expansion of the universe, the cosmological constant is not the only possible explanation. Other theories fall broadly into one of two categories: either modified gravity, or some new form of energy, such as a rolling

scalar field with a potential $V(\phi)$ (sometimes called *quintessence*). A variety of observational probes are underway to test or to constrain current theories. For a thorough review see [90] Chapter28.

There is a further school of thought that dark energy may be a relic of over-simplification in cosmological modelling, see for example [109, 118, 119], or bulk flow [120, 121]. These suggestions are presented as a substantial challenge to the standard cosmological model, but do not go so far as to propose an alternative to the FLRW approach.

9.3.5 Dark matter and structure formation

In ΛCDM the eponymous cold dark matter is non-relativistic and does not interact either with itself or with any particle in the Standard Model of particle physics[9]. It accounts for 84.4% of the matter density (26.4% of the critical density) of the universe [107] and, in the ΛCDM model, is essential for structure formation. Evidence for the existence of dark matter phenomena is compelling: galaxy rotation curves, gravitational lensing studies and 'the matter of the bullet cluster'. Yet its nature remains unknown. All the sources of evidence for dark matter are probes of gravitational potential. In other words, *dark matter* is the name given to gravitational potential that differs in magnitude or distribution from that estimated to arise from normal matter. (With hindsight, a less prescriptive name might have been more appropriate. Perhaps 'dark mass'.) The excess potential is in relation to that expected within the context of equilibrium statistical thermodynamics and quasi-Newtonian gravity in a FLRW framework with flat spacetime.

In dark matter research the conventional choice is between the possibilities that (a) the only interactions beetween dark matter and normal matter are gravitational; or (b) there is some interaction, however weak, between dark matter and the particles of the Standard Model of particle physics. Dark matter particle research commonly pursues the second of these possibilities. Much progress has been made in exploring the parameter space of a variety of models but no

[9] Some theories posit contributions from dark baryonic sources such as primeval black holes. In the cosmological context *baryonic* usually refers to matter made up of Standard Model particles, ignoring electrons (which have, relatively, negligible mass) and excluding neutrinos. There is also, of course, a contribution to dark matter from the masses of neutrinos, although in the Standard Model of particle physics neutrinos do not have mass.

definitive theory has emerged and there has not yet been any direct detection of dark matter particles. Section 27 of reference [107] gives a clear and concise summary of the current status of related research and open problems. (See also Snowmass [122]). To date, there is no specific evidence that dark matter has a particle solution.

Here I summarise the evidence for dark matter and consider how that evidence supports the case for the ΛCDM standard model - and how it can also be interpreted as a challenge to it.

Galaxy rotation curves

Luminous matter in galaxies, i.e. stars and ionised hot gas, can be directly detected. Observed galaxy rotation curves do not agree with expectations calculated based on the detectable matter and Newtonian gravity. This was first clearly shown in 1970 [123] and more extensively in work by Rubin et al [124]. Dark matter haloes must be assumed to make the data fit.

As early as 1933 Fritz Zwicky had already noticed an anomaly [125]. He reported that the average density of matter in the Coma cluster 'would have to be at least 400 times larger than that derived on the grounds of observations of luminous matter' in order to explain galaxy kinematics. I mention this early work because, despite numerical inaccuracies in the data Zwicky used, his paper sets out clearly the basis of the calculation, which is still commonly considered to be valid [126].

In his calculation Zwicky assumed a uniform, spherical density distribution of baryonic matter and used the virial theorem[10] with a Newtonian gravitational force law. He then calculated that the total potential energy in the system U should be

$$U = -\frac{3}{5}G\frac{M^2}{R} \tag{9.3.2}$$

where G is the gravitational constant, M the mass of luminous matter and R the radius of the cluster. Using the data at the time, the calculated value of

[10]The virial theorem, first devised by Rudolf Clausius (1822–1888) to describe motions of particles in thermodynamics, is predicated on energy equilibrium. To illustrate, for a static system in the absence of a surface pressure term its scalar form is simply $\langle E_{\text{tot}} \rangle = K + \frac{W}{2}$ where K is the kinetic energy of the system and W is the potential energy [127].

U would result in an average Doppler effect of 80 km/s, whereas the observed value was $1,000$ km/s or more.

At the time, Zwicky considered three alternatives to his assumptions: that virial theorem did not apply and all the available energy was kinetic rather than potential; that the Coma cluster was unstable and would in future disperse because of its high radial velocities; and that the apparent velocities were the result of distortion by gravitational redshift. He found none of these alternatives to be satisfactory. Zwicky did not consider alternatives to the Newtonian gravitational force law or to uniform density distribution.

Galaxy kinematics, i.e. using galaxy orbits as tracers of the underlying potential as did Zwicky, continue to be used. Lensing studies use observations of distortions of background galaxies to assess the intervening mass distribution. One of the most effective modern methods for measuring the masses of clusters employs X-ray and Sunrayev-Zeldovich (SZ) observations, which use measurements of the distribution of the intra-cluster medium to probe the potential as a function of radius. Boundary and other corrections are made to accommodate the fact that clusters are not in perfect equilibrium, but hydrostatic equilibrium is assumed. A useful review article is [128], where further references are given.

Each of the methods has underlying assumptions and possible biases which are increasingly well understood but are still the subject of active research, often using numerical simulations. These methods are all probes of gravitational potential: as would be expected, therefore, they give broadly similar results. Taken together with the estimated baryonic matter content of clusters, the proportions are considered to be $\sim$3 % stars, $\sim$12% ionised hot gas in the intracluster medium and $\sim$85% dark matter [128]. The fundamental basis for the analysis of these methods and for the estimation of baryonic matter content continues to be Newtonian gravity, and local evolution to approximate equilibrium is assumed.

Equation 9.3.2 bears no relation whatsover to the ten, interconnected, second-order differential GR field equations 9.1.2. So why is it expected that the Newtonian and virial theorem approach can produce reliable results for galaxy kinematics?

With this question in mind, an alternative approach to galaxy rotation curve

analysis has been reported, particularly by Cooperstock and Tieu [129], Balasin and Grumiller [130], and Ludwig [131]. Using a stationary, axially symmetric spacetime metric and the approximation of a co-rotating, pressureless, perfect fluid source for a rotating disk galaxy, it has been suggested that the GR gravitoelectromagnetism (GEM) mass-energy contribution[11] is sufficient to obviate the need for dark matter (for an infinitely thin disk [129], and for any rotating galaxy [130]). Ludwig [131] concluded that the GEM field produced by mass currents modifies galactic rotation curves notably at large distances, commenting, '... at large distances the Lorentz force due to the gravitomagnetic field effectively controls the mass equilibrium balance in view of the decaying centrifugal force. The field produced by the large disk of mass currents basically acts as a gravitomagnetic brake against the gravitational attraction.'

This work is not universally accepted and I do not cite it as definitive. However, at least one critic [132] mistakenly states that, 'it is widely accepted that the modelling of galaxy rotation curves in general relativity requires the inclusion of a dark matter halo in order to reproduce observations', when in fact, as we have seen, mdelling of galaxy rotation curves uses Newtonian gravity and virial theorem not GR. That paper also states, 'An immediate question regarding such claims is how such significant behaviours can have been consistently missed in the long history of numerical relativity.' The authors fail to recognise that state-of-the-art calculations in numerical relativity are confined to two-body scenarios and cannot be applied to galaxies [91].

The matter of the bullet cluster

'The matter of the bullet cluster' is a different class of evidence for dark matter. In 2006 Clowe et al presented weak lensing observations of the 1E0657-558 cluster merger [133]. At publication the paper caused immediate interest because it demonstrates spatial segregation of normal matter and inferred dark matter. Gravitational lensing maps produced by the team showed that gravitational potential in this merger does not trace the dominant normal matter mass component but rather has a centre of mass offset from the centre of the normal matter mass peaks at 8σ significance. The optical and lensing map produced in the study is reproduced in Figure 9.3.2.

[11] Gravitoelectromagnetism (GEM) is a general relativistic field effect associated with frame-dragging. I discuss it further in Section 12.4.4.

Figure 9.3.2: The Bullet Cluster
Image credit X-ray: NASA/CXC/CfA/M.Markevitch, Optical and lensing map: NASA/STScI, Magellan/U.Arizona/D.Clowe, Lensing map: ESO WFI

Clowe et al concluded that the observed displacement proves the presence of dark matter 'for the most general assumptions regarding the behaviour of gravity', and say that, 'Any non-standard gravitational force that scales with baryonic mass will fail to reproduce these observations.' However, they do not acknowledge that in GR, which is certainly a standard theory of gravity, gravity does not scale linearly with mass. In fact, the field equations 9.1.2 are nonlinear in both the spacetime metric and its first derivative[12]. Therefore, it is more correct to say that their findings do not prove the presence of dark matter, they demonstrate only that the distribution of gravitational potential differs from that calculated in Newtonian (or any other linear) theory. As far as I am aware, the relationship between the observed phenomena and any configuration of potential that might arise in nonlinear GR applied to a many-body, dynamical system has not yet been considered.

Specifically, in the bullet cluster the distribution of mass energy is seen to follow the component galaxy clusters as they draw apart. This is contrary to

[12]For a full explanation of this see Section 12.4.5

expectation in conventional analyses, which calculate that the majority of mass will remain with the retarded gas clouds that previously formed the intra-cluster media. It is interesting to note that the work on GEM-influenced galaxy kinematics [129–131] could point towards an explanation for this: the mass-energy in the interacting structures exceeds that of the non-interacting dust. I will revisit this argument in Chapter14.

Structure formation

One argument for the existence of dark matter is based on the success of sophisticated N-body simuations of cosmic evolution, for example the Virgo Consortium's EAGLE project [134] and the IllustrisTNG TNG50 project [135]. Such simulations, which use Newtonian theory for the accumulation of normal, visible matter, can only generate realistic structure if a framework of cold dark matter is built first, with a dominating mass energy, onto which the normal matter accretes [136]. A different way of looking at this is to notice that the modern N-body simulations require a framework of cold dark matter because without it there is insufficient time for the (linear) attractive force of Newtonian gravity to result in the structures we observe [136].

The argument that these simulations are evidence for the validity of ΛCDM model is, in a sense, reversible: they may demonstrate challenge to it. It could instead be argued that extant N-body simulations, which are implemented using the standard FLRW cosmology, fail to produce realistic structure unless gravitational potential from an unknown source, on a scale which far exceeds the mass-energy of known matter, is added in by hand.

9.3.6 Parameter fit, and tensions

ΛCDM embodies a fit of parameters carefully calibrated to establish consistency with observational data. This is, of course, the purpose of a model. However, if those parameters are then used to carry out analysis on the data, as is often the case, then there is a real risk of circular argument. Those analyses cannot in any way *test* the model, because they are already shaped by it. Then 'high-precision' really means that a precise fit of data and model is possible because the data is presented in a way that is dependent on the model.

Despite the high precision claimed for the model, tensions in fundamental ΛCDM

parameters do exist and show no signs of resolution [92, 137]. In particular:

- **Hubble tension**

  The disagreement between predictions of the Hubble constant $H_0 = H(z = 0)$ made by early-time probes in conjunction with the ΛCDM model on the one hand and, on the other hand, late-time, model-independent determinations made from local measurements of distances and redshifts. The inferred values H_0 are in tension at $4\sigma - 6\sigma$.

- **The S_8 tension**

  The S_8 parameter is a measure of the homogeneity of the Universe, defined as $S_8 \equiv (\Omega_{matter}/0.3)^{0.5}\sigma_8$ where σ_8 is the standard deviation of the density fluctuation in an $8h^{-1}$ Mpc radius sphere. There is disagreementat at 3σ significance between what is predicted by extrapolating the fluctuations in the CMB forward to the present day, and what is measured by multiple probes of the inhomogeneity in the nearby Universe.

9.4 Averaging, scale, and backreaction

The FLRW framework describes universes that are locally homogeneous and isotropic on all scales, not universes that are only statistically homogeneous and isotropic. Yet no-one claims that the real universe is locally homogeneous and isotropic: large variations in density clearly exist.

This *averaging problem* has been known for some time. A very clear explanation of the impact of scale and the physical foundations of the averaging problem is given by Wiltshire in [88]. Ellis [138] introduced the term *fitting problem* to describe the issue of finding a model that best descibes the average of the real universe with its complex structures.

It is suggested that the large local deviations of density mean that the average evolution may be far from the FLRW behaviour even above the homogeneity scale, if such a scale exists. The possibility that the observed change in average quantities from those of the ΛCDM model at later times is due to the formation of structures, an effect first studied by Shirokov and Fisher [139], is termed 'the backreaction conjecture' [140].

The backreaction conjecture and Wiltshire's related timescape cosmology [141] offer very interesting approaches. Both are nevertheless based firmly within the FLRW premise that the universe is expanding from a primordial and small Gaussian beginning, and that there is a well-defined mean density. The latter assumption is not necessarily valid in a complex system.

9.5 A new paradigm: complexity matters

In the preceding sections I have drawn attention to some of the challenges to the concordance model that are already being raised within the physics community. Such questioning is becoming more widespread, and was the subject of a 2024 meeting at the Royal Society [142]). To date, alternative suggestions either retain key assumptions (for example backreaction in the late universe [143] and timescape cosmology [88]) or, as in theories of modified gravity, do not attempt to replace the whole model. Yet each new generation of observational data, particularly recent observations by JWST [144], add to the need to review previously accepted views about the development of the early universe. The difficulty is, any new approach must be capable of fitting all the data at least as well as ΛCDM. This is far from trivial.

In the next chapter I begin to lay the foundations for an alternative approach to study of the universe by treating it as a complex system. Complex systems are physically different from those which are simple: they exhibit irregularity and do not reach equilibrium. However they can and often do evolve quasi-stable structures. The rigorous study of complexity is a growing interest in many scientific disciplines.

Chapter 10

Complexity

The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe.
P. W. Anderson (1972) [145]

It is more natural, or at least less ambiguous, to speak of complex behaviour *rather than complex systems.*
G Nicolis and Ilya Prigogine (1989) [146]

10.1 Introduction

In the latter decades of the 20th century it became widely recogised that systems exhibiting complex behaviour are ubiquitous in nature. Physics Nobel laureate Philip Anderson gave his seminal 1972 paper the title *More is different* [145], pointing out that systems involving many-body dynamics and nonlinear interaction behave in a way that is both qualitatively and quantitatively different from extrapolations of simple systems. In 2021, the Nobel Prize for Physics was awarded for work on understanding complex systems [147]. Complexity science is now mainstream in multiple disciplines but not, yet, in cosmology. In this chapter I will summarise some of the fundamental differences between, on the one hand, the simple behaviour described in classical mechanics and equilibrium thermodynamics and, on the other, complex behaviour. I will draw attention to the implications for cosmology and introduce some of the tools that have been

developed to analyse complex behaviour and structures. In the next chapter I go on to focus on self-organised criticality, a particular paradigm that has emerged from complexity science in physics as an explanation for the form of emergent structures.

10.2 Complex systems

What is meant by 'complex'? Despite widespread use of the term there is still no concise or consensus definition of what a *complex system* is. Ladyman et al [148] present a useful summary of the literature on this topic and suggest that the following features are necessary for a system to be complex:

- The system is an ensemble of many elements.
- There is interaction between the elements of the system. This feature is necessary because independent particles have no means of forming patterns or establishing order.
- Complex systems are those in which order emerges from disorder, i.e. there is *self-organisation*. This is by contrast with a simple system, such as a gas, in which many similar elements interact but do not generate organisation. Self-organised systems are considered by some to be synonymous with being *dissipative*, a term introduced by Nobel Laureate Ilya Prigogine [149]. A characteristic of dissipative systems is that they are far from equilibirum.

They go on to list further features, such as nonlinearity and feedback, that are common in complex systems. Although consistent with the literature, classifying systems in this way is not particularly rigorous in practice. For example, complex systems can exhibit intermittent chaos as well as order. At the opening of this chapter there is a quote from Nicolis and Prigogine's classic and influential text [146], in which they suggest that it is more natural to explore *complex behaviour* than to seek to define a complex system. Following that approach, I will not seek to present a definition of a complex system but, in the remainder of this section, will focus on features and behaviour. The principal reference for this section is *Exploring Complexity* [146], where the terminology is standardised and explained and multiple examples are given.

10.2.1 Dissipation and irreversibility

Dissipative systems are those which are open to an exchange of energy or material flows. In the fields of evolution, biology and chemisty there has long been an awareness that dissipation can result in increasing energy and complex structure. In physics, such systems have historically been studied primarily in the context of energy loss by a system, for example as a result of friction. A broader awareness of dissipation in diverse transport mechanisms first began to emerge in physics during the 20th century in the field of fluid mechanics [146]. It is now well-established in the context of complex systems, but the term 'dissipative' is still not commonly used in its technical sense.

To offer a specific example: physicists are comfortable with the notion of a damped pendulum as a dissipative system, its motion exhibiting gradual energy loss. There is less awareness of a forced pendulum as a dissipative system, yet it is equally so. In that case the energy transfer is from the environment to the pendulum system. The energy transfer could be constant, periodic or irregular, and the impact on the motion of the pendulum in each case would be different.

Dissipative systems are associated with reliance on external energy flows to maintain their organisation - a tornado is a good example.

In contrast to open, dissipative systems, classical dynamics is concerned with conservative, i.e. closed, systems. Total energy, total momentum and total angular momentum remain constant in closed systems. The related conservation laws are functions of acceleration, i.e. a second derivative with respect to time, rather than position or velocity. Since a transformation of time from positive to negative leaves acceleration invariant, the equations of motion are reversible. In contrast, dissipative systems are not isolated from their environment and their macroscopic description must use collective variables to define the state of the system at any instant, for example temperature, pressure, convection velocity. The equations of evolution of these variables are not invariant under a transformation from time to negative time and the evolution of the system is seen to be irreversible. Time symmetry is broken [146].

10.2.2 Instability and nonequilibrium

Equilibrium states are stable and, by definition, stationary: their properties do not vary with time. In mechanics, all points of a system in equilibrium have zero velocity and acceleration. Hydrostatic equilibrium and thermodynamic equilibrium both relate to a state in which the properties of a system are in *detailed balance* with its environment or, in the case of an isolated thermodynamic system, with itself. Detailed balance is a term used by Prigogine and others to refer to the property that for any process which induces a variation, there is an inverse process inducing a variation in an exactly opposite direction. There are no net fluxes across the system.

If on the contrary there is a nonvanishing flux between between a system and its environment, as is the case in a dissipative system, then differences can arise in state variables and the system is in a state of nonequilibirum. Differences in state variables can be transient or, if appropriate conditions are maintained, they can become permanent. Prigogine and his collaborators referred to such maintained conditions as *constraints* [146]. It is important to note that, even in the case of long-standing state variables, the state is not equilibrium.

In the presence of constraints, which may of course be entirely natural:

- Nonequilibrium states become susceptible to change. Variation is to be expected, as detailed balance does not hold.
- Such variations can give rise to *bifurcations* to new states that are fundamentally different from equilibrium. Here, the nontechnical sense of the term bifurcation and its mathematical meaning are consistent: a splitting into two branches of solution, which are different from each other.
- New phenomena which involve long-range correlations emerge. A classic example in thermal convection is the creation of Bénard cells in a layer of fluid subjected to the constraint of heating. Below a critical point, the distribution of the fluid's molecules remains homogeneous and uncorrelated. Temperature is the same across the entire system. As the temperature is increased beyond a critical point, however, ordered and highly correlated convection cells spontaneously form[1]. (See Chapter 1 of [146] for a full

[1] Further heating, beyond a second critical point, disrupts the orderly cells and turbulence ensues.

description of this well-known experiment.) Long-range correlations are a key feature of self-organised criticality, which is the subject of the next chapter.

Dissipative systems are not in detailed balance so persist in nonequilibrium states: they are not stationary but instead demonstrate evolution and transitions between states. Prigogine [149] extended the second law of thermodynamics to systems that are far from equilibrium, suggesting that ordered 'dissipative structures' could form from disorder in such conditions. These structures cannot exist independent of their environment [150].

10.2.3 Dynamical evolution of state: nonlinearity and feedback

We have already seen that dissipative systems are not in equililbrium: they evolve. The equations which describe the evolution of state variables of a dissipative system must be compatible with the conditions imposed by energy exchange with the system's environment and, in the case that the conditions are absent, must revert to a stationary, equilibrium solution.

Equations of state variable evolution do not have a generic form but are instead specific to each physical case. What such equations do share is that their solutions are often (but not always) nonlinear. In a nonlinear system the superposition principle does not apply: if solutions to the equations that describe the system are summed, they do not yield another solution. Multiplying a solution by a factor does not yield another solution. 'In a nonlinar system adding a small cause to one that is already present can induce dramatic effects that have no common measure with the amplitude of the cause.' pg 59 [146].

When a part of a nonequilibrium system interacts with another part, the new behaviour of the second part becomes an input to the subsequent behaviour of the first. This is feedback, which fuels nonlinearity.

10.2.4 Inhomogeneity: self-organisation and structure

A concise definition of *self-organisation* is this: 'Self-organisation is a process in which the organisation of a system, or its susceptibility to a constraining action, spontaneously increases, and this increase follows without control of any external

forces of the environment.' [150]

The key features of emergent structure an be summarised as:

- **Self-organisation**

  Self-organised structures in complex systems are evolving, inhomogeneous (i.e. irregular) and often fractal. This has significant consequences for the suitability of statistical methods used in their analysis, a topic which I explore further in Section 10.4.

- **Transitions and evolution**

  I began this chapter with a quote from Nicolis and Prigogine [146]. Although they declined in their classic book to give a definition of complexity, they commented that an essential feature of complex behaviour is the capacity to make transitions between different states, so that evolution plays an important role in observed behaviour.

- **Nonequilibrium**

  Complex systems may exhibit quasi-stability but they are neither stationary nor static and, in all cases, small differences can result in radically different outcomes as a complex system evolves. Structure in nonlinear, dynamical systems is far from equilibrium. This is not inconsistent with the self-organisation feature above. Indeed, Nicolis and Prigogine state, 'Nonequilibrium may be a source of order' [149] p.25.

- **High energy states**

  There is no tendency towards minimum-energy states. On the contrary, as in the example of the tornado mentioned earlier, high energy is expected. I expand on this point in Section 10.3.2.

10.2.5 Symmetry breaking and order

Just as time symmetry is broken in dissipative systems, the absence of homogeneity in nonequilibrium states results in the breaking of spatial symmetry. Nicolis and Prigogine comment, 'Broken symmetry accompanies the appearance of new properties that prompt us to charcterize the material as *ordered*' pg 41 [146].

10.2.6 Comments on chaos

Complexity and chaos are often conflated in common perception but, although they can be related by shared characteristics such as feedback and nonlinearity, they are different. In particular, neither chaos nor nonlinearity are necessary or sufficient for complexity [151].

Chaos theory began with Poincaré and his work on the 'three body problem' in Newtonian gravity. It is a branch of the theory of dynamical systems addressing systems whose time evolution is sensitively dependent on initial conditions. Mathematically, the property is captured by Lyapunov exponents and spectra[2].

The behaviour of chaotic systems is often indistinguishable from random behaviour because it is unpredictable yet they are, in fact, deterministic: their future state is completely fixed by their present state coupled with the laws that govern them [152].

10.3 Implications of a complex universe

10.3.1 A complex universe

The universe has many, interacting components; its interaction is nonlinear; and it evolves self-organised structure. Gravity, the mechanism for interaction, has infinite range, therefore cosmic structures must all be dissipative. It seems clear that the universe is a complex system and it is surprising that there is not a consensus that this is so.

Complexity has profound implications for the science of cosmology. In particular:

- A model of cosmic evolution that treats the universe as a homogenous and isotropic perfect fluid is not obviously justifiable, even if gravity were a linear interaction (which it is not).

[2]The Lyapunov exponent of a dynamical system is a quantity that characterizes the rate of separation of infinitesimally close trajectories. There is a spectrum of Lyapunov exponents, equal in number to the dimensionality of the phase space of the system, because the rate of separation can be different for different orientations of initial separation vector.

- There are significant implications in relation to emergent structure. This is essentially the subject of Chapter 11.
- Since complex systems do not attain equilibrium and do not establish minimum-energy states there are implications for the mass-energy of cosmic structures. The effect of complex interaction on mass-energy in general is outlined in Section 10.3.2. The impact for cosmic structures in particular is addressed in Chapter 14.
- The statistical tools of equilibrium thermodynamics are not appropriate for the analysis of complex systems. In Section 10.4 I signpost to alternative statistical methods that can be applied to irregular as well as regular distributions.

10.3.2 Complex interaction impacts on mass-energy

To illustrate the high-energy of complex interaction, and its associated impact on mass-energy, I report three examples. Two are physical, at very different scales: merging black holes and the proton. The third example is an N-body simulation based on simple rules.

Example 1: binary black hole merger

In an otherwise empty universe populated by two massive bodies initially at rest, Newtonian theory predicts that the bodies accelerate towards each other with simple equations of motion and the mass-energy of the system is a straightfoward function of the intrinsic masses of the two bodies.

In general relativity (GR) the situation is different. Even from rest and with no other interactions, no general equations of motion can be defined, since the spacetime curvature is subject to feedback and continual change. In GR, even a binary system exhibits complex behaviour. What, then, is the mass-energy of the binary system?

For gravitational-wave detector experiments it is necessary to answer this question with high precision, and several groups have worked on the problem in recent decades. Two complementary methods are used to approximate the nonlinear solution. For the initial stages of an inspiral the post-Newtonian approximation is used and this is brought together with a numerical relativity result for the

final infall phase (see Appendix H). Numerical relativity calculates solutions to 'relaxed' field equations which incorporate a pseudotensor[3] for field potential. This is an effective approach for two bodies which are very close to one another but it cannot be extrapolated generally because the pseudotensor would need to be restated for every applicable inertial frame. For a binary system, the post Newtonian and numerical relativity methods combine to generate a better-than-Newtonian model of the evolution of the binary system's mass-energy [153].

The first detection of a gravitational wave was reported by the LIGO collaboration in 2016 [154], from a binary black hole merger. The modelled initial black hole mass-energies were $36^{+5}_{-4}M_{\odot}$ and $29^{+4}_{-4}M_{\odot}$ with a final black hole mass-energy $62^{+4}_{-4}M_{\odot}$. Radiation as gravitational waves amounted to $3.0^{+0.5}_{-0.5}M_{\odot}$. This example demonstrates that there is net gravitational energy release when the system undergoes a transition from binary to a combined, single body. In other words, the mass-energy of the dynamical, interacting, two-body system exceeded the mass-energy of the final, unitary black hole.

Example 2: the proton

Let us now move on to the scale of nuclear and particle physics. The mass scale for all normal atomic matter in the universe is set by the proton. Unlike the mass of an electron, which is an elementary particle, the mass of the proton is not fully understood. In the concluding chapter of their classic text on particles and nuclei, Povh et al [155] comment that, 'The best description always seems to come from the framework of an "effective theory" chosen according to our experimental resolution. This is by no means a peculiarity of the complex systems of the strong interaction, but is a general property of many-body systems.'

In the Standard Model of particle physics protons are systems of three quarks confined in a self-interacting gluon field. Quarks have no internal structure and the rest-frame inertial mass of each quark, m_q, is attributed to interaction with the Higgs field. The strong force carriers, gluons, are also elementary particles but do not interact with the Higgs field so they are massless. However, the empirical rest mass of a proton is not $\sum m_q = 9.8$ MeV as might naively be expected, but is two orders of magnitude greater, $m_p = 938.27$ MeV. Only $\sim 1\%$

[3] A pseudotensor is an object which looks like a tensor but, unlike a real tensor, is not invariant under coordinate transformations.

of the proton mass is attributable to the Higgs mechanism.

Research in nuclear physics has shown that a part of the additional mass arises from purely quantum effects. However, the majority of it is a result of energetic dynamics and nonlinear (self) interaction, i.e. complexity. For the interested reader, I have briefly summarised in Appendix I relevant aspects of current proton mass research in the context of the Standard Model QCD Lagrangian.

Example 3: an N-body model

Tang et al [156] set out to examine the behaviour of a system with multiple effective degrees of freedom and complex dynamics using a model simple enough to be characterised theoretically. The model of an array of N balls of mass m, each connected to its neighbours by springs with a force constant K, was subject to a sinusoidal potential driven by a time-periodic square-wave force. They found that rather than equilibrium, the dynamics of the system selected configurations of minimal stability. Contrary to the expectation of equilibrium statistical mechanics, the peak of the probability density of total elastic energy of the chain was not at the energy minimum. In fact, the minimum-energy configuration was the least likely state to be observed and the maximum energy configurations were most heavily weighted.

10.4 Statistical methods for complex systems

10.4.1 Context

Statistical physics is the study of the macroscopic behaviour and properties of systems which have sufficiently large numbers of constituent components for statistics to be applied. In different terminology, it is the study of systems with many degrees of freedom.

In their thought-provoking book *Statistical Physics for Cosmic Structures* [85], published two decades ago, Gabrielli, Labini, Joyce and Pietronero begin by commenting that:

> Cosmology has, in particular in the development of its formalism since the 1970s, borrowed tools from statistical physics. The mathematical language used to describe correlations in galaxy catalogs, for

> example, was imported from the theory of liquids. Direct exchange or even collaboration between people in the two communities has, however, been rare. Cosmology has thus tended to seek instruments when necessary from statistical physics, but has not been very much in contact with or influenced by the many developments which have taken place in statistical physics in the last decades.
>
> Gabrielli et al (2005) [85], pg.1

It is a sobering thought that, in twenty years, very little has changed. There is still an emphasis in cosmology on traditional statistical methods that are valid only if the *a piori* assumption of homogeneity holds. Those methods do not enable that assumption to be tested.

The traditional tools of theoretical physics are analytical functions and differential equations. The efficacy of these tools is predicated on smooth functions and structures, with irregularities treated as perturbations or isolated singularities. By contrast, complex systems give rise to structures that are intrinsically irregular and nonanalytic. Figure 10.4.1 illustrates the distinction between regular and irregular, analytic and nonanalytic.

In recent decades renormalisation group theory and other new developments have occurred which take statistical physics into the realm of irregular distributions, scale transformations and nonequilibrium systems, including critical phenomena [158]. Does it matter that these new developments have not been incorporated into the statistical physics toolkit of cosmology? Already in the late 1990s Labini, Pietronero and others were arguing that it does matter [157]. I find their argument compelling, particularly as the alternative tools they suggest are equally applicable to homogeneous distributions and, indeed, can be used to *test* the homogeneity assumption.

Complex systems generate irregular measure distributions. For such systems, the tools of fractal analysis are appropriate but the tools of equilibrium statistical physics are not. That does not mean that complex systems always create mathematically perfect fractals - in nature that is not the case. In that sense, the impassioned debate within the physics community about whether or not the universe has fractal structure, which I touch on in the next chapter, is barely relevant. Rather, in relation to the efficacy of particular statistical methods,

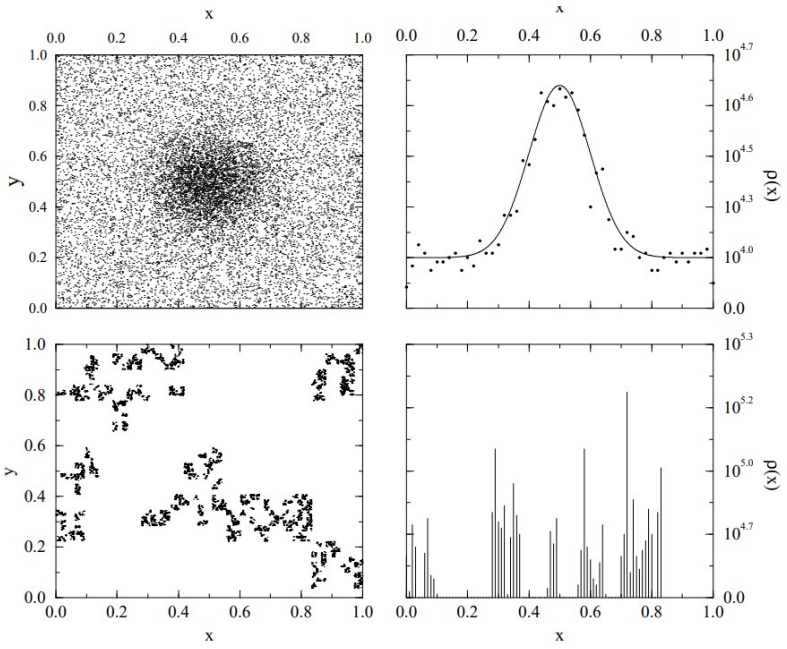


Figure 10.4.1: Example of analytical and nonanalytic structures. Top panels: (Left) A cluster in a homogeneous distribution. (Right) Density profile. In this case the fluctuation corresponds to an enhancement of a factor 3 with respect to the average density. Bottom panels: (Left) Fractal distribution in the two dimensional Euclidean space. (Right) Density profile. In this case the fluctuations are nonanalytical and there is no reference value, i.e. average density. The average density scales as a power law from any occupied point of the structure. (From [157])

a key question is this: does the total population we wish to analyse have a well-defined and positive mean density?

If the answer is 'yes', then it may be possible to use the standard tools of equilibrium statistical physics. This is especially so if the system is *statistically homogeneous*, i.e. its distribution does not change under spatial transformation. (We should note that statistical homogeneity is not the same as the more demanding quality of homogeneity.) A statistically homogeneous system can be analysed using the theory of Stationary Statistical Processes (SSPs), where *stationary* means invariant under spatial translation. If it is also *isotropic*, i.e. its

distribution does not change under rotation, that is even better. In either case, large enough samples of the total population can be expected to be representative of the whole, and the familiar two-point correlation function and its Fourier transform, the power spectrum, are suitable analysis tools.

If the answer is 'no', because the distribution is irregular and there is no well-defined, positive mean density, we enter into the regime of fractal analysis. This is irrespective of whether the distribution is a mathematically rigorous fractal or multi-fractal: the fact that it has no well-defined mean density is sufficient to invalidate the traditional tools. Samples are not representative of the whole, and two-point correlation functions are not only unreliable, they are mathematical nonsense and actively misleading. In those circumstances, the newer tools of fractal analysis are required.

If the answer is 'probably - our hypothesis is that the system is homogeneous', then it is important to test that hypothesis. Tools that assume the hypothesis is correct and are valid only if it is, cannot adequately fulfil the need.

The new tools for statistical physics in irregular systems do not, by themselves, shed any light on the physical processes driving the evolution of the universe. To bridge that gap, in Chapter 11 I draw attention to self-organised criticality, a concept that is mainstream in many disciplines in physics but not, yet, in cosmology.

10.4.2 Uniformity, regularity and traditional methods

The most simple hypothesis for representation of finite samples of a stochastic field is to use a uniform average background with imposed stochastic fluctuations. This framework, referred to as the *theory of stationary stochastic processes* (SSP), is well understood: fluctuations can be treated as perturbations around a mean field, and suitable perturbation theories can be developed. The approach is applicable to both (a) continuous processes, i.e. those in which the fluctuating signal in space is continuous and practically constant over sufficiently large regions; and (b) statistically stationary spatial measure distributions of discrete point particles, as long as there is a well-defined average.

As already mentioned, *stationary* refers to statistical homogeneity, a spatial

characteristic. It is equivalent to statistical translational invariance.

A particular mass or measure density field may be thought of as a single realisation of the SSP.

It is common to describe and analyse spatial mass distributions in cosmic structure by measuring the correlation between density fluctuations using the reduced two-point correlation function

$$\xi(r) \equiv \frac{\langle n(r)n(0)\rangle}{\langle n\rangle^2} - 1$$

In the case of a statistically homogeneous process the power spectrum $P(k)$ is the Fourier transform of the correlation function $\xi(r)$.

This two-point correlation function is suitable for analyses carried out in the context of standard cosmological models, which assume statistical homogeneity. The approach has routinely been applied to the temperature fluctuation field of the CMB radiation (see Section 9.3.3), and to the spatial distributions of cosmic structures.

There are, however, some density or measure fields for which the statistical tools of SSP give spurious and invalid results, in particular when the mean density $\langle n\rangle^2$ is not well defined. An important example is densities or measures which have fractal distributions. In such cases, the average density decays slowly from any point in the set but is asymptotically zero. Therefore, the estimator of the average mass density in any finite sample gives an infinite relative error with respect to the actual, zero value in the population. There is a further implication, that the distribution is *extremely* irregular. In fact, it is *singular* at every distribution point.

It is therefore essential that, before applying the simple and well-understood SSP statistical framework, the hypothesis that they are appropriate is tested. The tests cannot be carried out using SSP but must instead use statistical physics that can be applied to irregular distributions. These mathematical and statistical methods are usually called fractal analysis, which is the subject of the next section.

10.4.3 Irregular and nonanalytic structures: fractal analysis

In recent decades Pietronero, Labini and others have devoted substantial effort to defining and applying a statistical framework for studies of irregular, including fractal, structures (see for example [85, 159–167]. I briefly outline the nature of fractals in Appendix J, including the definition of 'fractal dimension' which is a key characteristic of the statistical behaviour of irregular distributions.

I have already commented that the average density of a fractal mass distribution is zero in the infinite volume limit. In that case, average density in a finite sample has no intrinsic meaning, because it depends on sample size. In practice, calculating a precise fractal dimension for a complex physical system using, for example, the Hausdorff dimension defined in Appendix J may be impossible. One can instead examine the *conditional density* which, for stochastic and isotropic distributions, is defined as [168]:

$$\langle n(\overrightarrow{r})\rangle_p = \frac{\langle n(r)n(0)\rangle}{\langle n(0)\rangle}$$

for spherical shells of radius r. The integrated condisional density gives the average density of particles in spheres of volume $V(r)$.

These measures can be used to identify the scale at which the regime of large fluctuations gives way to that of small fluctuations, i.e. the scale at which the average density becomes well-defined, if indeed it does.

To extend the two-point description of fractal mass distributions to three-point statistical properties the concept of lacunarity, i.e. void distribution, is introduced. Deeper characterization of strongly irregular distributions uses the concept of measure theory. Gabrielli et al provide a thorough exposition of these and other tools [85].

For completeness, I note that in respect of statistical mechanics for GR there are other approaches that can also be explored. These include, for example, work published on non-local kinetic and generalized hydrodynamic equations [169,170]

10.4.4 Spatial and temporal: the impact of GR

The preceding sections in this chapter summarise work by Labini, Pietronero, Gabrielli and others. In this section I state a new aspect of the problem of finding appropriate tools for the statistical analysis of our universe: the problem of variant time in GR. I address GR in some detail in Chapter 12. Here, I wish simply to point out that mass-energy distributions in the cosmos are, in the theory of GR, functions of both space and time. Time is not an invariant, background measure. One cannot take a single time slice and obtain 'accurate' analyses of spatial distributions on it. Spacetime, everywhere, is relative and subject to gravitational waves.

In the standard model of cosmology this aspect of time variance is ignored completely, relying on the model-dependent assumption that the effect is negligible because spacetime is, essentially, flat Euclidean space. In a different theoretical regime, which allows the possibility that the universe is not a smooth and homogeneous distribution but is instead irregular, the flat spacetime assumption cannot be consistent with the theory of GR.

I do not yet have an answer to offer with respect to how this can be addressed in a statistical framework. The problem is included in Section 15.1.3.

Chapter 11

Self-organised criticality

Self-organized criticality is a new way of viewing nature.
Per Bak (1996) [171], p.xi

The idea of self-organised criticality seems to me to be, not the right and unique solution ... but to have paradigmatic value, as the kind of generalisation which will characterize the next stage of physics.
P. W. Anderson (2011) [172], p.112

11.1 SOC: an explanation of emergent structure

In the preceding chapter I introduced complexity and commented briefly on statistical physics for complex systems. The tools of fractal analysis I referred to there facilitate analysis of complex structures but they do not by themselves offer explanations of dynamical physical phenomena. In this chapter I turn to another facet of statistical physics to make good this deficiency. I propose self-organised criticality (SOC) as an appropriate paradigm for study of the *behaviour and evolution* of structure in the complex, dynamical universe. I choose to describe SOC as a paradigm rather than as a model or a theory because it describes a broad family of processes which, whilst they share certain statistical characteristics, can arise within vastly different physical systems.

SOC emerged from the disciplines of statistical mechanics and condensed matter theory. It was introduced by Bak, Tang and Wiesenfeld (BTW) [1] in 1987 in a paper that now has more than ten thousand citations in peer reviewed journals in multiple disciplines.

SOC was originally presented as an explanation for two empirical observations of nature: that spatially extended forms often exhibit self-similar fractal structure [173] and that long-range correlations in time series are ubiquitous in diverse transport systems (see e.g. [174]). The existence of fractal structure in nature had been known since the pioneering work of Mandelbrot [175] but the mechanism by which these structures form was previously unknown. Mandelbrot's work implied a common dynamical origin for large and small events [113], as their statistical probability distribution functions are the same: Pareto-Levy functions. The probability of large events is given by the tails of those distributions, which fall off as power laws, i.e. much slower than Gaussian distributions. The time-series phenomenon, commonly termed *1/f noise*, is associated with low-frequency power spectra displaying power-law behaviour characterised by a power spectral density $P(f) \propto f^{-\alpha}$, where f is frequency, over vastly different time scales[1]. Bak and his collaborators argued that the spatial and temporal phenomena, both of which involve long-range correlations, are related. They presented a heuristic argument and simple numerical models.

The heuristic argument, which is independent of the details of any particular physical system, is compelling. Consider a many-body, dynamical and dissipative system with spatial degrees of freedom. As components interact in the presence of a slow-drive[2] energy source, structures emerge. These structures continue to evolve until they reach a minimally-stable, *critical* state. The local state is critical in the sense that it is related to phase transition: when the local energy accumulation breaches a threshold in the physics of that system, an abrupt cascade, or *avalanche* of energy release occurs. That *noise* propagates through the scaling clusters, disturbing the barely stable states and causing a cascade of energy dissipation on all length scales. The outcome is spatio-temporal correlations which manifest scale-free, fractal structure. Note that the critical

[1] Except for $\alpha = 0$, 'white noise', which represents random, uncorrelated fluctuations.

[2] Here *slow*-drive refers to an energy input that is slow in relation to intermittent and rapid cascades of energy release.

point is not the result of fine-tuning a parameter such as temperature but is instead an attractor reached by starting far from equilibrium.

Details of specific models and examples of SOC systems are given in, for example, Jensen [176], Preussner [177] and Aschwanden et al [178].

Historically, SOC has engendered some controversy [179]. This was largely driven by resistance to early claims that all systems in nature demonstrate SOC, as implied by the title of Bak's popular book *How Nature Works* [171], and by the failure to find the expected universality classes[3] [177]. As the concept has matured, however, SOC has fuelled substantial and fruitful research in complex systems in many fields, including solar and astrophysics [180] and plasmas [181]. SOC has also provided motivation for recent application in quantum fields, termed 'self-organised localisation' [182]. Very little has yet been written about the application of SOC at cosmological scales [178], although an interesting early paper by Smolin [183] does explicitly address the issue of criticality and self-organisation, in the context of both spiral galaxies and the large scale organisation of the universe.

11.1.1 Definitions

I first clarify the meaning of terms used in definitions of SOC:

- **Self-organisation**. This is not a new term introduced with SOC, it is a pre-existing concept in complexity science and is defined in Section 10.2.4. An alternative but consistent definition is give by Jensen [176]: 'An ability by certain nonequilibrium systems to develop structures and patterns in the absence of control or manipulation by an external agent'.
- **Criticality**. The technical origin of *criticality* lies in equilibrium thermodynamics, in the context of phase transitions. It refers to a state in which a disturbance decays algebraically with all members of the system influencing each other, as opposed to a disturbance in a non-critical system in which a disturbance falls off exponentially amongst local neighbours.

[3] In the context of ordinary critical phenomena certain dimensionless universal quantities such as critical exponents, the scaling function and ratios of amplitudes are independent of many details of a system. Systems that share universal features and share these characteristics are said to be in the same *universality class*.

Whilst criticality in SOC is also associated with phase transition, there are the following fundamental differences:

- In equilibrium thermodynamics, achieving a critical state requires fine-tuning of an appropriate parameter such as temperature. The critical point in dynamical SOC systems is not a result of tuning any parameter but is instead an attractor reached by starting far from equilibrium. The scaling properties[4] of the attractor are insensitive to the model parameters.
- Whereas there is a clear mathematical framework for calculating the statistical properties, including critical points, of equilibrium systems, there is no equivalent formalism to calculate the probablility that particular configurations of a dynamical system will occur. In SOC in particular, there is the problem that sharp thresholds in dynamics introduce highly nonlinear and nonanalytic terms in the equations of motion [176].
- In SOC, criticality does not mean that the system organises itself into a state where every local degree of freedom is close to some threshold [179].

- **Scale invariance** refers to invariance of a system or its fluctuations under scale transformations, which is the common usage in statistical physics. This is different from 'scale-invariant' used in cosmology to refer to, for example, the primordial power spectrum. The latter usage refers to a property of a specific system, in which the amplitude of fluctuations remains invariant as the 'horizon scale' increases [85].

It is now possible to define **Self-organised criticality (SOC)**. A modern definition of SOC which I consider to be useful is given by Preussner, and defines SOC as a phenomenon rather than a class of systems: 'SOC is regarded as ... scale invariance without external tuning of a control parameter, but with all the features of the critical point of an ordinary phase transition, in particular long range (algebraic) spatiotemporal correlations. In contrast to (other) instances of generic scale invariance it displays a separation of time scales, avalanches, is interaction dominated and contains nonlinearities.' [177]

[4]The scaling properties of an attractor include: fractal dimension, which quantifies how it scales with the size of the system (see Appendix J); correlation dimension, measuring the density of points on the attractor; and scale dependence.

Aschwanden et al use the following alternative but consistent definition: 'SOC is a critical state of nonlinear energy dissipation system that is slowly and continuously driven towards a critical value of a system-wide instability threshold, producing scale-free, fractal-diffusive, and intermittent avalanches with powerlaw-like size distributions.' [180]

11.1.2 Necessary conditions and phonomenological features

There is broad agreement about the *necessary generating conditions* for SOC. In 1998 Jensen [176] proposed that SOC behaviour would be observed in slowly-driven, interaction-dominated threshold systems (SDIDT). Watkins et al [179] noted that SDIDT need not exclusively be SOC but the conditions for SOC they proposed are effectively SDIDT:

- Nonlinear interaction in the form of thresholds.
- Intermittent avalanches, i.e. sudden and extreme releases of energy, which are to be expected in the presence of thresholds and slow driving.
- Separation of time scales, i.e. avalanches are distinct.

Implicit in the 1987 BTW paper [1], explicit in [113] and recognised in both of these definitions is the important point that the dynamical systems must involve interaction as well as spatial degrees of freedom for structure to emerge. For example, a quantum many-body system will relax from a non-Gaussian initial state to a Gaussian state under non-interacting dynamics - but recover its non-Gaussian condition when interaction is restored [184]. Similarly, in the temporal case, electrical systems produce 1/f noise when there is a current flowing but not otherwise [174].

The key *features of SOC phonomenology* are [179]:

- Non-trivial scaling without dependence on a control parameter.
- Spatio-temporal power law correlations.
- Apparent self-tuning to the critical point.

11.2 Is the universe a SOC system?

It cannot be claimed that all complex systems display SOC behaviour. There is no *sufficient* condition that guarantees that a system can be classified as exhibiting SOC, but there are generating conditions that are required and there is characteristic phenomenology. Therefore, I turn now to the question of whether SOC can be excluded as a viable paradigm for cosmology - in other words, whether any of the required conditions or the characteristic phenomenology is missing.

11.2.1 Generating conditions

We recall that the generating conditions for SOC behaviour are the presence of: a slow drive energy source; interaction-domination; thresholds and associated intermittent avalanches [176, 179].

Let us first consider the case for gravity as an energy source in the universe. Smolin [183] observes: 'While gravitationally bound systems may spend long periods of time in quasi equilibrium configurations, they do not reach true equilibrium states, characterized by maximal entropy. The reason is that they have practically inexhaustible sources of energy ... gravitationally bound systems such as galaxies can maintain significant flows of energy for cosmological time scales.' Furthermore, I will show in Chapter 12 that in the theory of GR energy is not conserved in any subsystem of the universe. Gravitational systems are dissipative.

The gravitational energy source is generally 'slow' drive because gravity is weak except in extreme conditions.

We observe intermittent energy release associated with thresholds: in an early review of SOC studies in astrophysical accretion systems Dendy et al [185] concluded that the potential sites for SOC in such systems are numerous. More generally, thresholds are designed into modern hydrodynamical simulations of galaxy evolution. They are required in order to simulate realistic structure (see, for example, the EAGLE (Evolution and Assembly of GaLaxies and their Environments) simulation [134, 186]). At this level, the specific physics of threshold feedback processes include Active Galactic Nuclei (AGN), exploding massive stars and core collapse supernovae, explosive events which limit the fraction of gas that forms stars as a galaxy evolves.

There is no obvious converse argument that thresholds and associated sudden energy release do not occur.

11.2.2 Characteristic phenomenology

Turning now to the phenomenology of SOC systems, the key features are [179]:

- Non-trivial scaling without dependence on a control parameter.
- Spatio-temporal power law correlations.
- Apparent self-tuning to the critical point.

Simply put, SOC systems evolve to self-organised fractal structure. The usual assumption, fundamental to FLRW cosmology and described in Chapter 9, is that the universe is isotropic and homogenous at large scales, not fractal. As we have seen in the previous chapter, the statistics of conventional cosmological modelling are only valid if this assumption holds. But does the universe in fact display fractal structure? The first person to pose the question was Mandelbrot himself in 1975, referenced in [187]. Using simple toy models, Mandelbrot predicted that not only clusters but also filaments, voids and walls would emerge in cosmic structure.

Fractals are usually characterised by their dimension. As I have explained in Section 10.4, fractal dimension is defined for an entire population: samples cannot be representative of a fractal structure. However, we must work with the data we have. An early study by Peebles [188] concluded that the space distribution of galaxies on scales less than about 15 Mpc h^{-1} could be described well as fractal with dimension $D = 1.23 \pm 0.04$, but that this was not the case at larger scales. Two decades later Labini et al [165], using the Sloan Digital Sky Survey, found inhomogeneities compatible with a continuation of fractal correlations but incompatible with a homogeneity scale smaller than 100 Mpc h^{-1}. They concluded that the evidence demonstrates that the large amplitude density fluctuations are limited only by sample sizes. Another quantitative study of average conditional density in the Sloan Digital Sky Survey Release 7 data concluded, 'Due to the scaling and data collapse we argue that the large scale galaxy distribution shows similarities with critical systems.' [189])

At larger scales, it has become clear that our universe does indeed exhibit

a pattern of clustering, filaments and voids. With each new release of data from sky surveys, the scale of identified structures has increased: in the 1986 CfA2 Redshift Survey the Great Wall was observed, a structure with size approximately 200 x 80 x 10 Mpch^{-1}, limited only by the sample size. In 2000, the Sloan Great Wall, some three times larger than the Great Wall, was identified [190]. The relative scale of these structures[5] is shown in Figure 11.2.1. In 2024 an ultra large scale structure described as the Big Ring, some 400 Mpch^{-1} in diameter, was seen in Mg II-absorber catalogues [191]. In 2025, using X-ray galaxy clusters to map matter density distribution, a 400 Mpch^{-1} structure, named Quipu, has been identified [192].

In 2021 De Marzo, Labini and Pietronero [167] used the properties of the Zipf-Mandelbrot law[6] to complement the standard correlation function analysis to examine the size distribution of superclusters of galaxies. Their conclusion was that galaxy superclusters are well described by a pure Zipf's law with no deviations and that currently available catalogues are not sufficiently large to spot a truncation in the power-law behaviour.

There are counter-arguments to these findings and some studies have concluded that the transition to homogeneity occurs at less than 100 Mpc h^{-1}. One such study, based on analysis of the WiggleZ Dark Energy Survey data, reports that for a survey of over $200,000$ blue galaxies in a cosmic volume of ~ 1 Gpc^3 h^{-3} the mean number of galaxies in spheres of comoving radius r is proportional to r^3 within 1 per cent at radii larger than 71 ± 8 Mpc h^{-1} at $z \sim 0.2$, 70 ± 5 Mpc h^{-1} at $z \sim 0.4$, 81 ± 5 Mpc h^{-1} at $z \sim 0.6$ and 75 ± 4 Mpc h^{-1} at $z \sim 0.8$ [193]. This work also provides useful references to other studies, including some of the papers I have cited in this section. For our purposes, the WiggleZ analysis is interesting but cannot be conclusive, since it explicitly uses a cosmological model to underpin the analysis: 'Certain parts of our analysis require the assumption

[5]Note that the characteristic scale, a measure defined as the scale at which fluctuations in the galaxy density field are rougly twice the value of the sample density, is $r_0 \sim 5$ Mpc/h. This scale is indicated by the small dot at the bottom of the image. To give physical meaning to r_0 in a finite sample it is necessary to verify that the average density is a well-defined concept: in a fractal structure it is not.

[6]The Zipf–Mandelbrot law (otherwise known as the Pareto-Zipf law or as Korčak's law) is a discrete power-law distribution generalised from Zipf's empirical law formulated using mathematical statistics that refers to the fact that for many types of data studied in the physical and social sciences, the rank-frequency distribution is an inverse relation.

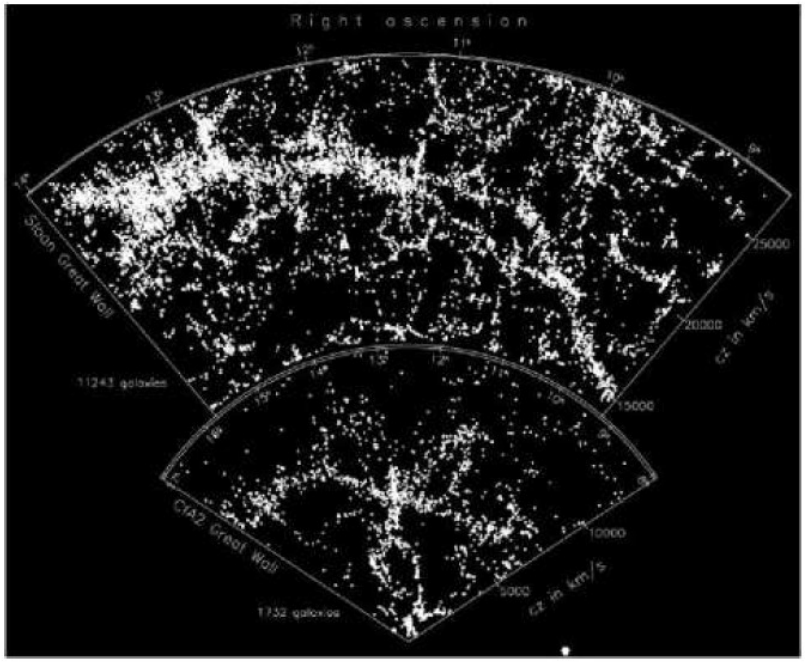


Figure 11.2.1:

Reproduced from Labini and Pietronero [164]: Latest progress in redshift surveys. SDSS Great Wall (2003) compared to CfA2 (1986) Great Wall at the same scale. Redshift distances cz are indicated. The small circle at the bottom has a diameter of 5 Mpc/h, the clustering length according to the standard interpretation of galaxy correlation. The SDSS slice is 4 degrees wide, the CfA2 slice is 12 degrees wide to make both slices approximately the same physical width at the two walls.

of a cosmological model and, implicitly, homogeneity (i.e. for converting WiggleZ redshifts to distances, correcting for the selection function, calculating the uncertainties using lognormal realisations, and finding the best-fitting bias). In these cases we use an input ΛCDM cosmology...' [193].

I comment that I am not seeking here to establish that the universe *is* fractal, only to explore whether there is valid statistical evidence that it is not. It would be an interesting piece of work to review the WiggleZ paper alongside analyses that conclude the scale of homogeneity is not yet apparent in other surveys but, as it stands, the WiggleZ analysis is ultimately dependent on the assumption of homogeneity. In particular, since the WiggleZ survey was not

large enough to contain spheres of the size at which homogeneity was claimed to set in, the protruding regions were populated with galaxies drawn from a random, i.e. homogenous, distribution.

Finally, I note that SOC does not suggest that the entire system is expected to be in a critical, i.e. fractal, state at all times.

11.2.3 Implications of the CMB

A common argument in support of the FLRW assumptions is the isotropy of the CMB at $z \sim 1100$ [104]. However, in the SOC framework: (a) initial conditions are not relevant to the emergence of structure so homogeniety and isotropy in the past are not evidence of an absence of fractal structure now; and (b) SOC does not suggest that all structure is at all times is in a critical state, only that it evolves to it.

11.2.4 Conclusion: SOC is not excluded as a possibility

I conclude that the conditions necessary to generate SOC-behaviour cannot be shown to be absent and that there is not yet any definitive evidence that SOC-consistent fractal phenomenology is not present in the universe.

11.3 Theoretical implications of SOC

There are some very interesting implications of SOC phenomenology. For my purposes the following are particularly significant:

- The emergence of structure is an inherent property of SOC and is independent of initial conditions.
- It is not the case, however, that all parts of a system are perpetually at a critical point [179].
- Mass-energy is expected to accumulate as configurations evolve to critical states.
- At the critical point, configurations with maximum mass-energy are statistically preferred [156]. This is directly contrary to the minimum energy expectation of a system in equilibrium.

- Avalanche dissipation occurs when a threshold is breached. Thereafter, the slow-drive accumulation again takes over.
- Since the avalanche dissipation from a critical point affects the entire system algebraically, long-range spatio-temporal correlations occur.
- There is no well-defined mass density in a fractal system: therefore samples cannot be representative of the population and equilibrium statistical tools are not applicable.
- We observe the universe from a point in spacetime which is special, in the sense that we are in the fractal.

11.4 Working with SOC

SOC can predict neither a specific system configuration nor the mass-energy of a galaxy or cluster. Yet, 'The power of SOC lies in the ability to both describe and explain a large variety of physical systems in a quantitative and physically-motivated manner' [194]. SOC is statistical physics. Crucially, unlike other statistical tools used in the analysis of structures, it also offers an explanation of how and why structures form in the universe.

SOC warns that the statistical tools applicable to homogeneity and equilibrium are not only inappropriate but actively misleading. Yet it is important to clarify that there is no outright, universally applicable mathematical formalism for SOC [176]. Numerical simulations exist for a number of classes of system, and a review of models, including the archetypal 'sand pile' model introduced by BTW in 1987, is given in Preussner [177]. I am not aware of any extant SOC model built with a general-relativity-like nonlinear interaction .

What is required in order to work with SOC, then, is a combination of:

1. Analysis tools that do not assume, *a priori*, large-scale homogeneity and distribution of matter around a well-defined mean. As we have seen in Section 10.4, appropriate tools do exist, broadly categorised as fractal analysis.
2. A paradigm which supports investigation of observational data without requiring exact solutions to GR field equations and without *a priori* as-

sumptions of specific initial conditions coupled with a linear lifecycle for the universe as a whole. This paradigm is supplied, I suggest, by SOC.

I discuss the next steps more fully in Chapter 15. In the meantime, I move on to address the theoretical implications of GR, when the theory is freed from the restrictions of the conventional FLRW framework.

Chapter 12

General relativity - back to basics

Spacetime tells matter how to move; matter tells spacetime how to curve.
J.A. Wheeler [195] pg 235

12.1 The case for general relativity

General relativity (GR) is the most complete description of the geometrical properties of the universe available and the theory has, so far, passed every experimental challenge [196].

GR has become so strongly associated with the FLRW framework that it is easy to forget that FLRW is not, actually, GR at all. The FLRW framework, which we described in Section 9.1, is a set of assumptions and approximations that allow solutions to the field equations to be found. The price paid is that GR is stripped of its fundamental nature: relativity in space and time and the effect of interaction are excluded. In this section we go back to basics and restore attention to the profound implications of the full theory.

12.2 Key principles and terms

The following key terms and principles are employed:

- **Equivalence principle**. It is always possible to find a frame of reference in which all local gravitational fields disappear.

- **Mass** Surprisingly, there is no rigorous and unambiguous definition of *mass* (see Jammer's works [197, 198] for interesting reviews of the topic). Nevertheless, physicists work with mass. In the Standard Model of particle physics mass is a function of coupling with the Higgs boson but, even at the still subatomic scale of the proton, we have seen in Section 10.3.2 that this is not a sufficient definition as only ~ 1 % of the proton mass is attributable to the Higgs mechanism. As a working definition we follow Misner et al and say 'Mass is the source of gravity' [86] pg. 404.

- **All forms of energy contribute to the gravitational field of an object** Einstein [199]. Thus mass and energy are equivalent.

- **Gravitational field**. We again follow Misner et al [86] and use the term *gravitational field* to mean both the geometry of spacetime and the various mathematical entities that collectively are associated with gravitation including the metric, the Riemann and Ricci curvature tensors, the covariant derivatives, connection coefficients and so on which are on the left-hand side of Einstein's field equations 9.1.2.

- **Matter**. Weyl said, 'We shall assign the term "matter" to that real thing, which is represented by the energy-momentum tensor' ([200] pg.203). Weyl meant, in the context of his time, the states and dynamics of 'real' normal matter and radiation fields. We have seen in Section 9.1 that Einstein's introduction of a cosmological constant term in equation 9.1.2 required the energy-momentum tensor to incorporate an additional source of energy, Λ, that the standard cosmological model associates with dark energy. And, of course, the standard model also requires non-interacting dark matter. When we refer to *matter* in this chapter we do so in the original spirit of Weyl.

- **Baryonic matter**. In cosmology it is conventional to refer to normal (as opposed to dark) matter as *baryonic* matter. More precisely, it refers to matter made up of Standard Model particles except for electrons (which have, relatively, negligible mass) and neutrinos (which, in the Standard Model, do not have mass). We prefer to avoid the term but, where it is quoted in this thesis, it is in that context.

- **Timepoint**. Usually described as an 'event' we define a *timepoint* to be an infinitesimal point in four-dimensional spacetime.
- **Timespan-region**. We call an extent of spacetime that is non-infinitesimal in either or both of space and time a *timespan-region*.

12.3 Einstein's field equations

In principle the core field equations are very simple[1] [2]:

$$\underset{\textit{spacetime geometry}}{G_{\mu\nu}} \;=\; \underset{\textit{stress-energy tensor}}{T_{\mu\nu}} \tag{12.3.1}$$

The devil is in the detail. The left-hand side, $G_{\mu\nu}$, is gravity, a function of the geometry of spacetime. The right-hand side, $T_{\mu\nu}$, is the source of gravity: a second-order tensor variously described as the stress-energy tensor, the energy-momentum tensor, or the matter-energy tensor.

More precisely, $G_{\mu\nu}$ is a function of the spacetime metric $g_{\mu\nu}$ and its first and second derivatives:

$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu}$ is spacetime curvature, with

$R_{\mu\nu}$ the contracted Riemann ('Ricci') curvature tensor

$$R_{\mu\nu} \equiv \sum_{\gamma} R^{\gamma}_{\mu\nu\gamma} \tag{12.3.2}$$

where

[1] A normalisation factor $\kappa = 8\pi G/c^4$ which ensures consistency with Newtonian gravity in the weak field limit is commonly included, as we showed in equation 9.1.1, but has been omitted here for simplicity. Due to general coordinate invariance a term $k\ g_{\mu\nu}$ may be added to the left-hand side, where k is a constant (and is, in equation 9.1.2, Λ). With reference to quantum field theory, the right-hand side may also be scaled by a factor $k'\ g_{\mu\nu}$ to accommodate the energy density of the vacuum, which is Lorentz invariant. None of these terms has any bearing on the points we are about to make.

[2] I have omitted the cosmological constant term $\Lambda g_{\mu\nu}$ which is the common depiction of the $k\ g_{\mu\nu}$ term referred to in the footnote above. The value of Λ has been shown to have negligible impact on star formation rate [201] and therefore is relevant only to the macroscopic evolution of the universe, which has been discussed in Section 9.3.4.

$$R^{\delta}_{\mu\nu\ \gamma} \equiv \frac{\partial \Gamma^{\delta}_{\mu\gamma}}{\partial x^{\nu}} - \frac{\partial \Gamma^{\delta}_{\mu\nu}}{\partial x^{\gamma}} + \sum_{\lambda} \Gamma^{\lambda}_{\mu\gamma}\Gamma^{\delta}_{\lambda\nu} - \sum_{\lambda} \Gamma^{\lambda}_{\mu\nu}\Gamma^{\delta}_{\lambda\gamma} \tag{12.3.3}$$

with connection coefficients

$$\Gamma^{\lambda}_{\mu\nu} = \frac{1}{2}\sum_{\sigma} g^{\lambda\sigma}\Big(\frac{\partial g_{\sigma\nu}}{\partial x^{\mu}} + \frac{\partial g_{\mu\sigma}}{\partial x^{\nu}} - \frac{\partial g_{\mu\nu}}{\partial x^{\sigma}}\Big) \tag{12.3.4}$$

R is the Ricci scalar $R \equiv \sum_{\mu\nu} g^{\mu\nu} R_{\mu\nu}$

$T_{\mu\nu}$ is the covariant form of $T^{\mu\nu}$, the rank-two, symmetric stress-energy tensor (or *energy-momentum tensor* which is a function of the energy density and energy flux of matter, including radiation, across the four dimensions of spacetime. Usually stated in its contravariant form:

$$T^{\mu\nu} = \begin{bmatrix} T^{00} & T^{01} & T^{02} & T^{03} \\ T^{10} & T^{11} & T^{12} & T^{13} \\ T^{20} & T^{21} & T^{22} & T^{23} \\ T^{30} & T^{31} & T^{32} & T^{33} \end{bmatrix} \tag{12.3.5}$$

The precise form of the tensor depends on what the spacetime contains. The following stress-energy tensors are familiar in cosmology:

- Vacuum:
$$T^{\mu\nu} = 0,\ \forall \mu, \nu$$
- Dust, a cloud of non-interacting particles with mass-energy density ρ all at rest in an isotropic rest frame:
$$T^{\mu\mu} = \rho;\ \mu = 0$$
$$T^{\mu\mu} = 0;\ \mu \neq 0$$
$$T^{\mu\nu} = 0;\ \mu \neq \nu$$
- Perfect fluid in equilibrium, with density ρ and pressure p, in an isotropic rest frame. In this frame the particles may be moving but their momenta cancel out.
$$T^{\mu\mu} = \rho;\ \mu = 0$$
$$T^{\mu\mu} = p;\ \mu \neq 0$$
$$T^{\mu\nu} = 0;\ \mu \neq \nu$$

Of course, pressure may not be isotropic and the stress exerted by a fluid need not be perpendicular to the surface on which it acts. The diagonal elements of the matrix are then anisotropic pressure, and the off-diagonal elements are the shear stress, which is what gives rise to name 'stress-energy' tensor.

The stress-energy tensor is, then, the total energy of the components of the spacetime. In practice it is not possible to define a realistic $T^{\mu\nu}$ for the universe or for any timespan-region that contains dynamical, interacting components, for reasons which we explore in the next section. Furthermore, even if it were possible to establish what $T^{\mu\nu}$ is at a particular instant that result would no longer be correct for the next, infinitesimal instant.

12.4 Implications of the theory

12.4.1 GR is fundamentally different

The GR field equations 12.3.1 describe a theory that is fundamentally different from other field theories. In particular:

- The theory of GR is built in Riemannian rather than Euclidean geometry. This distinction is familiar to mathematicians but not to many cosmologists, who work with FLRW-GR. In Euclidean space the shortest distance between two points is always a straight line. In Riemannian geometry, the extremal (shortest) line between two points is a geodesic. Any particle in a gravitational field follows a geodesic path, not a straight line. Einstein, Infeld and Hoffmann showed in 1938 [202] that the trajectories of massive particles, even so far apart that their gravitational interactions are weak and between which the vacuum field equations hold, cannot be arbitrary. Each particle must move along a geodesic in the spacetime determined by the others (see, for example, [203] pg 64). Therefore, wherever there is matter present, its particles follow geodesics and, '*The motions of matter can already be obtained from the vacuum field equations in the region outside the matter, without needing any details of the stress-energy tensor. This makes gravity very different from other interactions.*' [203] pg 65.

- The field is not only curvilinear, it is explicitly free of coordinate dependence. There is no background. In flat, Euclidean space we can feel confident that we understand the physical inerpretation of a chosen coordinate system.

In curved space that is no longer true: results may be difficult to interpret in terms of physical observables.

- Space and time are interchangeable and must be considered to be spacetime. Whereas spatial geodesics have line elements that are always positive definite that is not the case for spacetime geodesics.
- Spacetime curvature, aka the Einstein tensor $G_{\mu\nu}$, aka gravity, responds to all forms of energy. This includes not only inertial mass and electromagnetic fields but also the energy density of the gravitational field itself. Solving the equations of motion of a charged particle in an electromagnetic field requires two indepdenent steps, because the particle may be subject to forces other than the electromagnetic force: solve Maxwell's equations for the field and then use the Lorentz force law to find the acceleration in the field. Gravity, by contrast, sees everything: this is because any other forces that are present carry energy and therefore also couple with the gravitational field.

These differences have immense significance. In the rest of this chapter we state the principal implications.

12.4.2 No integral conservation of energy

In GR, energy is not conserved in any timespan-region. In the terminology of Chapter 10, any region in spacetime is and must be dissipative.

Conservation of energy is a fundamental principle of physics, arising from the invariance of physical laws under time translation. In the FLRW-GR framework time translation invariance is preserved, one of the assumptions necessary to make a solution to the field equations tractable. In the full theory of GR time is not independent and we must work with spacetime. Time translation invariance is broken.

The energy conservation condition in GR is simply that the covariant derivative of the stress-energy tensor, its divergence, vanishes:

$$\bigtriangledown_\mu T^{\mu\nu} = 0 \tag{12.4.1}$$

The divergence is a vector field that describes the source or sink of energy and momentum density. At any timepoint in spacetime it is zero. This does not help

us at all in finding some statement of conservation of energy in a timespan-region. In other field theories, for example electromagnetism, energy conservation can also be written in an integral form. Surely we can do the same for GR, and write a (spacetime) volume integral that represents the conserved gravitational 'charge' within that timespan-region?

The answer in a spcific case is yes, Gaussian flux integrals can indeed be generalised to GR. But, in general, the answer is no.

The explanation for this is as follows. In electromagnetic theory it is possible to establish the total conserved charge Q of a source by using a Gaussian flux integral over a closed 2-surface surrounding it (see, for example, Misner et al [86]pg 463):

$$Q = \frac{1}{4\pi} \oint E^j d^2 S_j = \frac{1}{4\pi} \oint F^{0j} d^2 S_j \tag{12.4.2}$$

Here E is the electric field, F is the electromagnetic field tensor and dS is an element of any limited surface. Similarly, in Newtonian gravity it is possible to establish the total mass M of a source via the Gaussian flux integral as a function of the Newtonian gravitational potential Φ

$$M = \frac{1}{4\pi} \oint \Phi_{,j} d^2 S_j \tag{12.4.3}$$

One can derive Gaussian flux integrals for the four-momentum and angular momentum of a gravitational source also in GR. However, the integrals can only be evaluated if (a) the closed surface of the integral is an asymptotically flat region surrounding the source; and (b) using asymptotically Minkowskian[3] coordinates. As Misner et al state very clearly:

[3]The Minkowski metric $\eta_{\mu\nu}$ with line element $ds^2 = -cdt^2 + dx^2 + dy^2 + dz^2$ is the metric of flat spacetime.

> 'In particular, *when evaluating the 4-momentum and angular momentum of a localized system, one must apply the flux integrals only in asymptotically Minkowskian coordinates. If such coordinates do not exist (spacetime not flat at infinity), one must completely abandon the flux integrals, and the quantities that rely on them for definition: the total mass, momentum, and angular momentum of the gravitating source...* Attempts to use [the flux integral] formulas in ways that lose sight of the Minkowski boundary conditions (and especially simply adopting them unmodified in curvilinear coordinates) easily and unavoidably produce nonsense.' (Misner et al [86] pg. 463)

The italics are the authors' own.

In other words, unless we require that spacetime is both asymptotically flat and that flat spacetime coordinates can be used to describe it, there is no integral conservation condition. The FLRW-GR of the standard cosmological model has conservation of energy in regions of spacetime. The full theory of GR does not.

The significance of this can hardly be overstated. In flat spacetime energy conservation is straightforward: in Newton's theory, mass is an intrinsic quantity that is itself conserved; in special relativity the mass m of a particle is not conserved but its energy E and its momentum p are both subject to conservation laws and are related to its mass by the equation $E^2 = p^2c^2 + m^2c^4$. In the dynamically curved spacetime of full GR, however, the stress-energy tensor $T^{\mu\nu}$ of equation 12.3.1 is not a conserved object. The conservation condition is instead that its covariant derivative vanishes, equation 12.4.1. Whilst a $T^{\mu\nu}$ can certainly be differentiated to obtain $\bigtriangledown_\mu T^{\mu\nu}$, the result cannot be integrated to give conservation laws over a timespan-region [204].

Every existent timespan-region is an open and dissipative system, not a closed and conservative one.

12.4.3 Spacetime is nowhere flat

Matter is represented by curvature, but not every curvature does represent matter; there may be curvature 'in vacuo'.

G Lemaitre in Schlipp (1949), p.440

The standard argument in cosmology is that the universe is almost perfectly flat, so conservation of energy applies. Yet no-one argues that spacetime is flat in the region of a black hole, for example. In the universe there are regions of relatively dense mass-energy and there are voids, on multiple scales. However flat the universe may be 'on average', or 'on sufficiently large scales', its four-dimensional topography cannot be said to be flat. If there are any sources at all in the universe, spacetime cannot be flat.

Specifically, in the absence of sources, i.e. $T_{\mu\nu} = 0$, the Ricci tensor in equation 12.3.1 must vanish. However, this does not necessarily mean that the Riemann tensor is zero. If it is not, then the metric tensor $g_{\mu\nu}$ must differ from the flat spacetime Minkowski metric $\eta_{\mu\nu}$. An example of such a non-Minkowskian metric in a vacuum is the Schwarzschild metric for the vacuum spacetime surrounding an isolated matter source. Lemaitre's quote above may therefore be rephrased as: matter has mass-energy but not all mass-energy is matter.

12.4.4 There is potential in the gravitational field

GR is a field theory so, although a body at any timepoint experiences no gravitational field in its own reference frame, in any timespan-region there arises a relative gravitational field potential.

The stress-energy tensor $T^{\mu\nu}$ in equation 12.3.1 includes only the energy of matter (which, in this case, would include the dark matter and dark energy of the standard cosmological model if that were the framework we were using) and does not include any purely gravitational potential [4]. Indeed, no tensor can be defined for gravitational energy density because, in compliance with the equivalence principle, it vanishes at each timepoint and so must vary with the co-ordinate frame[5]. In applying numerical relativity to binary mergers the field equations are supplemented by a pseudo-tensor to account for the gravitational field potential in that specific calculation [153].

[4] For a good explanation of the fact that there is no energy-momentum tensor for the field potential see Lambourne [87].

[5] There is an active field of research into practical and theoretical approaches to defining quasi-local energy momentum in GR, see for example Szabados [205] and references therein. That work does not invalidate the point that, in GR, non-infinitesimal physical spaces and times experience relative gravitational field potential.

Using the notation - althought not the linearised argument - in Misner et al [86] we will define a stress-energy pseudotensor for the gravitational field, $t^{\mu\nu}$, such that:

$$t^{\mu\nu} \equiv T^{\mu\nu}_{\text{eff}} - T^{\mu\nu} \tag{12.4.4}$$

Here $T^{\mu\nu}$ is the normal stress energy tensor but $T^{\mu\nu}_{\text{eff}}$, like $t^{\mu\nu}$, has no geometric, coordinate-free significance. Equation 12.4.4 is not a covariant tensor equation. Instead, $T^{\mu\nu}_{\text{eff}}$ is a coordinate-dependent object representing the total effective source of gravity influencing the evolution the spacetime metric for a specific timespan-region.

Gravitoelectromagnetism (GEM) and frame-dragging

The gravitational energy in GR is generated by mass currents in the gravitational field. This field theory nature of GR has invited analogy with electromagnetism, to the extent that the effect is usually called *gravitoelectromagnetism* (GEM) because of the formal analogy between the two field theories although it has, of course, nothing to do with electromagnetism. In his Princeton lectures of 1921 [206], Einstein pointed out that there are three sources of such currents: rotation of massive bodies, their linear motion, and their inertial mass. The currents are now usually described as *frame-dragging*.

Already in 1918 the question of the field effect of rotating bodies had been explored for a hollow sphere by Thirring and for a solid sphere by Lense and Thirring, both in the weak field approximation. (Translations of both papers are given in full in [207].) They concluded that, although small, the effect on the moons of Jupiter might be measurable. The 'Lense-Thirring effect' of frame dragging in rotational systems has indeed now been measured, for example in a binary pulsar system [208].

Calculations of the frame-dragging effect of single rotating bodies predict they are very small. By contrast, some studies based on modelling exact solutions of GR field equations in simplified galaxy systems have found that the GEM impact on such systems is large. Using a stationary, axially symmetric spacetime metric and the approximation of a co-rotating, pressureless, perfect fluid source for a rotating disk galaxy, it has been suggested that the GEM mass-energy contribution

is sufficient to obviate the need for dark matter (for an infinitely thin disk [129], and for any rotating galaxy [130]). These studies are not without their critics, so we do not present them as conclusive evidence of the scale of frame-dragging effects. They do demonstrate that such effects are mathematically consistent with the theory of GR [131].

12.4.5 Gravity does not scale linearly with mass

The field equations 12.3.1 are manifestly nonlinear in the metric $g_{\mu\nu}$ and its first derivative.

More fundamentally, there is a feedback loop between the spacetime metric terms on the left-hand side of the field equations and the stress-energy tensor on the right. As Wheeler said, 'Spacetime tells matter how to move; matter tells spacetime how to curve' [195]. This loop, which can clearly drive escalating interaction, is completely missing in the FLRW formulation.

Simply put, gravity does not scale linearly with mass.

12.4.6 Equilibrium cannot be established

Stability may arise at some scales in some timespan-regions of the system, but equilibrium cannot arise since the system cannot be stationary. Gravity responds to all forms of energy, therefore all gravitational systems are open and dissipative and it is not possible to attain a dynamical balance of forces. This is exactly the same argument as that used in Section 10.2.2 in the chapter on complex systems.

12.4.7 Relativity in space and time

Space and time are not absolute. From the perspective of any observer in space, it is straightforward to state that there is no meaning in the question, 'Am I moving away from that region or is it moving away from me?' Questions of relative time are more subtle. The ticking of a clock held in the hand of an observer who considers themselves to be stationary will not be synchronysed with the ticking of a clock held in another's hand unless they share a common Lorentz frame.

In full GR, time is nonlinear and the implications of removing the timeslice

approximation are profound. They include, as we have seen in Section 12.4.2, the extinction of a conservation of energy condition in any timespan-region. The 'timescape' implication has been explored extensively by Wiltshire, see [141] for an introduction to the topic.

12.4.8 Gravity is not a force

As a direct consequence of the equivalence principle, a body falling freely in its own reference frame experiences no force at all. It is therefore inappropriate to describe gravity as a force.

12.4.9 The gravitational field can repel as well as attract

Not only is gravity not a force in GR, the gravitational field is not always attractive. Although all known forms of matter satisfy the *dominant energy condition* [204]

$$T_{00} \geq |T_{\mu\nu}|, \quad \forall\ \mu, \nu$$

such that the mass-energy of that matter must be positive, there is no bound for the purely gravitational potential $t_{\mu\nu}$ described in Section 12.4.4.

To illustrate, see Figure 12.4.1. Consider a spherical timespan-region A, which contains a galaxy that is accumulating mass-energy as it forms. Let A be surrounded by a spherical 'near-zone' sufficiently thin that light (and gravitational waves) pass through it in a short timespan. A further zone, B, is an extended, vacuum spacetime such that light and gravitational waves take much longer to traverse the zone. Finally, let there be a zone C which contains the remainder of the universe. The energy conservation condition is that the divergence of the stress-energy tensor at each timepoint is zero, i.e. there is no background gravitational field at any timepoint. Hence, the accumulation of gravitational field potential in timespan-region A cannot increase total energy in the universe and must be offset somewhere. Gravitational waves can travel no faster than the speed of light. Therefore, in the scenario here, increasing field potential energy in A will impact the near-zone of A almost immediately but will take longer to flow into zones B and C. The gravitational field potential in the near-zone of A must have the opposite sign to that in A until that wave has time to disperse into B and C. In the near-zone of A gravity must be repulsive rather than attractive.

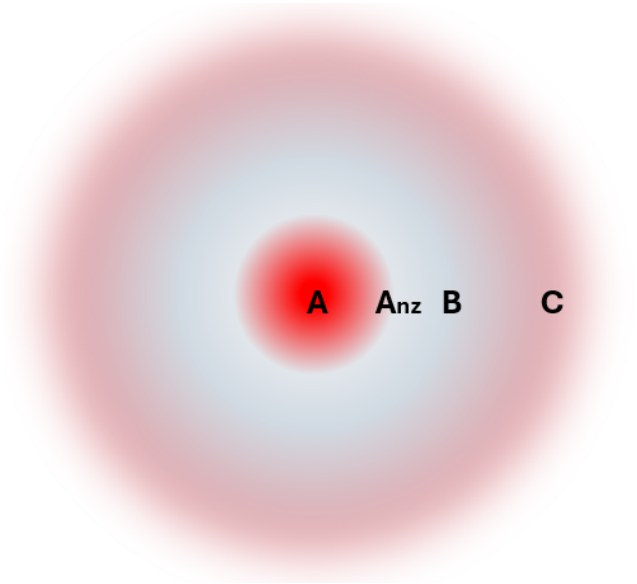


Figure 12.4.1: Schematic diagram of gravitational attraction and repulsion. Zone A is a deepening attractive potential well; The near zone of A has potential with an opposite sign, i.e. repellent; in zone B the gravitational wave is spreading outwards to equalise with the rest of the universe, zone C.

In Section 14.3.3 we comment further on the implication that under-dense regions are to be expected in the vicinity of cosmic structures, and refer to the repulsion effect of voids seen in studies of the CMB dipole repeller [209].

12.4.10 No general solution is possible

The first exact solution to Einsein's field equations 12.3.1 was found by Karl Schwarzshild in 1915, just a few weeks after their publication. The Schwarzshild solution employs a stationary, static, spherically symmetric metric to describe the vacuum spacetime outside a spherical mass. In modern times a number of exact solutions to the field equations are known (see [210] for a useful catalogue). However, physically-relevant solutions are specific to narrowly defined cases, such as the Kerr solution for a rotating black hole [211].

Why is it so difficult to find solutions that can be applied to widespread cosmological phenomena? There are problems of tractability and there are also fundamental barriers to solution which arise because of the nature of the theory:

- The feedback loop, $T^{\mu\nu} \rightarrow G^{\mu\nu} \rightarrow T^{\mu\nu} \rightarrow ...$, introduces perpetual change in each term of the system of equations.
- The stress-energy tensor cannot be defined for a realistic system, as discussed above.
- Even if a stress-energy tensor were to be constructed for some representation of the structure of matter, it would not lead to a general solution. Einstein, Infeld and Hoffmann made this clear [202]: 'Such energy-momentum tensors ... must be regarded as purely temporary and more or less phenomenological devices for representing the structure of matter, and their entry into the equations makes it impossible to determine how far the results obtained are independent of the particular assumption made concerning the constitution of matter ... Actually, the only equations of gravitation which follow without ambiguity from the fundamental assumptions of the general theory of relativity are the equations for empty space.'
- The field equations cannot uniquely determine the spacetime metric $g^{\mu\nu}$. The requirement that coordinate transformations must be possible means that any metric related to another solution must also be a valid solution. Hence, only six of the ten field equations are truly independent (see, for example, [212] pg.251).
- When it comes to solving the equations for a timespan-region we encounter the insuperable problem that the field equations of GR cannot be integrated, as explained above. In textbook terms, the equations are 'non-localisable' [86].

In the absence of any general solution, there are in principle three ways to approach the problem:

1. Choose $g_{\mu\nu}$ and solve for a $T_{\mu\nu}$.
2. Choose $T_{\mu\nu}$ and solve for $g_{\mu\nu}$.

3. Seek to identify both $g_{\mu\nu}$ and $T_{\mu\nu}$ from the underlying physical characteristics of the system, and explore the relationship.

In the search for physically relevant solutions the third method is usually employed. The FLRW framework of Section 9.1 is an example.

Chapter 13

Complexity in a model galaxy simulation

13.1 Introduction and purpose

In Chapter 14 I will address the application of SOC and GR to the universe. As a precursor to that, in this chapter I consider a model, five-body galaxy with two spatial dimensions. It is insufficient to demonstrate a key feature of complexity, self-organisation, as there are too few components. However, the interaction of the bodies with each other is expected to give rise to complex behaviour.

The aim of this work is to explore aspects of the following assertions:

1. Non-interacting dust or perfect fluid approximations are inadequate when modelling systems in which many-body interaction is present, since that interaction results in complex behaviour.

2. Interactions calculated in Newtonian gravity result in different behaviour from GR, even if the system is sub-relativistic and the spacetime is approximately flat.

3. In GR, the mass-energy of an interacting system exceeds the mass-energy of the rest-state of its constituent bodies.

To frame complexity specifically in this model, first consider a system of five bodies in which there is no interaction. This is a simple system and we can easily

calculate motion arising from any external force independently for each body. Let us now allow the five bodies to interact with each other through an attractive force which is proportional to mass, i.e. a linear interaction, and specify that one of the five bodies has a mass which far exceeds that of the other four. This is no longer a simple system but it can be modelled quite accurately as if it is, by treating the body with the dominating mass as the centre of the system and ignoring interactions between the small masses. Some adjustments will become necessary over time, but the model will produce results that are good for many purposes. This kind of approximation is routinely used in modelling our own solar system, for example.

If we instead specify that none of the five bodies has a dominant mass, and decline to ignore interactions between the bodies, the system exhibits complex behaviour. The equations of motion of the system have no analytical solutions that can be used to predict its evolution. This is so even if the interaction is completely linear. We must resort to numerical simulations of evolution from a specified initial configuration[1].

In the complex case, i.e. when interaction between the bodies is taken into account, I would like to address the above assertions by comparing the evolution of the model galaxy in the Newtonian gravity case with its evolution in the GR framework. However, whilst simulating the model galaxy dynamics with Newtonian gravity is straightforward, the same is not true for GR. I cannot carry out a simulation of a five-body system in GR, for the reasons outlined in Section 12.4.10. I therefore take the following approach:

- I first state the conventional approach to modelling a galaxy, which is to treat it as a simple, non-interacting system. The dynamics that arise in the model if it is approximated by a disk of noninteracting, pressureless dust contracting under its own gravity are summarised in Section 13.3.
- I then set out in Section 13.2 two different approaches to gravity: (a) linear Newtonian gravity; and (b) a linear approximation in which two features of GR are introduced. The first feature is that mass-energy is the source of gravity and the second is that energy is not subject to a conservation

[1] Numerical methods have identified that there are *simple choreographies* for certain N-body problems in the Newtonian potential case. These require very particular initial conditions [213]

condition within the model system. I use the term 'GR approximation' for this framework, although it represents only a small step towards GR.

- I present quantitative simulations in Section 13.4 in which the five bodies interact with each other as a complex system, in each of the two linear approaches to gravity.
- The extension of the comparison to a full, nonlinear GR regime is addressed qualitatively in Section 13.5.
- I discuss the findings in Section 13.6.

13.2 Gravity in the model

13.2.1 Newtonian gravity

In Newtonian theory the source of gravity is intrinsic mass and, in flat spacetime, total energy in the region of the simulation must be constant

$$E_{\text{total}} = -E_{\text{potential}} + E_{\text{kinetic}} = \text{constant}$$

Any increase in kinetic energy is offset by a corresponding change in potential energy and vice versa. For i bodies of mass m_i the total energy in the system is thus

$$E_{\text{sys tot}} = -G\sum_{i\neq j}\frac{m_i m_j}{r_{ij}} + \frac{1}{2}\sum_i m_i|\mathrm{v}_i|^2 = \text{constant} \tag{13.2.1}$$

where G is Newton's gravitational constant, r_{ij} is the distance between bodies i and j and $\mathbf{v}_i$ is each body's velocity.

The source of gravity in this theory is invariant mass. I therefore calculate equations of motion arising from the Newtonian force on each particle

$$\mathbf{F}_{\mathrm{N}i} = G\sum_{i\neq j}\frac{m_i m_j}{|r_{ij}^2|}\hat{\mathbf{r}}_{ij} \tag{13.2.2}$$

and a constant system mass that is simply

$$M_{\text{sysN}} = \sum_i m_i \tag{13.2.3}$$

13.2.2 Linear approximation with aspects of GR

The 'GR approximation' that I use in the simulation is based on only two GR-related features: I use mass-energy as the source of gravity and I do not impose an energy conservation condition within the system. Specifically, in the numerical simulation I exclude GR's nonlinearity and inherent feedback, I employ discrete steps of invariant time and space[2], and I ignore the impact of gravitational field potential within the timespan-region of the system (Section 12.4.4). These considerations are reintroduced in the qualitative evaluation in Section 13.5.

I am not using the stress-energy tensor $T^{\mu\nu}$ of the field equations in my calculations but we do want to understand how it relates to the mass-energy source I will define. For this I look to the stress-energy tensor for a point particle (see, for example, [203] pg. 60):

$$T^{\mu\nu} = m \int \frac{dz^\mu}{ds} \frac{dz^\nu}{ds} \delta^{(4)}(x - z(s)) ds \tag{13.2.4}$$

In flat spacetime the integral can be carried out to obtain, in normal Cartesian co-ordinates, the standard four-momentum density for a point particle [203]

$$T_0^\mu = m \frac{dz^\mu}{ds} \delta^{(3)}(x^i - z^i(s)) \tag{13.2.5}$$

This is usually written as (E, p^1, p^2, p^3), so T^{00} is the energy density E.

The mass-energy source of gravity for the model then is an estimate of T^{00}, which I will call T. On experimental grounds, it is known that the relativistic version of gravitational mass includes all forms of energy [214]. To derive a workable approximation for total mass-energy I borrow from the post-Newtonian[3] formalism for gravitational interaction, which is described for N-bodies in Chapter 9 of Poisson and Will [212]:

1. Poisson and Will first note that the behaviour of a perfect fluid is governed by the continuity equation $\partial_t \rho^* + \partial_j(\rho^* v^j) = 0$, where $\rho^* \equiv \sqrt{(-g)}\gamma\rho$ with $\gamma \equiv u^0/c$ is the conserved mass density and v^j is the fluid's velocity

[2] I am well aware that in flat spacetime energy is conserved. Here the point is that the aim of this simulation is to move towards GR, which is not a flat spacetime regime.

[3] Post-Newtonian theory is defined as 'the theory of weak-field gravity within the near zone, and of the slowly moving systems that generate it and respond to it.' pg 371 [212].

field defined with respect to the time coordinate t. As usual, g is the determinant of the matrix of components of the metric $g_{\mu\nu}$. Expressions for the fluid's conserved mass-energy M, its total momentum P^j and its centre-of-mass position R^j are then derived. The results of this analysis are applied to a collection of separated bodies (retaining the conservation of energy condition).

2. For a system of N bodies labelled with the index $A = 1, 2, ..., N$ the fluid density is expressed as

$$\rho^* = \sum_A \rho^*_A \tag{13.2.6}$$

with ρ^*_A zero everywhere except within the volume occupied by A, and the total mass energy of body A is

$$M_A \equiv m_A + \frac{1}{c^2}\Big(\mathcal{T}_A + \Omega_A + E^{\rm int}_A\Big) + \mathcal{O}(c^{-4}) \tag{13.2.7}$$

where, in A's co-moving frame:

$$\begin{aligned}
m_A &\equiv \int_A \rho^* d^3x \text{ is the material mass of A} \\
\mathcal{T}_A &\equiv \frac{1}{2}\int_A \rho^* \bar{v}^2 d^3\bar{x} \text{ is the kinetic energy of A} \\
\Omega_A &\equiv -\frac{1}{2}G\int_A \frac{\rho^*\rho^{*'}}{|\bar{x}-\bar{x}'|} d^3\bar{x}' d^3\bar{x} \text{ is gravitational potential energy of A} \\
E^{int}_A &\equiv \int_A \rho^* \Pi d^3\bar{x} \text{ is A's internal energy (Π is internal energy per unit mass)}
\end{aligned} \tag{13.2.8}$$

In GR we must have $\Omega_A = 0$ in A's co-moving frame and in my model of point particles with mass m we also have $E^{int}_A = 0$. Thus the total mass energy of body A reduces to material mass plus kinetic energy. For my purposes, then, I say

$$T_i \approx m_i + \frac{1}{c^2}\frac{1}{2}m_i v_i^2 \tag{13.2.9}$$

For a system of i bodies total mass-energy is then given by

$$E_{\text{sys tot}} \equiv \sum_i T_i = \sum_i (m_i + \tfrac{1}{c^2}\tfrac{1}{2} m_i v_i^2) \tag{13.2.10}$$

I now denote $\mathbf{F}_{\text{GR}i}$ to be the source of gravity felt by particle i in the model. I use equation 13.2.9 as the expression of mass and write the equivalent of Newtonian force equation 13.2.2 as:

$$\mathbf{F}_{\text{GR}i} = G \sum_{j \neq i} \left(\frac{(m_i + \tfrac{1}{c^2}\tfrac{1}{2} m_i v_i^2)(m_j + \tfrac{1}{c^2}\tfrac{1}{2} m_j v_j^2)}{|r_{ij}^2|} \hat{\mathbf{r}}_{ij} \right) \tag{13.2.11}$$

The mass-energy of the system is calculated as

$$M_{\text{sysGR}} = E_{\text{sys tot}} = \sum_i (m_i + \tfrac{1}{c^2}\tfrac{1}{2} m_i \mathbf{v}_i^2) \tag{13.2.12}$$

13.3 Simple system: no interaction between particles

In the case that interaction between randomly-distributed particles in a two-dimensional space is ignored, the usual approach to gravity modelling is to treat the system as a homogeneous, axisymmetric disk and to calculate the radial collapse of that disk.

In Newtonian gravity this problem can be solved analytically but in GR numerical methods are required [215]. In either case the disk contracts, ultimately to a black hole. For example, Abrahams et al (ibid) report the simulation of collapse of a cold homogeneous disk, the disk analogue of the spherical Oppenheimer-Snyder collapse, considering both black hole formation and gravitational wave radiation.

Simply put, a simple system contracts under gravity. In terms of the linear equations 13.2.2 and 13.2.11 that are employed below, the only difference between Newton's theory and my linear approximation of GR would be that in the latter the collapse would happen faster.

13.4 Complex system: particles interact with each other

13.4.1 A numerical simulation

In the simulation I work with i point particles, where $i = 5$. Each particle has a mass of 10^3kg. Their initial configuration is random, set in a 100m $\times$ 100m x,y plane. The total simulation time is $T = 10^6$s and time steps are $t = \Delta T = 10^{-2}$s. I use a gravitational constant $G' = 2 \times 10^{-9}\text{Nm}^2\text{kg}^{-1}$). Each particle has a randomly assigned initial velocity on the x,y plane in the range $-0.001c$ ms$^{-1} < v < 0.001c$ ms^{-1}. Since the initial velocities are random, there is no imposed net angular momentum.

The units of mass, space and time I have assigned are described as 'kilograms', 'metres' and 'seconds' respectively. Apart from the time unit 'seconds', however, these are not to be read as normal physical units, since the scale of the simulation has been adjusted to accommodate $c = 1$ and a practicable run time. The scaling impact of assigning $c = 1$ is as follows:

- Inertial mass 10^3 kg remains unaffected. For reference, a physically realistic mass of a star is substantially higher, $\mathcal{O}(10^{30})$ kg.
- Distance $x, y = (1 \times \text{c} \times \text{s})$m in the simulation has a physical equivalent $x, y \approx 10^8$ m. Thus a 100m $\times$ 100m grid in the simulation, i.e. physical equivalent $10^{10}\text{m} \times 10^{10}$m, is substantially smaller than a physically realistic spatial extent of a galaxy, which could be $\mathcal{O}(10^{15})$m to $\mathcal{O}(10^{25})$m
- Velocity $v = 0.001c$ in my simulation scale is $v = 0.001\text{ms}^{-1}$ and the un-scaled equivalent would be $v \approx 10^5\text{ms}^{-1}$. This is consistent with a realistic average velocity of a star in a galaxy, $\mathcal{O}(10^5)\text{ms}^{-1}$.
- The units of force are kg ms^{-2}. Newtonian gravitational force as defined in equation 13.2.2 has the gravitational constant G and mass in the numerator and the square of distance in the denominator. In the GR approximation, equation 13.2.11, the numerator also has a velocity-dependent term (with no units, as it is v/c). I have set a scale of G' $\mathcal{O}(10^{-9})$. The c-related scaling of the force arises from the squared distance: in my simulation, the demoninator is $\mathcal{O}(1)$, whereas the physical scale is $\mathcal{O}(10^{16})$. However, the

simulation mass scale of 10^3kg, which appears in the numerator, is several orders of magnitude smaller than a realistic mass scale.

- The outcome of scaling with respect to acceleration is that the simulation runs with higher values for acceleration than are strictly realistic in a physical galaxy. This 'speeding-up' of the simulation enables it to produce results within a manageable run-time but does not affect the nature of the outcome.

I use discrete time steps and model the simple differential equations for acceleration and velocity using Newtonian dynamics and numerical methods. Numerical solutions are, of course, approximations. My consideration of the reliability of the simulation is in Appendix K.1.

13.4.2 Expectation

The Newtonian and approximate GR formulations I have defined, equations 13.2.2 and 13.2.11 respectively, are very similar when velocities are small. I know that in a chaotic phase preceding order in a complex system small changes to inputs can produce very large differences in behaviour so, despite the almost negligible change in input, I expect to see disproportionate differences in the dynamical behaviour of the five bodies in the two different regimes.

The equations for mass-energy, Newtonian 13.2.3 and GR approximation 13.2.12, are defined such that Newtonian mass is constant, under conservation of energy, and mass in the GR approximation is always greater than rest mass. Clearly, I will see this result. I do not know in advance how the GR-approximation mass-energy will fluctuate over time.

13.4.3 Outcome

From the initial, random distribution of five particles in Figure 13.4.1 I calculate position, velocity, acceleration and mass-energy values for each of the particles at time steps of $\Delta T = t = 0.1s$ up to a total elapsed time $T = 10^6 s$, in both the Newtonian case and the GR approximation, using equations 13.2.2 and 13.2.11 respectively as the source of gravity.

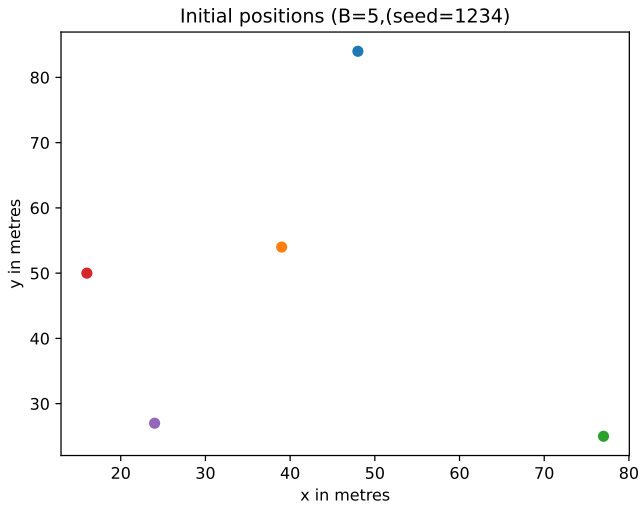


Figure 13.4.1:
Random initial configuration of five massive point particles, random seed = 1234

Dynamical behaviour

The simulation, which retains complex system interaction in both the Newtonian case and the GR approximation, does not demonstrate contraction: see Figure 13.4.2.

The range of the GR-approximation and Newtonian outcomes in Figure 13.4.2 is very different so, for comparison purposes, I present a zoom to a 200×200m scale in both cases in Figure 13.4.3.

In both cases the systems demonstrate complex behaviour. On closer inspection, as shown in Figure 13.4.4, the calculations are nearly indistinguishable in early times. This is consistent with the commonly held assumption that, in a sub-relativistic, weak-field system the Newtonian calculation is a good approximation for gravity. However, over a longer period there is a divergence between the calculations and, once that divergence has occurred, the difference between the two sets of calculations grows rapidly. The final positions of the five particles, in Figure 13.4.5, are very different.

What is happening where the calculations diverge? With this particular initial configuration Figure 13.4.4 shows that the divergence occurs at time $T \sim 3.8 \times 10^5$s. I examine the dynamics of the GR approximation more closely in Figure 13.4.6 by adding time markers to the position evolution of the GR approximation.

Figure 13.4.6a allows us to identify two points at which particles experience reversals of direction in the GR approximation that do not occur in the Newtonian calculation. One of these, involving the particles labelled with purple and green, is clearly a gravitational 'sling-shot' effect that occurs at time $T \sim 3.77 \times 10^5$s, see Figure 13.4.6b. At time $T \sim 3. \times 10^5$s the particle labelled red is decelerating sharply and then changes trajectory towards the high-velocity green-labelled particle. These effects are not abnormal, but they are different from the Newtonian calculation. The shifts in velocity are clear in the velocity magnitude plot in Figure 13.4.7. In that plot, the sharp reversals of velocity as the green- and purple-labelled particles interact with each other, particularly bewteen $T \sim 2.55 \times 10^5$s and $T \sim 3.77 \times 10^5$s, appear as steep oscillations.

In Appendix K.2.1 I present simulation results for four other sets of random initial conditions. In all cases, divergences occur abruptly and then grow over time. The simulation is 10^6s: I am not dealing with values that are large relative to the universe. Over cosmic time the two sets of calculations would bear no relationship to each other or, it must be said, to any simple-system approximation.

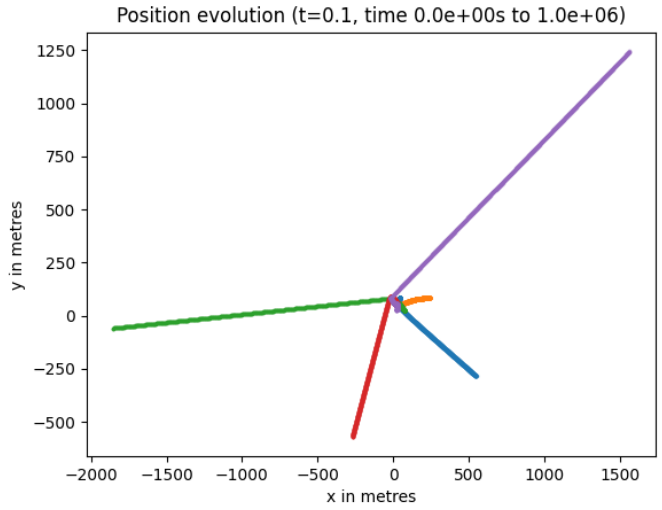
Of course, in a complex system, slightly different initial conditions would also result in very different outcomes, But, again, Newtonian calculations and the GR approximation would diverge for that alternate set of initial conditions.

The difference between the GR approximation and the Newtonian calculation, both in a complex system, arises from the fact that the source of gravity is mass-energy and the conservation of energy condition is not and should not be present in the GR approximation. Which leads us to consider next what this means for the mass-energy of the individual particles and of the system as a whole.

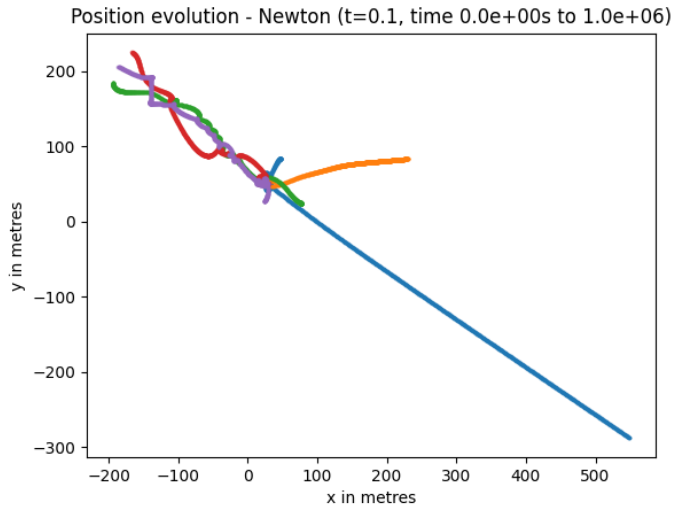
Mass-energy

In the GR approximation, mass-energy is sensitive to velocity. We have seen the changes in velocity magnitude in Figure 13.4.7. The impact on particle mass is shown in Figure 13.4.8 and on system mass in Figure 13.4.9. In the Newtonian case, conservation of energy ensures total mass-energy does not change. The solution to GR approximation equation 13.2.11, on the other hand, manifests a variable mass-energy profile, until effective particle interactions stop.

In Figure 13.4.9, and in the case of other random initial conditions in Appendix K.2.2, the mass-energy of the system always exceeds the intrinsic mass of the constituent particles. Indeed this must be so with the approximation defined in equation 13.2.11.



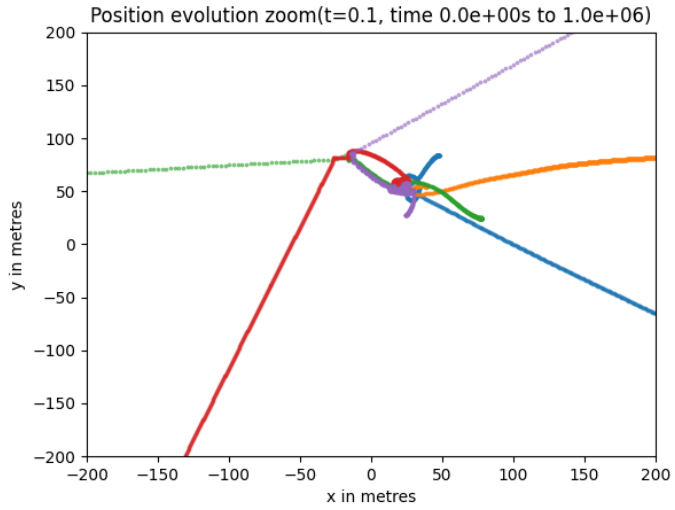
(a) GR approximation



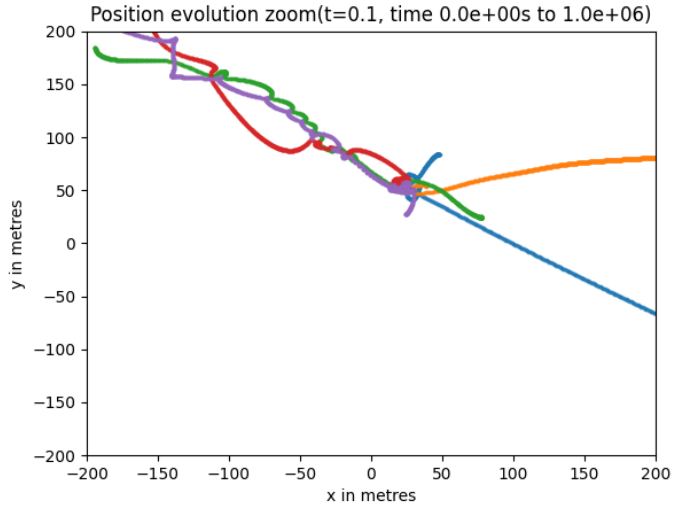
(b) Newtonian

Figure 13.4.2: Position evolution, full scale. Elapsed time $T = 10^6 s$, interval $t = 0.1 s$.)

.



(a) GR approximation



(b) Newtonian

Figure 13.4.3: Position evolution, zoom to 200m x 200m. Elapsed time $T = 10^6 s$, interval $t = 0.1s$.)

.

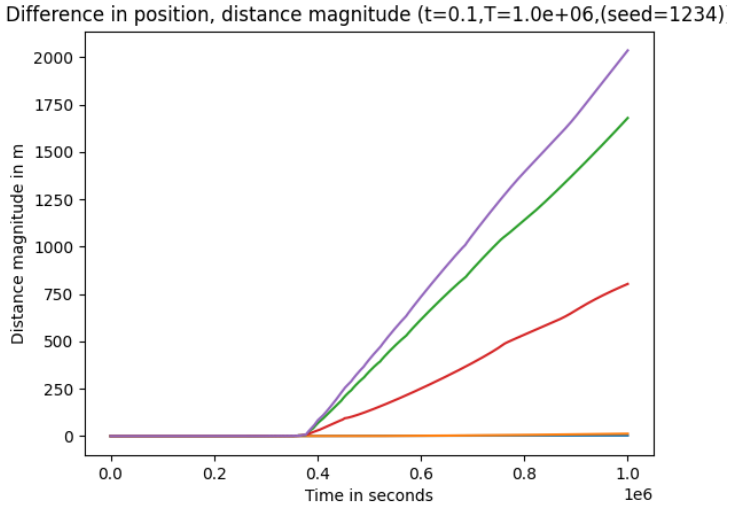
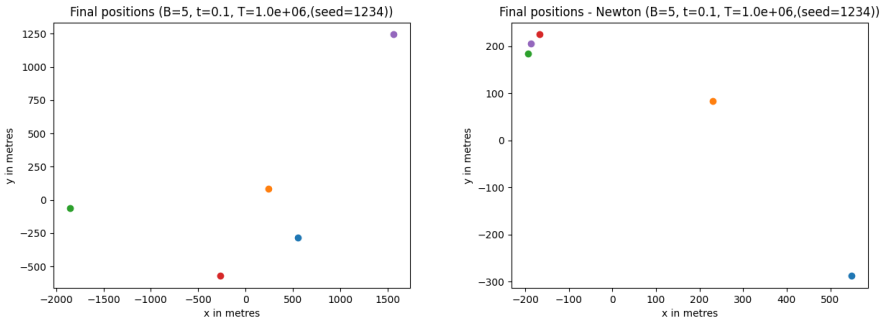


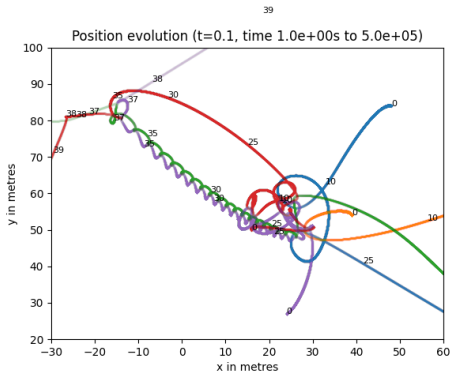
Figure 13.4.4: Difference between the GR approximation and the Newtonian positions over time. Elapsed time $T = 10^6 s$, interval $t = 0.1 s$.)

.



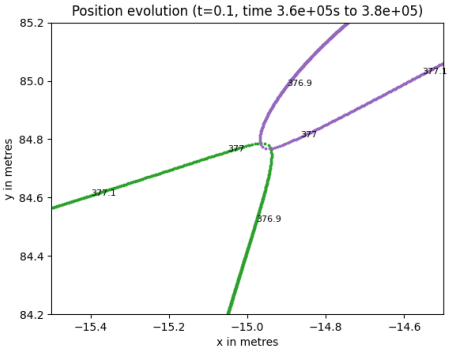
(a) GR approximation (b) Newtonian

Figure 13.4.5: Final position after time $T = 10^6$s.



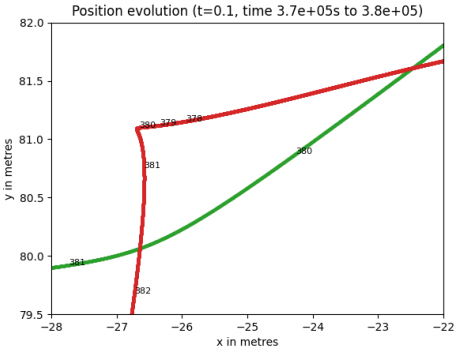
(a) Zoom to centre, timestamp units of $10^4 s$.

.



(b) Zoom to purple/green transition, timestamp units of $10^3 s$.

.



(c) Zoom to red transition, timestamp units of $10^3 s$.

.

Figure 13.4.6: Position evolution in the GR approximation, zoom to transition points and add time stamp.

.

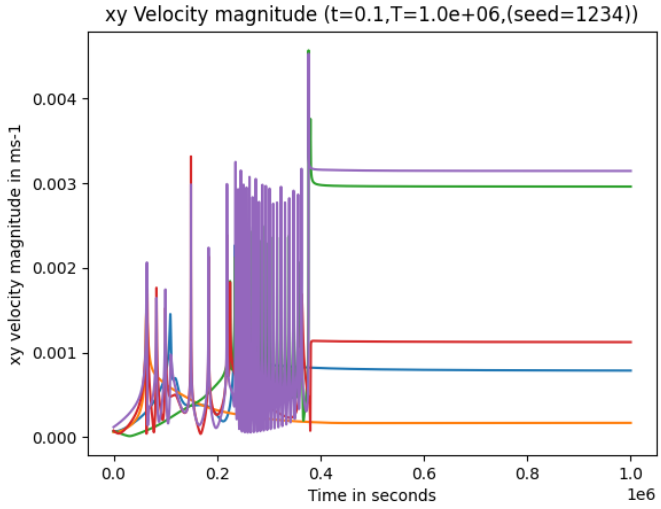
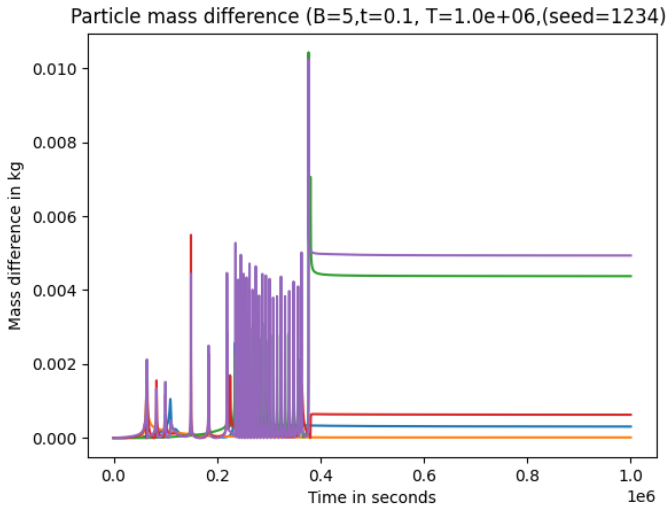
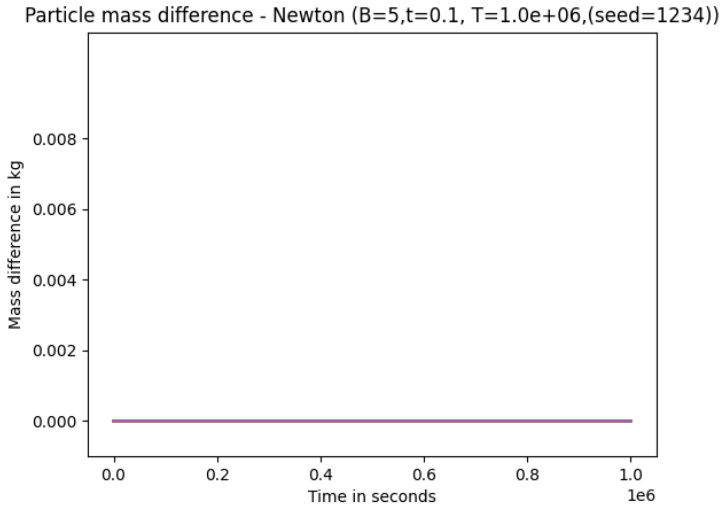


Figure 13.4.7: GR approximation, velocity magnitude. Total elapsed time $T = 10^6$s, time interval $t = \Delta T = 0.1$s.

.

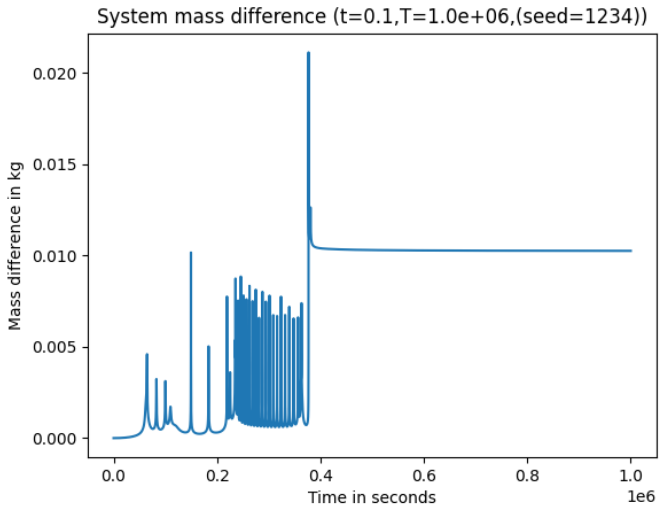


(a) GR approximation

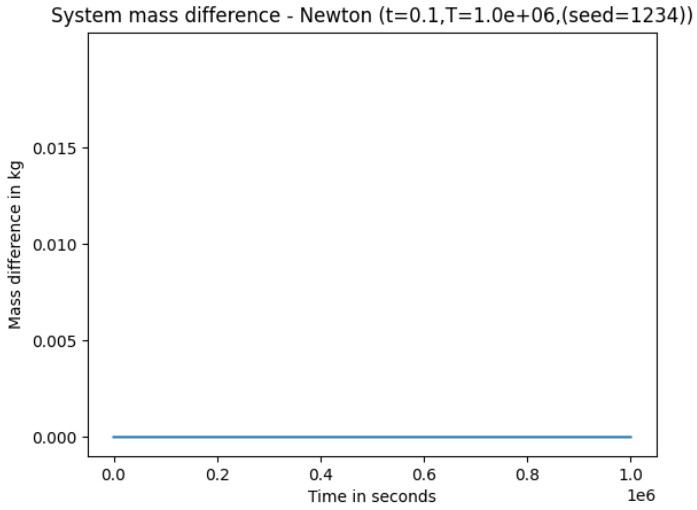


(b) Newtonian

Figure 13.4.8: Particle mass difference. Elapsed time $T = 10^6$s, time interval $t = \Delta T = 0.1s$.



(a) GR approximation



(b) Newtonian

Figure 13.4.9: System mass difference. Elapsed time $T = 10^6$s, time interval $t = \Delta T = 0.1s$.

13.5 Impact of full GR interaction

Retaining the physical premise of the model galaxy, i.e. point particles with equal intrinsic mass, the Newtonian simulation I employed in Section 13.2.1 is complete but the GR approximation in Section 13.2.2 is far from the end of the story. There are elements of the right hand side of the field equations, the source of gravity, that are missing. What would happen if the approximation were to be improved? My approach to this question must be qualitative rather than quantitative, for the reasons outlined in Section 12.4.10.

In GR, each individual particle, in its own inertial frame, experiences no force at all. I have chosen to consider five particles together, as a system, and have imposed co-ordinates for that purpose. I have calculated positions and velocities for each particle *relative* to the others in the chosen co-ordinate system. Because of these choices, if I wish to understand the dynamics and mass-energy of all five particles and the entire system I must consider:

1. The spacetime metric and stress-energy tensor at the timepoints inhabited by each particle, see Section 13.5.1.
2. The spacetime metric throughout the timespan-region within which the particles exist, including those timepoints in which there is no particle, see Section 13.5.2.
3. The significance of covariant time, Section 13.5.3.

13.5.1 Particle at a timepoint

The field equations 12.3.1 require a spacetime metric with a stress-energy tensor source and, at a timepoint, the relationship expressed in the equation is exact. There is a minimum value for the stress-energy tensor, since all known forms of matter satisfy the dominant energy condition $T_{00} \geq |T_{\mu\nu}|, \quad \forall\ \mu, \nu$ [204] such that the mass-energy of that matter must be positive. However, the field equations 12.3.1 impose no maximum, since the spacetime metric can move freely.

The mass-energy approximation defined for my simulation T in equation 13.2.11, i.e. a mass plus kinetic energy framework. This does not incorporate energy arising as a result of anisotropic pressure and sheer stress, i.e. off-diagonal terms in $T_{\mu\nu}$, which could increase the mass-energy content of the stress-energy tensor.

I therefore assert that the impact of a more realistic stress-energy tensor source would be to increase rather than decrease the mass-energy of the system.

13.5.2 System occupying a timespan-region

I have described in Section 12.4.4 that, in the GR framework, within any timespan-region of spacetime there exists gravitational field potential $t^{\mu\nu}$ relative to the rest of the universe. This gravitational energy is also a source of mass, but it is not included in the stress-energy tensor and cannot be written as a tensor, because it is co-ordinate dependent. In order to determine the total mass-energy of a system in a specified timespan-region, A, I would like to be able to write and to solve a system of equations of the form

$$\text{“ } \int_{\mathrm{A}} G_{\mu\nu} = \int_{\mathrm{A}} T_{\mu\nu} + \int_{\mathrm{A}} t_{\mu\nu} \text{ ”}$$

but I cannot: neither $G_{\mu\nu}$ nor $T_{\mu\nu}$ are integrable, and $t_{\mu\nu}$ is a coordinate-dependent object that cannot be written as a tensor. The simulation in this paper uses equation 13.2.12, a simple sum of the mass-energies of five individual particles, as an approximation for the GR mass-energy of the system. It completely ignores the mass-energy of the gravitational field itself. We must ask, how much mass-energy am I omitting?

In Section 12.4.4 on gravitoelectromagnetism and frame-dragging, I referred to work by Cooperstock et al [129] and Balasin et al [130] which suggests that the GEM mass-energy contribution is sufficient to obviate the need for dark matter.

In [130], it is interesting that the authors concluded also that the ‘Newtonian approximation breaks down in an extended rotating region, even though it is valid locally everywhere.’ Ludwig [131] further concluded that the GEM field produced by mass currents modifies galactic rotation curves notably at large distances, commenting, ‘... at large distances the Lorentz force due to the gravitomagnetic field effectively controls the mass equilibrium balance in view of the decaying centrifugal force. The field produced by the large disk of mass currents basically acts as a gravitomagnetic brake against the gravitational attraction.’

These authors did not take a complex system approach to the problem but instead modelled exact solutions to the GR-field equations using a stationary, axially symmetric spacetime metric and the approximation of a co-rotating, pressureless, perfect fluid source for a rotating disk galaxy. These papers are not, therefore, a direct answer to the question of 'How much mass-energy am I omitting?'. But GR does predict mass-energy in the gravitational field and these studies do lend weight to the consideration that its effect may be substantial in dense regions.

13.5.3 Time

The simulation and analysis accommodates time variance in the sense that I acknowledge the absence of energy conservation within the GR model, but otherwise it is based on the treatment of time in slices. This is itself an approximation, as discussed in Section 12.4.7.

13.6 Discussion

I have asserted that galaxies are complex systems, whatever the theory of gravity. Based on the maturing understanding of complex systems in modern physics I do not expect simplifying assumptions to do a good job of modelling galaxies made up of interacting bodies. Similarly, linear Newtonian gravity cannot be a good approximation for highly nonlinear GR except in special circumstances. With the aid of a model simulation I have demonstrated that:

- Treating a model galaxy as a complex, i.e. an intreacting, many-body system, produces results that are different from any simple, non-interacting approximation.
- The approximate GR and the Newtonian dynamics are initially similar but diverge irredemiably over time. How much time it will take for the difference to become significant cannot be predicted, and divergences can be abrupt.
- The absence of energy conservation within the system in the approximate GR case results not only in divergent dynamics but also higher mass-energy than the Newtonian calculation.

- The qualitative extension in Section 13.5 supports an expectation that increasing the accuracy of the GR approximation would increase rather than reduce the difference between the two simulations and would magnify the mass-energy of the system still further.

A point-particle, five-body model is hardly a physical system. Could it be that in larger and more realistic systems the differences between simple and complex, or between Newtonian calculations and GR, would be less significant? In other words, over a vast population could these results cancel out? There is an active and interesting field of research into averaging and backreaction effects from structure, see for example [118,216,217]. Yet, for the reasons outlined in Chapter 10, there are good reasons to think that neither simplifying nor averaging are the answer.

Chapter 14

SOCGR in the universe

If ... we grant that clumpiness in the distribution of matter in the universe is a basic property of fundamental importance for cosmology and not merely a local nuisance that can be ignored in the grand smoothed-out view, we must pay much more attention than we have thus far to the possible consequences of this situation.

G de Vaucoulers (1970) [218], p.1211

In Chapter 11 we considered the necessary generating conditions for and the characteristic phenomenology of SOC systems, and concluded that SOC cannot currently be excluded as a possible paradigm for study of the universe. Chapter 12 presented text-book GR, freed from the simplifying assumptions of the FLRW framework. In this chapter I bring the two together to address the big questions in cosmology. I distinguish between *implications* of the framework, which cannot necessarily be tested, and *predictions* arising from the framework. The latter cannot be stated in precise numerical terms because they are necessarily statistical in nature, but they are capable of test and challenge.

14.1 SOCGR implications and predictions: the basis

I begin by summarising the key features of the SOC paradigm from Section 11.3 and the theory of GR from Section 12.4, since they form the basis of this

chapter.

1. SOC
    - 1.1 The emergence of structure is an inherent property of SOC and the characteristics of the emergent structure are independent of initial conditions.
    - 1.2 The system dynamics are irreversible: time symmetry is broken.
    - 1.3 Mass-energy accumulates as configurations evolve to critical states, but it is not the case that all parts of a system are perpetually at a critical point.
    - 1.4 At the critical point, configurations with maximum energy are statistically preferred (contrary to the minimum energy expectation of a system in equilibrium).
    - 1.5 When a local physical threshold is breached, there is a phase transition and avalanche dissipation occurs, affecting the entire system algebraically. This results in long-range spatio-temporal correlations.
    - 1.6 There is no well-defined mass density in a fractal system: therefore samples cannot be representative of the population and statistical tools applicable in systems exhibiting equilibrium and regularity are not valid.
    - 1.7 We observe the universe from a point and time in space which is in the fractal.
2. GR
    - 2.1 Gravity does not scale linearly with mass-energy.
    - 2.2 The field equations 12.3.1 embody a feedback loop, and equilibrium is not possible except in a vacuum or single-body universe.
    - 2.3 There is potential in the gravitational field, with associated gravito-electromagnetic (GEM) phenomena including frame-dragging.
    - 2.4 In low density regions the gravitational field can repel as well as attract.
    - 2.5 The gravitational field cannot be integrated over a timespan-region. Therefore, there is no conservation condition that applies to mass-energy in a timespan-region.
    - 2.6 Time, as well as space, is fundamentally relative.

14.2 Implications of SOCGR

14.2.1 Initial conditions are irrelevant

For any system in which the SOC paradigm applies, i.e. a system in which there are many interacting components with a slow-drive energy source and some physical mechanism(s) which introduce thresholds, initial conditions are not relevant to the evolution of structure. There is no initial condition that must be special or tuned in any way.

We cannot, by any analysis, rewind the present configuration of matter and energy to uncover a unique starting point.

14.2.2 The 'arrow of time' is irreversible

Not only are we unable, by any analysis, to rewind the present configuration of matter and energy to uncover a unique starting point - we cannot rewind at all. There is no analysis that can recover a previous state and there is no mechanism at all which can restore a past configuration. This is a general property of complex systems, as evolving structure clearly distinguishes an irreversible relationship between now and what has gone before.

This does not mean that we cannot understand the system, but it does mean that we must adopt a different approach from that traditionally employed when dealing with reversible mechanics.

14.2.3 The universe may have no beginning

In SOC, there is no suggestion that the universe emerges from a singularity at the beginning of time. Rather, the picture is one of perpetual change.

Nevertheless, SOC is not incompatible with aspects of the Big Bang. In the SOC paradigm, catastrophic avalanches are expected to occur with non-vanishing probability and no limit to their potential extent. In principle, therefore, the Big Bang scenario with which we are so familiar could be the outcome of such an event affecting the whole system: in other words, our universe could be the ongoing flow of a total structural reformation event. With this idea, there may or may not be a mathematical singularity associated with the beginning of the flow,

but it is possible to assert that there is no reason to assume that the beginning of this flow is a beginning of all phenomena. Equally, the universe we can observe and study may not represent the entirety of all phenomena: there may be an eternally-evolving 'super-universe' of which our particular flow is just one region.

14.2.4 The universe has no objective age

Time is tricky in GR. Even if time was everywhere flat - and in GR it is not, as eloquently argued by Wiltshire [141] - there is no way to establish how old the universe is. In fact, in the full implications of GR, the age of the universe is not a well-defined question.

At a deeper level, taken together with the SOC implication that the universe may have no beginning, it may simply be that the universe has no age at all.

Of course, this does not prevent us from carrying out valuable analysis based on *relative* time and age - as long as we are careful to take into consideration the implications of GR-associated variability.

14.2.5 There is no Hubble tension

The Hubble constant is a feature of the FLRW framework. It is only defined in that context, as described in Section 9.3.6. In SOCGR there is no such parameter and no valid comparison to be made between early- and late-time expansion.

14.2.6 The horizon, isotropy and flatness problems vanish

In Section 9.3.2 I described the horizon and isotropy problems, associated with correlations across the entire sky which should not be possible within the past light cone of the present horizon size. In the SOCGR paradigm we lose the condition of a particular time duration which limits causal connection, and we gain a mechanism which generates long-range spatio-temporal correlations, with no need for any inflationary mechanism.

Likewise, the flatness problem is a relic of FLRW and hence vanishes in the SOCGR framework: in its complex reality, the universe is a shifting topography of interacting gravitational waves. It is nowhere flat.

In fact, in SOCGR all cosmologial fine-tuning problems disappear.

14.3 Predictions of SOCGR

In this section I summarise those predictions of the SOCGR framework which are, in principle, testable. In each case I first explain the prediction and then refer to tests and evidence. In Chapter 15 I make suggestions for further work in this regard.

14.3.1 There is emergent structure which tends towards scale-free, fractal configurations

1. **Structure perpetually emerges, tending towards scale-free, fractal configurations.**

   This is a core feature of SOC.

   In Section 11.2.2 I discussed the question of whether the universe manifests fractal characteristics. This issue is not yet fully determined but it can be tested. Testing must use statistical tools that are not based on the *a priori* assumptions of homogenity and the existence of a well-defined mean density.

2. **In present and future sky surveys will show that ever-larger structures exist.**

   This is an expectation based on the scale-free nature of fractal structures.

   As discussed in Section 11.2.2, so far this prediction has been fulfilled.

3. **Spatio-temporal correlations extend throughout the entire universe and such correlations will be manifest in, for example, nascent fractal structure in the CMB.**

   The emergence of spatio-temporal correlations in SOC structures is a fundamental feature of the paradigm. (Of course, the temporal aspect of the correlation may be difficult to come to grips with in the GR theory of gravity.)

The high degree of spatial correlation observed in the universe, cause of the isotropy problem in the standard cosmological model (see Section 9.3.2), is supporting evidence for this aspect of the SOCGR framework.

It will be interesting to use, explicitly, statistical methods for irregular structures to reanalyse sky surveys and CMB data.

14.3.2 There is excess mass-energy in cosmic structures (a.k.a. dark matter)

1. **The mass-energy of structures such as galaxies and clusters will exceed the Newtonian expectation.**

   This prediction arises as follows:

   - Mass-energy accumulates under a slow-drive energy source. In GR, the stress-energy tensor accumulates the dynamical mass-energy generated in SOC processes and can do this until some physical threshold is reached, since it is unconstrained by $G_{\mu\nu}$ which is nonlinear and unbounded (although Einstein found there may be a gravitational minimum for spatial extent, beyond which additional energy results in renewed expansion - see Einstein 1939 [219]).
   - There is no conservation of energy in any timespan-region, since the field equations cannot be integrated.
   - In addition to the stress-energy tensor there is gravitational field energy within the timespan-region relative to other timespan regions. This energy cannot be written as a tensor.
   - Gravity does not scale linearly with mass-energy: the field equations are nonlinear in the spacetime metric and its first derivative and they also embody a positive feedback loop. As a result of the feedback loop in the field equations it is to be expected that, as the energy of matter and radiation in the stress-energy tensor increases, the gravitational field will also increase the system's mass-energy by drawing down potential from the region external to the timespan region occupied by the system. In other words, increasing kinetic and stress energy leads to *increasing* rather

than decreasing gravitational field potential. In this framework, the greater the interaction the greater the drawdown of potential.

SOCGR does not deny the possibility of new particles, but they are not required to explain the excess potential usually described as dark matter.

This prediction is instantly recognisable as an expectation of dark matter phenomena. All the compelling and self-consistent galaxy rotation curve and gravitational lensing evidence for dark matter, described in Section 9.3.5, supports the argument.

2. **Given two regions in which the sum of masses of components is the same, one of which is highly interactive and another which is not (for example, dust, gas or a highly dispersed galaxy), the total mass-energy of the highly-interactive region will always exceed that of the mass-energy of the low interaction region.**

   I have described above the SOCGR mechanism whereby complex systems of interacting bodies accrue increasing mass-energy. In a system of non-interacting bodies, i.e. a simple system, there is no such mechanism.

   In Section 9.3.5 we noticed that the matter of the bullet cluster is evidence that the mass-energy associated with colliding clusters tracks the galaxies in those clusters rather than the intra-cluster gas, even though by conventional assumptions the gas is estimated to have greater intrinsic mass[1].

   Studies of correlations between structure characteristics and dark matter distribution have been underway for some time, see for example [220]. It will be interesting to revisit them from the perspective of SOCGR.

   Specifically, statistical analysis of galaxy characteristics, using tools that are appropriate for irregular, nonequilibrium systems, should

[1] As an aside I note in relation to the bullet cluster that, in GR, clusters are expected to occupy deep, asymmetric and dynamical wells of gravitational potential and a proportion of their mass-energy is derived from that field potential. If we think of each potential well as a minimally-stable 'standing wave' in the gravitational field then the meeting of two such waves must result in interference. It is not unreasonable that the profile of the potential should mimic that of shock waves as is the case in the bullet cluster.

demonstrate a relationship between the degree of interaction and the mass-energy gap.

I note the recently reported findings that the ultra-diffuse dwarf galaxy FCC 224 is deficient of dark matter [221, 222].

3. **There will be variability in the profile of excess gravitational potential in cosmic structure. Statistically, high mass-energy systems are expected to predominate.**

   In SOC systems, configurations with maximum energy are preferred at the critical point. Although it is not the case that all systems are at all times at the critical point, the fact that the slow-energy-drive accumulation phase in SOC structure formation takes place over extended periods whilst the intermittent cascades of energy release which are triggered by the breach of a physical threshold are rapid suggests that there will be a statistical bias towards high- and maximum-energy systems in the population of cosmic structures.

   Therefore, statistical analysis of galaxy characteristics, using tools that are appropriate for irregular, nonequilibrium systems, should demonstrate that high mass-gap systems predominate.

   It will be interesting to see if there are correlations between mass-gap and stage of evolution and/or shape of galaxy structures.

14.3.3 Adjacent to structures there are voids

1. **Regions of underdensity will be present adjacent to and related to dynamical cosmic structures.**

   This prediction arises as follows:

   - The dynamic frame-dragging of gravitational potential energy into wells incorporating dense regions of energetic interaction must result in a compensatory change in the gravitational field outside the well. This is because the field potential only exists in a timespan-region relative to the rest of the universe: for each *timepoint* in the universe there is no potential, and there is a conservation condition $\nabla_\mu T^{\mu\nu} = 0$.
   - I argue that, because gravitational waves can travel no faster than the speed of light, it is likely that the frame-dragging effect

of potential being drawn into the energetic region will establish a wave peak that is a higher potential than that in the surrounding timespan-region. See Figure 12.4.1.

- The unexpectedly high radial velocities observed in clusters is not unstable precisely because the clusters are unable to disperse: they are constrained by deep potential wells, the peak of which is higher than the potential in the background region. These wells serve the same function as halos of dark matter in conventional terminology.
- This profile, which is particular to GR, allows the quasi-stability in structure to extend over long periods.

That voids are closely related to structure is supported by the findings of the recent Dark Energy Survey's Y3 reconstructed weak lensing convergence mass maps [223], which are weighted projections of the primarily dark matter density field in the foreground of observed galaxies. The team found underdensities in tomographic slices of the galaxy catalogue of which they said, 'On average, these tunnel-like voids correspond to density minima that are compensated by an overdense zone in their surroundings'. An extended underdensity consistent with a super-void with radius $R_v \sim 250$ Mpc/h is also reported [224] (pg 18).

Closer to home, an investigation of a galaxy underdensity covering 90% of the sky in the region around the around the Milky Way is reported in [225].

14.3.4 Distant structures will appear to accelerate away (a.k.a dark energy)

1. **Observations of distant structures in the universe will indicate that they are accelerating away from us.**

   SOCGR makes no prediction about whether the universe is undergoing overall spatial expansion or, if it is, whether that expansion is accelerating or not. In fact, since no boundaries to the universe are known (in SOCGR, there is not even a boundary at an initial time), and since time and space are all relative, the question of universal or general expansion is meaningless in this framework.

SOCGR does predict that an observer in one timespan-region of the universe will see acceleration in distant objects that differs from that which would be calculated by an observer in another timespan-region. This is a consequence of fractal structure.

SOCGR predicts that observers in relatively dense regions of the universe will calculate that all distant objects are accelerating away from them. This is because:

- Structure formation leads to a deepening gravitational potential well, as described in Section 14.3.2.
- Time is not linear: in the region of a deep and deepening potential well, time proceeds normally for observers within that Lorentz frame but slows down for a distant observer (this concept is familiar to us from the extreme case of a black hole). For the observer within the well, objects outside it are subject to relative time compression.

This prediction is, essentially, a statement that the observation of apparent expansion conventionally described as dark energy is to be expected - but is not necessarily expansion. Thus the observations that distant Type I supernovae do appear to be accelerating away from us as observers, described in Section 9.3.4, are also evidence for SOCGR.

Chapter 15

Next steps and conclusion

The fact that we are traditionally accustomed to think in terms of smooth or analytical structures has a crucial effect on the type of questions we ask and on the methods we use to answer them.
Gabrielli, Labini, Joyce, Pietronero (2005) [85], p.5

Often complex structures arise from processes which are strongly out of equilibrium and dissipative ... A fundamental notion in this context is that of irregularity, one which is completely new to physics... All the previous 'regular fluid-like' concepts and theoretical methods lose their meaning and, in order to give a proper characterization of the properties of this sytem, one has to look at it from a new perspective.
Gabrielli, Labini, Joyce and Pietronero (2005) [85], pg. 3

15.1 Ways forward

15.1.1 A new and multi-disciplinary approach

SOC is a paradigm rather than a single mathematical theory, and non-FLRW GR has no general solution. Therefore, the SOCGR framework I have introduced is essentially an invitation to ask new questions and to employ the developing tools and techniques designed for work with irregular, complex, nonequilibrium systems to revisit the vast and growing available data. This work must necessarily

be multi-disciplinary, involving people with expertise in cosmology, complexity science, modern statistical physics, fluid dynamics and vortices, modelling and simulations. It is likely that machine learning will play a valuable role in future.

The aim must be to analyse existing and new data in ways which *test* the assumptions of our models as well as explore best fit scenarios. For example, the Dark Energy Survey Year 3 results [226] conclusion has the statement, 'we dropped the two highest redshift bins of the MagLim sample as they contributed to a very poor fit to all models considered in this paper'. I do not comment on this treatment in particular, but note that in future new models may make sense of old data.

I suggest that it would be valuable to incorporate awareness of complexity science and training in the techniques of non-equilibirum statistical physics in graduate taught and research programmes.

15.1.2 Working with statistical physics for complex structures

I concur with Gabrielli et al [85] that it is essential that the new tools of statistical physics for complex systems should come to be employed in cosmology.

Purely analytical methods cannot support research in this field alone: numerical and experimental work is necessary, in co-ordinated, inter-disciplinary approaches. Gabrielli et al [85] suggest such efforts can be characterised at three levels:

1. **Mathematical and geometrical methods**, including those of fractal geometry introduced by Mandelbrot [227].
2. **Physical models**, including computer simulations generating structures by self-organisation. Gabrielli et al list a number of such models and Preussner's work [177] provides a portfolio of SOC-specific approaches.
3. **Developing theoretical understanding**. Scaling theory derived from applications in critical phenomena has been used in phenomenological approaches to complex systems, allowing identification of relations between properties. On the other hand, specific mechanisms for structure formation

do not appear to allow generalisation in fundamental theories. Any possible specific theory must accommodate the facts that these systems are far from equilibrium and have irreversible dynamics. This is where SOC provides a useful starting point.

15.1.3 Create a clear programme of work

A first step will be to draw up a clear and well-defined programme of work. The following are not intended to be complete proposals, nor are they comprehensive.

- As far as I am aware, there has been no SOC-based examination of the cosmos as a whole, but there has been work which uses the tools of statistical physics for irregular and nonequilibrium systems, for example [162–168]. It would be helpful to conduct a thorough review of these papers, in collaboration with their authors, to understand them fully and explore ways in which they can be extended.

- It would be interesting to review alternative views alongside this work: in particular, a thorough understanding of the different conclusions of the work cited in the preceding paragraph and the analysis of the WiggleZ survey [193] is desirable.

- Carry out a critical review of the WMAP paper [228] to clarify how this analysis is model-dependent and what the implications and alternatives are.

- A review of Galaxy characteristics and correlations in the new perspective of structure emerging from a slow-drive, nonlinear energy source punctuated by period cascades of release would be interesting.

- Employ tools used in fluid dynamics and the study of vortices to explore how gravitational potential might behave in a frame-dragging, deepening potential well.

- A SOC mechanism predicts that there should be nascent fractal structure in the CMB. I am aware of two studies that have analysed CMB data and found evidence of fractal dimension in the CMB, [229] in 2011 and [230] in 2020, and there may be others. It would be interesting to do a full literature review and revisit these analyses.

- For all of the above, a good understanding of the impact on distance estimates of both time variance and the topography of gravitational waves in a structure tending towards fractal is essential. This represents a very substantial field of work, revisiting the construction of the cosmological distance ladder.

There are further inferences, i.e. tentative predictions, that can be drawn from the framework. Clarifying these is a stream of work in its own right. It seems likely that, for example:

- All structures will tend towards rotation, and there will be spiral-like signatures in regions of strong interaction. This suggestion arises primarily from GR in a many-body system: in particular, from the phenomenon of frame-dragging in the gravitational field, described in Section 12.4.4. As bodies interact they follow geodesic paths relative to each other, with positive feedback leading to spirals and rotation.
- Emerging structures as they mature will tend towards being flat, i.e. oriented in a plane.
- GR will result in a predominance of spiral structure for galaxies.
- In very dense, strongly interacting regions, jets of energy release will occur. This is a likely consequence of the physical impact of gravitational slingshot when highly-energetic, dynamical bodies travel so close to each other than they each experience strong deflection.

As an aside, although in the SOCGR paradigm the present state of the system can tell us nothing about its past evolution, the latter statement is not so precise in relation to the universe, since to look out into the cosmos is to look back in time.

15.2 On worldviews and how they shape our research

'*One of the main tasks of cosmology is to measure the density of the Universe, and how this is divided between dark matter and baryons.*' From Chapter 22 [90]. This is a statement from the Big Bang, FLRW worldview of cosmology. The question is not only not relevant in the alternative paradigm of SOCGR, it is

not valid, since the density of the universe is not defined when we view it as a complex system. The way in which we view our universe has a huge impact on the questions we believe we can ask, and the scope of the answers that we will admit into consideration.

A *worldview* may be described as 'a largely unconscious but generally coherent set of presuppositions and beliefs that every person has which shape how we make sense of the world'[1]. This is related to the idea of a *paradigm* introduced by Kuhn in his influential book *The Structure of Scientific Revolutions* [231]: '(Paradigms) I take to be universally recognised scientific achievements that for a time provide model problems and solutions to a community of practitioners'.

In the introduction I posed the question: do we want to solve an equation, or do we want to understand the universe? In a very real sense this is a question about worldview and about paradigm. In the western history of sciences, and of physics in particular, there is a widespread presupposition that it is only by solving equations that we can understand the Universe. We have seen that the drive to solve the field equations of GR for the universe as a whole has led to a grand standard model that simplifies and smooths, posits a beginning and an end for all things, an age of the universe of order 13 billion years, and which carries in its wake considerable unknowns, uncertainties, tensions and singularities.

In this thesis I have attempted to demonstrate that there are options for the questions we ask and for the scientific methods we use. That there is, in effect, an alternative worldview available which can lead to paradigms that are no less rigorously scientific than the FLRW exact GR solution.

As a final word, it is becoming ever more clear that humanity's collective worldview must shift from one in which our society, our environmennt and our climate are perceived as stable background processes which we can perturb at will around a well-defined equilibrium state and instead realise that we will owe our continued existence to a respectful relationship with the complex, interconnected web of a reality in which there is no uniformity and no equilibrium. Every action we take matters: a small push here, or there, cannot be relied upon to have a small impact. Perhaps Physics as a discipline, as it begins to focus beyond

[1] This form of the definition is taken from OxfordReference.com

the traditional tools of simplification, can contribute to this shift in collective worldview.

15.3 Conclusion

Physics in the 20th century solved the problems of constucting hierarchical levels which obeyed clear-cut generalizations within themselves: atomic and molecular theory, nuclear physics, quantum chromodynamics, electroweak theory, quantum many-body theory, classical hydrodynamics, molecular biology ... In the 21st century one revolution that can take place is the construction of generalizations which jump and jumble the hierarchies, or generalizations which allow scale-free or scale-transcending phenomena. The paradigm for the first is broken symmetry, for the second self-organised criticality.

P. W. Anderson (2011) [172], p.112

I have argued that the universe is a complex system and that it is not possible to model it effectively by choosing simplifications that negate its complex reality. Even if the universe proves to be statistically homogeneous and isotropic at large scales, the simplifying assumptions of the standard cosmological model, the FLRW framework, do not apply: the statistical validity of that framework is assured only in the case of super-homogeneity. In other words, it requires that the universe is everywhere and at all times smoothly homogeneous and flat. Very clearly, it is not.

We need to analyse the vast data we have accumulated in this and the last century using different statistical tools, so assumptions can be tested rather than taken as given. Fortunately, such tools have begun to be developed in recent decades. They are not yet mainstream in cosmology, but there are groups who have used them to great effect.

New tools in statistical physics cannot, by themselves, tell us about the dynamical processes which generate the evolution of structures. They cannot explain *how* emergent structure evolves in complex systems. For that we need a new paradigm, and I have suggested that SOC is a powerful place to start. There are numerous SOC models but, even if there were one which seemed perhaps relevant

to the entire universe - and there is no such model - the extreme nonlinearity of gravity cannot be adequately implemented as an interaction algorithm. However, the heuristic story which underpins SOC is a perfect example of an alternative way of looking at the questions of cosmology. What happens if we remove the need for linear evolution from a singular beginning? If we look at the CMB data without analysing it within a framework of assumed homogeneity and isotropy, what do we see? If our carefully constructed distance ladder is revisited in the context of a timescape, what impact is there on our assessment of the distance in time and space of the phenomena we observe?

Without equations that can be solved or a specific model that we can write as a simulation for the universe as a whole, it is still possible to do effective work that will lead to an ever-greater understanding of the universe we inhabit. As a bonus, SOCGR predicts the observed phenomena commonly described as dark matter and dark energy without invoking new matter or fields.

In conclusion, there is much we do not yet understand about mass and its role in structure formation. This is exciting - there is a lot to be done. The standard, Big Bang cosmology puts us at the centre of the universe with everything in all directions expanding away from us. Ptolemy's model of the solar system put Earth in the centre: it was able, using complex adjustments, to predict the movements of the planets. Copernicus' heliocentric model which replaced it, and new data from telescope observations, opened up the way for Kepler's simplifying laws. In the modern era, the physics community has created extraordinary theoretical, experimental and technological advances and has accumulated vast data of staggering precision. Perhaps we are at the point where we can proceed to move beyond the open problems of 20th century cosmology to find new questions that belong to the next generation.

Appendices to Part I

Appendix A

Catalogue of 6D tree amplitudes

A.1 Polarisation and helicity in 6D amplitudes

Our convention is that all momenta are outgoing, hence an incoming momentum P is $-P$ in calculations.

In this project we consider only external gluons and Higgs, and internal gluon and scalar loop momenta in amplitude calculations. Scalars are unaffected by direction. Gluons, however, have four helicity states in 6D (see page 38). Since these cannot be assigned a simple $+, -$ it is not valid simply to reverse helicity for a negative momentum: instead, it is sufficient to multiply the spinor associated with the negative momentum by a factor of i.

This adjustment to the spinors in the amplitude calculation is made in gghj module **spinors**.

A.2 6D 3-point analytic amplitudes used in calculation

3-point - all gluon

Source: [55]

$$A_3^0(1^g_{a\dot{a}}, 2^g_{b\dot{b}}, 3^g_{c\dot{c}}) = i\Gamma_{abc}\tilde{\Gamma}_{\dot{a}\dot{b}\dot{c}} \tag{A.2.1}$$

where the tensors

$$\begin{aligned}\Gamma_{abc} &= u_{1a}u_{2b}w_{3c} + u_{1a}w_{2b}u_{3c} + w_{1a}u_{2b}u_{3c}\\ \tilde{\Gamma}_{\dot{a}\dot{b}\dot{c}} &= \tilde{u}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{w}_{3\dot{c}} + \tilde{u}_{1\dot{a}}\tilde{w}_{2\dot{b}}\tilde{u}_{3\dot{c}} + \tilde{w}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{u}_{3\dot{c}}\end{aligned} \tag{A.2.2}$$

are written in terms of the 2x1 spinors $u, \tilde{u}$ satisfying the following properties defined on a cyclic order $\{ijk\}$:

$$\begin{aligned}u_{ia}\tilde{u}_{jb} &= \langle i_a|j_b\rangle\\ u_{ja}\tilde{u}_{ib} &= -\langle j_a|i_b\rangle\end{aligned} \tag{A.2.3}$$

and $w, \tilde{w}$ are the inverse of $u, \tilde{u}$, for which we also impose momentum conservation. These objects are described in full in Section 5.4.2.

3-point - 1 higgs 2 gluons

Source: [49]:

$$A_3^0(H, 1_{a\dot{a}}, 2_{b\dot{b}}) = i\langle 1_a|2_{\dot{b}}]\langle 2_b|1_{\dot{a}}] \tag{A.2.4}$$

$$\textbf{h1g2} \quad = i\ \textbf{sp62}(\textbf{sp6d}_{12}, \textbf{sp6d}_{21})$$

3-point - 1 higgs 2 scalars

Source: [49]:

$$A_3^0 = -is_{12} \tag{A.2.5}$$

$$\textbf{h1s2} \quad = -i\ \textbf{sp6sij}_{12}$$

3-point - 2 scalars 1 gluon

Source: [56] Appendix B:

$$A^0(1^{\phi^{1,2}}, 2^{\phi_{1,2}}, 3^g_{c\dot{c}}) = \frac{-i}{2s_{r3}}\langle 3_c|(1-2)|r|3_{\dot{c}}] \tag{A.2.6}$$

where r is a massless refernce vector satisfying $s_{r3} \neq 0$.

A.3 6D 4-point analytic amplitudes used in calculation

4-point - all gluon

Source: [55]

$$A_4^0 = \frac{-i}{st}\langle 1_a 2_b 3_c 4_d\rangle[1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}}] \tag{A.3.1}$$

Repeated in [45] Section II C and in [56] Appendix B:

$$A_4^0(1^g_{a\dot{a}}, 2^g_{b\dot{b}}, 3^g_{c\dot{c}}, 4^g_{d\dot{d}}) = \frac{-i}{s_{12}s_{23}}\langle 1_a 2_b 3_c 4_d\rangle[1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}}] \tag{A.3.2}$$

$$\textbf{g4} \quad = \frac{-i}{\textbf{sp6sij}_{12}\textbf{sp6sij}_{23}}\ \textbf{sp64d}(1,2,3,4,a,b,c,d)\ \textbf{sp64td}(1,2,3,4,a,b,c,d)$$

We can choose to construct this 4-point amplitude from the 3-point using BCFW, using equation 4.3.1 and selecting 1 and 2 for the shift. There is only one Feynman diagram, see Figure 5.3.1, and A_4^0 thus takes the form:

$$A_4^0 = -\frac{i}{k^2}A_{3L}A_{3R} \tag{A.3.3}$$

4-point - 3 gluons 1 higgs

Source: [49]:

$$\begin{aligned} A_4^0(H, 1^g_{a\dot{a}}, 2^g_{b\dot{b}}, 3^g_{c\dot{c}}) = {} & \frac{-i}{s_{12}s_{23}^2 s_{13}^2} \\ & \times \{(s_{12}m_H^2 + s_{13}s_{23})\langle 1_a|\ \not{k}_3|2_b\rangle[1_{\dot{a}}|\ \not{k}_3|2_{\dot{b}}]\langle 3_c|\ \not{k}_2\ \not{k}_1|3_{\dot{c}}] \\ & - s_{23}(s_{13}m_H^2 + s_{12}s_{23})\langle 1_a|3_{\dot{c}}]\langle 3_c|1_{\dot{a}}]\langle 2_b|\ \not{k}_1\ \not{k}_3|2_{\dot{b}}] \\ & + s_{13}(s_{23}m_H^2 + s_{12}s_{13})\langle 2_b|3_{\dot{c}}]\langle 3_c|2_{\dot{b}}]\langle 1_a|\ \not{k}_2\ \not{k}_3|1_{\dot{a}}] \\ & - s_{12}\langle 1_a|\ \not{k}_2\ \not{k}_3|1_{\dot{a}}]\langle 2_b|\ \not{k}_1\ \not{k}_3|2_{\dot{b}}]\langle 3_c|\ \not{k}_2\ \not{k}_1|3_{\dot{c}}]\} \end{aligned} \tag{A.3.4}$$

4-point - 2 gluons 2 scalars

Source: [56] Appendix B:

$$A_4^0(1^g_{a\dot{a}}, 2^g_{b\dot{b}}, 3^{\phi_{1,2}}, 4^{\phi_{1,2}}) = \frac{-i}{4s_{12}s_{23}}\langle 1_a 2_b 3_e 3^e\rangle[1_{\dot{a}} 2_{\dot{b}} 3_{\dot{e}} 3^{\dot{e}}] \tag{A.3.5}$$

4-point - 1 gluon 1 higgs 2 scalars

Source: [49]:

$$A_4^0(H, 1^g_{a\dot{a}}, 2^s, 3^s) = -i\Big(\frac{1}{s_{23}} + \frac{m_H^2}{s_{12}s_{13}}\Big)\langle 1_a| \not{k}_3 \not{k}_2|1_{\dot{a}}] \tag{A.3.6}$$

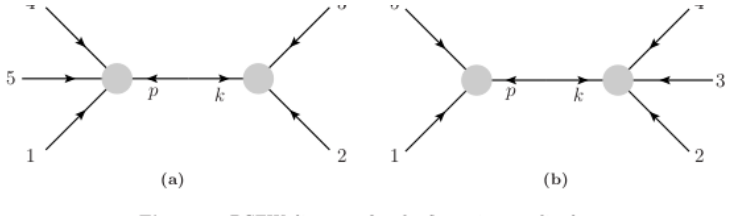
A.4 6D higher-point amplitudes from BCFW - examples

In this Section we retain the p, k notation for the intermediate momentum that is used in the original sources.

5-point - all gluon

Source: [55]

Using BCFW equation 4.3.1 to construct this tree amplitude there are two diagrams (figure 2 from [55]):

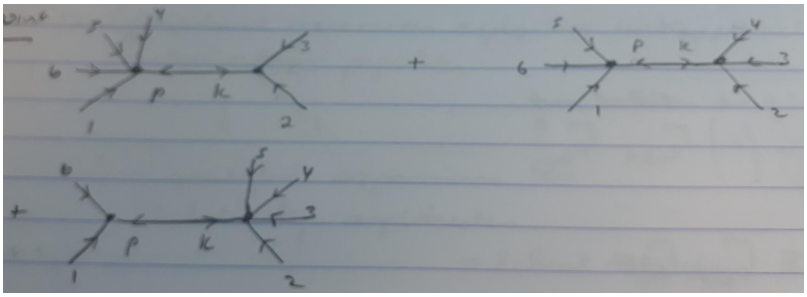


And A_5^0, stated here for simplicity without the $X^{a\dot{a}}$ and spinor little group indices which are shown in full in Section 5.4.3, thus takes the form:

$$A_5^0 = -\frac{i}{k_{23}^2}A_{4L}A_{3R} + \frac{i}{k_{15}^2}A_{3L}A_{4R} \tag{A.4.1}$$

6-point - all gluon

Using equation 4.3.1 relations and the three diagrams:

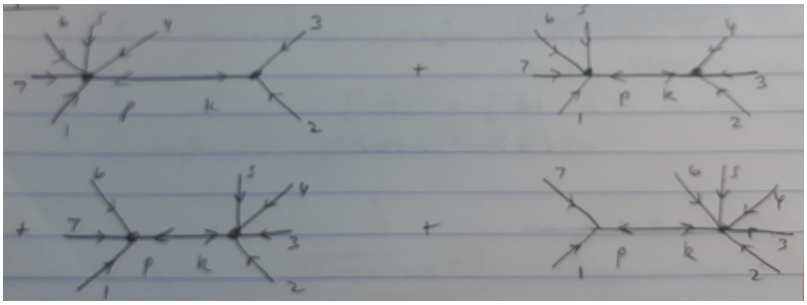


We have the form:

$$A_6^0 = -\frac{i}{k_{23}^2} A_{5L} A_{3R} - \frac{i}{k_{234}^2} A_{4L} A_{4R} - \frac{i}{k_{16}^2} A_{3L} A_{5R} \tag{A.4.2}$$

7-point - all gluon

Using equation 4.3.1 relations and the four diagrams:



$$A_7^0 = -\frac{i}{k^2} A_{6L} A_{3R} - \frac{i}{k^2} A_{5L} A_{4R} - \frac{i}{k^2} A_{4L} A_{5R} - \frac{i}{k^2} A_{3L} A_{6R} \tag{A.4.3}$$

A.5 6D analytic amplitude expressions available for use in testing

5-point - all gluon

Source: [55]

$$A_5^0 = -\frac{i}{k^2} A_{4L} A_{3R} + \frac{i}{k^2} A_{3L} A_{4R} = \frac{1}{s_{12} s_{23} s_{34} s_{45} s_{51}} (\mathcal{A} + \mathcal{D}) \tag{A.5.1}$$

where

$$\mathcal{A}_{a\dot{a}b\dot{b}c\dot{c}d\dot{d}e\dot{e}} = \langle 1_a| \not{p}_2 \not{p}_3 \not{p}_4 \not{p}_5|1_{\dot{a}}]\langle 2_b 3_c 4_d 5_e\rangle[2_{[a]}3_{[b]}4_{[d]}5_{[e]}]\text{+cyclic permutations} \tag{A.5.2}$$

and

$$\begin{aligned}\mathcal{D}_{a\dot{a}b\dot{b}c\dot{c}d\dot{d}e\dot{e}} = &\langle 1_a(2.\tilde{\Delta}_2)_{\dot{b}}\langle 2_b 3_c 4_d 5_e\rangle[1_{\dot{a}}3_{\dot{b}}4_{\dot{d}}5_{\dot{e}}] + \langle 3_c(4.\tilde{\Delta}_4)_{\dot{d}}\langle 1_a 2_b 4_d 5_e\rangle[1_{\dot{a}}2_{\dot{b}}3_{\dot{c}}5_{\dot{e}}]\\ &+ \langle 4_d(5.\tilde{\Delta}_5)_{\dot{e}}\langle 1_a 2_b 3_c 5_e\rangle[1_{\dot{a}}2_{\dot{b}}3_{\dot{c}}4_{\dot{d}}] - \langle 3_c(5.\tilde{\Delta}_5)_{\dot{e}}\langle 1_a 2_b 4_d 5_e\rangle[1_{\dot{a}}2_{\dot{b}}3_{\dot{c}}4_{\dot{d}}]\\ &- ([1_{\dot{a}}(2.\Delta_2)_b\rangle[2_{\dot{b}}3_{\dot{c}}4_{\dot{d}}5_{\dot{e}}]\langle 1_a 3_c 4_d 5_e\rangle + [3_{\dot{c}}(4.\Delta_4)_d\rangle[1_{\dot{a}}2_{\dot{b}}4_{\dot{d}}5_{\dot{e}}\rangle\langle 1_a 3_c 4_d 5_e\rangle\\ &+ [4_d(5.\Delta_5)_e\rangle[1_{\dot{a}}2_{\dot{b}}3_{\dot{c}}5_{\dot{e}}]\langle 1_a 2_b 3_c 4_d\rangle - [3_c(5.\Delta_5)_e\rangle[1_{\dot{a}}2_{\dot{b}}4_{\dot{d}}5_{\dot{e}}]\langle 1_a 2_b 3_c 4_d\rangle)\end{aligned} \tag{A.5.3}$$

with

$$\begin{aligned}\Delta_1 &= \langle 1| \not{p}_2 \not{p}_3 \not{p}_4 - \not{p}_4 \not{p}_3 \not{p}_2|1\rangle\\ \tilde{\Delta}_1 &= [1| \not{p}_2 \not{p}_3 \not{p}_4 - \not{p}_4 \not{p}_3 \not{p}_2|1]\end{aligned} \tag{A.5.4}$$

and other Δ_i defined by cyclic permutations.

There are two contributions, one in the s_{23} channel and the other in the s_{51} channel ([45] Section III B). Evaluating the diagrams gives

$$\begin{aligned}A_5^0(1^g_{a\dot{a}}, 2^g_{b\dot{b}}, 3^g_{c\dot{c}}, 4^g_{d\dot{d}}, 5^g_{e\dot{e}}) = &\frac{-i}{s_{45}s_{5\hat{1}}s_{23}}(\langle \hat{1}_a \hat{2}_b 4_d 5_e\rangle u_{3_c} + \langle \hat{1}_a 3_c 4_d 5_e\rangle u_{\hat{2}_b})\\ &\times([\hat{1}_{\dot{a}}\hat{2}_{\dot{b}}4_{\dot{d}}5_{\dot{e}}]\tilde{u}_{3_{\dot{c}}} + [\hat{1}_{\dot{a}}3_{\dot{c}}4_{\dot{d}}5_{\dot{e}}]\tilde{u}_{\hat{2}_{\dot{b}}})\\ &\frac{-i}{s_{34}s_{51}s_{\hat{2}3}}(\langle \hat{2}_b \hat{1}_a b 4_d 3_c\rangle u_{5_e} + \langle \hat{2}_b 5_e 4_d 3_c\rangle u_{\hat{1}_a})\\ &\times([\hat{2}_{\dot{b}}\hat{1}_{\dot{a}}4_{\dot{d}}3_{\dot{c}}]\tilde{u}_{5_{\dot{e}}} + [\hat{2}_{\dot{b}}5_{\dot{e}}4_{\dot{d}}3_{\dot{c}}]\tilde{u}_{\hat{1}_{\dot{a}}})\end{aligned} \tag{A.5.5}$$

Note that Section V G. on pg 14 [49] comments that the [55] expression was found through BCFW recursion and repeated use of the 5-point Shouten identity, but he finds it convenient to use an intermediate form before the use of the Shouten identity:

$$
\begin{aligned}
A_5^0(1^g_{a\dot{a}}, 2^g_{b\dot{b}}, 3^g_{c\dot{c}}, 4^g_{d\dot{d}}, 5^g_{e\dot{e}}) = & \frac{-i}{s_{12}s_{23}^2 s_{34}s_{45}s_{15}} \\
& \times \Big(\langle 1_a 2_b 3_c 4_d \rangle [1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}}] \\
& \times (s_{14} \langle 5_e | \not{p}_1 \not{p}_3 \not{p}_2 \not{p}_4 | 5_{\dot{e}}] - s_{13} \langle 5_e | \not{p}_1 \not{p}_4 \not{p}_2 \not{p}_4 | 5_{\dot{e}}] \\
& \quad + s_{12} \langle 5_e | \not{p}_1 \not{p}_4 \not{p}_3 \not{p}_4 | 5_{\dot{e}}]) \\
& - \langle 1_a 2_b 3_c 4_d \rangle [1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 5_{\dot{e}}] \\
& \times (s_{14} \langle 5_e | \not{p}_1 \not{p}_3 \not{p}_2 \not{p}_5 | 4_{\dot{d}}] - s_{13} \langle 5_e | \not{p}_1 \not{p}_4 \not{p}_2 \not{p}_5 | 4_{\dot{d}}] \\
& \quad + s_{12} \langle 5_e | \not{p}_1 \not{p}_4 \not{p}_3 \not{p}_5 | 4_{\dot{d}}]) \\
& + \langle 1_a 2_b 3_c 5_e \rangle [1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 5_{\dot{e}}] \\
& \times (s_{14} \langle 4_d | \not{p}_5 \not{p}_1 \not{p}_2 \not{p}_3 | 5_{\dot{e}}] - s_{13} \langle 4_d | \not{p}_5 \not{p}_1 \not{p}_2 \not{p}_4 | 5_{\dot{e}}] \\
& \quad + s_{12} \langle 4_d | \not{p}_5 \not{p}_1 \not{p}_3 \not{p}_4 | 5_{\dot{e}}]) \\
& + s_{15} \langle 1_a 2_b 3_c 5_e \rangle [1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 5_{\dot{e}}] \langle 4_d | \not{p}_5 \not{p}_1 \not{p}_2 \not{p}_3 | 4_{\dot{d}}] \\
& - s_{45} \langle 2_b 3_c 4_d 5_e \rangle [2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}} 5_{\dot{e}}] \langle 1_a | \not{p}_5 \not{p}_4 \not{p}_3 \not{p}_2 | 1_{\dot{a}}] \\
& + s_{45} \langle 2_b 3_c 4_d 5_e \rangle [1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}}] \langle 1_a | \not{p}_2 \not{p}_3 \not{p}_4 \not{p}_1 | 5_{\dot{e}}] \\
& + s_{45} \langle 2_b 3_c 4_d 5_e \rangle [1_{\dot{a}} 2_{\dot{b}} 3_{\dot{c}} 5_{\dot{5}}] \langle 1_a | \not{p}_5 \not{p}_1 \not{p}_2 \not{p}_3 | 4_{\dot{d}}] \\
& - s_{45} \langle 1_a 2_b 3_c 4_d \rangle [2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}} 5_{\dot{5}}] \langle 5_e | \not{p}_1 \not{p}_4 \not{p}_3 \not{p}_2 | 1_{\dot{a}}] \\
& - s_{45} \langle 1_a 2_b 3_c 5_e \rangle [2_{\dot{b}} 3_{\dot{c}} 4_{\dot{d}} 5_{\dot{5}}] \langle 4_d | \not{p}_3 \not{p}_2 \not{p}_1 \not{p}_5 | 1_{\dot{a}}] \Big)
\end{aligned}
\tag{A.5.6}
$$

A.6 4D analytic amplitude expressions available for use in testing

For complex momenta:

$$
A_3^{\text{tree}}(1_g^-, 2_g^-, 3_g^+) = i \frac{\langle 12 \rangle^4}{\langle 12 \rangle \langle 23 \rangle \langle 31 \rangle} \tag{A.6.1}
$$

$$
A_3^{\text{tree}}(1_g^+, 2_g^+, 3_g^-) = -i \frac{[12]^4}{[12][23][31]} \tag{A.6.2}
$$

For all $n \geq 4$:

$$
A_n^{\text{tree}}(1_g^\pm, 2_g^+, ..., n_g^+) = A_n^{\text{tree}}(1_g^\pm, 2_g^-, ..., n_g^-) = 0 \tag{A.6.3}
$$

$$A_n^{\text{tree}}(1_g^+, 2_g^+, ...i_g^-, ..., j_g^-, ...n_g^+) = i\frac{\langle ij\rangle^4}{\langle 12\rangle\langle 23\rangle...\langle n1\rangle} \tag{A.6.4}$$

With quarks:

$$A_n^{\text{tree}}(1_q^-, 2_q^+, ...i_g^-, ..., n_g^+) = i\frac{\langle 1i\rangle^2\langle 2i\rangle}{\langle 12\rangle\langle 23\rangle...\langle n1\rangle} \tag{A.6.5}$$

A.7 6D amplitudes with quarks for future work

3-point - 2 quarks, 1 gluon

Source: [56] Appendix B:

$$A^0(1_a^q, 2_b^q, 3_{c\dot{c}}^g) = \frac{i}{s_{r3}}\langle 1_a 2_b 3_c r_x\rangle\langle r_x|3_{\dot{c}}] \tag{A.7.1}$$

where r is a massless reference vector satisfying $s_{r3} \neq 0$

4-point - 2 quarks, 2 gluons

Source: [45] Section III C:

$$A_4^0(1_{a\dot{a}}^g, 2_{b\dot{b}}^g, 3_c^q, 4_d^q) = \frac{-i}{s_{12}s_{23}}\langle 1_a 2_b 3_c 4_d\rangle[1_{\dot{a}} 2_{\dot{b}} 3_{\dot{e}} 3^{\dot{e}}] \tag{A.7.2}$$

where the repeated index $\dot{e}$ is summed.

Source: [56] Appendix B:

$$A_4^0(1_{a\dot{a}}^g, 2_{b\dot{b}}^g, 3_c^q, 4_d^q) = \frac{-i}{2s_{12}s_{23}}\langle 1_a 2_b 3_c 4_d\rangle[1_{\dot{a}} 2_{\dot{b}} 3_{\dot{e}} 3^{\dot{e}}] \tag{A.7.3}$$

3-point - 2 quarks, 1 scalar

Source: [56] Appendix B:

$$\begin{aligned} A^0(1^{\phi_1}, 2_b^q, 3_c^q) &= \frac{i}{\sqrt{2}}\langle 1_a|2_{\dot{b}}] \\ A^0(1^{\phi_2}, 2_b^q, 3_c^q) &= \frac{i}{\sqrt{2}}\langle 1_a|\gamma_5|2_{\dot{b}}] \end{aligned} \tag{A.7.4}$$

4-point - 2 quarks, 2 scalars

Source: [49]:

$$A_4^0(1^s, 2^s, 3_c^q, 4_d^q) = \frac{i(s_{13} - s_{23})}{4s_{12}s_{23}}\langle 1_a 1^a 3_c 4_d\rangle \tag{A.7.5}$$

4-point - 2 quarks, 1 gluon, 1 scalar

Source: [49]:

$$A_4^0(1_{a\dot{a}}^g, 2_b^q, 3^s, 4_d^q) = -\frac{i}{4\sqrt{2}s_{12}s_{23}}\langle 1_a 2_b 3_c 3^c\rangle[1_{\dot{a}} 3^{\dot{c}} 3_{\dot{c}} 4_{\dot{d}}] \tag{A.7.6}$$

Appendix B

Examples of cuts: all-gluon loop

This appendix includes a sketch of the cuts for the all-gluon loop. Collection of the full family of cuts from existing sources is available in Armstrong [8] with code LoopCuts.

12345
12354
12435
12453
12534
12543
13245
13254
13425
13452
13524
13542
14235
14253
14325
14352
14523
14532
15234
15243
15324
15342
15423
15432
21345
21354
21435
21453
21534
21543
23145
23154
23415
23451
23514
23541
24135
24153
24315
24351
24513
24531
25134
25143
25314
25341
25413
25431
31245
31254
31425
31452
31524
31542
32145
32154
32415
32451
32514
32541
34125
34152
34215
34251
34512
34521
35124
35142
35214
35241
35412
35421
41235
41253
41325
41352
41523
41532
42135
42153
42315
42351
42513
42531
43125
43152
43215
43251
43512
43521
45123
45132
45213
45231
45312
45321
51234
51243
51324
51342
51423
51432
52134
52143
52314
52341
52413
52431
53124
53142
53214
53241
53412
53421
54123
54132
54213
54231
54312
54321

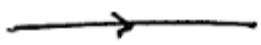
24
25
34
35
45
145
234
235
245
345

Appendix C

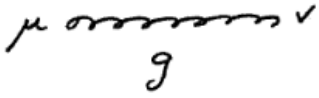
Colour-ordered Feynman rules

Possible vertices with gluons are defined by the Lagrangian, $\mathcal{L}_H^{\text{int}} = \frac{C}{2} H \text{tr} G_{\mu\nu} G^{\mu\nu}$. This generates vertices involving two, three or four gluons.

The full set of rules required for the calculation is [12], [61], [49]:

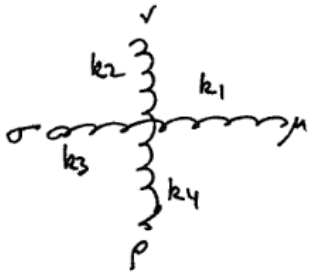


$\frac{i}{\not{p}-m_\psi+i0}$

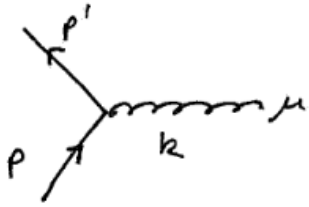


$\frac{i}{k^2+i0}\left(g^{\mu\nu}+(\xi-1)\frac{k^{mu}k^{nu}}{k^2}\right)$

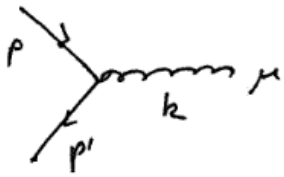
$-\frac{i}{\sqrt{2}}\Big(g^{\mu\nu}(k_1-k_2)^\sigma+g^{\nu\sigma}(k_2-k_3)^\mu+g^{\sigma\mu}(k_3-k_1)^\nu\Big)$



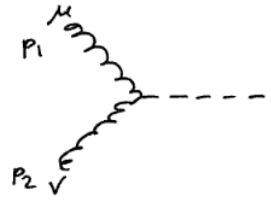
$ig^{\mu\sigma}g^{\nu\rho}-\frac{i}{2}(g^{\mu\nu}g^{\rho\sigma}+g^{\mu\rho}g^{\sigma\nu})$



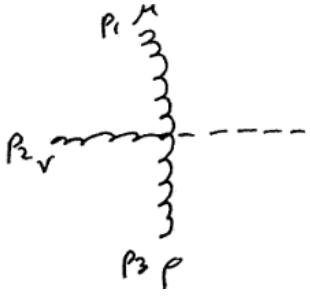
$\frac{i}{\sqrt{2}}\gamma^\mu$



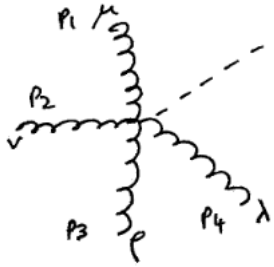
$-\frac{i}{\sqrt{2}}\gamma^\mu$



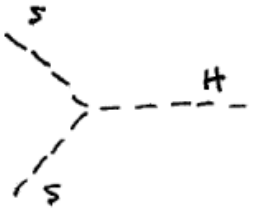
$$i\mathrm{A}\delta^{ab}(g^{\mu\nu}p_1p_2 - p_1^\nu p_2^\mu)$$



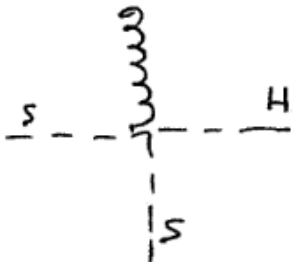
$$-\mathrm{A}g\Big[(p_1-p_2)^\rho g^{\mu\nu} + (p_2-p_3)^\mu g^{\nu\rho} + (p_3-p_1)^\nu g^{\rho\mu}\Big]$$



$$-\mathrm{A}g^2\Big[(g^{\mu\rho}g^{\nu\sigma} - g^{\mu\sigma}g^{\nu\rho}) + (g^{\mu\nu}g^{\rho\sigma} - g^{\mu\sigma}g^{\nu\rho}) + (g^{\mu\nu}g^{\rho\sigma} - g^{\mu\rho}g^{\nu\sigma})\Big]$$



$$-2ik_1.k_2 = -is_{12}$$



$$-i\sqrt{2}(k_1 - k_2)^\mu$$

Appendix D

Lie algebra

D.1 Introduction

We examine here the impact of transformations on the 6-D spinor representation used in our calculations for two reasons:

- In Section 4 we have chosen a specific representation of six-dimensional spinors because it reduces to the familiar four-dimensional spinor representation when momenta are four-dimensional. We wish to demonstrate that the chosen spinor representation is, in fact, Lorentz invariant.
- For practical purposes our suite of test momenta, from randrot in Appendix F, have been generated by rotation from four dimensions into six dimensions. If we simply apply the normal prescription to generate the six-dimensional spinors from these transformed momenta sets the two columns of the resulting spinors emerge with a non-trivial distortion: the two columns are not independently Lorentz invariant. In order to test our amplitude calculations we need to be able to generate the six-dimensional spinors from rotated momenta so that they retain Lorentz invariance and can be used to compare against the results of calculations using the original four-dimensional momenta values.

We do this by examining the associated Lie algebras for the rotations and boosts sub groups of the Lorentz group.

D.2 Background

For continuous (Lie) groups it is useful to consider the infinitesimal generators of the group, which form a structure known as a Lie algebra (pg 96) [232].

$SO(2)$ is the group of proper rotations in two dimensions, all abut the same axis, which we take as the z axis.

$SO(3)(SU(2))$ is the group of all proper rotations in three dimensions. (pg 101) [232]

The group is specified by its law of composition: how two rotations combine to make a third. The structure formed by the infinitesimal generators X_i is known as an algebra, which is in the first place a vector space since (complex) linear combinations of the X_i are again generators. there is, however, an additional composition law given by commutation: for any two generators X, Y the commutator $[X, Y]$ is also a generator.

The Lie algebra of the group is the commutator relation $[X_a, X_b] = if_{abc}X_c$ for some constants f_{abc}. The commutator in the algebra plays a role similar to the multiplication law for the group. (page 47) [233]

Gran-Schmidt procedure

Given any basis $\mathbf{f}_i$ of V it is possible to construct an orthonormal basis $\mathbf{e}_i$ satisfying

$$(\mathbf{e}_i, \mathbf{e}_j) = \delta_{ij} \tag{D.2.1}$$

by the 'Gran-Schmidt' procedure as follows:

- First construct $\mathbf{e}_1$ by normalising f_1

$$\mathbf{e}_1 = \mathbf{f}_1/||\mathbf{f}_1|| \tag{D.2.2}$$

- Then construct $\mathbf{e}_2$ from $\mathbf{f}_2$ by subtracting its 'component' along $\mathbf{e}_1$ to make it orthogonal to $\mathbf{e}_1$ and then normalising

$$\mathbf{e}_2 = (\mathbf{f}_2 - (\mathbf{e}_1, \mathbf{f}_2)e_1)/||...|| \tag{D.2.3}$$

- The next vector $\mathbf{e}_3$ is constructed from $\mathbf{f}_3$ by subtracting off its components along both $\mathbf{e}_1$ and $\mathbf{e}_2$ and then normalising, and so on.

D.3 Building the Lie algebras

The python implementation follows in the form of a Jupyter notebook.

In [1]:
```
import numpy as np
import sys
sys.path.append('./tests/')
import scipy.linalg
from scipy.linalg import det

import randrot as rr
import spinors as sp
from utils import *
```

Building the lie algebra for rotations

We define matrices for the Lie algebra of rotations in the $SU(4)$ representation:

$$\sigma^{ij}^A_B = \frac14\left(\tilde\gamma^{i}^{AC}\gamma^{j}_{CD} - \tilde\gamma^j^{AC}\gamma^{i}_{CD}\right)$$

$$\tilde\sigma^{ij}_A^B = \frac14\left(\gamma^{i}_{AC}\tilde\gamma^{j}^{CD} - \gamma^j_{AC}\tilde\gamma^{i}^{CD}\right)$$

In [2]:
```
def sigma(mu,nu):
    return 0.25 * 1* (np.dot(sigt6[mu][0], sig6[nu][0]) -  np.dot(sigt6[nu][0], sig6[mu][0]))
```

In [3]:
```
def sigmat(mu,nu):
    return 0.25 * 1 *(np.dot(sig6[mu][0], sigt6[nu][0]) -  np.dot(sig6[nu][0], sigt6[mu][0]))
```

The elements of the algebra are labelled by two indices, the two axis that will get rotated. It is a 10-dimensional.

In [4]:
```
sindices = [ (i,j) for i in range(1,6) for j in range(1,6) if i<j]
```

In [5]:
```
# an alternative definition of sigma is through e^{ijklmn}g_k g_l g_m g_n
def osigma(i,j):
    ret = np.zeros((4,4), dtype='complex128')
    for k in range(6):
        for l in range(6):
            for m in range(6):
                for n in range(6):
                    sign = (permutationSign([0,1,2,3,4,5],[i,j,k,l,m,n]))
                    ret +=  sign * np.dot(gammast[k],np.dot(gammas[l],np.dot(gammast[m],gammas[n])))
    return ret/48

for i, j in sindices:
    #print ("sigma and osigma(",i,",",j,")")
    #print (sigma(i,j))
    #print ("osigma(i,j)")
    #print(osigma(i,j))
    assert np.allclose(sigma(i,j),osigma(i,j))
```

This is the algebra for the fundamental representation of rotations in R^5.

In [6]:
```
def w(i,j):
    ret = np.zeros((5,5))
    if i == j :
        return ret
    ret[i, j] = -1
    ret[j, i] = 1
    return ret
```

In [7]:
```
#for i in range(5):
#    for j in range(5):
#        print("indices ", i,j)
#        print(w(i,j))
```

The sigma and sigma tilde matrices are sort of orthonormal to each other, so we can use them to project linear combinations onto components.

In [8]:
```
#print (('{:7} ' *11).format('   ..... ', *sindices))
print (('{:} ' *11).format('   ..... ', *sindices))
for i in sindices:
    row = []
    for j in sindices:
        row.append(np.trace(np.dot(sigmat(*i).T, sigma(*j))))
    #print (('{:7} '*11).format(i,*row))
    print (('{:} '*11).format(i,*row))
```

```
  .....  (1, 2) (1, 3) (1, 4) (1, 5) (2, 3) (2, 4) (2, 5) (3, 4) (3, 5) (4, 5)
(1, 2) (1+0j) 0j 0j 0j 0j 0j 0j 0j 0j 0j
(1, 3) 0j (1+0j) 0j 0j 0j 0j 0j 0j 0j 0j
(1, 4) 0j 0j (1+0j) 0j 0j 0j 0j 0j 0j 0j
(1, 5) 0j 0j 0j (1+0j) 0j 0j 0j 0j 0j 0j
(2, 3) 0j 0j 0j 0j (1+0j) 0j 0j 0j 0j 0j
(2, 4) 0j 0j 0j 0j 0j (1+0j) 0j 0j 0j 0j
(2, 5) 0j 0j 0j 0j 0j 0j (1+0j) 0j 0j 0j
(3, 4) 0j 0j 0j 0j 0j 0j 0j (1+0j) 0j 0j
(3, 5) 0j 0j 0j 0j 0j 0j 0j 0j (1+0j) 0j
(4, 5) 0j 0j 0j 0j 0j 0j 0j 0j 0j (1+0j)
```

In [9]:
```
windices = [ (i,j) for i in range(5) for j in range(5) if i<j]
#print (('{:7} '*11).format('   ..... ', *windices))
print (('{:} '*11).format('   ..... ', *windices))
for i in windices:
    row = []
    for j in windices:
        row.append(np.trace(np.dot(w(*i).T, w(*j))))
    #print (('{:7} '*11).format(i,*row))
    print (('{:} '*11).format(i,*row))
```

```
  .......  (0, 1) (0, 2) (0, 3) (0, 4) (1, 2) (1, 3) (1, 4) (2, 3) (2, 4) (3, 4)
(0, 1) 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
(0, 2) 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
(0, 3) 0.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
(0, 4) 0.0 0.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0
(1, 2) 0.0 0.0 0.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0
(1, 3) 0.0 0.0 0.0 0.0 0.0 2.0 0.0 0.0 0.0 0.0
(1, 4) 0.0 0.0 0.0 0.0 0.0 0.0 2.0 0.0 0.0 0.0
(2, 3) 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0 0.0 0.0
(2, 4) 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0 0.0
(3, 4) 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0
```

In [10]:
```python
def projSigma(m):
    ret = [ 1 * np.trace(np.dot(m, sigmat(*ss).T)) for ss in sindices]
    return np.array(ret)

def projSigmat(m):
    ret = [1 * np.trace(np.dot(m, sigma(*ss).T)) for ss in sindices]
    return np.array(ret)

for ind, i in enumerate(sindices):
    target = np.zeros(10)
    target[ind] = 1
    assert np.allclose(projSigma(sigma(*i)), target)

for ind, i in enumerate(sindices):
    target = np.zeros(10)
    target[ind] = 1
    assert np.allclose(projSigmat(sigmat(*i)), target)
```

In [11]:
```python
#for i in sindices:
#    row = []
#    for j in sindices:
#        c,s = sigmaComm(i,j)
#        print("i ", i, "j ", j)
#        print("c", c)
#        print("s", s)
```

In [12]:
```python
def projW(m):
    indices = [ (i,j) for i in range(5) for j in range(5) if i<j]
    ret = [0.5* np.trace(np.dot(m, w(*ss).T)) for ss in windices]
    return np.array(ret)

for ind, i in enumerate(windices):
    target = np.zeros(10)
    target[ind] = 1
    assert np.allclose(projW(w(*i)), target)
```

In [13]:
```python
def sigmaComm(i,j):
    pro = projSigma(comm(sigma(*i),sigma(*j)))
    nz = np.nonzero(pro)
    assert len(nz)<=1
    if len(nz[0]) == 0:
        return 0.0, None
    #print nz
    c = pro[nz[0]][0]
    assert c.imag == 0
    return c.real,sindices[nz[0][0]]

def sigmatComm(i,j):
    pro = projSigmat(comm(sigmat(*i),sigmat(*j)))
    nz = np.nonzero(pro)
    assert len(nz)<=1
    if len(nz[0]) == 0:
        return 0.0, None
    #print nz
    c = pro[nz[0]][0]
    assert c.imag == 0
    return c.real,sindices[nz[0][0]]
```

In [14]:
```python
def wComm(i,j):
    pro = projW(comm(w(*i),w(*j)))
    nz = np.nonzero(pro)
    assert len(nz)<=1
    if len(nz[0]) == 0:
        return 0.0, None
    #print nz
    c = pro[nz[0]][0]
    assert c.imag == 0
    return c.real,windices[nz[0][0]]
```

This checks the commutation relations in the two albebras are the same:

In [15]:
```python
print('='*10, ' sigma ', '='*10)
#print (('{:7} '*11).format('....... ', *sindices))
print (('{:} '*11).format('....... ', *sindices))

for i in sindices:
    row = []
    for j in sindices:
        c, s = sigmaComm(i,j)
        if s is None:
            row.append('.')
        else:
            row.append('{} s_{}{}'.format(int(c),*s))
    #print (('{:7} '*11).format(i,*row))
    print (('{:} '*11).format(i,*row))

print('='*10, ' sigmat ', '='*10)
#print (('{:7} '*11).format('....... ', *sindices))
print (('{:} '*11).format('....... ', *sindices))
for i in sindices:
    row = []
    for j in sindices:
        c, s = sigmatComm(i,j)
        if s is None:
            row.append('.')
```

```
            else:
                row.append('{} s_{}{}'.format(int(c),*s))
        #print (('{:7} '*11).format(i,*row))
        print (('{:} '*11).format(i,*row))
print('='*10, ' w ', '='*10)

#print (('{:7} '*11).format('....... ', *windices))
print (('{:} '*11).format('....... ', *windices))
for i in windices:
    row = []
    for j in windices:
        c, s = wComm(i,j)
        if s is None:
            row.append('.')
        else:
            row.append('{} s_{}{}'.format(int(c),*s))
    #print (('{:7} '*11).format(i,*row))
    print (('{:} '*11).format(i,*row))
```

```
==========  sigma  ==========
.......  (1, 2) (1, 3) (1, 4) (1, 5) (2, 3) (2, 4) (2, 5) (3, 4) (3, 5) (4, 5)
(1, 2) . 1 s_23 1 s_24 1 s_25 -1 s_13 -1 s_14 -1 s_15 . . .
(1, 3) -1 s_23 . 1 s_34 1 s_35 1 s_12 . . -1 s_14 -1 s_15 .
(1, 4) -1 s_24 -1 s_34 . 1 s_45 . 1 s_12 . 1 s_13 . -1 s_15
(1, 5) -1 s_25 -1 s_35 -1 s_45 . . . 1 s_12 . 1 s_13 1 s_14
(2, 3) 1 s_13 -1 s_12 . . . 1 s_34 1 s_35 -1 s_24 -1 s_25 .
(2, 4) 1 s_14 . -1 s_12 . -1 s_34 . 1 s_45 1 s_23 . -1 s_25
(2, 5) 1 s_15 . . -1 s_12 -1 s_35 -1 s_45 . . 1 s_23 1 s_24
(3, 4) . 1 s_14 -1 s_13 . 1 s_24 -1 s_23 . . 1 s_45 -1 s_35
(3, 5) . 1 s_15 . -1 s_13 1 s_25 . -1 s_23 -1 s_45 . 1 s_34
(4, 5) . . 1 s_15 -1 s_14 . 1 s_25 -1 s_24 1 s_35 -1 s_34 .
==========  sigmat  ==========
.......  (1, 2) (1, 3) (1, 4) (1, 5) (2, 3) (2, 4) (2, 5) (3, 4) (3, 5) (4, 5)
(1, 2) . 1 s_23 1 s_24 1 s_25 -1 s_13 -1 s_14 -1 s_15 . . .
(1, 3) -1 s_23 . 1 s_34 1 s_35 1 s_12 . . -1 s_14 -1 s_15 .
(1, 4) -1 s_24 -1 s_34 . 1 s_45 . 1 s_12 . 1 s_13 . -1 s_15
(1, 5) -1 s_25 -1 s_35 -1 s_45 . . . 1 s_12 . 1 s_13 1 s_14
(2, 3) 1 s_13 -1 s_12 . . . 1 s_34 1 s_35 -1 s_24 -1 s_25 .
(2, 4) 1 s_14 . -1 s_12 . -1 s_34 . 1 s_45 1 s_23 . -1 s_25
(2, 5) 1 s_15 . . -1 s_12 -1 s_35 -1 s_45 . . 1 s_23 1 s_24
(3, 4) . 1 s_14 -1 s_13 . 1 s_24 -1 s_23 . . 1 s_45 -1 s_35
(3, 5) . 1 s_15 . -1 s_13 1 s_25 . -1 s_23 -1 s_45 . 1 s_34
(4, 5) . . 1 s_15 -1 s_14 . 1 s_25 -1 s_24 1 s_35 -1 s_34 .
==========  w  ==========
.......  (0, 1) (0, 2) (0, 3) (0, 4) (1, 2) (1, 3) (1, 4) (2, 3) (2, 4) (3, 4)
(0, 1) . 1 s_12 1 s_13 1 s_14 -1 s_02 -1 s_03 -1 s_04 . . .
(0, 2) -1 s_12 . 1 s_23 1 s_24 1 s_01 . . -1 s_03 -1 s_04 .
(0, 3) -1 s_13 -1 s_23 . 1 s_34 . 1 s_01 . 1 s_02 . -1 s_04
(0, 4) -1 s_14 -1 s_24 -1 s_34 . . . 1 s_01 . 1 s_02 1 s_03
(1, 2) 1 s_02 -1 s_01 . . . 1 s_23 1 s_24 -1 s_13 -1 s_14 .
(1, 3) 1 s_03 . -1 s_01 . -1 s_23 . 1 s_34 1 s_12 . -1 s_14
(1, 4) 1 s_04 . . -1 s_01 -1 s_24 -1 s_34 . . 1 s_12 1 s_13
(2, 3) . 1 s_03 -1 s_02 . 1 s_13 -1 s_12 . . 1 s_34 -1 s_24
(2, 4) . 1 s_04 . -1 s_02 1 s_14 . -1 s_12 -1 s_34 . 1 s_23
(3, 4) . . 1 s_04 -1 s_03 . 1 s_14 -1 s_13 1 s_24 -1 s_23 .
```

Here we check that the commutation relations match what we expect for $SO(5)$

In [16]:
```
def delta(i,j):
    if i == j:
        return 1
    else:
        return 0


def predictedComm(i, j, matrixFn):
    ret = np.zeros_like(matrixFn(1,2))
    ret += delta(i[0], j[0]) * matrixFn(i[1], j[1])
    ret -= delta(i[0], j[1]) * matrixFn(i[1], j[0])
    ret -= delta(i[1], j[0]) * matrixFn(i[0], j[1])
    ret += delta(i[1], j[1]) * matrixFn(i[0], j[0])
    return ret
```

In [17]:
```
for i in windices:
    for j in windices:
        assert np.allclose(predictedComm(i, j, w), comm(w(*i),w(*j)))
```

In [18]:
```
for i in sindices:
    for j in sindices:
        assert np.allclose(predictedComm(i, j, sigma) , comm(sigma(*i),sigma(*j)))
for i in sindices:
    for j in sindices:
        assert np.allclose(predictedComm(i, j, sigmat) , comm(sigmat(*i),sigmat(*j)))
```

In [19]:
```
def makeLorenzRotation(params):
    ws = np.array([w(*i) for i in windices])
    L = np.sum(np.multiply(params[:,np.newaxis, np.newaxis] , ws) , axis =0)
    R = scipy.linalg.expm(L)
    return R
```

In [20]:
```
def makeSpinorRotations(params):
    sigmas = np.array([sigma(*i) for i in sindices])
    L = np.sum(np.multiply(params[:,np.newaxis, np.newaxis] , sigmas) , axis =0)
    U = scipy.linalg.expm(L)
    sigmats = np.array([sigmat(*i) for i in sindices])
    L = np.sum(np.multiply(params[:,np.newaxis, np.newaxis] , sigmats) , axis =0)
    Ut = scipy.linalg.expm(L)
    return U, Ut
```

Building lieAlgebra for boosts

We define boost transformation matrices for i = 1 to 5

In [21]:
```
# Define generators
def boostGen(i):
    # i is index for boost
```

```
    Y = np.zeros([6,6], dtype = complex)
    Y[0,i] = -1j
    Y[i,0] = -1j
    return Y
```

In [22]:
```
def boostMatrix(chi, i):
    # chi is boost
    # i is index for boost
    B = np.eye(6)
    B[0,0] = np.cosh(chi)
    B[i,i] = np.cosh(chi)
    B[0,i] = -np.sinh(chi)
    B[i,0] = -np.sinh(chi)
    return B
```

In [23]:
```
def lorentzBoost(p,chi,i):
    # p is momentum to be boosted
    # chi is boost
    # i is index for boost
    ret = np.zeros(6)
    B = boostMatrix(chi,i)
    ret += np.dot(B,p.T)
    return ret
```

We expect commutator for boosts on i and j axes to be a rotation around the same axes: $$ [Y_i,Y_j,] = W_{ij} $$ It is necessary to account for the shift in indices: Y is 6 x 6 with indices 0 to 5. Excluding the i and j = 0 means we use only indices 1 to 5. We map to W which are 5 x 5 matrices with indices 0 to 5.

In [24]:
```
# Define the boost commutator
def boostComm(i,j):
    Yi = boostGen(i)
    Yj = boostGen(j)
    return comm(Yi,Yj)
```

In [25]:
```
# Test commutation relation gives expected result
for i in range(5):
    for j in range(5):
        Ycomm = boostComm(i,j)
        Wij = w(i-1,j-1)
        for a in range(1,5):
            for b in range(1,5):
                assert Ycomm[a,b] == Wij[a-1,b-1]
```

Transformations

In [26]:
```
p = rr.getRandom6Momentum(1235)
q = refq(p)
```

In [27]:
```
params = np.zeros(10)
params[0] = np.pi/2

params = rr.randomDirection(10)

R = makeLorenzRotation(params)
assert np.allclose(np.dot(R, R.T), np.eye(5))
assert np.isclose(np.linalg.det(R), 1.0)

(prot, qrot) = rr.rotate6D([p,q], rotation=R)

U, Ut = makeSpinorRotations(params)
assert np.allclose(np.dot(U, U.T.conj()), np.eye(4))
assert np.isclose(np.linalg.det(U), 1.0)

assert np.allclose(np.dot(Ut, Ut.T.conj()), np.eye(4))
assert np.isclose(np.linalg.det(U), 1.0)
```

In [28]:
```
gammas_rot = np.array([np.dot(Ut, np.dot(g, U.T.conj())) for g in gammas])
gammast_rot = np.array([np.dot(U, np.dot(g, Ut.T.conj())) for g in gammast])
```

In [29]:
```
p_contracted = p*np.array([1,-1,-1,-1,-1,-1])
pslash_rot = np.array([pp* gg for pp,gg in zip(p_contracted, gammas_rot)])
pslash_rot = np.sum(pslash_rot ,axis=0)
pslash_rot = slashedMomentum(p, gammas_rot)
pslasht_rot = slashedMomentum(p, gammast_rot)
```

Contracting

- the original p with rotated gamma matrices
- the rotated p with the original matrices

is equivalent

In [30]:
```
assert np.allclose(pslash_rot,slash(prot))
assert np.allclose(pslasht_rot,slasht(prot))
```

Here I check that the rotated spinor is a solution of the rotated dirac equation, but they are not the same as basis as the building the spinor from the rotated momentum.

In [31]:
```
s = sp.Spinor6(p).sp6d()
st = sp.Spinor6(p).sp6td()
s_rot = sp.Spinor6(prot).sp6d()
```

In [32]:
```
assert np.allclose(np.dot(pslash_rot,s_rot), np.zeros((4,2)))
```

In [33]:
```
assert np.allclose(np.dot(pslash_rot, np.dot(U,s)), np.zeros((4,2)))
```

In [34]:
```
np.allclose(np.dot(U,s), s_rot)
```

Out[34]:
```
False
```

Using these transformation rules the spinor products are truly invariant:

In [35]:
```
for i in range(20):
    test_params = rr.randomDirection(10)
    test_U, test_Ut = makeSpinorRotations(test_params)
    test_p1, test_p2 = rr.getRandom6Momenta(2)
    sprod_of_rotated_spinors = sp.sp62_all(
        test_U.dot(sp.Spinor6(test_p1).sp6d()),
        test_Ut.dot(sp.Spinor6(test_p2).sp6td())
        )
    orig_sprod = sp.sp6_dd(test_p1, test_p2)
    assert np.allclose(sprod_of_rotated_spinors, orig_sprod)
```

The spinor product transforms from

$$\left\langle \lambda\mu\right]_{a\dot b} = \lambda_a^A \tilde\mu_{A,\dot b}$$

to

$$\left\langle \lambda'\mu'\right]_{a\dot b} = \lambda'_a^A \tilde\mu'_{A,\dot b} = U^A_B\lambda_a^B \tilde U_A^C \tilde\mu_{C,\dot b} = U^A_B \tilde U_A^C \;\lambda_a^B \tilde\mu_{C,\dot b}$$

so we see that we must have the property

$$U^A_B \tilde U_A^C = \delta_B^C$$

In [36]:
```
assert np.allclose(Ut.T.dot(U), np.eye(4))
```

This property has an equivalent condition on the Lie algebra:

$$\left(\tilde \sigma^{ij}\right)^T = - \sigma^{ij}$$

In [37]:
```
for i,j in sindices:
    assert np.allclose(sigmat(i,j).T, - sigma(i,j))
```

Reconstruction of the momentum from the spinors

In [38]:
```
def pFromS(s):
    res= np.zeros(6, dtype='complex128')
    for i in range(6):
        prod = (np.dot(s.T, np.dot(gammas[i], s)))
        component = -0.25*(prod[1,0]-prod[0,1])
        assert np.isclose(prod[1,0], -prod[0,1])
        res[i] = component
    return res

def pFromSt(st):
    res= np.zeros(6, dtype='complex128')
    for i in range(6):
        prod = (np.dot(st.T, np.dot(gammast[i], st)))
        component = -0.25*(prod[0,1]-prod[1,0])
        res[i] = component
    return res

assert np.allclose(pFromS(s), p)
assert np.allclose(pFromSt(st), p)

#assert np.allclose(pFromS(us),prot)
#assert np.allclose(pFromSt(ust),prot)
```

Covariant definition of the basis for 6D spinors

We use

$$C_4 = i \gamma^0\gamma^1\gamma^2\gamma^3$$

The definition of the spinors is such that

$$\not q |s\rangle$$

is an eigenvector of C_4 with eigenvalue ∓ 1 for the first and second column respectively. We see that below:

In [39]:
```
c4 = 1j * gammas[0].dot(gammast[1]).dot(gammas[2]).dot(gammast[3])
q = np.array(refq(p))
qs = np.dot(slash(q),s)
proj = np.dot(c4,qs)
qs_rec = reciprocal(qs)
chopComplex(np.dot(qs_rec, proj))
```

Out[39]:
```
array([[-1.+0.j,  0.+0.j],
       [ 0.+0.j,  1.+0.j]])
```

This is also the case for the spinor constructed for the rotated momentum:

In [40]:
```
qs_rot = np.dot(slash(q),s_rot)
proj_rot = np.dot(c4,qs_rot)
s_rot_rec = reciprocal(qs_rot)
chop(np.dot(s_rot_rec, proj_rot ))
```

Out[40]:
```
array([[-1.+0.j,  0.+0.j],
       [ 0.+0.j,  1.+0.j]])
```

For the transformed spinor $U|s\rangle$ the columns are not eigenvectors of C_4:

In [41]:
```
us = np.dot(U, s)
qus = np.dot(slash(q),us)
proj_rot = np.dot(c4,qus)
qus_rec = reciprocal(qus)
S = chopComplex(np.dot(qus_rec, proj_rot ))
S,det(S)
```

Out[41]:
```
(array([[-0.49281518+0.j        , -0.17968734-0.85137868j],
        [-0.17968734+0.85137868j,  0.49281518+0.j        ]]),
 (-1-2.7755575615628914e-17j))
```

But the determinant is -1 so it is consistent with having eigenvalues 1 and -1.

We define projectors onto the positive and negative eigenvectors of C_4:

In [42]:
```
Pplus = 0.5*(np.eye(4)+c4)
Pminus = 0.5*(np.eye(4)-c4)
```

In [43]:
```
assert np.allclose(Pplus.dot(Pplus), Pplus)
assert np.allclose(Pminus.dot(Pminus), Pminus)
assert np.allclose(Pminus + Pplus, np.eye(4))
```

In [44]:
```
c4_rot = 1j * gammas_rot[0].dot(gammast_rot[1]).dot(gammas_rot[2]).dot(gammast_rot[3])
```

General construction

Here I construct the spinors from general principle: as solution of hte dirac equation and the labelling of the first and second vector is given by the eigenvalues of an operator. This fixes the columns up to a phase per column.

In [45]:
```
def getSpinors(p, q, Pp, Pm):
    assert np.allclose(Pp + Pm, np.eye(4))
    assert np.allclose(np.dot(Pp, Pp), Pp)
    assert np.allclose(np.dot(Pm, Pm), Pm)
    assert np.allclose(np.dot(Pm, Pp), np.zeros_like(Pp))
    assert np.allclose(np.dot(Pp, Pm), np.zeros_like(Pp))
    ev, vs = np.linalg.eig(slash(p))
    sols = vs.T[np.isclose(ev, 0.0)]
    # check the solutions are solutions...
    assert np.allclose(chopComplex(slash(p).dot(sols.T)), np.zeros_like(sols.T))
    # make sure the vectors are orthonormal
    if not np.allclose( sols.dot(sols.T.conj()), np.eye(2)):
        b1 = sols[0]
        b1 = b1/np.sqrt(b1.dot(b1.T.conj()))
        b2 = sols[1]
        b2 = b2 - np.dot(b2, b1.conj().T) * b1
        b2 = b2/np.sqrt(b2.dot(b2.T.conj()))
    else:
        b1, b2 = sols
    bp = np.array([b1,b2])
    # now they must be orthonormal:
    assert np.allclose(chopComplex(np.dot(bp,bp.T.conj())), np.eye(2))
    # check they are still a solution of the dirac equation:
    assert np.allclose(chopComplex(np.dot(slash(p),bp.T)), np.zeros_like(bp.T))
    # calculate the components of the vectors with appropriate eigenvector of P_{\pm}q
    comp_minus = scipy.linalg.null_space(bp.dot(np.dot(Pp,slash(q))).T)
    comp_plus = scipy.linalg.null_space(bp.dot(np.dot(Pm,slash(q))).T)
    # check there is only one vector:
    #print scipy.linalg.svd(bp.dot(np.dot(Pp,slash(q))).T)
    #print scipy.linalg.svd(bp.dot(np.dot(Pm,slash(q))).T)
    #print comp_plus.shape
    #print comp_plus.shape, comp_minus.shape
    assert comp_plus.shape == (2,1)
    assert comp_minus.shape == (2,1)
    comp_plus = comp_plus[:,0]
    comp_minus = comp_minus[:,0]
    # get the two vectors for the two columns of the spinor
    c1 = bp.T.dot(comp_plus)
    c2 = bp.T.dot(comp_minus)
    final = np.array([c1,c2]).T
    # check they are still solutions of the dirac equation
    assert np.allclose(chopComplex(slash(p).dot(final)),np.zeros((4,2)))
    # check they re orthnormal
    assert np.allclose(final.T.conj().dot(final), np.eye(2))
    ratios = pFromS(final)/p
    #print ratios
    assert np.allclose(ratios[np.isfinite(ratios)],ratios[0])
    scale = 1.0/np.sqrt( ratios[0] )
    final = final*scale
    assert np.allclose(pFromS(final), p)
    return final
```

In [46]:
```
# Define tilde spinors
#def getSpinorsTilde(p,q,Pp,Pm):
#    sp = getSpinors(p,q,Pp,Pm)
#    t1 = sp[:,0]
#    t2 = sp[:,1]
#    final = np.array([t1,t2]).T
#    final = np.matrix.conjugate(final)
#    return final
```

In [47]:
```
params = rr.randomDirection(10)
#params[8:] = [0,0]
R = makeLorenzRotation(params)
(prot, qrot) = rr.rotate6D([p,q], rotation=R)
U, Ut = makeSpinorRotations(params)

s = sp.Spinor6(p).sp6d()
st = sp.Spinor6(p).sp6td()
s_rot = sp.Spinor6(prot).sp6d()
us = np.dot(U, s)
```

In [48]:
```
sp1 = getSpinors(p, q, Pplus, Pminus)
phase1 = sp1[2,0].conj()/np.abs(sp1[2,0])
phase2 = sp1[1,1].conj()/np.abs(sp1[1,1])
final = np.array([phase1*sp1[:,0], phase2*sp1[:,1]]).T
assert np.allclose(final, s)
```

In [49]:
```
sp2 = getSpinors(prot, q, Pplus, Pminus)
phase1 = sp2[2,0].conj()/np.abs(sp2[2,0])
phase2 = sp2[1,1].conj()/np.abs(sp2[1,1])
```

```
final = np.array([phase1*sp2[:,0], phase2*sp2[:,1]]).T
assert np.allclose(final, s_rot)
```

In [50]:
```
Pplus_rot = np.dot(Ut, np.dot(Pplus, Ut.T.conj()))
Pminus_rot = np.dot(Ut, np.dot(Pminus, Ut.T.conj()))
```

In [51]:
```
assert np.allclose(Pplus_rot.dot(Pplus_rot), Pplus_rot)
assert np.allclose(Pminus_rot.dot(Pminus_rot), Pminus_rot)
assert np.allclose(Pminus_rot + Pplus_rot, np.eye(4))
```

In [52]:
```
sp_rot = getSpinors(prot, qrot, Pplus_rot, Pminus_rot)
phase1 = sp_rot[0,0]/us[0,0]
phase2 = sp_rot[0,1]/us[0,1]
final = np.array([sp_rot[:,0]/phase1, sp_rot[:,1]/phase2]).T
assert np.allclose(final, us)
```

Expected change

In [53]:
```
def change(p, params):
    q = refq(p)
    R = makeLorenzRotation(params)
    (prot, qrot) = rr.rotate6D([p,q], rotation=R)

    U, Ut = makeSpinorRotations(params)
    spp = sp.Spinor6(p).sp6d()  # getSpinors(p, q, Pplus, Pminus)
    sppt = sp.Spinor6(p).sp6td()  # getSpinors(p, q, Pplus, Pminus)
    #print(sp)
    sp_rot_2 = sp.Spinor6(prot).sp6d() #getSpinors(prot, q, Pplus, Pminus)
    spt_rot_2 = sp.Spinor6(prot).sp6td() #getSpinors(prot, q, Pplus, Pminus)

    usp = np.dot(U,spp)
    utstp = np.dot(Ut,sppt)
    transMatrix = np.dot(reciprocal(sp_rot_2), usp)
    transMatrixt = np.dot(reciprocal(spt_rot_2), utstp)

    assert np.isclose(det(transMatrix), 1.0)
    assert np.allclose(transMatrix.T.conj().dot(transMatrix), np.eye(2))
    assert np.allclose(usp, sp_rot_2.dot(transMatrix))

    assert np.isclose(det(transMatrixt), 1.0)
    assert np.allclose(transMatrixt.T.conj().dot(transMatrixt), np.eye(2))
    assert np.allclose(utstp, spt_rot_2.dot(transMatrixt))

    return transMatrix, transMatrixt
```

In [54]:
```
params = rr.randomDirection(10)
```

In [55]:
```
p1, p2 = rr.getRandom6Momenta(2, seed=1212)
pxy = np.array([1,0,0,0.6,0.8,0])
R = makeLorenzRotation(params)
U, Ut = makeSpinorRotations(params)
p1rot, pxyrot = rr.rotate6D([p1, pxy], rotation = R)
tmp1, tmtp1 = change(p1, params)
tmxy, tmtxy = change(pxy, params)
sprod = sp.sp6_dd(p1,pxy)
```

In [56]:
```
sprod_rot = sp.sp6_dd(p1rot,pxyrot)
sprod_rot
```

Out[56]:
```
array([[-0.84209556-0.52203865j, -0.30999959-1.10718329j],
       [ 0.30999959-1.10718329j, -0.84209556+0.52203865j]])
```

In [57]:
```
assert np.allclose(tmp1.T.dot(sprod_rot).dot(tmtxy), sprod)
```

Test Lie algebra for boost

In [58]:
```
#Obtain a 6d phase space point
ps = rr.phaseSpacePoint6(4,2,1234)
p1,p2,p3,p4 = ps
q = np.array(refq(p1))
c4 = 1j * gammas[0].dot(gammas[1]).dot(gammas[2]).dot(gammast[3])
Pplus = 0.5 * (np.eye(4) + c4)
Pminus = 0.5 * (np.eye(4) - c4)
chi = 1
```

In [59]:
```
# Boost it
b1 = lorentzBoost(p1,chi,1)
b2 = lorentzBoost(p2,chi,1)
b3 = lorentzBoost(p3,chi,1)
b4 = lorentzBoost(p4,chi,1)
qb = lorentzBoost(q,chi,1)
```

Check that spinor product agrees to a phase

In [60]:
```
spp1 = sp.Spinor6(p1).sp6d()
spp2 = sp.Spinor6(p2).sp6d()
spp3 = sp.Spinor6(p3).sp6d()
spp4 = sp.Spinor6(p4).sp6d()

spb1 = getSpinors(b1,qb,Pplus,Pminus)
spb2 = getSpinors(b2,qb,Pplus,Pminus)
spb3 = getSpinors(b3,qb,Pplus,Pminus)
spb4 = getSpinors(b4,qb,Pplus,Pminus)

for a in range(2):
    for b in range(2):
        for c in range(2):
            for d in range(2):
                sp64_orig  = sp.sp64(spp1,spp2,spp3,spp4,a,b,c,d)
                sp64_boost = sp.sp64(spb1,spb2,spb3,spb4,a,b,c,d)
                assert abs(sp64_orig) - abs(sp64_boost) < 1e-13
```

Appendix E

Derivations

E.1 Flattened vector

For a massless 6-vector P we can define an equivalent massive 4-vector p^4 in terms of an arbitrary null reference vector q as follows:

$$p^4 = p^\flat + \rho q \tag{E.1.1}$$

squaring both sides

$$(p^4)^2 = (p^\flat)^2 + 2p^\flat \rho q + \rho^2 q^2 = m^2 \tag{E.1.2}$$

Using $(p^\flat)^2 = 0 = q^2$ and using equation E.1.1 to express $p^\flat$ in terms of p^4 we have

$$2(p^4 - \rho q)\rho q = m^2 \tag{E.1.3}$$

and as before $q^2 = 0$, so

$$\begin{aligned} 2p^4 \rho q &= m^2 \\ \rho &= \frac{m^2}{2p^4.q} \end{aligned} \tag{E.1.4}$$

and hence equation E.1.1 gives

$$p^\flat = p^4 - \frac{m^2}{2p^4.q} q \tag{E.1.5}$$

Appendix F

Jupyter notebooks

F.1 utils

Function		
g2	$g^{\mu\nu}$	Metric is mainly negative, +−... 2x2
g4	$g^{\mu\nu}$	4x4
g6	$g^{\mu\nu}$	6x6
g8	$g^{\mu\nu}$	8x8
MP4		Minkowski products
MP6		
refq	q	Arbitrary null reference vector
pflat	$p^\flat$	Flattened 4-vector
lev2u	$\epsilon^{\alpha\beta}$	Levi-Civita tensors
lev2d	$\epsilon_{\alpha\beta}$	
lev3	$\epsilon^{\alpha\beta\gamma} = \epsilon_{\alpha\beta\gamma}$	
lev4	$\epsilon^{\alpha\beta\gamma\delta} = \epsilon_{\alpha\beta\gamma\delta}$	
sig4	$\sigma^i_{\alpha\dot{\beta}}, \sigma_i^{\alpha\dot{\beta}}$	4D Pauli matrices
sigt4	$\sigma^{i\alpha\dot{\beta}}, \sigma_{i\alpha\dot{\beta}}$	
sig6	Σ^i_{AB}	6D sigma matrices, Cheung and O'Connell convention
sigt6	$\tilde{\Sigma}^{iAB}$	
gam4	γ^i	4D gamma matrices, Weyl (chiral) basis, 4x4

Function		
gam6	Γ^i	6D gamma matrices, 8x8
slash4	$p_{\alpha\dot{\alpha}} = p_\mu \sigma^\mu_{\alpha\dot{\alpha}}$	4-momentum contraction with Pauli matrices
slash4t	$p^{\dot{\alpha}\alpha} = \tilde{\sigma}^{\dot{\alpha}\alpha}_\mu p^\mu$	
slash	$P_{AB} = P_\mu \Sigma^\mu_{AB}$	6-momentum contraction with Sigma matrices
slasht	$P^{AB} = P^\mu \Sigma^{\mu AB}$	
sij4	s_{ij}	4D Mandelstam
sij6	S_{ij}	6D Mandelstam

utils: utilities

For gghj package

- IN: None
- PROCESS:
  - Sets up the metric, levi-civita sigma and gamma matrices for 4 and 6 dimensions
  - Provides Minkowski products

Conventions follow Cheung and O'Connell 2009, use Weyl basis for 4d sigma matrices.
$ p_{\mu} = (p0, p1, p2, ...) $, and $ p^{\mu} = (p0, -p1,-p2, ...)$
Work with $p_{\mu} $ unless otherwise stated.

Import standard packages

In [1]:
```
import numpy as np
import itertools as it
import sympy
import math
from cmath import sqrt
from sympy.physics.quantum import tensorproduct
from scipy.linalg import det
```

Definitions

Minkowski metric

Metric is mainly negative: +----...

In [2]:
```
g2 = np.diagflat([1, -1])
g4 = np.diagflat([1, -1, -1, -1])
g6 = np.diagflat([1, -1, -1, -1, -1, -1])
g8 = np.diagflat([1, -1, -1, -1, -1, -1, -1, -1])
```

Minkowski products

In [3]:
```
def MP4(p, q):
    return p[0] * q[0] - np.dot(p[1:4] , q[1:4])
def MP6(p, q):
    return p[0] * q[0] - np.dot(p[1:6] , q[1:6])
```

Function refq: arbitrary reference vector

Obtain arbitrary null vector q and check that q.p is not zero:

In [4]:
```
def refq(p):
    refq_original = (5,0,3,4,0,0)
    refq_nozeros = (abs(sqrt(14)),1,2,3,0,0)
    refq_nozerosscaled = (abs(sqrt(56)),2,4,6,0,0)
    refq_randm = (2.000000000000001 +0.j, -0.7628747983169564+0.j, -0.0504656249392774+0.j,
                  -1.8481004471598275+0.j,  0.                 +0.j,  0.                 +0.j)
    refq_fromsam = (2.17027666842246047, 0.27345005368269976,
                    2.00302541595091689, -0.78943971821443050,0,0)
    refq_min = (1,0,-1,0,0,0)
    refq = refq_min

    # Now check that p.q !+ 0
    pq = p[0] * refq[0]
    for i in (1,2,3):
        pq -= p[i] * refq[i]
    if pq == 0:
        # If it is, generate alternate refq
        refq = refq_randm

    return refq
```

Function pflat: flat four-momentum

We also define here the 'flat' 4-momentum associated with any 6-vector with $ p4,p5 \neq 0 $ in relation to a null reference vector q, such that $ p^{\flat 2} = 0 $:

- **pflat** = (4)p - [(4)p^2 / ((6)P + q)^2] * q

In [5]:
```
def pflat(p):
    pflat = np.zeros((6), dtype = complex)

    # Check p is not already a flat 4-momentum:
    if p[4] != 0 or p[5] != 0:
        q = refq(p)

        # Calculate numerator
        #num = Mom(p).pmass() * Mom(p).pmasst()
        num = p[0]**2 - p[1]**2 - p[2] **2 - p[3]**2

        # Calculate denominator
        pq = p[0] * q[0]
        for i in (1,2,3):
            pq -= p[i] * q[i]
        den = 2 * pq

        # Calculate pflat
        pflat = np.zeros((6), dtype = complex)
        for i in (0,1,2,3):
            pflat[i] = p[i] - (num / den) * q[i]
```

```
    else:
        # if p is already a 4-momentum
        for i in range (6):
            pflat[i] = p[i]

    assert abs(MP6(pflat,pflat)) < 1e-12

    return pflat
```

In [6]:
```
# Check numerically the same as Mathematica implementation
dp1d = np.array([sqrt(13*13 + 3*3 + 7*7 + 2*2 + 5*5), 13,-3,-7,2,5])
assert abs(np.all(pflat(dp1d) - np.array([14.8846,13,-1.88462,-7,0,0]))) < 1e-10
```

Levi-Civita tensors - lev

Levi-Civita tensors are:

- **lev2u** is 2D levi-civita^ab = [0 1][-1 0]
- **lev2d** is 2D levi-civita_ab = -lev2u

In [7]:
```
lev2u = np.array([[0, 1], [-1, 0]])
```

In [8]:
```
lev2d = - lev2u
```

In [9]:
```
lev3 = np.zeros((3, 3, 3), dtype = complex)
perm3 = np.zeros((3))     # initialise permutations
l30   = 0                   # define indices
l31   = 1
l32   = 2
for l3y in (it.permutations([l30, l31, l32])):
    np.put(perm3, [0, 1, 2],l3y)
    l3x = (np.sign(perm3[1] - perm3[0]) * np.sign(perm3[2] - perm3[0])
                * np.sign(perm3[2] - perm3[1]))
    lev3[l3y] = l3x
```

In [10]:
```
lev4 = np.zeros((4, 4, 4, 4), dtype = complex)
perm4 = np.zeros((4))
l40   = 0
l41   = 1
l42   = 2
l43   = 3
for l4y in (it.permutations([l40, l41, l42, l43])):
    np.put(perm4, [0, 1, 2, 3], l4y)
    l4x =  (np.sign(perm4[1] - perm4[0]) * np.sign(perm4[2] - perm4[0])
           * np.sign(perm4[3] - perm4[0]) * np.sign(perm4[2] - perm4[1])
           * np.sign(perm4[3] - perm4[1]) * np.sign(perm4[3] - perm4[2]))
    lev4[l4y] = l4x
```

Standard Pauli matrices - sig4

We use the following convention for Pauli matrices:

- **sig40d** = $ \sigma_0 $ = **1** = [(1, 0), (0, 1)]
- **sig41d** = $ \sigma_1 $ = [(0, 1), (1, 0)]
- **sig42d** = $ \sigma_2 $ = [(0, -i), (i, 0)]
- **sig43d** = $ \sigma_3 $ = [(1, 0), (0, -1)]

(If we need to distinguish between little group indices up and down, see H&P app B)

In [11]:
```
sig40d = np.identity(2, dtype = complex)
sig41d = np.array([[0, 1],[1, 0]], dtype = complex)
sig42d = np.array([[0,-1j],[+1j, 0]], dtype = complex)
sig43d = np.array([[1, 0],[0, -1]], dtype = complex)

sig4d = ([sig40d], [sig41d], [sig42d], [sig43d])
```

For sigmatilde4 with $ \mu $ index down:

- **sigt40d** = $ \tilde{\sigma}_0 $ = sig40
- **sigt41d** = $ \tilde{\sigma}_1 $ = -sig41
- **sigt42d** = $ \tilde{\sigma}_2 $ = -sig42
- **sigt43d** = $ \tilde{\sigma}_3 $ = -sig43

In [12]:
```
sigt40d = sig40d
sigt41d = -sig41d
sigt42d = -sig42d
sigt43d = -sig43d

sigt4d = ([sigt40d], [sigt41d], [sigt42d], [sigt43d])
```

We also define $ \sigma^{\mu} $

In [13]:
```
sig40u = sig40d
sig41u = -sig41d
sig42u = -sig42d
sig43u = -sig43d

sig4u = ([sig40u], [sig41u], [sig42u], [sig43u])
```

In [14]:
```
sigt40u = sigt40d
sigt41u = -sigt41d
sigt42u = -sigt42d
sigt43u = -sigt43d

sigt4u = ([sigt40u], [sigt41u], [sigt42u], [sigt43u])
```

Sigma matrices in 6d - sig6

The sigma6 matrices are antisymmetric matrices which for convenience are simply related to a standard choice of gamma matrices in 4d. The specific form of these matrices here is as Appendix A of Cheung and O'Connell 2009:

- **sig60u** = $ \Sigma^0_{AB} = i \sigma_1 \otimes \sigma_2 $
- **sig61u** = $ \Sigma^1_{AB} = i \sigma_2 \otimes \sigma_3 $
- **sig62u** = $ \Sigma^2_{AB} = -\sigma_2 \otimes \sigma_0 $
- **sig63u** = $ \Sigma^3_{AB} = -i \sigma_2 \otimes \sigma_1 $
- **sig64u** = $ \Sigma^4_{AB} = -\sigma_3 \otimes \sigma_2 $
- **sig65u** = $ \Sigma^5_{AB} = i \sigma_0 \otimes \sigma_2 $

Note that $ \Sigma^{\mu}_{AB} $ (upper $ \mu $ index) is a product of $ \sigma_{\mu} $ (lower index). It has the AB indices down, following C&O'C.

With tilde matrices (AB index up) given by:

- **sigt60u** = $ \tilde{\Sigma}^{0AB} = -\Sigma^0_{AB}$ = -sig60u
- **sigt61u** = $ \tilde{\Sigma}^{1AB} = \Sigma^1_{AB}$ = sig61u
- **sigt62u** = $ \tilde{\Sigma}^{2AB} = -\Sigma^2_{AB}$ = -sig62u
- **sigt63u** = $ \tilde{\Sigma}^{3AB} = \Sigma^3_{AB}$ = sig63u
- **sigt64u** = $ \tilde{\Sigma}^{4AB} = -\Sigma^4_{AB}$ = -sig64u
- **sigt65u** = $ \tilde{\Sigma}^{5AB} = \Sigma^5_{AB}$ = sig65u

In [15]:
```
sig60u = 1j * np.kron(sig41d, sig42d)
sig61u = 1j * np.kron(sig42d, sig43d)
sig62u = -np.kron(sig42d, sig40d)
sig63u = -1j * np.kron(sig42d, sig41d)
sig64u = -np.kron(sig43d, sig42d)
sig65u = 1j * np.kron(sig40d, sig42d)
sig6u = ([sig60u], [sig61u], [sig62u], [sig63u],[sig64u], [sig65u])
```

In [16]:
```
sigt60u = -sig60u
sigt61u = sig61u
sigt62u = -sig62u
sigt63u = sig63u
sigt64u = -sig64u
sigt65u = sig65u
sigt6u = ([sigt60u], [sigt61u], [sigt62u], [sigt63u],[sigt64u], [sigt65u])
```

We also define $ \Sigma_{\mu}$ and $ \tilde{\Sigma}_{\mu}$

In [17]:
```
sig60d = sig60u
sig61d = -sig61u
sig62d = -sig62u
sig63d = -sig63u
sig64d = -sig64u
sig65d = -sig65u
sig6d = ([sig60d], [sig61d], [sig62d], [sig63d],[sig64d], [sig65d])
```

In [18]:
```
sigt60d = sigt60u
sigt61d = -sigt61u
sigt62d = -sigt62u
sigt63d = -sigt63u
sigt64d = -sigt64u
sigt65d = -sigt65u
sigt6d = ([sigt60d], [sigt61d], [sigt62d], [sigt63d],[sigt64d], [sigt65d])
```

In [19]:
```
#print('sig6u')
#print(sig6u)
#print()
#print('sigt6u')
#print(sigt6u)
#print()
#print('sig6d')
#print(sig6d)
#print()
#print('sigt6d')
#print(sigt6d)
#print()
```

Slash momenta

- **slash(p)** = $ P_{AB} = P_{\mu} \Sigma^{\mu}_{AB}$
- **slasht(p)** = $ P^{AB} = P_{\mu} \tilde{\Sigma}^{\mu AB}$

4d contraction with Pauli matrices (used in testing) also defined here:

- **psig4d** = p_alpha, alphadot = p_mu sigma4^mu_alpha, alphadot
- **psig4u** = p^alphadot, alpha = sigma4tilde_mu^alphadot, alpha p^mu
  (note: sigmatilde_mu^little group up shares definition of sigma^mu_little group in invar)

In [20]:
```
gammas = np.array([ss[0] for ss in sig6u])
gammast = np.array([ss[0] for ss in sigt6u])
gammasd = np.array([ss[0] for ss in sig6d])
gammastd = np.array([ss[0] for ss in sigt6d])


#print(gammas) #These are definitely the same as DM Mathematica implementation
```

In [21]:
```
def slashedMomentum(p, gammaMatrices):

    #Original version used p^mu, but this is incorrect:
    #p_contracted = p*np.array([1,-1,-1,-1,-1,-1])
    #pslash = np.array([pp* gg for pp,gg in zip(p_contracted, gammaMatrices)])

    pslash = np.array([pp* gg for pp,gg in zip(p, gammaMatrices)])
    pslash = np.sum(pslash ,axis=0)
    return pslash
```

In [22]:
```
def slash(p):
    return slashedMomentum(p, gammas)
```

In [23]:
```
def slasht(p):
    return slashedMomentum(p, gammast)
```

And required for testing 4D spinors:

In [24]:
```
gammas4 = np.array([ss[0] for ss in sig4d])
gammast4 = np.array([ss[0] for ss in sigt4d])
```

In [25]:
```
def slashed4Momentum(p, gammaMatrices):
    #p_contracted = p*np.array([1,-1,-1,-1])
    #pslash = np.array([pp* gg for pp,gg in zip(p_contracted, gammaMatrices)])

    pslash = np.array([pp* gg for pp,gg in zip(p, gammaMatrices)])
    pslash = np.sum(pslash ,axis=0)
    return pslash
```

In [26]:
```
def slash4(p):
    return slashed4Momentum(p, gammas4)
```

In [27]:
```
def slasht4(p):
    return slashed4Momentum(p, gammast4)
```

In [28]:
```
# Check numerically the same as Mathematica implementation

dp1d = np.array([sqrt(13*13 + 3*3 + 7*7 + 2*2 + 5*5), 13,-3,-7,2,5])
dp1d4 = np.array([sqrt(13*13 + 3*3 + 7*7), 13,-3,-7])


dPSlash = np.array([[0, 1j*dp1d[4]+dp1d[5], dp1d[1]+1j*dp1d[2], dp1d[0]-dp1d[3]],
                    [-1j*dp1d[4]-dp1d[5], 0, -dp1d[0]-dp1d[3], -dp1d[1]+1j*dp1d[2]],
                    [-dp1d[1]-1j*dp1d[2], dp1d[0]+dp1d[3], 0, -1j*dp1d[4]+dp1d[5]],
                    [-dp1d[0]+dp1d[3], dp1d[1]-1j*dp1d[2], 1j*dp1d[4]-dp1d[5], 0]])
assert np.allclose(slash(dp1d)-dPSlash, 0)

dPSlasht = np.array([[0, -1j*dp1d[4]+dp1d[5], dp1d[1]-1j*dp1d[2], -dp1d[0]-dp1d[3]],
                    [1j*dp1d[4]-dp1d[5], 0, dp1d[0]-dp1d[3], -dp1d[1]-1j*dp1d[2]],
                    [-dp1d[1]+1j*dp1d[2], -dp1d[0]+dp1d[3], 0, 1j*dp1d[4]+dp1d[5]],
                    [dp1d[0]+dp1d[3], dp1d[1]+1j*dp1d[2], -1j*dp1d[4]-dp1d[5], 0]])
assert np.allclose(slasht(dp1d)-dPSlasht, 0)

dPSlash4 = np.array([[dp1d4[0]+dp1d4[3], dp1d4[1]-1J*dp1d4[2]],
                    [dp1d4[1]+1J*dp1d4[2], dp1d4[0]-dp1d4[3]]])
assert np.allclose(slash4(dp1d4)-dPSlash4, 0)
```

Mandelstam

- S_ij = (P_i + P_j)^2 = 2 P_i.P_j
  sij6 = 2P_i.P_j

- s_(ij) = (p_i + p_j)^2 = 2.p_i.p_j = p_i^(alpha alphadot).p_j_(alpha alphadot)
  sij4 = 2p_i.p_2

In [29]:
```
def sij6(p1,p2):
    return 2 * MP6(p1,p2)
```

In [30]:
```
def sij4(p1,p2):
    return 2 * MP4(p1,p2)
```

Other utilities

In [31]:
```
# Commutator
def comm(m1, m2):
    return np.dot(m1,m2) - np.dot(m2,m1)
```

In [32]:
```
def permutationSign(new, orig):
    assert len(new) == len(orig)
    n = len(new)
    m = np.zeros( (n,n) , dtype=int)
    for i, j  in zip(new, orig):
        m[i, j] = 1
    return int(det(m))
```

In [33]:
```
def chop(a):
    a[np.abs(a)<1e-12] = 0
    return a
```

In [34]:
```
def chopComplex(a):
    ret = chop(np.real(a))+ 1j * chop(np.imag(a))
    return ret
```

In [35]:
```
def reciprocal(vs):
    '''
        Return a matrix with the reciprocal vectors as row, such that reciprocal(vs).dot(vs) == 1
    '''
    v2 = np.dot(vs.T.conj(), vs)
    assert np.isclose( v2[0,1], 0.0  )
    assert np.isclose( v2[0,1], 0.0  )
    assert np.isclose( v2[1,1], v2[0,0]  )
    rec = vs/v2.diagonal()
    rec = rec.T.conj()
    assert np.allclose(np.dot(rec,vs), np.eye(2))
    return rec
```

In [36]:
```
def dotprod(u,w):
    # Calculate dot product of two 2x1 matrices
    res = 0
    for i in (0,1):
        res += u[i] * w[i]
```

F.2 randrot

	Function	
Rotations		Set up function required for rotation
	randomDirection	
	getBasis	
	getRandomRotation	
	getRandomRotationWithFixedVector	
	rotate6D	
Random momenta		
	getRandom4Momenta	
	getRandom6Momentum	
	getRandom6Momenta	
	random4Momentum6D	With trailing zeros
	random4Momenta6d	
Phase space point		Make a random phase space point
	boost	
	boostedMomenta	
	boosted6Momenta	(not required)
	phaseSpacePoint4	
	phaseSpacePoint46	With trailing zeros
	phaseSpacePoint6	
Import from S@M		
	momset4d	Import 100 sets of 4-particle, 4D phase space points generated by S@M
Build Lie algebras		
	sigma	
	sigmat	
	sindices	
	w	
	comm	
	windices	
	makeLorenzRotation	
	makeSpinorRotations	
	pFromS	Reconstruct momentum from spinors
	pFromSt	

	Function	
Covariant basis		
	c4	
	Pplus	
	Pminus	
General construction		
	getSpinors	Obtain spinors from rotated momentum sets

randrot: random momenta, phase space points and rotated spinors

For gghj package

- IN: None
- PROCESS:
  - Generates random four-momenta
  - Rotates to 6D to create random six-momenta
  - Generates random 4D phase space points
  - Rotates to 6D to create random 6D phase space points
  - Generates rotated spinors for the rotated momenta using lieAlgebra relations

Import standard packages

In [1]:
```
import numpy as np
import sympy
import scipy
import scipy.linalg
from scipy.linalg  import det
import math
from cmath import sqrt
```

Import other gghj modules

In [2]:
```
from utils import *
```

Rotations

First set up functions required for rotation

In [3]:
```
def randomDirection(n):
    # n is number of randomDirections required
    # used in rotated spinor generation

    while True:
        p = np.random.uniform(-1,1,n)
        r = np.sum(p**2)
        if r <=1:
            break
    p /= np.sqrt(r)
    return p
```

In [4]:
```
def getBasis(preexisting):
    # preexisting is a momentum

    ndim = len(preexisting[0])
    refs = np.eye(ndim)
    vs = list(preexisting)
    nMissing = ndim - len(preexisting)
    ind = 0
    while len(vs) < ndim:
        v = np.empty(ndim)
        v[:] = refs[ind]
        ind += 1
        #print(v)
        for ivb, vb in enumerate(vs):
            v = v - np.dot(v, vb) * vb
            #print(v)
            for vbb in vs[:ivb]:
                assert np.isclose(0,np.dot(v,vbb))
        if np.allclose(v,np.zeros(ndim)):
            continue
        v /= np.sqrt(sum(v**2))
        vs.append(v)
    return vs
```

In [5]:
```
def getRandomRotation(ndim):
    rds = [randomDirection(idim) for idim in range(ndim, 0 , -1) ]
    vs = [rds[0]]
    for istep in range(1,ndim):
        m = np.array(getBasis(vs))
        nv = np.dot(rds[istep], m[istep:])
        vs.append(nv)

    rot = np.array(vs)
    assert np.allclose(np.dot(rot,rot.T), np.eye(ndim))
    return rot
```

In [6]:
```
def getRandomRotationWithFixedVector(ndim, p):
    pnorm = p / np.sqrt(np.sum(p*p))
    S = np.array(getBasis([pnorm]))
    assert np.allclose(np.dot(S,S.T), np.eye(ndim))

    r = getRandomRotation(ndim-1)
    R = np.eye(ndim)
    R[1:,1:] = r

    rot = np.dot(S.T,np.dot(R,S))
    assert np.allclose(np.dot(rot,rot.T), np.eye(ndim))
    assert np.allclose(np.dot(rot,p), p)
    return rot
```

Define rotation rotate into 6D

In [7]:
```
def rotate6D(ps, seed=None, keepFixed=None, rotation=None):
    if seed:
```

```
        np.random.seed(seed)
    if keepFixed is None:
        if rotation is None:
            r = getRandomRotation(5)
        else :
            r=rotation
           # print("r = rotation")
            #print(r)
    else:
        r = getRandomRotationWithFixedVector(5, keepFixed)
    ret = []
    for p in ps:
        new = np.zeros([6], dtype = complex)
        new[0] = p[0]
        #print("new 0", new)
        new[1:] = np.dot(r, p[1:])
        #print("new 1:", new)
        ret.append(new)
        #print(ret)
        assert abs(MP6(p,p)) < 1e-13
    return ret
```

Random momenta

In [10]:
```
def getRandom4Momenta(n, seed = None):
    # n random momenta in 4D, n can be 1, not connected as a phase space point
    # equivalent to rambo

    if seed:
        np.random.seed(seed)
    E = np.random.uniform(0.2,2.0, n)
    c = np.random.uniform(-1,1, n)
    phi = np.random.uniform(0, np.pi, n)
    s = np.sqrt(1 - c**2)
    momenta = np.array([E, E * s * np.sin(phi), E * s * np.cos(phi), E * c])
    return momenta.T
```

In [11]:
```
def getRandom6Momentum(seed=None):
    # A single random momentum obtained by rotation from 4D
    if seed:
        np.random.seed(seed)
    E = np.random.uniform(0.2,2.0)
    rot = getRandomRotation(5)
    ref = np.array([E,0,0,0,0])
    vec = np.dot(rot,ref)
    ret = np.empty(6)
    ret[0] = E
    ret[1:] = vec
    assert abs(MP6(ret,ret)) < 1e-13
    return ret
```

In [12]:
```
def getRandom6Momenta(n, seed=None):
    if seed:
        np.random.seed(seed)
    ret = np.empty( (n, 6) )
    for i in range(n):
        ret[i] = getRandom6Momentum()
    return ret
```

In [13]:
```
def random4Momentum6D(seed=None):
    if seed:
        np.random.seed(seed)
    E = np.random.uniform(0.2,2.0)
    rot = getRandomRotation(3)
    ref = np.array([E,0,0])
    vec = np.dot(rot,ref)
    ret = np.empty(6)
    ret[0] = E
    ret[1:4] = vec
    ret[4] = 0.0
    ret[5] = 0.0
    assert abs(MP6(ret,ret)) < 1e-13
    return ret
```

In [14]:
```
def random4Momenta6D(n, seed=None):
    if seed:
        np.random.seed(seed)
    ret = np.empty( (n, 6) )
    for i in range(n):
        ret[i] = random4Momentum6D()
    return ret
```

Make a random phase space point

In [15]:
```
# Brought in from my makePSP

def boost(q,x,gamma,b):
    # all parmeters are from posMomenta:
    # q     = one of the momenta from getRandomMomenta
    # x     = required total energy / total 'mass' of random set
    # gamma = sum of q[0] terms / total 'mass' of random set
    # b     = array(-qi / total 'mass') for i = 1,2,3

    p0 = x*(gamma*q[0] + np.dot(b,q[1:]))
    p = np.array(q[1:])
    p += b*q[0]
    f = 1 / (1 + gamma)
    f *= np.dot(b,q[1:])
    p += b*f
    p *= x
    return (p0,) + tuple(p)
```

In [16]:
```
def boostedMomenta(E,n, seed = None):
    # Obtain n light-like momenta which sum to energy E
```

```
    if seed:
        np.random.seed(seed)
    qs = getRandom4Momenta(n,seed)
    Q = np.array([ sum([q[j] for q in qs]) for j in range(4)])
    M = math.sqrt(MP4(Q,Q))
    b = np.array( [-q/M for q in Q[1:]])
    x = E / M
    gamma = Q[0] / M
    boostM = [boost(q,x,gamma,b) for q in qs]
    assert (abs(MP6(p,p)) < 1e-13 for p in boostM)
    return boostM
```

In [17]:
```
def boosted6Momenta(E,n, seed = None):
    # Obtain n light-like momenta which sum to energy E
    # Not required

    if seed:
        np.random.seed(seed)

    qs = getRandom6Momenta(n, seed)
    Q = np.array([ sum([q[j] for q in qs]) for j in range(6)])
    M = math.sqrt(MP6(Q,Q))
    b = np.array( [-q/M for q in Q[1:]])
    x = E / M
    gamma = Q[0] / M
    boostM = [boost(q,x,gamma,b) for q in qs]
    return boostM
```

In [18]:
```
def phaseSpacePoint4(n,m, seed = None):
    # n is the total number of momenta
    # m is the total number of negative momenta
    # seed is optional, may be useful for testing

    if n < 2 or m < 2:
        return(print("Error, both n and m must be >= 2"))

    if seed:
        np.random.seed(seed)
    E = n/2
    pneg = boostedMomenta(E, m, seed)
    neg = np.array([-1,1,1,1])
    pneg = np.multiply(pneg,neg)
    ppos = boostedMomenta(E, n-m)
    # No seed passed to ppos to avoid pneg and ppos using same momenta
    psp = np.concatenate((pneg, ppos), axis=0)

    # Check this is a valid phase space point
    checkpsum = np.zeros(4)
    for p in psp:
        for i in range(4):
            checkpsum[i] += p[i]
    assert np.allclose(checkpsum, np.zeros(4))

    return psp
```

In [19]:
```
def phaseSpacePoint46(n,m, seed = None):
    #Obtain a 4D phase space point and add trailing zeros

    if seed:
        np.random.seed(seed)

    psp = np.zeros((n,6))
    #print(psp)
    psp4 = phaseSpacePoint4(n,m,seed)
    #print(psp4)
    for i in range(n):
        for j in range(4):
            psp[i,j] += psp4[i,j]
    return psp
```

In [20]:
```
def phaseSpacePoint6(n,m, seed = None):
    # n is the total number of momenta
    # m is the total number of negative momenta

    if n < 2 or m < 2:
        return(print("Error, both n and m must be >= 2"))

    if seed:
        np.random.seed(seed)

    E = n/2
    pneg = boosted6Momenta(E, m, seed)
    neg = np.array([-1,1,1,1,1,1])
    pneg = np.multiply(pneg,neg)
    ppos = boosted6Momenta(E, n-m)
    return np.concatenate((pneg, ppos), axis=0)
```

Import 100 sets of 4-particle, 4d phase space points generated by S@M

In [21]:
```
# Import 4D phase space points from S@M and reconfigure with trailing zeros
# Used in testing module: tree

def momset4d():
    momset4d = np.zeros((100,4,6))
    rawsam = np.loadtxt("sam_momenta_sets.txt")

    n = 0
    for a in range (100):
        row = 0
        for row in range(4):
            col = 0
            for col in range(4):
                momset4d[a,row,col] = rawsam[n]
                n += 1
                col += 1
```

```
            momset4d[a,row,4] = 0
            momset4d[a,row,5] = 0
            row += 1
        a += 1
    return momset4d

sam4p = momset4d()
```

Using lieAlgebra for rotated spinors

Building the lie algebras

We define matrices for the Lie algebra of rotations in the $SU(4)$ representation:

$$\sigma^{ij}^A_B = \frac14\left(\tilde\gamma^{i}^{AC}\gamma^{j}_{CD} - \tilde\gamma^j^{AC}\gamma^{i}_{CD}\right)$$

$$\tilde\sigma^{ij}_A^B = \frac14\left(\gamma^{i}_{AC}\tilde\gamma^{j}^{CD} - \gamma^j_{AC}\tilde\gamma^{i}^{CD}\right)$$

In [22]:
```
def sigma(mu,nu):
    return 0.25 * 1* (np.dot(sigt6[mu][0], sig6[nu][0]) -  np.dot(sigt6[nu][0], sig6[mu][0]))
```

In [23]:
```
def sigmat(mu,nu):
    return 0.25 * 1 *(np.dot(sig6[mu][0], sigt6[nu][0]) -  np.dot(sig6[nu][0], sigt6[mu][0]))
```

The elements of the algebra are labelled by two indices, the two axis that will get rotated. It is a 10-dimensional.

In [24]:
```
sindices = [ (i,j) for i in range(1,6) for j in range(1,6) if i<j]
```

This is the algebra for the fundamental representation of rotations in R^5.

In [25]:
```
def w(i,j):
    ret = np.zeros((5,5))
    if i == j :
        return ret
    ret[i, j] = -1
    ret[j, i] = 1
    return ret
```

In [26]:
```
def comm(m1,m2):
    return np.dot(m1,m2) - np.dot(m2,m1)
```

In [27]:
```
windices = [ (i,j) for i in range(5) for j in range(5) if i<j]
```

In [28]:
```
def makeLorenzRotation(params):
    ws = np.array([w(*i) for i in windices])
    L = np.sum(np.multiply(params[:,np.newaxis, np.newaxis] , ws) , axis =0)
    R = scipy.linalg.expm(L)
    return R
```

In [29]:
```
def makeSpinorRotations(params):
    sigmas = np.array([sigma(*i) for i in sindices])
    L = np.sum(np.multiply(params[:,np.newaxis, np.newaxis] , sigmas) , axis =0)
    U = scipy.linalg.expm(L)
    sigmats = np.array([sigmat(*i) for i in sindices])
    L = np.sum(np.multiply(params[:,np.newaxis, np.newaxis] , sigmats) , axis =0)
    Ut = scipy.linalg.expm(L)
    return U, Ut
```

Reconstruction of the momentum from the spinors

DO NOT USE - CORRECT 6d FORUMULATION NOW IN utils.ipynb

In [30]:
```
def pFromS(s):
    res= np.zeros(6, dtype='complex128')
    for i in range(6):
        prod = (np.dot(s.T, np.dot(gammas[i], s)))
        component = -0.25*(prod[1,0]-prod[0,1])
        assert np.isclose(prod[1,0], -prod[0,1])
        res[i] = component
    return res

def pFromSt(st):
    res= np.zeros(6, dtype='complex128')
    for i in range(6):
        prod = (np.dot(st.T, np.dot(gammast[i], st)))
        component = -0.25*(prod[0,1]-prod[1,0])
        res[i] = component
    return res

#assert np.allclose(pFromS(s), p)
#assert np.allclose(pFromSt(st), p)
```

Covariant definition of the basis for 6D spinors

We use

$$C_4 = i \gamma^0\gamma^1\gamma^2\gamma^3$$

The definition of the spinors is such that

$$\not q |s\rangle$$

is an eigenvector of C_4 with eigenvalue ∓ 1 for the first and second column respectively. We see that below:

In [31]:
```
c4 = 1j * gammas[0].dot(gammast[1]).dot(gammas[2]).dot(gammast[3])
```

We define projectors onto the positive and negative eigenvectors of C_4:

In [32]:
```
Pplus = 0.5*(np.eye(4)+c4)
Pminus = 0.5*(np.eye(4)-c4)
```

General construction of rotated spinors

Notes re using this general construction:

For a single momentum, to use getSpinors we will require the following steps

- params = randomDirection(10)
- R = makeLorenzRotation(params)
- U, Ut = makeSpinorRotations(params)
- Pplus_rot = np.dot(Ut, np.dot(Pplus, Ut.T.conj()))
- Pminus_rot = np.dot(Ut, np.dot(Pminus, Ut.T.conj()))

- specify a momentum, p
- specify a reference vector, q

- (prot,qrot) = rotate6D([p,q], rotation = R)

- For getSpinors specify: p = prot, q = qrot, Pp = Pplus_rot, Pm = Pminus_rot

For a phase space point, to use getSpinors for all momenta we will require the following steps

- params to Pminus_rot as above

- specify a phase space point ps = phaseSpacePoint46
- p1,p2,p3,p4,... = ps
- specify a reference vector, q
- q = np.array(refq(p1))

- r1, r2, r3,r4,... = rotate6D(ps, rotation = R)
- qr = rotate6D([q], rotation = R)
- qrot = qr[0]

- For getSpinors specify: p = r1, q = qrot, Pp = Pplus_rot, Pm = Pminus_rot
- Then p = r2, q = qrot ... etc

In [33]:
```python
def getSpinors(p, q, Pp, Pm):

    assert np.allclose(Pp + Pm, np.eye(4))
    assert np.allclose(np.dot(Pp, Pp), Pp)
    assert np.allclose(np.dot(Pm, Pm), Pm)
    assert np.allclose(np.dot(Pm, Pp), np.zeros_like(Pp))
    assert np.allclose(np.dot(Pp, Pm), np.zeros_like(Pp))
    ev, vs = np.linalg.eig(slash(p))

    sols = vs.T[np.isclose(ev, 0.0)]

    # check the solutions are solutions...
    assert np.allclose(chopComplex(slash(p).dot(sols.T)), np.zeros_like(sols.T))
    # make sure the vectors are orthonormal
    if not np.allclose( sols.dot(sols.T.conj()), np.eye(2)):
        b1 = sols[0]
        b1 = b1/np.sqrt(b1.dot(b1.T.conj()))
        b2 = sols[1]
        b2 = b2 - np.dot(b2, b1.conj().T) * b1
        b2 = b2/np.sqrt(b2.dot(b2.T.conj()))
    else:
        b1, b2 = sols
        print("b1, b2 = sols")
    bp = np.array([b1,b2])

    # now they must be orthonormal:
    assert np.allclose(chopComplex(np.dot(bp,bp.T.conj())), np.eye(2))
    # check they are still a solution of the dirac equation:
    assert np.allclose(chopComplex(np.dot(slash(p),bp.T)), np.zeros_like(bp.T))

    # calculate the components of the vectors with appropriate eigenvector of P_{\pm}q
    comp_minus = scipy.linalg.null_space(bp.dot(np.dot(Pp,slash(q))).T)
    comp_plus = scipy.linalg.null_space(bp.dot(np.dot(Pm,slash(q))).T)
    # check there is only one vector:
    assert comp_plus.shape == (2,1)
    assert comp_minus.shape == (2,1)

    comp_plus = comp_plus[:,0]
    comp_minus = comp_minus[:,0]
    # get the two vectors for the two columns of the spinor
    c1 = bp.T.dot(comp_plus)
    c2 = bp.T.dot(comp_minus)
    final = np.array([c1,c2]).T
    ## check they are still solutions of the dirac equation
    assert np.allclose(chopComplex(slash(p).dot(final)),np.zeros((4,2)))
    # check they re orthnormal
    assert np.allclose(final.T.conj().dot(final), np.eye(2))
    ratios = pFromS(final)/p
    assert np.allclose(ratios[np.isfinite(ratios)],ratios[0])
    scale = 1.0/np.sqrt( ratios[0] )
    final = final*scale
    assert np.allclose(pFromS(final), p)
    return final
```

In [34]:
```python
# THIS IS NOT TESTED (AND NOT USED) ***************************************

def getTildeSpinors(p, q, Pp, Pm):

    assert np.allclose(Pp + Pm, np.eye(4))
    assert np.allclose(np.dot(Pp, Pp), Pp)
    assert np.allclose(np.dot(Pm, Pm), Pm)
    assert np.allclose(np.dot(Pm, Pp), np.zeros_like(Pp))
    assert np.allclose(np.dot(Pp, Pm), np.zeros_like(Pp))
    ev, vs = np.linalg.eig(slasht(p))

    sols = vs.T[np.isclose(ev, 0.0)]

    # check the solutions are solutions...
    assert np.allclose(chopComplex(slasht(p).dot(sols.T)), np.zeros_like(sols.T))
    # make sure the vectors are orthonormal
    if not np.allclose( sols.dot(sols.T.conj()), np.eye(2)):
        b1 = sols[0]
        b1 = b1/np.sqrt(b1.dot(b1.T.conj()))
        b2 = sols[1]
        b2 = b2 - np.dot(b2, b1.conj().T) * b1
        b2 = b2/np.sqrt(b2.dot(b2.T.conj()))
    else:
        b1, b2 = sols
        print("b1, b2 = sols")
    bp = np.array([b1,b2])

    # now they must be orthonormal:
    assert np.allclose(chopComplex(np.dot(bp,bp.T.conj())), np.eye(2))
    # check they are still a solution of the dirac equation:
    assert np.allclose(chopComplex(np.dot(slash(p),bp.T)), np.zeros_like(bp.T))

    # calculate the components of the vectors with appropriate eigenvector of P_{\pm}q
    comp_minus = scipy.linalg.null_space(bp.dot(np.dot(Pp,slash(q))).T)
    comp_plus = scipy.linalg.null_space(bp.dot(np.dot(Pm,slash(q))).T)
    # check there is only one vector:
    assert comp_plus.shape == (2,1)
    assert comp_minus.shape == (2,1)

    comp_plus = comp_plus[:,0]
    comp_minus = comp_minus[:,0]
    # get the two vectors for the two columns of the spinor
    c1 = bp.T.dot(comp_plus)
    c2 = bp.T.dot(comp_minus)
    final = np.array([c1,c2]).T
    ## check they are still solutions of the dirac equation
    assert np.allclose(chopComplex(slash(p).dot(final)),np.zeros((4,2)))
    # check they re orthnormal
    assert np.allclose(final.T.conj().dot(final), np.eye(2))
    ratios = pFromS(final)/p
    assert np.allclose(ratios[np.isfinite(ratios)],ratios[0])
    scale = 1.0/np.sqrt( ratios[0] )
    final = final*scale
    assert np.allclose(pFromS(final), p)
```

F.3 spinors

Class	*Function*		
Mom	pplus	$p_0 + p_3$	Momentum components of spinors
	pminus	$p_0 - p_3$	
	pperp	$p_1 + ip_2$	
	pperm	$p_1 - ip_2$	
	pmasst	$p_5 + ip_4$	
	pmass	$p_5 - ip_4$	
Spinor2	la	$\lambda_\alpha, \|i\rangle$	Weyl spinors
	lat	$\tilde{\lambda}_{\dot{\alpha}}, [i\|$	
	cla	$\lambda^\alpha, \langle i\|$	
	clat	$\tilde{\lambda}^{\dot{\alpha}}, \|i]$	
Spinor4	up	$u_+ = v_- = \begin{bmatrix} \lambda_\alpha \\ 0 \end{bmatrix}, \|p\rangle$	Spinor solutions of 4D Dirac
	um	$u_- = v_+ = \begin{bmatrix} 0 \\ \tilde{\lambda}^{\dot{\alpha}} \end{bmatrix}, \|p]$	
	ubp	$\bar{u}_+ = \bar{v}_- = \begin{bmatrix} 0 & \tilde{\lambda}_{\dot{\alpha}} \end{bmatrix}, [p\|$	
	ubm	$\bar{u}_- = \bar{v}_+ = \begin{bmatrix} \lambda^\alpha & 0 \end{bmatrix}, \langle p\|$	
Sp6Ka	ka	κ	κ components of 6D spinors
	kat	$\tilde{\kappa}$	
	kap	κ'	
	kapt	$\tilde{\kappa}'$	
Spinor6	sp6d	Λ_a^A	6D spinors
	sp6td	$\tilde{\Lambda}_{A\dot{a}}$	
	sp6u	Λ^{Aa}	
	sp6tu	$\tilde{\Lambda}_A^{\dot{a}}$	
	sp22	$\langle ij\rangle = \lambda^\alpha \lambda_\alpha$ etc	Weyl spinor contractions
	sp62	$\langle i^a \|j_{\dot{b}}] = [j_{\dot{b}}\|i^a\rangle = \Lambda_i^{Aa}\tilde{\Lambda}_{jA\dot{b}}$ etc	6D spinor contractions
	sp64	$\langle i_a, j_b, k_c, l_d\rangle$	
	sp6s1	$\langle i_a\| \not{P} \|j_b\rangle$	6D spinor strings
	sp6s2	$\langle i_a\| \not{P}_1 \not{P}_2 \|i^{\dot{a}}]$	
	sp6s3	$\langle i_a\| \not{P}_1 \not{P}_2 \not{P}_3 \|j_b\rangle$	
	sp6s6	$\langle i_a\| \not{P}_j \not{P}_k \not{P}_l - \not{P}_l \not{P}_k \not{P}_j \|i_a\rangle$	

Class	*Function*		
	pol	$\epsilon^{\mu}_{a\dot{a}}$	Polarisation vector

spinors: 6D spinor formalism

For gghj package

- IN: Receives from module **tree** calls for specified spinor-related values associated with specified 6D momenta.
- FUNCTIONALITY:
  - Calculates 6D spinor products and spinor strings.
  - According to the Cheung and O'Connell (2009) formalism.
- OUT: Returns requested spinor-related values to **tree**.

Import standard packages

In [1]:
```
import numpy as np
import sympy
import scipy
from cmath import sqrt
```

Import other gghj modules

In [2]:
```
from utils import *           # metric, levi-civita, sigma and gamma matrices, Minkowski products etc
import randrot as rr          # random momenta generator to replace randm and testm
```

Class Mom: Momentum spinor components and contraction

Convention:

- $ p_{\mu} $ = **p** = (p0, p1, p2, p3, p4, p5);
- $ p^{\mu} $ = **pu** = (p0, -p1, -p2, -p3, -p4, -p5)

Calculate momentum components of spinors:

- **pplus** = $ p_0 + p_3 $
- **pminus** = $ p_0 - p_3 $
- **pperp** = $ p_1 + ip_2 $
- **pperm** = $ p_1 - ip_2 $
- **pmasst** = $ p_5 + ip_4 $
- **pmass** = $ p_5 - ip_4 $

In [3]:
```
class Mom:
    def __init__(self, p):
        self._p      = p
        self.c       = (self._p)
        self._pplus  = None
        self._pminus = None
        self._pperp  = None
        self._pperm  = None
        self._pmasst = None
        self._pmass  = None
        self._psig6d = None
        self._psig6u = None
        self._psig4d = None
        self._psig4u = None

    def __str__(self):
        out = "Mom: Momentum spinor components and contractions with sigma matrices\
               for momentum{}".format(','.join([str(cc) for cc in self.c]))
        return out
    def __repr__(self):
        return "Mom({})".format(','.join([str(cc) for cc in self.c]))


    def pplus(self):
        self._pplus = self._p[0] + self._p[3]
        return self._pplus

    def pminus(self):
        self._pminus = self._p[0] - self._p[3]
        return self._pminus

    def pperp(self):
        self._pperp = self._p[1] + 1j * self._p[2]
        return self._pperp

    def pperm(self):
        self._pperm = self._p[1] - 1j * self._p[2]
        return self._pperm

    def pmass(self):
        self._pmass = self._p[5] - 1j * self._p[4]
        return self._pmass

    def pmasst(self):
        self._pmasst = self._p[5] + 1j * self._p[4]
        return self._pmasst
```

Class Spinor2: Weyl spinors

Four-momentum can be represented as a bispinor, using massless chiral and anti-chiral spinors. Notation (is per DM Mathematica):

- $ p^{\alpha \dot{\alpha}}_{\mu} = p_{\mu} \sigma^{\mu \alpha \dot{\alpha}} = \tilde{\lambda} ^{\dot{\alpha}} \lambda^{\alpha}$ **= lat la**
- $ p_{\mu \alpha \dot{\alpha}} = p_{\mu} \sigma^{\mu}_{\alpha \dot{\alpha}} = \lambda_{\alpha} \tilde{\lambda}_{\dot{\alpha}}$ **= cla clat** = $ \epsilon_{\alpha \beta} \lambda^{\beta} \epsilon_{\dot{\alpha} \dot{\beta}} \tilde{\lambda}^{\dot{\beta}}$

So:

- **la**: | i > lambda^alpha
- **lat**: [i | lambdatilde^alphadot
- **cla**: < i | lambda_alpha
- **clat**: | i] lambdatilde_alphadot

Comments:

- Handling division by zero: if the plus component of SixMomentum is zero then a denominator with this component will result in 'divide by zero' error. However, all spinors with plus in the denomenator also have a factor of plus in the numerator, hence safe to give answer zero.
- All Spinor2 are represented as 2 x 1 matrices.

In [4]:
```
class Spinor2:

    # This version has the square root demoninator explicit in each calculation

    def __init__(self, p):
        self._p    = p
        self.c     = (self._p)
        self._la   = None
        self._lat  = None
        self._cla  = None
        self._clat = None

    def __str__(self):
        out = "Spinor2: 2d chiral/anti-chiral spinors with 4-momentum \
               components{}".format(','.join([str(cc) for cc in self.c]))
        return out
    def __repr__(self):
        return "Spinor2({})".format(','.join([str(cc) for cc in self.c]))


    def la(self):
        if self._la is None:
            self._la = np.zeros((2,1), dtype = complex)
            if Mom(self._p).pplus() != 0:
                self._la[0] = Mom(self._p).pplus()
                self._la[1] = Mom(self._p).pperp()
        return self._la / sqrt(Mom(self._p).pplus())

    def lat(self):
        if self._lat is None:
            self._lat = np.zeros((2,1), dtype = complex)
            if Mom(self._p).pplus() != 0:
                self._lat[0] = Mom(self._p).pplus()
                self._lat[1] = Mom(self._p).pperm()
        return self._lat / sqrt(Mom(self._p).pplus())

    def cla(self):
        if self._cla is None:
            self._cla = np.zeros((2,1), dtype = complex)
            if Mom(self._p).pplus() != 0:
                self._cla[0] = -Mom(self._p).pperp()
                self._cla[1] = Mom(self._p).pplus()
        return self._cla / sqrt(Mom(self._p).pplus())

    def clat(self):
        if self._lat is None:
            self._clat = np.zeros((2,1), dtype = complex)
            if Mom(self._p).pplus() != 0:
                self._clat[0] = -Mom(self._p).pperm()
                self._clat[1] = Mom(self._p).pplus()
        return self._clat / sqrt(Mom(self._p).pplus())
```

Class Sp6Ka: construct kappa components of 6D spinors

The p4, p5-related components required to construct 6-d spinors are (Bern 2011 pg 4):

- **ka** = kappa = (p5 - ip4) / < cla_p la_q >
- **kat** = kappatilde = (p5 + ip4) / [lat_q clat_p]
- **kap** = kappaprime = (p5 + ip4) / < cla_p la_q >
- **kapt** = kappaprimetilde = (p5 - ip4) / [lat_q clat_p]

(Test for denominator = 0 is applied but this should never occur because reference vector is defined as one for which q.p != 0.)

In [5]:
```
# Products spaa and spbb now defined consistently with Bern et al

class Ka6:
    def __init__(self, p):
        self.p     = p
        self.pflat = pflat(p)
        self.q     = refq(p)
        self._spaa = sp22(Spinor2(self.q).cla(), Spinor2(self.pflat).la())
        self._spbb = sp22(Spinor2(self.pflat).lat(), Spinor2(self.q).clat())
        self.c     = (self.p, self.pflat, self.q, self._spaa, self._spbb)
        self._ka   = None
        self._kat  = None
        self._kap  = None
        self._kapt = None

    def __str__(self):
        out = "Sp6Ka: Kappa components of 6D spinors from\
               #being {}".format(','.join([str(cc) for cc in self.c]))
        return out
    def __repr__(self):
        return "Sp6Ka({})".format(','.join([str(cc) for cc in self.c]))


    def ka(self):
        num = Mom(self.p).pmass()
        if self._spaa == 0: self._ka = 0
        else: self._ka = num / self._spaa
```

```
        return self._ka

    def kat(self):
        num = Mom(self.p).pmasst()
        if self._spbb == 0: self._kat = 0
        else: self._kat = num / self._spbb
        return self._kat

    def kap(self):
        num = Mom(self.p).pmasst()
        if self._spaa == 0: self._kap = 0
        else: self._kap = num / self._spaa
        return self._kap

    def kapt(self):
        num = Mom(self.p).pmass()
        if self._spbb == 0: self._kapt = 0
        else: self._kapt = num / self._spbb
        return self._kapt
```

Class Spinor6: spinor solutions of 6D Dirac equation

The $ \Lambda_a $ and $ \Lambda_{\dot{a}} $ solutions are found by solving the 6D Dirac equation. The $ a, \dot{a} $ indices are raised using $ \epsilon^{ab} $.

6d massless spinors are then expressed in terms of massless 4d spinors via the expressions

- **sp6d** $ =\ \Lambda^A_a \ = \begin{bmatrix} \kappa' \mu_{\alpha} & \lambda_{\alpha}\\ \tilde{\lambda}^{\dot{\alpha}} & -\tilde{\kappa}' \tilde{\mu}^{\dot{\alpha}}\end{bmatrix} = \begin{bmatrix} \text{kap cla(q)} & \text{cla(pflat)} \\ \text{lat(pflat)} & \text{-kapt lat(q)} \end{bmatrix}$
- **sp6td** $ = \tilde{\Lambda}_{A \dot{a}} = \begin{bmatrix} \kappa \mu^{\alpha} & \lambda^{\alpha}\\ \tilde{\lambda}_{\dot{\alpha}} & -\tilde{\kappa} \tilde{\mu}_{\dot{\alpha}}\end{bmatrix} = \begin{bmatrix} \text{ka la(q)} & \text{la(pflat)} \\ \text{clat(pflat)} & -\text{kat clat(q)} \end{bmatrix}$
- **sp6u** $ =\ \Lambda^{Aa} = \ \epsilon^{ab}\Lambda^{Ab} $
- **sp6tu** $ = \tilde{\Lambda}_{A}^{\dot{a}} \ = \epsilon^{\dot{a}\dot{b}}\tilde{\Lambda}_{A\dot{b}} $

The two columns in sp6, sp6t correspond to the little group index a and adot taking value 1 or 2 respectively. The A index represents the rows.

The 6d massless condition is p^2 = 4p^2 - p4^2 - p5^2 = 4p^2 - mass x masstilde = 0. Therefore these massless 6d spinors can represent 4-momenta via a combination of massless 4d spinors.

In the special case that SixMomentum has p4, p5 = 0, i.e. 4-momentum is massless, the ka-type components are zero and the 6-d spinors simplify to 4-d spinors.

Note that the Bern et al definitions differ:

- $ =\ \Lambda^A_a \ = \begin{bmatrix} -\kappa \mu_{\alpha} & \lambda_{\alpha}\\ \tilde{\lambda}^{\dot{\alpha}} & \tilde{\kappa} \tilde{\mu}^{\dot{\alpha}}\end{bmatrix} = \begin{bmatrix} \text{-ka cla(q)} & \text{cla(pflat)} \\ \text{lat(pflat)} & \text{kat lat(q)} \end{bmatrix}$
- $ = \tilde{\Lambda}_{A \dot{a}} = \begin{bmatrix} \kappa' \mu^{\alpha} & \lambda^{\alpha}\\ -\tilde{\lambda}_{\dot{\alpha}} & \tilde{\kappa}' \tilde{\mu}_{\dot{\alpha}}\end{bmatrix} = \begin{bmatrix} \text{kap la(q)} & \text{la(pflat)} \\ \text{-clat(pflat)} & \text{kapt clat(q)} \end{bmatrix}$

The differences in $ \kappa $ definitions are:

- $ m \leftrightarrow \tilde{m} $
- $ \langle \mu \lambda \rangle \leftrightarrow \langle \lambda \mu \rangle $, i.e. factor of -1
- $ [\lambda \mu] \leftrightarrow [\mu \lambda] $, similarly a factor of -1

Bern also has a factor of -1 in column 1 of $ \tilde{\Lambda}_{A \dot{a}} $
Coding the Bern et al definitons in the conventions used in gghj does not solve the Dirac equation.

In [6]:
```
class Spinor6:

    def __init__(self, p):
        self.p      = p
        self.pflat  = pflat(p)
        self.q      = refq(p)
        self.la_p   = Spinor2(self.pflat).la()
        self.lat_p  = Spinor2(self.pflat).lat()
        self.cla_p  = Spinor2(self.pflat).cla()
        self.clat_p = Spinor2(self.pflat).clat()
        self.la_q   = Spinor2(self.q).la()
        self.lat_q  = Spinor2(self.q).lat()
        self.cla_q  = Spinor2(self.q).cla()
        self.clat_q = Spinor2(self.q).clat()
        self._sp6d  = None
        self._sp6td = None
        self._sp6u  = None
        self._sp6tu = None

    #def __str__(self):
        #out = "Spinor6: 6-d spinors expressed in terms of massless 4-d spinors\
              #with momenta{}".format(','.join([str(cc) for cc in self.c]))
        #return out
    #def __repr__(self):
        #return "Spinor6({})".format(','.join([str(cc) for cc in self.c]))

    def sp6(self):
        if self._sp6d is None:
            self._sp6d      =  np.zeros((4,2), dtype = complex)
            self._sp6d[0,0] =  (Ka6(self.p).kap() * self.cla_q[0])[0]
            self._sp6d[1,0] =  (Ka6(self.p).kap() * self.cla_q[1])[0]
            self._sp6d[2,0] =  (self.lat_p[0])[0]
            self._sp6d[3,0] =  (self.lat_p[1])[0]
            self._sp6d[0,1] =  (self.cla_p[0])[0]
            self._sp6d[1,1] =  (self.cla_p[1])[0]
            self._sp6d[2,1] =  -(Ka6(self.p).kapt() * self.lat_q[0])[0]
            self._sp6d[3,1] =  -(Ka6(self.p).kapt() * self.lat_q[1])[0]

        if self._sp6u is None:
            self._sp6u      =  np.matmul(self._sp6d,lev2u)
```

```
        return self._sp6d, self._sp6u

    def sp6t(self):
        if self._sp6td is None:
            self._sp6td      =  np.zeros((4,2), dtype = complex)
            self._sp6td[0,0] =  (Ka6(self.p).ka() * self.la_q[0])[0]
            self._sp6td[1,0] =  (Ka6(self.p).ka() * self.la_q[1])[0]
            self._sp6td[2,0] =  (self.clat_p[0])[0]
            self._sp6td[3,0] =  (self.clat_p[1])[0]
            self._sp6td[0,1] =  (self.la_p[0])[0]
            self._sp6td[1,1] =  (self.la_p[1])[0]
            self._sp6td[2,1] =  (-Ka6(self.p).kat() * self.clat_q[0])[0]
            self._sp6td[3,1] =  (-Ka6(self.p).kat() * self.clat_q[1])[0]

        if self._sp6tu is None:
            self._sp6tu      = np.matmul(self._sp6td,lev2u)

        return self._sp6td, self._sp6tu
```

Function definitions, including spinor products

Functions momSpinors: obtain spinors from positive or negative momenta

Our convention is all momenta treated as outgoing. Incoming momenta, therefore, are negative. For all-gluon amplitudes, the prescription for a spinor for a negative momentum is:
$$ \Lambda_{(-p)} = i \Lambda_{(p)} $$

In [7]:
```
def posS(p):
    spd,spu = Spinor6(p).sp6()
    sptd,sptu = Spinor6(p).sp6t()
    return spd,spu,sptd,sptu

def negS(p):
    spd,spu = Spinor6(-p).sp6()
    sptd,sptu = Spinor6(-p).sp6t()
    spd *= 1j
    spu *= 1j
    sptd *= 1j
    sptu *= 1j
    return spd,spu,sptd,sptu

def momSpinors(p):
    if p[0] < 0:
        spd, spu,sptd,sptu = negS(p)
    else:
        spd,spu,sptd,sptu = posS(p)
    return spd,sptd,spu,sptu
```

Functions pFromS6D: reconstruct momentum from spinors

To obtain momenta from spinors use the following:
$$ p_{\mu} = - \frac{1}{4} \langle p_a | \Sigma_{\mu} | p_b \rangle \epsilon^{ab} = - \frac{1}{4} [p_{\dot{a}}|\tilde{\Sigma}_{\mu}|p_{\dot{b}}] \epsilon^{\dot{a}\dot{b}} $$

NOTE THIS REPLACES pFromS and pFromSt in randrot, which do not use correct formula for 6D spinors

In [8]:
```
def pFromS6D(s):
    res= np.zeros(6, dtype='complex128')
    for i in range(6):
        prod1 = np.dot(s,-lev2u)   # lev2u is epsab, here need epsba so -lev2u
        prod2 = np.dot(s.T, np.dot(gammastd[i], prod1))
        component = -0.25*np.trace(prod2)
        res[i] = component
    return res

def pFromSt6D(st):
    res= np.zeros(6, dtype='complex128')
    for i in range(6):
        prod1 = np.dot(st,-lev2u)    # lev2u is epsab, here need epsba so -lev2u
        prod2 = np.dot(st.T, np.dot(gammasd[i], prod1))
        component = -0.25*np.trace(prod2)
        res[i] = component
    return res
```

Functions sp22: contractions of two Weyl spinors

This function takes 2D spinors as arguments.

For two particles i and j we calculate contractions of Weyl spinors with the following definitions (H&P p6 16, 175):

- < ij > = - < ji > = lambda_i^alpha x lambda_j_alpha
- [ij] = - [ji] = lambdatilde_i_alphadot x lambdatilde_j^alphadot
  etc

Bern et al clarify: $ \langle ij \rangle = \epsilon^{\alpha \beta} \lambda_{i \beta} \lambda_{j \alpha} $, $ \ \ \ [ij] = \epsilon_{\dot{\alpha} \dot{\beta}} \tilde{\lambda}^{\dot{\beta}} \tilde{\lambda}^{\dot{\alpha}} $, $ \ \ \ \langle ij \rangle [ji] = s_{ij} $

In [9]:
```
def sp22(sp2i, sp2j):
    sp22 = sp2i[0] * sp2j[0] + sp2i[1] * sp2j[1]
    return sp22
```

In [10]:
```
def spaa(p,q):
    return sp22(Spinor2(p).cla(), Spinor2(q).la())
```

In [11]:
```
def spbb(p,q):
    return sp22(Spinor2(p).lat(), Spinor2(q).clat())
```

Functions sp62: contractions of two 6D spinors

Takes as arguments two 6D spinors.

For two particles i and j we calculate 2 x 2 6D spinor inner products with the following definitions:

- < i^a | j_bdot] = Lambda_i^Aa x Lambdatilde_j_Abdot = [j_bdot | i^a >
- < i_a | j^bdot] = Lambda_ia^A x Lambdatilde_jA^bdot = [j^bdot | i_a >
- < i_a | j_bdot] = Lambda_ia^A x Lambdatilde_jAbdot = [j_bdot | i_a >
- < i^a | j^bdot] = Lambda_i^Aa x Lambdatilde_jA^bdot = [j^bdot | i^a >

sp62(ij) = **sp6xx**(p_i) **sp6xx**(p_j)

In [12]:
```
def sp62(sp6i, sp6j, a, b):
    sp62 = 0
    for A in (0,1,2,3):
        sp62 += sp6i[A,a] * sp6j[A,b]
    return sp62
```

In [13]:
```
# For all helicities:

def sp62_all_old(sp6i, sp6j):
    sp62 = np.zeros((2,2), dtype = complex)
    for i in (0,1,2,3):
        sp62[0,0] += sp6i[i,0] * sp6j[i,0]
        sp62[1,0] += sp6i[i,1] * sp6j[i,0]
        sp62[0,1] += sp6i[i,0] * sp6j[i,1]
        sp62[1,1] += sp6i[i,1] * sp6j[i,1]
    return sp62

def sp62_all(sp6i, sp6j):
    sp62 = np.matmul(np.transpose(sp6i),sp6j)
    return sp62
```

Shorthand, specified by momenta:

In [14]:
```
def sp6_dd(p1, p2):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd2,sptd2,spu2,sptu2 = momSpinors(p2)
    return sp62_all(spd1,sptd2)
```

Functions sp64: contractions of four 6D spinors

Takes as arguments four 6D spinors and their specified helicities.

- < i_a j_b k_c l_d > = levi4_ABCD. Lambda_i^A_a. Lambda_j^B_b. Lambda_k^C_c. Lambda_l^D_d
- < i^a j^b k^c l^d > = levi4_ABCD. Lambda_i^Aa. Lambda_j^Bb. Lambda_k^Cc. Lambda_l^Dd

sp64 = tensor contraction of outer product of [**lev4**_ABCD . **sp6**(A,i) . **sp6**(B,j) . **sp6**(C,k) . **sp6**(D,l)]

In [15]:
```
def sp64(sp6i,sp6j,sp6k,sp6l,a,b,c,d):
    # This function calculates a single number, for specified helicities
    # Add an error exit here for a,b,c,d not 0,1
    sp64 = 0
    for A in range(4):
        for B in range(4):
            for C in range(4):
                for D in range(4):
                    if lev4[A,B,C,D] == 1:
                        sp64 += (sp6i[A,a] * sp6j[B,b] * sp6k[C,c] * sp6l[D,d])
                    if lev4[A,B,C,D] == -1:
                        sp64 -= (sp6i[A,a] * sp6j[B,b] * sp6k[C,c] * sp6l[D,d])
    return sp64
```

Functions sp6s: spinor strings

These functions take two 6D spinors and a specified number of 6-momenta (contracted with Sigma matrices)

- < 1_a | P2slash | 3_b > = Lambda^A_a P^AB Lambda^B_b

  sp6s1 = **sp6d**(p1) **slash(p2)** **sp6d**(p3)
- < i_a | Pslash_3 Pslash_4 | j^a]

  sp6s2 = **sp6d**(p1) **slash**(p2) **slasht**(p3) **sp6td**(p4)

  etc

In [16]:
```
def sp6s1(p1,p2,p3,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd3,sptd3,spu3,sptu3 = momSpinors(p3)
    sp1 = spd1
    sl2 = slash(p2)
    sp3 = spd3
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(sl2,sp3[:,0])
    spm[:,1] = np.matmul(sl2,sp3[:,1])

    return sp62(sp1, spm, a, b)
```

In [17]:
```
def sp6st1(p1,p2,p3,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd3,sptd3,spu3,sptu3 = momSpinors(p3)
    sp1 = sptd1
    sl2 = slash(p2)
    sp3 = sptd3
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(sl2,sp3[:,0])
    spm[:,1] = np.matmul(sl2,sp3[:,1])

    return sp62(sp1, spm, a, b)
```

In [18]:
```
def sp6s2(p1,p2,p3,p4,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd4,sptd4,spu4,sptu4 = momSpinors(p4)
```

```
    sp1 = spd1
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sp4 = spd4
    step1 = np.matmul(sl2,sl3)
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(step1,sp4[:,0])
    spm[:,1] = np.matmul(step1,sp4[:,1])

    return sp62(sp1, spm, a, b)
```

In [19]:
```
def sp6s3(p1,p2,p3,p4,p5,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd5,sptd5,spu5,sptu5 = momSpinors(p5)
    sp1 = spd1
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sl4 = slash(p4)
    sp5 = spd5
    step1 = np.matmul(sl2,np.matmul(sl3,sl4))
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(step1,sp5[:,0])
    spm[:,1] = np.matmul(step1,sp5[:,1])

    return sp62(sp1, spm, a, b)
```

In [20]:
```
def sp6st3(p1,p2,p3,p4,p5,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd5,sptd5,spu5,sptu5 = momSpinors(p5)
    sp1 = sptd1
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sl4 = slash(p4)
    sp5 = sptd5
    step1 = np.matmul(sl2,np.matmul(sl3,sl4))
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(step1,sp5[:,0])
    spm[:,1] = np.matmul(step1,sp5[:,1])

    return sp62(sp1, spm, a, b)
```

In [21]:
```
def sp6s4(p1,p2,p3,p4,p5,p6,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd6,sptd6,spu6,sptu6 = momSpinors(p6)
    sp1 = spd1
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sl4 = slash(p4)
    sl5 = slasht(p5)
    sp6 = spd6
    step1 = np.matmul(sl3,np.matmul(sl4,sl5))
    step2 = np.matmul(sl2,step1)
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(step2,sp6[:,0])
    spm[:,1] = np.matmul(step2,sp6[:,1])

    return sp62(sp1, spm, a, b)
```

In [ ]:
```
def sp6s5(p1,p2,p3,p4,p5,p6,p7,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd7,sptd7,spu7,sptu7 = momSpinors(p7)
    sp1 = spd1
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sl4 = slash(p4)
    sl5 = slasht(p5)
    sl6 = slash(p6)
    sp7 = spd7
    step1 = np.matmul(sl4,np.matmul(sl5,sl6))
    step2 = np.matmul(sl2,np.matmul(sl3,step1))
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(step2,sp7[:,0])
    spm[:,1] = np.matmul(step2,sp7[:,1])

    return sp62(sp1, spm, a, b)
```

In [ ]:
```
def sp6st5(p1,p2,p3,p4,p5,p6,p7,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd6,sptd6,spu6,sptu6 = momSpinors(p7)
    sp1 = sptd1
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sl4 = slash(p4)
    sl5 = slasht(p5)
    sl6 = slash(p6)
    sp7 = sptd7
    step1 = np.matmul(sl4,np.matmul(sl5,sl6))
    step2 = np.matmul(sl2,np.matmul(sl3,step1))
    spm = np.empty([4,2], dtype = complex)
    spm[:,0] = np.matmul(step2,sp7[:,0])
    spm[:,1] = np.matmul(step2,sp7[:,1])

    return sp62(sp1, spm, a, b)
```

In [ ]:
```
def sp6s6(p1,p2,p3,p4,p5,p6,p7,p8,a,b):
    spd1,sptd1,spu1,sptu1 = momSpinors(p1)
    spd8,sptd8,spu8,sptu8 = momSpinors(p8)
    sp1 = spd8
    sl2 = slash(p2)
    sl3 = slasht(p3)
    sl4 = slash(p4)
    sl5 = slasht(p5)
    sl6 = slash(p6)
    sl7 = slasht(p7)
    sp8 = sptd8
    step1 = np.matmul(sl5,np.matmul(sl6,sl7))
    step2 = np.matmul(sl3,np.matmul(sl4,step1))
```

F.4 treewithspinors

Class	*Function*		
	g4	$A_4(1_{a\dot{a}}, 2_{b\dot{b}}, 3_{c\dot{c}}, 4_{d\dot{d}})$	All-gluon 4-point amplitude
	auX	$x^a \tilde{x}^{\dot{a}}$	Auxiliary matrix
	pol	$\mathcal{E}^{\mu}_{a\dot{a}}$	Polarisation vector
	rv	r^{AB}	Shift vector
	zPole	z_{2j}	Shift parameter z
ZSpinor	zsp1	$\Lambda^a_{\hat{1}}, \Lambda_{\hat{1}a}$	Spinors for shifted momenta
	zsp2	$\Lambda^b_{\hat{2}}, \Lambda_{\hat{2}b}$	
	zspt1	$\tilde{\Lambda}^{\dot{a}}_{\hat{1}}, \tilde{\Lambda}_{\hat{1}\dot{a}}$	
	zspt2	$\tilde{\Lambda}^{\dot{b}}_{\hat{2}}, \tilde{\Lambda}_{\hat{2}\dot{b}}$	
	pHatZs	$\hat{p}_1, \hat{p}_1$	Shifted momenta from Zspinors
	pHat	$\hat{p}_1, \hat{p}_1$	Shifted momenta from momenta
	spFor3g	$\Lambda_{\hat{1}a} ... \Lambda_{\hat{2}j}$	Spinors for three-point momenta
	uutilde	$u, \tilde{u}, w, \tilde{w}$	2×1 components of spinor products
	g3Gamma	$\Gamma_{abc}\tilde{\Gamma}_{\dot{b}\dot{c}}$	Components of 3-point amplitude
	g3Amp	$A_3(1_{a\dot{a}}, 2_{b\dot{b}}, 3_{c\dot{c}})$	Three-point amplitude

tree_with _spinors

Import standard packages

In [1]:
```
import numpy as np
from cmath import sqrt
from cmath import exp
import scipy
from scipy.sparse import csr_matrix # Used to find non-zero elements in g3 normalisation
from matplotlib import pyplot as plt
```

Import other gghj modules

In [2]:
```
from utils import *
import spinors as sp
import randrot as rr # random momenta and phase space points for testing
```

Set up data for preliminary testing

Momenta

In [3]:
```
# As used in Mathematica check
dp1d = np.array([sqrt(13*13 + 3*3 + 7*7 + 2*2 + 2*2), 13,-3,-7,2,-2])
dp2d = np.array([sqrt(9*9 + 7*7 + 5*5 + 1 + 7*7), -9,7,-5,1,-7])
```

Phase space points

In [4]:
```
def psp_from_rr46(seed):
    # A 4D with trailing zeros phase space point from randrot
    psp_for_test = rr.phaseSpacePoint46(4,2,seed)
    psp1 = psp_for_test[0,:]
    psp2 = psp_for_test[1,:]
    psp3 = psp_for_test[2,:]
    psp4 = psp_for_test[3,:]

    return psp1,psp2,psp3,psp4

def psp_from_rr6(seed):
    # A 6D phase space point from randrot
    psp_for_test = rr.phaseSpacePoint6(4,2,seed)
    psp1 = psp_for_test[0,:]
    psp2 = psp_for_test[1,:]
    psp3 = psp_for_test[2,:]
    psp4 = psp_for_test[3,:]

    return psp1,psp2,psp3,psp4

def psp_from_SaM(seed):
    # A 4D phase space point from S@M with trailing zeros
    # Note that there are only 100 of these in the set, so seed cannot exceed 100

    if seed > 100:
        print('psp_from_S@M seed outside range 100, replaced by 1')
        seed = 1

    psp1 = rr.momset4d()[seed,0,0:]
    psp2 = rr.momset4d()[seed,1,0:]
    psp3 = rr.momset4d()[seed,2,0:]
    psp4 = rr.momset4d()[seed,3,0:]

    return psp1,psp2,psp3,psp4

def psp_from_DMtensors():
    psp1 = np.array([np.sqrt(1+1+4+1+(1.5)**2)*-1,1,1,2,1,1.5])
    psp2 = np.array([np.sqrt(1+1+4+1+1),-1,-1,-2,-1,1])
    psp3 = np.array([np.sqrt(4+1+1+1+1)*-1,2,-1,-1,1,-1])
    psp4 = np.array([np.sqrt(4+1+1+1+1.5**2),-2,1,1,-1,-1.5])
    #print(psp1)
    #print(psp2)
    #print(psp3)
    #print(psp4)
    for psp in (psp1,psp2,psp3,psp4):
        assert np.allclose(MP6(psp,psp),0)
        #print(MP6(psp,psp))

    return psp1,psp2,psp3,psp4


psp1_1,psp1_2,psp1_3,psp1_4 = psp_from_rr46(1234)
psp2_1,psp2_2,psp2_3,psp2_4 = psp_from_SaM(1)
psp3_1,psp3_2,psp3_3,psp3_4 = psp_from_rr6(1234)
psp4_1,psp4_2,psp4_3,psp4_4 = psp_from_DMtensors()
```

In [5]:
```
np.savez('rr6.npz',p1=psp3_1, p2=psp3_2, p3=psp3_3, p4=psp3_4)
np.savez('rr46.npz',p1=psp1_1, p2=psp1_2, p3=psp1_3, p4=psp1_4)
np.savez('sam.npz',p1=psp2_1, p2=psp2_2, p3=psp2_3, p4=psp2_4)
```

Load the S@M-generated amplitude sets

For testing, 3 helicity configurations for each phase space point

In [6]:
```
# Load the S@M-generated amplitude sets: 3-helicity configurations for each phase space point

def ampset4d():
    ampset4d = np.zeros((100,3,1), dtype = complex)
    rawsam = np.loadtxt("sam_amplitude_sets.txt")
    n = 0
```

```
    for a in range (100):
        for row in range(3):
            ampset4d[a,row,0] = -rawsam[n,1] + 1j *rawsam[n,0]
            n += 1
    return ampset4d
```

Tree amplitudes - 4-point

The formulae used here for 4-point amplitudes are given in the Catalogue appendix:

- for **g4** see D.2.1
- for **g3h1** see D.2.2
- for **g2s2** see D.2.3
- for **g1h1s2** see D.2.4

All gluon 4-point

In [7]:
```
# Set up g4 to calculate for all helicities using momSpinors(p)

def g4(p1,p2,p3,p4):
    spd1,sptd1,spu1,sptu1 = sp.momSpinors(p1)
    spd2,sptd2,spu2,sptu2 = sp.momSpinors(p2)
    spd3,sptd3,spu3,sptu3 = sp.momSpinors(p3)
    spd4,sptd4,spu4,sptu4 = sp.momSpinors(p4)
    res = np.zeros([2,2,2,2,2,2,2,2], dtype = complex)

    for a in (0,1):
        for ad in (0,1):
            for b in (0,1):
                for bd in (0,1):
                    for c in (0,1):
                        for cd in (0,1):
                            for d in (0,1):
                                for dd in (0,1):
                                    if a==ad and b==bd and c==cd and d==dd:
                                        res[a,ad,b,bd,c,cd,d,dd] += (sp.sp64(spd1,spd2,spd3,spd4,a,b,c,d) *
                                                                     sp.sp64(sptd1,sptd2,sptd3,sptd4,ad,bd,cd,dd))
    res *= - 1j / (sij6(p1,p2) * sij6(p2,p3))
    return(res)

# Set up g4 to calculate for specified helicities using momSpinors(p)

def g4_hel(p1,p2,p3,p4,a,ad,b,bd,c,cd,d,dd):
    spd1,sptd1,spu1,sptu1 = sp.momSpinors(p1)
    spd2,sptd2,spu2,sptu2 = sp.momSpinors(p2)
    spd3,sptd3,spu3,sptu3 = sp.momSpinors(p3)
    spd4,sptd4,spu4,sptu4 = sp.momSpinors(p4)

    sp64  = sp.sp64(spd1,spd2,spd3,spd4,a,b,c,d)
    sp64t = sp.sp64(sptd1,sptd2,sptd3,sptd4,ad,bd,cd,dd)
    res = sp64 * sp64t
    res *= (-1j / (sij6(p1,p2) * sij6(p2,p3)))
    return(res)
```

In [8]:
```
# TESTS ARE ALSO IN TESTING SECTION

# First check that g4 and g4_hel give the same results

def test_g4_1(p1,p2,p3,p4):
    testg4 = g4(p1,p2,p3,p4)
    for a in (0,1):
        for ad in (0,1):
            for b in (0,1):
                for bd in (0,1):
                    for c in (0,1):
                        for cd in (0,1):
                            for d in (0,1):
                                for dd in (0,1):
                                    if a==ad and b==bd and c==cd and d==dd:
                                        testg4hel = g4_hel(p1,p2,p3,p4,a,ad,b,bd,c,cd,d,dd)
                                        assert abs(testg4hel) == abs(testg4[a,ad,b,bd,c,cd,d,dd])


# Test g4 for 4D momenta agrees to amplitudes calculated in S@M (abs value)

# Load the S@M-generated amplitude sets: 3-helicity configurations for each phase space point

def ampset4d():
    ampset4d = np.zeros((100,3,1), dtype = complex)
    rawsam = np.loadtxt("sam_amplitude_sets.txt")
    n = 0
    for a in range (100):
        for row in range(3):
            ampset4d[a,row,0] = -rawsam[n,1] + 1j *rawsam[n,0]
            n += 1
    return ampset4d

def test_g4_2():
    samamp = ampset4d()
    for n in range(100):
        p0 = rr.sam4p[n,0,0:]
        p1 = rr.sam4p[n,1,0:]
        p2 = rr.sam4p[n,2,0:]
        p3 = rr.sam4p[n,3,0:]

        samamp_mmpp = samamp[n,0,0]
        g4_mmpp = g4_hel(p0,p1,p2,p3,1,1,1,1,0,0,0,0)
        assert np.allclose(samamp_mmpp,g4_mmpp)

        samamp_mpmp = samamp[n,1,0]
        g4_mpmp = g4_hel(p0,p1,p2,p3,1,1,0,0,1,1,0,0)
        assert np.allclose(samamp_mpmp,g4_mpmp)

        samamp_mppm = samamp[n,2,0]
```

```
        g4_mppm = g4_hel(p0,p1,p2,p3,1,1,0,0,0,0,1,1)
        assert np.allclose(samamp_mppm,g4_mppm)

def test_g4_3():
    # Use test data set psp2_1....
    samamp = ampset4d()
    p0 = rr.sam4p[1,0,0:]
    p1 = rr.sam4p[1,1,0:]
    p2 = rr.sam4p[1,2,0:]
    p3 = rr.sam4p[1,3,0:]

    samamp_mmpp = samamp[1,0,0]
    g4_mmpp = g4_hel(p0,p1,p2,p3,1,1,1,1,0,0,0,0)
    assert np.allclose(samamp_mmpp,g4_mmpp)

    samamp_mpmp = samamp[1,1,0]
    g4_mpmp = g4_hel(p0,p1,p2,p3,1,1,0,0,1,1,0,0)
    assert np.allclose(samamp_mpmp,g4_mpmp)

    samamp_mppm = samamp[1,2,0]
    g4_mppm = g4_hel(p0,p1,p2,p3,1,1,0,0,0,0,1,1)
    assert np.allclose(samamp_mppm,g4_mppm)

    print('test_g4_3')
    print('momenta')
    print(p0)
    print(p1)
    print(p2)
    print(p3)
    print('samamp_mmpp')
    print(samamp_mmpp)
    print('g4_mmpp = g4_hel(p0,p1,p2,p3,1,1,1,1,0,0,0,0)')
    print(g4_mmpp)
    print('samamp_mpmp')
    print(samamp_mpmp)
    print('g4_mmpp = g4_mpmp = g4_hel(p0,p1,p2,p3,1,1,0,0,1,1,0,0)')
    print(g4_mpmp)
    print('samamp_mmpp')
    print(samamp_mppm)
    print('g4_mppm = g4_hel(p0,p1,p2,p3,1,1,0,0,0,0,1,1)')
    print(g4_mppm)


test_g4_1(psp2_1,psp2_2,psp2_3,psp2_4)
test_g4_2()
test_g4_3()
```

```
test_g4_3
momenta
[ 1.23419873 -0.88976435 -0.60409097 -0.60550805  0.          0.        ]
[1.22926242 0.95260605 0.62170184 0.4659556  0.         0.         ]
[-1.17281121  0.50708293 -0.67729778 -0.8121704   0.          0.        ]
[-1.29064994 -0.56992463  0.65968691  0.95172284  0.          0.        ]
samamp_mmpp
(-0.5929259092487204-2.619816501532103j)
g4_mmpp = g4_hel(p0,p1,p2,p3,1,1,1,1,0,0,0,0)
(-0.5929259092487206-2.619816501532103j)
samamp_mpmp
(0.7478378985765556-0.7489166144556619j)
g4_mmpp = g4_mpmp = g4_hel(p0,p1,p2,p3,1,1,0,0,1,1,0,0)
(0.7478378985765558-0.7489166144556625j)
samamp_mmpp
(0.23370192933897643+0.2897991611863344j)
g4_mppm = g4_hel(p0,p1,p2,p3,1,1,0,0,0,0,1,1)
(0.23370192933897677+0.2897991611863346j)
```

BCFW

Functions to implement 6d BCFW shift between $ P_1 $ and $ P_2 $:

- Create an auxiliary matrix $ X^{a \dot{a}} = x^a \tilde{x}^{\dot{a}} $ where $ x^a, \tilde{x}^{\dot{a}} $ are arbitrary
- Define polarisation vector for $ P_1 $ with respect to $ P_2 $
- Calculate a null vector r which will satisfy $ P_1 . r = P_2 . r = 0 $
- Calculate sum of momenta, $ P_{2j} = P_2 + ... + P_j $
- Calculate $ z_{2j} $, the location of the pole at $ P_{2j}(z)^2 = 0:\ z_{2j} = \dfrac{-P^2_{2_j}}{2r.P_{2j}} $
- Calculate shifted spinors fr $ \hat{P}_1 = P_1 + zr, \hat{P}_2 = P_2 - zr $ using C&O'C eqns 5.7-10
- Use spinors to calculate unshifted amplitude as
  $ X^{a \dot{a}}A^{tree}_n(0) = \sum^{n-1}_{j=3} \sum_h X^{a \dot{a}} L(\hat{P}_2, ... , P_j, -\hat{P}^{-h}_{2j}) \times \frac{1}{P^2_{2j}}A_R(\hat{P}^h_{2j}, P_{j+1}, ... , \hat{P}_1)|_{z = z_{2j}} $

Auxiliary matrix

xuu: $ X^{a \dot{a}} = x^a \tilde{x}^{\dot{a}} $
xdd: $ X_{a \dot{a}} = x_a \tilde{x}_{\dot{a}} $
xud: $ X^{a}_{ \dot{a}} = x^a \tilde{x}_{\dot{a}} $
xdu: $ X_{a}^{\dot{a}} = x_a \tilde{x}^{\dot{a}} $

where $ x^a $ is arbitrary

In [9]:
```
def auXx():
    # Return separate x, xt matrices

    # Specify an arbitrary matrix for $ x^a, xt^a $
    xu  = np.array([1,0])
    xtu = np.array([1,0])
    # Lower indices
    xd = np.matmul(xu,lev2d)
    xtd = np.matmul(xtu,lev2d)

    return xu,xtu,xd,xtd
```

```
def auX():
    # Return X = x xt full auxiliary matrix

    # Obtain x components
    xu, xtu, xd, xtd = auXx()

    Xuu = np.outer(xu,xtu)
    assert np.linalg.det(Xuu) == 0

    Xud = np.outer(xu,xtd)
    assert np.linalg.det(Xud) == 0

    Xdd = np.outer(xd,xtd)
    assert np.linalg.det(Xdd) == 0

    Xdu= np.outer(xd,xtu)
    assert np.linalg.det(Xdu) == 0

    return Xuu, Xud, Xdd, Xdu
```

Polarisation vector

Originally copied in from spinors.py but now amended here to ensure p1.r = r.p2 = 0

With a null reference vector q, such that $p.q \neq 0$, there are associated spinors $ |q^a \rangle \langle q^b|\epsilon_{ab}$ and $q = |q_{\dot{a}}] [q_{\dot{b}}|\epsilon^{\dot{a}\dot{b}}$. The polarisation vectors are then defined by Cheung and O'Connell to be

$$ \mathcal{E}^{\mu}_{a \dot{a}} = \frac{1}{\sqrt{2}} \langle p_a|\Sigma^{\mu}|q_b\rangle (\langle q_b | p^{\dot{a}}])^{-1}$$ $$\mathcal{E}^{\mu}_{a \dot{a}} = \frac{1}{\sqrt{2}} (\langle p^a | q_{\dot{b}}])^{-1} [q_{\dot{b}}|\tilde{\Sigma}^{\mu}|p_{\dot{a}}] $$

Where $ \langle p^a | q_{\dot{b}}])^{-1} = -\dfrac{\langle p_a | q^{\dot{b}}]}{2 p.q}$

In [10]:
```
def pol(p1,p2):

    pol = np.zeros([4,6], dtype = complex)

    # Set up spinors
    p1_spd,p1_sptd,p1_spu,p1_sptu = sp.momSpinors(p1)
    p2_spd,p2_sptd,p2_spu,p2_sptu = sp.momSpinors(p2)

    # Obtain inverse of spinor product
    #inv21 = (-1/sij6(p1,p2))*sp.sp62_all(p2_spu,p1_sptd)
    inv21 = (1/sij6(p1,p2))*sp.sp62_all(p2_spu,p1_sptd)

    # Calculate 4 reference vectors, one for each helicity combination

    row = 0
    for a in (0,1):
        for ad in (0,1):
            for mu in range(6):
                #sig = gammas[mu]
                #sig = gammast[mu]
                sig = gammastd[mu]
                #sig = gammasd[mu]
                for b in (0,1):
                    pol[row,mu] += sp.sp62(p1_spd,np.matmul(sig,p2_spd),a,b) * inv21[b,ad]
            # Check each vector is null
            assert abs(MP6(pol[row],pol[row])) < 1e-14
            row += 1
    return pol/sqrt(2)
```

Shift vector

Vector for the shift is a null vector with the properties of a polarisation vector: $ r^{AB} = \frac{1}{\sqrt{2}}X^{a \dot{a}} (\mathcal{E}^{AB}_1)_{a \dot{a}} $

For the purposes of our BCFW shifted momenta we will use the polarisation of momentum p1 and choose the reference spinor for the polarisation vector to be Λ_2.

In [11]:
```
def rv(p1,p2):

    # Obtain auxiliary matrix
    Xuu, Xud, Xdd, Xdu = auX()
    #Xuu = np.array([[1,1],[1,1]])

    # Obtain polarisation vectors for all helicity combinations
    polv = pol(p1,p2)

    # Build shift vector
    rmu = np.zeros([6], dtype = complex)
    for mu in range(6):
        rmu[mu] += polv[0,mu] * Xuu[0,0]
        rmu[mu] += polv[1,mu] * Xuu[0,1]
        rmu[mu] += polv[2,mu] * Xuu[1,0]
        rmu[mu] += polv[3,mu] * Xuu[1,1]

    assert np.allclose(MP6(rmu,rmu),0)
    assert np.allclose(MP6(p1,rmu),0)
    assert np.allclose(MP6(rmu,p2),0)

    return rmu/sqrt(2)
```

z and location of pole

- Calculate location of pole $ z_{2j} = \dfrac{-P^2_{2_j}}{2r.P_{2j}} $, where $ P_{2j} = P_2 + ... + P_j $

In [12]:
```
def p2j(*args):
    # args are momenta
    p2j = np.zeros([6], dtype = complex)
    for p in args:
        p2j += p
```

```
    return p2j
```

In [13]:
```
def zPole(rv, p2j):
    # rv is shift vector, args are momenta
    # Calculate zPole
    return -MP6(p2j,p2j) / (2 * MP6(rv,p2j))
```

In [14]:
```
# Function is for test

def zGen(z0,t):
    # z0 is zPole(rv,p1,p2)
    theta = 2
    res = z0 + t * exp(1j * theta)
    return res
```

In [15]:
```
# Obtain shift parameters for test data

r_1 = rv(psp1_1,psp1_2)
p2j_1 = p2j(psp1_2,psp1_3)
z_1 = zPole(r_1,p2j_1)

r_2 = rv(psp2_1,psp2_2)
p2j_2 = p2j(psp2_2,psp2_3)
z_2 = zPole(r_2,p2j_2)

r_3 = rv(psp3_1,psp3_2)
p2j_3 = p2j(psp3_2,psp3_3)
z_3 = zPole(r_3,p2j_3)

r_4 = rv(psp4_1,psp4_2)
p2j_4 = p2j(psp4_2,psp4_3)
z_4 = zPole(r_4,p2j_4)
```

Spinors for BCFW shift

NEW DEFINITIONS

ZSpinors rewritten using Bern definitions and $ \epsilon $ to raise/lower indices

$$ \Lambda^{Aa}_{\hat{1}} \ \ = \Lambda^{Aa}_1 - \frac{z}{s_{12}} X^{a}_{\dot{a}} [1^{\dot{a}} | 2^b \rangle \Lambda^A_{2b} $$

$$ \Lambda^{Ab}_{\hat{2}} \ \ = \Lambda^{Ab}_2 - \frac{z}{s_{12}} X^{a}_{\dot{a}} \Lambda^A_{1a} [1^{\dot{a}} | 2^b \rangle$$

$$ \tilde{\Lambda}_{\hat{1} A \dot{a}} \ \ = \tilde{\Lambda}_{1A \dot{a}} + \frac{z}{s_{12}} X^{a}_{\dot{a}} \langle 1_a | 2_{\dot{b}}] \tilde{\Lambda}^{\dot{b}}_{2A} $$

$$ \tilde{\Lambda}_{\hat{2}A \dot{b}} \ \ = \tilde{\Lambda}_{2A \dot{b}} + \frac{z}{s_{12}} X^{a}_{\dot{a}} \tilde{\Lambda}_{1A}^{\dot{a}} \langle 1_a | 2_{\dot{b}}] $$

In [16]:
```
class ZSpinors:
    def __init__(self,p1,p2,z):
        # For amplitude calculations, z is zPole
        self.p1 = p1
        self.p2 = p2
        self.z  = z
        # Obtain unshifted p1,p2 spinors
        self.spd1, self.sptd1, self.spu1, self.sptu1 = sp.momSpinors(self.p1)
        self.spd2, self.sptd2, self.spu2, self.sptu2 = sp.momSpinors(self.p2)
        # Bring in auxiliary matrix components
        self.Xuu,self.Xud, self.Xdd, self.Xdu = auX()

        self._zspu1  = None
        self._zspd1 = None
        self._zsptu1  = None
        self._zsptd1  = None
        self._zspu2  = None
        self._zspd2  = None
        self._zsptu2  = None
        self._zsptd2  = None

    def zsp1(self):
        if self._zspu1 is None:
            self.spinprod1 = sp.sp62_all(self.sptu1,self.spu2)
            self.spinprod2 = np.matmul(self.spinprod1,np.transpose(self.spd2))
            self.spinprod3 = np.transpose(np.matmul(self.Xud,self.spinprod2))
            self._zspu1 = self.spu1 + (self.z / sij6(self.p1,self.p2)) * self.spinprod3
            self._zspd1 = np.matmul(self._zspu1,lev2d)
        return self._zspu1, self._zspd1

    def zsp2(self):
        if self._zspu2 is None:
            self.spinprod1 = np.matmul(np.transpose(self.Xud), np.transpose(self.spd1))
            self.spinprod2 = sp.sp62_all(self.sptu1,self.spu2)
            self.spinprod3 = np.matmul(np.transpose(self.spinprod1), self.spinprod2)
            self._zspu2 = self.spu2 + (self.z / sij6(self.p1,self.p2)) * self.spinprod3
            self._zspd2 = np.matmul(self._zspu2,lev2d)
        return self._zspu2, self._zspd2

    def zspt1(self):
        if self._zsptd1 is None:
            self.spinprod1 = sp.sp62_all(self.spd1,self.sptd2)
            self.spinprod2 = np.matmul(np.transpose(self.Xud), self.spinprod1)
            self.spinprod3 = np.matmul(self.spinprod2,np.transpose(self.sptu2))
            self._zsptd1 = self.sptd1 - (self.z / sij6(self.p1,self.p2)) * np.transpose(self.spinprod3)
            self._zsptu1 = np.matmul(self._zsptd1,lev2u)
        return self._zsptu1, self._zsptd1

    def zspt2(self):
        if self._zsptd2 is None:
            self.spinprod1 = np.matmul(self.Xud, np.transpose(self.sptu1))
            self.spinprod2 = sp.sp62_all(self.spd1,self.sptd2)
            self.spinprod3 = np.matmul(np.transpose(self.spinprod1),self.spinprod2)
            self._zsptd2 = self.sptd2 - (self.z / sij6(self.p1,self.p2)) * self.spinprod3
            self._zsptu2 = np.matmul(self._zsptd2,lev2u)
        return self._zsptu2, self._zsptd2
```

In [17]:
```
def test_ZSpinor6_dirac(p1,p2,z):

    zspu1,zspd1 = ZSpinors(p1,p2,z).zsp1()
    zspu2,zspd2 = ZSpinors(p1,p2,z).zsp2()
    zsptu1,zsptd1= ZSpinors(p1,p2,z).zspt1()
    zsptu2,zsptd2= ZSpinors(p1,p2,z).zspt2()

    p1h = sp.pFromS6D(zspd1)
    p2h = sp.pFromS6D(zspd2)

    assert np.allclose(np.matmul(slasht(p1h), zspd1),0, atol = 1e-10)
    assert np.allclose(np.matmul(slasht(p2h), zspd2),0, atol = 1e-10)
    assert np.allclose(np.matmul(slash(p1h), zsptd1),0, atol = 1e-10)
    assert np.allclose(np.matmul(slash(p2h), zsptd2),0, atol = 1e-10)
    assert np.allclose(np.matmul(slasht(p1h), zspu1),0, atol = 1e-10)
    assert np.allclose(np.matmul(slasht(p2h), zspu2),0, atol = 1e-10)
    assert np.allclose(np.matmul(slash(p1h), zsptd1),0, atol = 1e-10)
    assert np.allclose(np.matmul(slash(p2h), zsptd2),0, atol = 1e-10)

def test_ZSpinor6_pFromS(p1,p2,z):

    zspu1,zspd1 = ZSpinors(p1,p2,z).zsp1()
    zspu2,zspd2 = ZSpinors(p1,p2,z).zsp2()
    zsptu1,zsptd1= ZSpinors(p1,p2,z).zspt1()
    zsptu2,zsptd2= ZSpinors(p1,p2,z).zspt2()

    assert np.allclose(sp.pFromS6D(zspu1),sp.pFromS6D(zspd1))
    assert np.allclose(sp.pFromSt6D(zsptu1),sp.pFromSt6D(zsptd1))
    assert np.allclose(sp.pFromS6D(zspu1),sp.pFromSt6D(zsptd1))
    assert np.allclose(sp.pFromS6D(zspu2),sp.pFromS6D(zspd2))
    assert np.allclose(sp.pFromSt6D(zsptu2),sp.pFromSt6D(zsptd2))
    assert np.allclose(sp.pFromS6D(zspu2),sp.pFromSt6D(zsptd2))


# Use 6D momenta with z = 1

test_ZSpinor6_dirac(dp1d,dp2d,z=1)
test_ZSpinor6_pFromS(dp1d,dp2d,z=1)

# Now use cases with z != 1

test_ZSpinor6_dirac(psp1_1,psp1_2,z_1)
test_ZSpinor6_dirac(psp2_1,psp2_2,z_2)
test_ZSpinor6_dirac(psp3_1,psp3_2,z_3)
test_ZSpinor6_dirac(psp4_1,psp4_2,z_4)

test_ZSpinor6_pFromS(psp1_1,psp1_2,z_1)
test_ZSpinor6_pFromS(psp2_1,psp2_2,z_2)
test_ZSpinor6_pFromS(psp3_1,psp3_2,z_3)
test_ZSpinor6_pFromS(psp4_1,psp4_2,z_4)
```

Shifted momenta

Obtain shifted momenta from the ZSpinors

In [18]:
```
def pHatZs(p1,p2,z):

    zspu1,zspd1 = ZSpinors(p1,p2,z).zsp1()
    zspu2,zspd2 = ZSpinors(p1,p2,z).zsp2()
    zsptu1,zsptd1 = ZSpinors(p1,p2,z).zspt1()
    zsptu2,zsptd2 = ZSpinors(p1,p2,z).zspt2()

    p1h_Zs = sp.pFromS6D(zspd1)
    p2h_Zs = sp.pFromS6D(zspd2)
    p1h_Zst = sp.pFromSt6D(zsptd1)
    p2h_Zst = sp.pFromSt6D(zsptd2)

    # Check p1h, p2h from spinors and tilde spinors are consistent
    assert np.allclose(p1h_Zs,p1h_Zst)
    assert np.allclose(p2h_Zs,p2h_Zst)

    # Check p1h,p2h are null and momentum is conserved
    assert np.allclose(MP6(p1h_Zs,p1h_Zs),0)
    assert np.allclose(MP6(p2h_Zs,p2h_Zs),0)
    assert np.allclose(p1h_Zs + p2h_Zs, p1 + p2)
    # Same check for p1h,p2h from tilde spinors
    assert np.allclose(MP6(p1h_Zst,p1h_Zst),0)
    assert np.allclose(MP6(p2h_Zst,p2h_Zst),0)
    assert np.allclose(p1h_Zst + p2h_Zst, p1 + p2)

    return p1h_Zs, p2h_Zs
```

Alternatively, with a shift between p_i and p_j the shifted momenta are given by
$$ \hat{p}_i = p_i - zX^{a \dot{a}} \mathcal{E}_{i a \dot{a}} $$ $$ \hat{p}_j = p_j + zX^{a \dot{a}} \mathcal{E}_{i a \dot{a}} $$
where $ z $ is the shift, $ X $ is the auxiliary vector $ X^{a \dot{a}} = x^a \tilde{x}^{\dot{a}} $ and $ \mathcal{E} $ is the polarisation vector.

In [19]:
```
def pHat(p1,p2,rv,z):
    pHat1 = p1 - z * rv
    pHat2 = p2 + z * rv

    assert np.allclose(MP6(pHat1,pHat1),0)
    assert np.allclose(MP6(pHat2,pHat2),0)
    assert np.allclose(pHat1 + pHat2, p1 + p2)

    return pHat1,pHat2
```

In [20]:
```
def test_pHat(p1,p2,z,r):
    # Obtain phat from ZSpinors
    p1h_Zs, p2h_Zs = pHatZs(p1,p2,z)
    # Calculate phat
    p1h_calc, p2h_calc = pHat(p1,p2,z,r)
    # Check they are the same
    assert np.allclose(p1h_Zs,p1h_calc)
    assert np.allclose(p2h_Zs,p2h_calc)
```

```
test_pHat(psp1_1,psp1_2,z_1,r_1)
test_pHat(psp2_1,psp2_2,z_2,r_2)
test_pHat(psp3_1,psp3_2,z_3,r_3)
test_pHat(psp4_1,psp4_2,z_4,r_4)
```

Tree amplitudes - 3-point

All gluon 3-point

All-gluon 3-point amplitude per Cheung and O'Connell:
$$ A_3(1_{a \dot{a}}, 2_{b \dot{b}}, 3_{c \dot{c}}) = i \Gamma_{abc} \tilde{\Gamma}_{\dot{a} \dot{b} \dot{c}} $$
where
$$ \Gamma_{abc} = u_{1a}u_{2b}w_{3c} + u_{1a}w_{2b}u_{3c} + w_{1a}u_{2b}w_{3c}$$
$$ \tilde{\Gamma}_{\dot{a} \dot{b} \dot{c}} = \tilde{u}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{w}_{3\dot{c}} + \tilde{u}_{1\dot{a}}\tilde{w}_{2\dot{b}}\tilde{u}_{3\dot{c}} + \tilde{w}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{u}_{3\dot{c}} $$
And for the right hand side when we use bcfw we need little group index up for the shift vector: $$ A_3(1_{a \dot{a}}, 2_{b \dot{b}}, 3^{e \dot{e}}) = i \Gamma_{ab}\ ^{e} \tilde{\Gamma}_{\dot{a} \dot{b}}\ ^{ \dot{e}} $$
where
$$ \Gamma_{ab}\ ^{e} = u_{1a}u_{2b}w_3^e + u_{1a}w_{2b}u_3^e + w_{1a}u_{2b}w_3^e$$
$$ \tilde{\Gamma}_{\dot{a} \dot{b}}\ ^{\dot{e}} = \tilde{u}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{w}_3^{\dot{e}} + \tilde{u}_{1\dot{a}}\tilde{w}_{2\dot{b}}\tilde{u}_3^{\dot{e}} + \tilde{w}_{1\dot{a}}\tilde{u}_{2\dot{b}}\tilde{u}_3^{\dot{e}} $$

Obtain spinors for left- and right-hand amplitudes

For spinor representation of shifted leg we calculate the internal momentum $ \hat{p}_{2j} $ and obtain the relevant spinors from sp.momSpinors.

The split is:
$$\text{lhs } = \hat{p}_1 \pm \hat{p}_{2j} +p_4 $$ $$ \text{rhs } = \hat{p}_2 + p_3 \mp \hat{p}_{2j} $$

Note that the sign of $ \hat{p}_{2j} $ is opposite for the two sides but sp.momSpinors will handle this by applying
$$ \Lambda_{-\hat{p}_{2j}} = i\Lambda_{\hat{p}_{2j}}$$

In [21]:
```
# This function uses p2jh extracted from spinors, not momenta

def spFor3g(p1,p2,p3,p4):

    # Obtain r, p2j (unhatted)  and z
    rv12 = rv(p1,p2)
    p2j12_right = p2j(p2,p3)
    z12 = zPole(rv12,p2j12_right)

    # Obtain spinors for shifted momenta, p1h, p2h
    zs1u,zs1d = ZSpinors(p1,p2,z12).zsp1()
    zst1u,zst1d= ZSpinors(p1,p2,z12).zspt1()
    zs2u,zs2d = ZSpinors(p1,p2,z12).zsp2()
    zst2u,zst2d= ZSpinors(p1,p2,z12).zspt2()

    #Check these spinors all return consistent shifted momenta
    assert np.allclose(sp.pFromS6D(zs1d),sp.pFromS6D(zs1u))
    assert np.allclose(sp.pFromS6D(zs1d),sp.pFromSt6D(zst1d))
    assert np.allclose(sp.pFromS6D(zs1d),sp.pFromSt6D(zst1u))
    assert np.allclose(sp.pFromS6D(zs2d),sp.pFromS6D(zs2u))
    assert np.allclose(sp.pFromS6D(zs2d),sp.pFromSt6D(zst2d))
    assert np.allclose(sp.pFromS6D(zs2d),sp.pFromSt6D(zst2u))

    # Obtain spinors for unshifted momenta p3 and p4
    s3d,st3d,s3u,st3u = sp.momSpinors(p3)
    s4d,st4d,s4u,st4u = sp.momSpinors(p4)
    # Check unshifted spinors return expected momenta
    assert np.allclose(sp.pFromS6D(s3d),p3)
    assert np.allclose(sp.pFromS6D(s3d),sp.pFromS6D(s3u))
    assert np.allclose(sp.pFromS6D(s3d),sp.pFromSt6D(st3d))
    assert np.allclose(sp.pFromS6D(s3d),sp.pFromSt6D(st3u))
    assert np.allclose(sp.pFromS6D(s4d),p4)
    assert np.allclose(sp.pFromS6D(s4d),sp.pFromS6D(s4u))
    assert np.allclose(sp.pFromS6D(s4d),sp.pFromSt6D(st4d))
    assert np.allclose(sp.pFromS6D(s4d),sp.pFromSt6D(st4u))

    # Calculate shifted internal momentum
    r_p2jh = sp.pFromS6D(zs2d) + p3
    r_p2jht = sp.pFromSt6D(zst2d) + p3
    l_p2jh = sp.pFromS6D(zs1d) + p4
    l_p2jht = sp.pFromSt6D(zst1d) + p4

    # Check tilde and non-tilde spinor source produces same result
    assert np.allclose(r_p2jh, r_p2jht)
    assert np.allclose(l_p2jh, l_p2jht)
    # Check right-hand calculation is equal and opposite to left-hand
    assert np.allclose(r_p2jh, -l_p2jh)

    # Obtain the spinors from p2jh momentum
    r_sp2jhd, r_sp2jhtd, r_sp2jhu, r_sp2jhtu = sp.momSpinors(r_p2jh)
    l_sp2jhd, l_sp2jhtd, l_sp2jhu, l_sp2jhtu = sp.momSpinors(l_p2jh)

    # Check they produce the correct momentum
    assert np.allclose(sp.pFromS6D(r_sp2jhd),sp.pFromSt6D(r_sp2jhtd))
    assert np.allclose(sp.pFromS6D(r_sp2jhd),sp.pFromS6D(r_sp2jhu))
    assert np.allclose(sp.pFromS6D(r_sp2jhd),sp.pFromSt6D(r_sp2jhtu))

    assert np.allclose(sp.pFromS6D(l_sp2jhd),sp.pFromSt6D(l_sp2jhtd))
    assert np.allclose(sp.pFromS6D(l_sp2jhd),sp.pFromS6D(l_sp2jhu))
    assert np.allclose(sp.pFromS6D(l_sp2jhd),sp.pFromSt6D(l_sp2jhtu))

    # ADD TEST TO ENSURE FOR ALL 3point COMBINATIONS pi.pj =0
    assert np.allclose(MP6(sp.pFromS6D(zs2d),p3),0)
    assert np.allclose(MP6(p3,sp.pFromS6D(r_sp2jhd)),0)
    assert np.allclose(MP6(sp.pFromS6D(r_sp2jhd),sp.pFromS6D(zs2d)),0)
    assert np.allclose(MP6(sp.pFromS6D(l_sp2jhd),p4),0)
    assert np.allclose(MP6(p4,l_p2jh),0)
```

```
    assert np.allclose(MP6(l_p2jh,sp.pFromS6D(l_sp2jhd)),0)



    # Alternatively, calculate shifted interal momentum from pHat
    p1Hat, p2Hat = pHat(p1,p2,rv12,z12)
    r_p2jh_calc = p2Hat + p3
    l_p2jh_calc = p1Hat + p4
    assert np.allclose(r_p2jh_calc, -l_p2jh_calc)
    # Obtain spinors from mom.spinors
    r_sp2jhd_alt, r_sp2jhtd_alt, r_sp2jhu_alt, r_sp2jhtu_alt = sp.momSpinors(r_p2jh_calc)
    l_sp2jhd_alt, l_sp2jhtd_alt, l_sp2jhu_alt, l_sp2jhtu_alt = sp.momSpinors(l_p2jh_calc)

    # Check spinors return correct momentum
    assert np.allclose(sp.pFromS6D(r_sp2jhd_alt),r_p2jh_calc)
    assert np.allclose(sp.pFromS6D(l_sp2jhd_alt),l_p2jh_calc)

    # Test to ensure all 3point combinations pi.pj =0
    assert np.allclose(MP6(p2Hat,p3),0)
    assert np.allclose(MP6(p3,r_p2jh_calc),0)
    assert np.allclose(MP6(r_p2jh_calc,p2Hat),0)
    assert np.allclose(MP6(p1Hat,l_p2jh_calc),0)
    assert np.allclose(MP6(l_p2jh_calc,p4),0)
    assert np.allclose(MP6(p4,p1Hat),0)

    return (zs1d,zst1d,zs2d,zst2d,s3d,st3d,s4d,st4d,r_sp2jhd, r_sp2jhtd, l_sp2jhd, l_sp2jhtd)
```

In [22]:
```
def test_spFor3g(p1,p2,p3,p4):
    #r_p2jh= spFor3g(p1,p2,p3,p4)
    zs1d,zst1d,zs2d,zst2d,s3d,st3d,s4d,st4d,r_sp2jhd, r_sp2jhtd, l_sp2jhd, l_sp2jhtd = spFor3g(p1,p2,p3,p4)

#print('psp set 1')
test_spFor3g(psp1_1,psp1_2,psp1_3,psp1_4)
#print('psp set 2')
test_spFor3g(psp2_1,psp2_2,psp2_3,psp2_4)
#print('psp set 3')
test_spFor3g(psp3_1,psp3_2,psp3_3,psp3_4)
#print('psp set 4')
test_spFor3g(psp4_1,psp4_2,psp4_3,psp4_4)
```

Find u,v and utilde,vtilde for S(2) spinors

Per Cheung and O'Connell: Since we cannot invert $ \langle i|j] $ because det$\langle i|j] = -2p_i p_j = 0 $ we instead express the rank one matrix $ \langle i_a | j_{\dot{b}}] $ as the product of two 2x1 matrices such that $$u_{ia} \tilde{u}_{j \dot{b}} = \langle i_a | j_{\dot{b}}] $$

In [23]:
```
# IMPORT getuutilde(spinors) FROM tensors.ipynb

def uutilde(s1,s2,s3,st1,st2,st3):

    '''strategy is to make the spinor product s12 diagonal, from there we can deduct the transformed versions
    of its two constituents as spinors with one zero constituent u1, ut2. Then we extract the other ut3 from s13 adn u3
    from s32 and then we can get ut1 and u2 from s21 and s31 respectively.
    '''

    # Obtain spinor products
    s12  = sp.sp62_all(s1,st2)
    s23 = sp.sp62_all(s2,st3)
    s31  = sp.sp62_all(s3,st1)
    # Reverse ordering
    s21  = sp.sp62_all(s2,st1)
    s13 = sp.sp62_all(s1,st3)
    s32  = sp.sp62_all(s3,st2)

    #print('s12')
    #print(s12)
    #print('s23')
    #print(s23)
    #print('s31')
    #print(s31)
    #print()
    #print('s21')
    #print(s21)
    #print('s32')
    #print(s32)
    #print('s13')
    #print(s13)

    eig_res = np.linalg.eig(s12)
    if np.isclose(eig_res.eigenvalues[1],0) :
        # swap the eigenvalues adnd eigenvectors
        S = np.zeros((2,2), dtype=complex)
        orig = np.array(eig_res.eigenvectors)
        S[:,0] = orig[:,1]
        S[:,1] = orig[:,0]
    else:
        S = eig_res.eigenvectors

    Sinv = np.linalg.inv(S)

    s12_trans = (Sinv@s12@S)
    s13_trans = (Sinv@s13@S)
    s21_trans = (Sinv@s21@S)
    s23_trans = (Sinv@s23@S)
    s31_trans = (Sinv@s31@S)
    s32_trans = (Sinv@s32@S)

    val = s12_trans[1,1]

    u1_trans = np.array([[0], [np.sqrt(val)]])
    ut2_trans = np.array([[0], [np.sqrt(val)]])
    ut3_trans = np.array([[s13_trans[1,0]],[s13_trans[1,1]]])*(-1/np.sqrt(val))  # negative because of the revers order
    u3_trans = np.array([[s32_trans[0,1]],[s32_trans[1,1]]])*(-1/np.sqrt(val))
    if np.allclose(u3_trans[0,0],0):
        ut1_trans = np.array([[s31_trans[1,0]],[s31_trans[1,1]]]) /u3_trans[1,0]
    else:
        ut1_trans = np.array([[s31_trans[0,0]],[s31_trans[0,1]]]) /u3_trans[0,0]
```

```
if np.allclose(ut1_trans[0,0],0):
    u2_trans = np.array([[s21_trans[0,1]],[s21_trans[1,1]]])  / -ut1_trans[1,0]
else:
    u2_trans = np.array([[s21_trans[0,0]],[s21_trans[1,0]]])  / -ut1_trans[0,0]

u1 = (S@u1_trans)
u2 = (S@u2_trans)
u3 = (S@u3_trans)

ut1 = (Sinv.T@ut1_trans)
ut2 = (Sinv.T@ut2_trans)
ut3 = (Sinv.T@ut3_trans)

#positive order
assert np.allclose(np.outer(u1,ut2), s12)
assert np.allclose(np.outer(u2,ut3), s23)
assert np.allclose(np.outer(u3,ut1), s31)

# reverse order negative

assert np.allclose(-np.outer(u2,ut1), s21)
assert np.allclose(-np.outer(u1,ut3), s13)
assert np.allclose(-np.outer(u3,ut2), s32)

u1_0, u1_1 = u1[0,0],u1[1,0]
u2_0, u2_1 = u2[0,0],u2[1,0]
u3_0, u3_1 = u3[0,0],u3[1,0]

if np.allclose(u1_0,0):
    w1_0, w1_1 = 1/u1_1,0
else:  w1_0, w1_1 = 0,1/u1_0
w1 = np.array([[w1_0], [w1_1]])

if np.allclose(u2_0,0):
    w2_0, w2_1 = 1/u2_1,0
else:  w2_0, w2_1 = 0,1/u2_0
w2 = np.array([[w2_0], [w2_1]])

if np.allclose(u3_0,0):
    w3_0, w3_1 = 1/u3_1,0
else: w3_0, w3_1 = 0,1/u3_0
w3pre = np.array([[w3_0], [w3_1]])

## OBTAIN MOMENTUM SUM
# Raise indices on spinors
s1u = np.matmul(s1,lev2u)
s2u = np.matmul(s2,lev2u)
s3u = np.matmul(s3,lev2u)
momsum = np.zeros([4,1],dtype = complex)
for a in (0,1):
    momsum[:,0] += w1[a,0] * s1u[:,a] + w2[a,0] * s2u[:,a] + w3pre[a,0] * s3u[:,a]
if np.allclose(momsum,0):
    #print('w momsum works without adjustment')
    w3 = w3pre
else:
    den = np.zeros([4,1],dtype = complex)
    for a in (0,1):
        den [:,0] += u1[a,0] * s1u[:,a]
    factor = (momsum[0,0]/(den[0,0]) ) # u1*la1 = u2*la2 = u3*la3
    w3 = w3pre - factor*u3

# Add a check that momcon condition is now met
momsum2 = np.zeros([4,1],dtype = complex)
for a in (0,1):
    momsum2[:,0] += w1[a,0] * s1u[:,a] + w2[a,0] * s2u[:,a] + w3[a,0] * s3u[:,a]
assert np.allclose(momsum2,0)

ut1_0, ut1_1 = ut1[0,0],ut1[1,0]
ut2_0, ut2_1 = ut2[0,0],ut2[1,0]
ut3_0, ut3_1 = ut3[0,0],ut3[1,0]

if np.allclose(ut1_0,0):
    wt1_0, wt1_1 = 1/ut1_1,0
else:  wt1_0, wt1_1 = 0,1/ut1_0
wt1 = np.array([[wt1_0], [wt1_1]])

if np.allclose(ut2_0,0):
    wt2_0, wt2_1 = 1/ut2_1,0
else:  wt2_0, wt2_1 = 0,1/ut2_0
wt2 = np.array([[wt2_0], [wt2_1]])

if np.allclose(ut3_0,0):
    wt3_0, wt3_1 = 1/ut3_1,0
else: wt3_0, wt3_1 = 0,1/ut3_0
wt3pre = np.array([[wt3_0], [wt3_1]])

# Raise indices on tilde spinors
st1u = np.matmul(st1,lev2u)
st2u = np.matmul(st2,lev2u)
st3u = np.matmul(st3,lev2u)

# Calculate pre-normalised momentum sum
momsumt = np.zeros([4,1], dtype = complex)
for ad in (0,1):
    momsumt[:,0] += wt1[ad,0] * st1u[:,ad] + wt2[ad,0] * st2u[:,ad] + wt3pre[ad,0] * st3u[:,ad]
if np.allclose(momsumt,0):
    #print('wt momsumt works without adjustment')
    wt3 = wt3pre
else:
    dent = np.zeros([4,1],dtype = complex)
    if np.allclose(ut1_0,0):
        for ad in (0,1):
            dent [:,0] += ut1[ad,1] * st1u[:,ad]
        factort = momsumt[0,0]/dent[0,0] # u1*la1 = u2*la2 = u3*la3
        wt3 = wt3pre - factort*ut3
    else:
        for ad in (0,1):
            dent [:,0] += ut1[ad,0] * st1u[:,ad]
        factort = momsumt[0,0]/dent[0,0] # u1*la1 = u2*la2 = u3*la3
```

```
            wt3 = wt3pre - factort*ut3

    # Confirm momcon condition is now met
    momsumt2 = np.zeros([4,1], dtype = complex)
    for ad in (0,1):
        momsumt2[:,0] += wt1[ad,0] * st1u[:,ad] + wt2[ad,0] * st2u[:,ad] + wt3[ad,0] * st3u[:,ad]
    assert np.allclose(momsumt2,0)

    return u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3
```

1. Test uut meets consistency relation C&O'C equation 4.6

With the minus signs specified for sp13,sp21,sp32.
This test is carried out as assertion in uutilde construction.

2. Test uw inverse relation C&O'C equation 4.10

$$ u_{ia}w_{ib} - u_{ib} w_{ia} = \epsilon_{ab} $$ $$ \tilde{u}_{i\dot{a}}\tilde{w}_{i\dot{b}} - \tilde{u}_{i\dot{b}} \tilde{w}_{i\dot{a}} = \epsilon_{\dot{a}\dot{b}} $$

3. Test derived from momentum condition C & O'C equation 4.9 gives

$$ \tilde{u}_2^{\dot{b}} |2_{\dot{b}}] = \tilde{u}^{\dot{c}}_3 |3_{\dot{c}}],\ \ \ \tilde{u}_1^{\dot{a}} |1_{\dot{a}}] = \tilde{u}^{\dot{c}}_3 |3_{\dot{c}}] $$ $$ u^a_i \langle i_a| = u^a_j \langle j_a | \ \ \ \forall i,j$$

4. Test wwt3R

Calculated wwt3R should meet the condition C&O'C equation 6.5

$$ w_{-\hat{2j}}. w_{\hat{2j}} = \dfrac{1}{ u_{-\hat{2j}}. u_{\hat{2j}}} $$

5.

Expressing right-hand wwt in relation to left-hand uut in equation 6.6 as follows:

$$ w_{-\hat{2j}} = \dfrac{u_{\hat{2j}}}{\sqrt{-s}} $$ $$ \tilde{w}_{-\hat{2j}} = \dfrac{\tilde{u}_{\hat{2j}}}{\sqrt{-s}} $$

where, per equation 6.3

$$ -s = \tilde{u}_{-\hat{2j}} . \tilde{u}_{\hat{2j}} u_{-\hat{2j}} . u_{\hat{2j}} = \tilde{u}_{-\hat{2j}}^{\dot{e}} \tilde{u}_{\hat{2j}\dot{e}} u_{-\hat{2j}}^e u_{\hat{2j}e}$$

ACTUALLY, IN OUR CASE IT IS THE -p2j THAT IS ON THE LEFT AND +p2j ON THE RIGHT.
As long as approach is internally consistent this should not matter.

6. Test normalised uut spinors meet properties C&O'C equation 6.4

$$ u_{\hat{p}_{2j}}.w_{-\hat{p}_{2j}} = \tilde{u}_{\hat{p}_{2j}}.\tilde{w}_{-\hat{p}_{2j}}= w_{\hat{p}_{2j}}.u_{-\hat{p}_{2j}}= \tilde{w}_{\hat{p}_{2j}}.\tilde{u}_{-\hat{p}_{2j}}= 0 $$

7. Test wwt spinors meet strong momentum conservation C&O'C equation 4.12

$$ |w_1.1 \rangle + |w_2.2 \rangle + |w_3.3 \rangle = 0 $$

Explicitly

$$ w_1^a \Lambda^A_{1a} + w_2^a \Lambda^A_{2a} + w_3^a \Lambda^A_{3a} = 0 $$ $$ \tilde{w}_{1\dot{a}} \Lambda^{\dot{a}}_{1A} + \tilde{w}_{2\dot{a}} \Lambda^{\dot{a}}_{2A} + \tilde{w}_{3\dot{a}} \Lambda^{\dot{a}}_{3A} = 0 $$

THIS CONDITION IS USED TO CONSTRUCT THE w AND wt IN uutilde

In [24]:
```
# Obtain left- and right-hand spinors,u,w for testing

def uutilde_Lset(p1,p2,p3,p4):

    zs1d,zst1d,zs2d,zst2d,s3d,st3d,s4d,st4d,r_sp2jhd, r_sp2jhtd, l_sp2jhd, l_sp2jhtd = spFor3g(p1,p2,p3,p4)

    s1 = s4d
    st1 = st4d
    s2 = zs1d
    st2 = zst1d
    s3 = l_sp2jhd
    st3 = l_sp2jhtd
    u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3 = uutilde(s1,s2,s3,st1,st2,st3)
    return (s1,st1,s2,st2,s3,st3,u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3)

def uutilde_Rset(p1,p2,p3,p4):

    zs1d,zst1d,zs2d,zst2d,s3d,st3d,s4d,st4d,r_sp2jhd, r_sp2jhtd, l_sp2jhd, l_sp2jhtd = spFor3g(p1,p2,p3,p4)

    s1 = zs2d
    st1 = zst2d
    s2 = s3d
    st2 = st3d
    s3 = r_sp2jhd
    st3 = r_sp2jhtd
    u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3 = uutilde(s1,s2,s3,st1,st2,st3)
    return (s1,st1,s2,st2,s3,st3,u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3)
```

In [25]:
```
def test_uutilde(p1,p2,p3,p4):

    s1L,st1L,s2L,st2L,s3L,st3L,u1L, u2L, u3L, ut1L, ut2L, ut3L, w1L, w2L,\
        w3L, wt1L, wt2L, wt3L = uutilde_Lset(p1,p2,p3,p4)
    s1R,st1R,s2R,st2R,s3R,st3R,u1R, u2R, u3R, ut1R, ut2R, ut3R, w1R, w2R,\
        w3R, wt1R, wt2R, wt3R = uutilde_Rset(p1,p2,p3,p4)

    # Raise indices on u3,etc
    u3Lu = np.matmul(lev2u,u3L)
    ut3Lu = np.matmul(lev2u,ut3L)
    w3Lu = np.matmul(lev2u,w3L)
    wt3Lu = np.matmul(lev2u,wt3L)
    u3Ru = np.matmul(lev2u,u3R)
    ut3Ru = np.matmul(lev2u,ut3R)
```

```
w3Ru = np.matmul(lev2u,w3R)
wt3Ru = np.matmul(lev2u,wt3R)

u1Lu = np.matmul(lev2u,u1L)
u2Lu = np.matmul(lev2u,u2L)
u1Ru = np.matmul(lev2u,u1R)
u2Ru = np.matmul(lev2u,u2R)
ut1Lu = np.matmul(lev2u,ut1L)
ut2Lu = np.matmul(lev2u,ut2L)
ut1Ru = np.matmul(lev2u,ut1R)
ut2Ru = np.matmul(lev2u,ut2R)

t= -2*MP6(p1,p3)
s = 2*MP6(p1,p2)

# 2. Test inverse condition

print('left inverse calc')
for u,w in ((u1L,w1L),(u2L,w2L),(u3L,w3L),(ut1L,wt1L),(ut2L,wt2L),(ut3L,wt3L)):
    eab = np.zeros([2,2], dtype = complex)
    for a in (0,1):
        for b in (0,1):
            eab[a,b] += (u[a,0] * w[b,0] - u[b,0] * w[a,0])
    if np.allclose(eab,lev2d): print('is allclose lev2d')
    else:
        if np.allclose(eab,lev2u): print('is allclose lev2u')
        else: print('test fails, eab:', eab.round(4))
print()
print('right inverse calc')
for u,w in ((u1R,w1R),(u2R,w2R),(u3R,w3R),(ut1R,wt1R),(ut2R,wt2R),(ut3R,wt3R)):
    eab = np.zeros([2,2], dtype = complex)
    for a in (0,1):
        for b in (0,1):
            eab[a,b] += (u[a,0] * w[b,0] - u[b,0] * w[a,0])
    if np.allclose(eab,lev2d): print('is allclose lev2d')
    else:
        if np.allclose(eab,lev2u): print('is allclose lev2u')
        else: print('test fails, eab:', eab.round(4))

# 3. Test from momentum condition

print()
print('test conservation consistency, tildes:')
res1L = np.zeros([4,1],dtype = complex)
res2L = np.zeros([4,1],dtype = complex)
res3L = np.zeros([4,1],dtype = complex)
for ad in (0,1):
    res1L[:,0] += ut1Lu[ad,0] * st1L[:,ad]
    res2L[:,0] += ut2Lu[ad,0] * st2L[:,ad]
    res3L[:,0] += ut3Lu[ad,0] * st3L[:,ad]
assert np.allclose (res1L,res2L)
print('res1L = res2L')
assert np.allclose(res1L, -res3L)
print('res1L, res2L = -res3L')

res1R = np.zeros([4,1],dtype = complex)
res2R = np.zeros([4,1],dtype = complex)
res3R = np.zeros([4,1],dtype = complex)
for ad in (0,1):
    res1R[:,0] += ut1Ru[ad,0] * st1R[:,ad]
    res2R[:,0] += ut2Ru[ad,0] * st2R[:,ad]
    res3R[:,0] += ut3Ru[ad,0] * st3R[:,ad]
assert np.allclose (res1R,res2R)
print('res1R = res2R')
assert np.allclose(res1L, -res3L)
print('res1R, res2R = -res3R')

print()
print('test conservation consistency, non-tildes:')
resu1L = np.zeros([4,1],dtype = complex)
resu2L = np.zeros([4,1],dtype = complex)
resu3L = np.zeros([4,1],dtype = complex)
for a in (0,1):
    resu1L[:,0] += u1Lu[a,0] * s1L[:,a]
    resu2L[:,0] += u2Lu[a,0] * s2L[:,a]
    resu3L[:,0] += u3Lu[a,0] * s3L[:,a]
assert np.allclose (resu1L,resu2L)
print('resu1L = resu2L')
assert np.allclose(resu1L, -resu3L)
print('resu1L, resu2L = -resu3L')

print()
print('test conservation consistency, non-tildes:')
resu1R = np.zeros([4,1],dtype = complex)
resu2R = np.zeros([4,1],dtype = complex)
resu3R = np.zeros([4,1],dtype = complex)
for a in (0,1):
    resu1R[:,0] += u1Ru[a,0] * s1R[:,a]
    resu2R[:,0] += u2Ru[a,0] * s2R[:,a]
    resu3R[:,0] += u3Ru[a,0] * s3R[:,a]
assert np.allclose (resu1R,resu2R)
print('resu1R = resu2R')
assert np.allclose(resu1R, -resu3R)
print('resu1R, resu2R = -resu3R')


# 4. Test equation 6.5
print()
print('test equation 6.5 w3L.w3Ru = 1/u3L.u3Ru fails:')
print(dotprod(w3L,w3Ru))
print(1/dotprod(u3L,u3Ru))
#assert np.allclose(dotprod(w3L,w3R),1/dotprod(u3L,u3R))

# 6. Test equation 6.4
print()
print('test equation 6.4:')
for (u,w) in ((u3Ru,w3L),(ut3Ru,wt3L),(w3Ru,u3L),(wt3Ru,ut3L)):
    if np.allclose(dotprod(u,w),0): print('pass')
    else: print('fail, uRwL is not zero:  ',dotprod(u,w))
```

```
test_uutilde(psp1_1,psp1_2,psp1_3,psp1_4)
test_uutilde(psp2_1,psp2_2,psp2_3,psp2_4)
test_uutilde(psp3_1,psp3_2,psp3_3,psp3_4)
test_uutilde(psp4_1,psp4_2,psp4_3,psp4_4)
```

```
left inverse calc
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u

right inverse calc
is allclose lev2d
is allclose lev2d
is allclose lev2d
is allclose lev2d
is allclose lev2d
is allclose lev2d

test conservation consistency, tildes:
res1L = res2L
res1L, res2L = -res3L
res1R = res2R
res1R, res2R = -res3R

test conservation consistency, non-tildes:
resu1L = resu2L
resu1L, resu2L = -resu3L

test conservation consistency, non-tildes:
resu1R = resu2R
resu1R, resu2R = -resu3R

test equation 6.5 w3L.w3Ru = 1/u3L.u3Ru fails:
[0.39366437-0.30826671j]
[-0.39366437+0.30826671j]

test equation 6.4:
pass
pass
pass
pass
left inverse calc
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u

right inverse calc
is allclose lev2d
is allclose lev2d
is allclose lev2d
is allclose lev2d
is allclose lev2d
is allclose lev2d

test conservation consistency, tildes:
res1L = res2L
res1L, res2L = -res3L
res1R = res2R
res1R, res2R = -res3R

test conservation consistency, non-tildes:
resu1L = resu2L
resu1L, resu2L = -resu3L

test conservation consistency, non-tildes:
resu1R = resu2R
resu1R, resu2R = -resu3R

test equation 6.5 w3L.w3Ru = 1/u3L.u3Ru fails:
[-0.3031506-0.27116091j]
[0.3031506+0.27116091j]

test equation 6.4:
pass
pass
pass
pass
left inverse calc
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u

right inverse calc
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u

test conservation consistency, tildes:
res1L = res2L
res1L, res2L = -res3L
res1R = res2R
res1R, res2R = -res3R

test conservation consistency, non-tildes:
resu1L = resu2L
resu1L, resu2L = -resu3L

test conservation consistency, non-tildes:
resu1R = resu2R
resu1R, resu2R = -resu3R

test equation 6.5 w3L.w3Ru = 1/u3L.u3Ru fails:
```

```
[0.+0.j]
[0.15313761-0.65282566j]

test equation 6.4:
fail, uRwL is not zero:   [-4.11505847+3.64146593j]
fail, uRwL is not zero:   [-0.00715336+0.78259544j]
fail, uRwL is not zero:   [0.13628738+0.12060238j]
fail, uRwL is not zero:   [0.01167882+1.27769266j]
left inverse calc
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u

right inverse calc
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u
is allclose lev2u

test conservation consistency, tildes:
res1L = res2L
res1L, res2L = -res3L
res1R = res2R
res1R, res2R = -res3R

test conservation consistency, non-tildes:
resu1L = resu2L
resu1L, resu2L = -resu3L

test conservation consistency, non-tildes:
resu1R = resu2R
resu1R, resu2R = -resu3R

test equation 6.5 w3L.w3Ru = 1/u3L.u3Ru fails:
[0.+0.j]
[-0.04730239-0.77804917j]

test equation 6.4:
fail, uRwL is not zero:   [0.20095488+0.45056291j]
fail, uRwL is not zero:   [0.72621823-1.74638784j]
fail, uRwL is not zero:   [-0.82565037+1.85119881j]
fail, uRwL is not zero:   [-0.20300944-0.48819101j]
```

Calculate $ \Gamma_{abc} $ and $ \tilde{\Gamma}_{\dot{a} \dot{b} \dot{c}} $

In [27]:
```
def g3Gamma(u1,u2,u3,ut1,ut2,ut3,w1,w2,w3,wt1,wt2,wt3):

    gam = np.zeros([2,2,2], dtype = complex)
    gamt = np.zeros([2,2,2], dtype = complex)
    for a in (0,1):
        for b in (0,1):
            for c in (0,1):
                gam[a,b,c]     += (u1[a,0] * u2[b,0] * w3[c,0] +
                                   u1[a,0] * w2[b,0] * u3[c,0] +
                                   w1[a,0] * u2[b,0] * u3[c,0])
    for ad in (0,1):
        for bd in (0,1):
            for cd in (0,1):
                gamt[ad,bd,cd] += (ut1[ad,0] * ut2[bd,0] * wt3[cd,0] +
                                   ut1[ad,0] * wt2[bd,0] * ut3[cd,0] +
                                   wt1[ad,0] * ut2[bd,0] * ut3[cd,0])

    return gam, gamt
```

Calculate left- and right-hand 3-point amplitudes

$$ A_3(1_{a \dot{a}},2_{b \dot{b}}, 3_{c \dot{c}}) = i\Gamma_{abc} \tilde{\Gamma}_{\dot{a} \dot{b} \dot{c}} $$

In [28]:
```
def g3Amp(p1,p2,p3,p4):

    # Get the spinors for momenta: external, shifted and propagator
    zs1d,zst1d,zs2d,zst2d,s3d,st3d,s4d,st4d,r_sp2jhd, r_sp2jhtd, l_sp2jhd, l_sp2jhtd = spFor3g(p1,p2,p3,p4)

    # Assign left- and right-hand spinors........

    # Left-hand cyclical momenta are p4,p1h,p2jh1
    Ls1 = s4d
    Lst1 = st4d
    Ls2 = zs1d
    Lst2 = zst1d
    Ls3 = l_sp2jhd
    Lst3 = l_sp2jhtd

    # Right-hand cyclical momenta are p2h,p3,p2jhr
    Rs1 = zs2d
    Rst1 = zst2d
    Rs2 = s3d
    Rst2 = st3d
    Rs3 = r_sp2jhd
    Rst3 = r_sp2jhtd

    # Obtain the left- and right-hand uut and wwts
    Lu1, Lu2, Lu3, Lut1, Lut2, Lut3, Lw1, Lw2,  Lw3, Lwt1, Lwt2, Lwt3 = uutilde(Ls1,Ls2,Ls3,Lst1,Lst2,Lst3)
    Ru1, Ru2, Ru3, Rut1, Rut2, Rut3, Rw1, Rw2,  Rw3, Rwt1, Rwt2, Rwt3 = uutilde(Rs1,Rs2,Rs3,Rst1,Rst2,Rst3)

    # Raise the index on right-hand u3, W3
    Ru3u = np.matmul(lev2u,Ru3)
    Rut3u = np.matmul(lev2u,Rut3)
    Rw3u = np.matmul(lev2u,Rw3)
    Rwt3u = np.matmul(lev2u,Rwt3)

    # Obtain $\Gamma$ and $\tilde{\Gamma}$
```

```
    Lgam,Lgamt = g3Gamma(Lu1, Lu2, Lu3, Lut1, Lut2, Lut3, Lw1, Lw2,  Lw3, Lwt1, Lwt2, Lwt3)
    Rgam,Rgamt = g3Gamma(Ru1, Ru2, Ru3u, Rut1, Rut2, Rut3u, Rw1, Rw2,  Rw3u, Rwt1, Rwt2, Rwt3u)

    # Calculate left-hand amplitude
    g3L = np.zeros([2,2,2,2,2,2], dtype = complex)
    for d in (0,1):
        for dd in (0,1):
            for a in (0,1):
                for ad in (0,1):
                    for e in (0,1):
                        for ed in (0,1):
                            g3L[d,dd,a,ad,e,ed] += Lgam[d,a,e] * Lgamt[dd,ad,ed]
    #g3L *= 1j

    # Calculate right-hand amplitude
    g3R = np.zeros([2,2,2,2,2,2], dtype = complex)
    for b in (0,1):
        for bd in (0,1):
            for c in (0,1):
                for cd in (0,1):
                    for e in (0,1):
                        for ed in (0,1):
                            g3R[b,bd,c,cd,e,ed] += Rgam[b,c,e] * Rgamt[bd,cd,ed]
    #g3R *= 1j

    return g3L,g3R
```

Calculate the BCFW 4-point from left and right g3

Cheung & O'Connell notation $$ x^a \tilde{x}^{\dot{a}} A_{4;a \dot{a} b ...} (p_1,p_2,...) = \sum_{L,R} \sum_{c \dot{c}}\Big(-\dfrac{i}{k^2} \Big) x^a \tilde{x}^{\dot{a}} A_{a \dot{a} c \dot{c}}(\hat{p}_1(z^*),...,\hat{k}) A_{b \dot{b}}^{c \dot{c}}(\hat{p}_2(z^*),...,-\hat{k}) $$

Bern notation $$ A^{tree}_n(0) = \sum_{j=3}^{n-1} \sum_{h} A_L(\hat{p}_2,...,p_j,-\hat{P}_{2j}^{-h}) \times \dfrac{i}{P^2_{2j}} A_R(\hat{P}_{2j}^{(h)},p_{j+1},...,\hat{p}_1)|_{z=z_{2j}} $$

Cheung & O'Connell notation (4-point) $$ x^a \tilde{x}^{\dot{a}} A_{4;a \dot{a} b \dot{b} c \dot{c} d \dot{d}} = \dfrac{i}{t} x^a \tilde{x}^{\dot{a}}A_{L;a \dot{a} e \dot{e} d \dot{d}} A_{R;b \dot{b} c \dot{c}}\ ^{e \dot{e}} $$

Bern notation (4-point) $$ $$

With our choice of auxiliary matrix $x^a \tilde{\}^{\dot{a}}$ only the $a = 1, \dot{a}=1$ survives and the factor is 1.

We need to sum over $ e, \dot{e} $ index:

In [29]:
```
def bcfw4_uncontracted(p1,p2,p3,p4):

    # Obtain left- and right-hand amplitudes
    g3L,g3R = g3Amp(p1,p2,p3,p4)

    # Left-hand momenta and indices are p4(d,dd), p1h(a,ad), p2jl(e,ed)
    # Right hand momenta and indices are p2h(b,bd), p3(c,cd), p2jr(e,ed)
    # Right-hand e,ed indices are already raised in uutilde function above

    # Calculate product with sum over e and edot
    res = np.zeros([2,2,2,2,2,2,2,2], dtype = complex)
    for a in (0,1):
        for ad in (0,1):
            for b in (0,1):
                for bd in (0,1):
                    for c in (0,1):
                        for cd in (0,1):
                            for d in (0,1):
                                for dd in (0,1):
                                    for e in (0,1):
                                        for ed in (0,1):
                                            res[a,ad,b,bd,c,cd,d,dd] += (g3L[d,dd,a,ad,e,ed]
                                                                         * g3R[b,bd,c,cd,e,ed])

    return res * 1j / sij6(p2,p3)
```

In [30]:
```
def bcfw4(p1,p2,p3,p4):

    # Obtain uncontracted bcfw amplitude:
    g4res = bcfw4_uncontracted(p1,p2,p3,p4)

    # Contract over a, adot =1
    g4 = np.zeros([2,2,2,2,2,2,2,2], dtype = complex)
    a = 1
    ad = 1
    for b in (0,1):
        for bd in (0,1):
            for c in (0,1):
                for cd in (0,1):
                    for d in (0,1):
                        for dd in (0,1):
                            g4[a,ad,b,bd,c,cd,d,dd] += g4res[a,ad,b,bd,c,cd,d,dd]
    return g4
```

In [75]:
```
def test_bcfw4_raw(p1,p2,p3,p4):
    # Just check it runs
    fourpoint_bcfw = bcfw4(p1,p2,p3,p4)
    fourpoint_g4 = g4(p1,p2,p3,p4)
    print()
    print()
    print('FOURPOINT BCFW')
    print(fourpoint_bcfw.round(4))
    print()
    print()
    print('FOURPOINT g4')
    print(fourpoint_g4.round(4))

print('psp set 2')
test_bcfw4_raw(psp2_1,psp2_2,psp2_3,psp2_4)
```

```
psp set 2


FOURPOINT BCFW
[[[[[[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]


     [[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]]



    [[[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]


     [[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]]



   [[[[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]


     [[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]]



  [[[[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]


     [[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]]



  [[[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
```

```
     [ 0.    +0.j      0.    +0.j    ]]]]


   [[[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]]]]]



[[[[[[[ 0.    +0.j      0.    +0.j    ]
       [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]]


   [[[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]]]


  [[[[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]]


   [[[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]]]]



[[[[[[ 0.    +0.j      0.    +0.j    ]
       [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 4.817 -4.8239j  0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]]


   [[[[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]

     [[ 0.    +0.j      0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]]

    [[[-1.1869+4.1113j  0.    +0.j    ]
      [ 0.    +0.j      0.    +0.j    ]]
```

```
      [[ 0.    +0.j      0.    +0.j    ]
       [ 0.    +0.j      0.    +0.j    ]]]]]



    [[[[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[-1.1869+4.1113j  0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]


     [[[[-0.5929-2.6198j  0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]


      [[[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]

       [[ 0.    +0.j      0.    +0.j    ]
        [ 0.    +0.j      0.    +0.j    ]]]]]]]]

FOURPOINT g4
[[[[[[[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]


     [[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j      0.5929-2.6198j]]]]


    [[[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]


     [[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]]]



   [[[[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]


     [[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]]


    [[[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.7478-0.7489j]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]


     [[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.2337+0.2898j -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]]]]



  [[[[[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]

      [[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]]


     [[[-0.    +0.j     -0.    +0.j    ]
       [-0.    +0.j     -0.    +0.j    ]]
```

```
    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]


  [[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]]


 [[[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]


  [[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]]]]


[[[[[[[-0.    +0.j     -0.    +0.j    ]
      [-0.    +0.j     -0.    +0.j    ]]

     [[-0.    +0.j     -0.    +0.j    ]
      [-0.    +0.j     -0.    +0.j    ]]]


    [[[-0.    +0.j     -0.    +0.j    ]
      [-0.    +0.j     -0.    +0.j    ]]

     [[-0.    +0.j     -0.    +0.j    ]
      [-0.    +0.j     -0.    +0.j    ]]]]


  [[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]]


 [[[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]


  [[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
```

```
      [-0.    +0.j     -0.    +0.j    ]]

     [[-0.    +0.j     -0.    +0.j    ]
      [-0.    +0.j     -0.    +0.j    ]]]]]]



[[[[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j      0.2337+0.2898j]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[ 0.7478-0.7489j -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]


  [[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]]


 [[[[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]


  [[[[-0.5929-2.6198j -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]


   [[[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]

    [[-0.    +0.j     -0.    +0.j    ]
     [-0.    +0.j     -0.    +0.j    ]]]]]]]]
```

In [47]:

```
# Load the S@M-generated amplitude sets: 3-helicity configurations for each phase space point

def ampset4d():
    ampset4d = np.zeros((100,3,1), dtype = complex)
    rawsam = np.loadtxt("sam_amplitude_sets.txt")
    n = 0
    for a in range (100):
        for row in range(3):
            ampset4d[a,row,0] = -rawsam[n,1] + 1j *rawsam[n,0]
            n += 1
    return ampset4d

def test_g4_2():
    samamp = ampset4d()
    for n in range(100):
        p0 = rr.sam4p[n,0,0:]
        p1 = rr.sam4p[n,1,0:]
        p2 = rr.sam4p[n,2,0:]
        p3 = rr.sam4p[n,3,0:]

        samamp_mmpp = samamp[n,0,0]
        g4_mmpp = g4_hel(p0,p1,p2,p3,0,0,1,1,1,1,0,0)
        assert abs(samamp_mmpp - g4_mmpp) < 1e13

        samamp_mpmp = samamp[n,1,0]
        g4_mpmp = g4_hel(p0,p1,p2,p3,0,1,0,1,1,0,1,0)
        assert abs(samamp_mpmp - g4_mpmp) < 1e13

        samamp_mppm = samamp[n,2,0]
        g4_mppm = g4_hel(p0,p1,p2,p3,0,1,1,0,1,0,0,1)
        assert abs(samamp_mppm - g4_mppm) < 1e13
```

F.5 Addendum: alternative code

The alternative implementation referenced in Section 5.5 has the following modules:

Module	
4Dspinors.ipynb	Implements 4D spinors
spinor_products.py	Finds 6D spinor products
tensors.py	Implements tensor management
tensors.ipynb	Calculates 6D BCFW
momenta_x11.ipynb	Checks BCFW 4-point consistent with S@M)
	(or other input data sets)

In [1]:
```
import numpy as np
from tensors import *
from lorentz import makeOnShell
```

In [2]:
```
mom = Tensor([np.sqrt(14),1,2,3], [Index(Lorenz, "mu", False)])
```

In [3]:
```
def sigmas4(mu_down, ad_down, a_down):
    return Tensor([sig0, sig1, sig2, sig3], [Index(Lorenz, mu_down, False), Index(spd, ad_down, False), Index(sp, a_down, Fals
```

In [4]:
```
pslash = contract(sigmas4('mu','ad','a'), mom.raiseIndex('mu'), 'mu')
pslasht = contract(sigmasT4('mu','a','ad'), mom.raiseIndex('mu'), 'mu')
```

In [5]:
```
def lam2u(p, a_label):
    pd = p.lowerIndex(p.indices[0]).data
    data = np.array(
        [pd[0] + pd[3], pd[1] + 1j * pd[2]]
    )/np.sqrt(pd[0] + pd[3])

    return Tensor(data, [Index(sp, a_label, True)])

def lamt2u(p, ad_label):
    pd = p.lowerIndex(p.indices[0]).data
    data = np.array(
        [pd[0] + pd[3], pd[1] - 1j * pd[2]]
    )/np.sqrt(pd[0] + pd[3])

    return Tensor(data, [Index(spd, ad_label, True)])
```

In [6]:
```
la = lam2u(mom, 'a')
lat = lamt2u(mom, 'ad')
```

In [7]:
```
assert np.allclose(contract(pslash, la, 'a').data, 0)
assert np.allclose(contract(pslash, lat, 'ad').data, 0)
```

In [9]:
```
assert np.allclose(cartesianProd(la,lat).data, pslasht.data)
assert np.allclose(cartesianProd(la.lowerIndex('a'),lat.lowerIndex('ad')).reorder(pslash.indices).data, pslash.data)
```

In [10]:
```
pslasht
```

Out[10]:
```
Tensor(data=[[6.74165739+0.j 1.        -2.j]
 [1.        +2.j 0.74165739+0.j]], indices=['a↑', 'ad↑'])
```

In [12]:
```
cartesianProd(la.lowerIndex('a'),lat.lowerIndex('ad')).reorder(pslash.indices)
```

Out[12]:
```
Tensor(data=[[ 0.74165739+0.j -1.        +2.j]
 [-1.        -2.j  6.74165739+0.j]], indices=['ad↓', 'a↓'])
```

In [13]:
```
reconstructed = 0.5*contract(cartesianProd(la,lat), sigmas4('mu','ad','a'), 'a').contract('ad')
```

In [14]:
```
assert np.allclose(reconstructed.data , mom.data)
```

```
from tensors import *

def lat6(p, A_label, ad_label):
    if np.allclose(p.data[4:], 0):
        print("Using lat64D for lat6")
        return lat64D(p, A_label, ad_label)

    pd = p.lowerIndex('mu').data
    ss = np.sqrt(pd[3] + (pd[0]**2 + pd[1]**2 + 2 * pd[0] * pd[2] + pd[2]**2 + pd[3]**2)/( 2
* (pd[0] + pd[2])))

    ss2 = np.sqrt((pd[0]**2 + pd[1]**2 +
 2 * pd[0] * (pd[2] + pd[3]) + (pd[2] + pd[3])**2)/(pd[0] + pd[2]))

    ss3 = np.sqrt((4 * pd[0]**2 + 4 * pd[2] * pd[3] + 4 * pd[0] * (pd[2] + pd[3]) -
 2 * (pd[4]**2 + pd[5]**2))/(pd[0] + pd[2]))

    ss4 = np.sqrt(pd[3] - (-2* pd[0]**2 - 2 * pd[0]* pd[2] + pd[4]**2 + pd[5]**2)/(
 2 * (pd[0] + pd[2])))

    x = (ss2 * (1j * pd[4] + pd[5]))/(np.sqrt(2) * (pd[0] - 1j* pd[1] + pd[2] +
   pd[3]))
    y = (ss2 * (pd[4] - 1j* pd[5]))/(
 np.sqrt(2) * (pd[0] - 1j* pd[1] + pd[2] + pd[3]))

    aa = -(((pd[0] - 1j* pd[1] + pd[2] + pd[3]) * (pd[4] + 1j* pd[5]))/(
 ss3 * (pd[0] + pd[2])))

    bb = (ss4 * (pd[0] - pd[3]) + ((-pd[1] + 1j * pd[2]) * (pd[1] + (
    1j* (2 * pd[0] * pd[2] + 2 * pd[2]**2 + pd[4]**2 + pd[5]**2))/(
    2 * (pd[0] + pd[2])))/ss4)/(1j * pd[4] + pd[5])

    data = [[x,-((pd[1] + 1j*(pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(2.*(pd[0]
+ pd[2])))/ss)],
            [y,ss],
            [ss,aa],
        [(pd[1] - 1j*(pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(2.*(pd[0] +
pd[2])))/ss,bb]]


    return Tensor(data, [Index(sp6, A_label, False), Index(spd, ad_label,False)])

def la6(p, A_label, a_label):
    if np.allclose(p.data[4:], 0):
        print("Using la64D for la6")
        return la64D(p, A_label, a_label)
    pd = p.lowerIndex('mu').data
    ss = np.sqrt(pd[3] + (pd[0]**2 + pd[1]**2 + 2 * pd[0] * pd[2] + pd[2]**2 + pd[3]**2)/( 2
* (pd[0] + pd[2])))
    I = 1j

    ss2 = np.sqrt((pd[0]**2 + pd[1]**2 +
 2 * pd[0] * (pd[2] + pd[3]) + (pd[2] + pd[3])**2)/(pd[0] + pd[2]))

    ss3 = np.sqrt((4 * pd[0]**2 + 4 * pd[2] * pd[3] + 4 * pd[0] * (pd[2] + pd[3]) -
```



```
 2 * (pd[4]**2 + pd[5]**2))/(pd[0] + pd[2]))

    ss4 = np.sqrt(pd[3] - (-2* pd[0]**2 - 2 * pd[0]* pd[2] + pd[4]**2 + pd[5]**2)/(
 2 * (pd[0] + pd[2])))

    x = -((ss2 * (pd[4] + I * pd[5]))/(np.sqrt(2)* (pd[0] - I * pd[1] + pd[2] + pd[3])))

    y = -I*x

    aa = -((I *(ss4 * (pd[0] - pd[3]) - ((pd[1] - I * pd[2]) * (pd[1] + (
       I * (2 * pd[0] * pd[2] + 2 * pd[2]**2 + pd[4]**2 + pd[5]**2))/(
       2 * (pd[0] + pd[2])))/ss4))/(pd[4] + I * pd[5]))

    bb = I*aa

    data= [
        [x, ss],
        [y, (  pd[1] + I * (pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(
      2 * (pd[0] + pd[2])))/ ss],
      [-((
   pd[1] - I * (pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(
       2 * (pd[0] + pd[2])))/ss), aa],
       [ss, bb]]
    return Tensor(data, [Index(sp6, A_label, True), Index(sp, a_label,True)])



def lat64D(p, A_label, ad_label):

    pd = p.lowerIndex('mu').data
    ss = np.sqrt(pd[3] + (pd[0]**2 + pd[1]**2 + 2 * pd[0] * pd[2] + pd[2]**2 + pd[3]**2)/( 2
* (pd[0] + pd[2])))

    x = 0
    y = 0
    aa = 0
    bb = 0

    data = [[x,-((pd[1] + 1j*(pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(2.*(pd[0]
+ pd[2])))/ss)],
            [y,ss],
            [ss,aa],
        [(pd[1] - 1j*(pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(2.*(pd[0] +
pd[2])))/ss,bb]]


    return Tensor(data, [Index(sp6, A_label, False), Index(spd, ad_label,False)])

def la64D(p, A_label, a_label):
    pd = p.lowerIndex('mu').data
    ss = np.sqrt(pd[3] + (pd[0]**2 + pd[1]**2 + 2 * pd[0] * pd[2] + pd[2]**2 + pd[3]**2)/( 2
* (pd[0] + pd[2])))
    I = 1j
```

```
    x = 0

    y = 0

    aa = 0
    bb = 0

    data= [
        [x, ss],
        [y, (  pd[1] + I * (pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(
      2 * (pd[0] + pd[2])))/ ss],
      [-((
   pd[1] - I * (pd[2] - (-pd[0]**2 + pd[1]**2 + pd[2]**2 + pd[3]**2)/(
       2 * (pd[0] + pd[2])))/ss), aa],
       [ss, bb]]
    return Tensor(data, [Index(sp6, A_label, True), Index(sp, a_label,True)])



def momFromLa(la, mu='mu'):
    la_L = la.relabel(la.indices[0].label, 'Aintern').relabel(la.indices[1].label,
'aintern').raiseIndex('aintern')
    la_R = la.relabel(la.indices[0].label, 'Bintern').relabel(la.indices[1].label,
'bintern').raiseIndex('bintern')
    shifted = -0.25 * c(c(c(la_L, sigma6(mu, 'Aintern', 'Bintern'), 'Aintern'), la_R,
'Bintern'), eps2('aintern',
'bintern',raised=False),'aintern').contract('bintern').lowerIndex(mu)
    return shifted

def momFromLat(lat, mu='mu'):
    lat_L = lat.relabel(lat.indices[0].label, 'Aintern').relabel(lat.indices[1].label,
'adintern').lowerIndex('adintern')
    lat_R = lat.relabel(lat.indices[0].label, 'Bintern').relabel(lat.indices[1].label,
'bdintern').lowerIndex('bdintern')
    shifted = -0.25 * c(c(c(lat_L, sigma6t(mu, 'Aintern', 'Bintern'), 'Aintern'), lat_R,
'Bintern'), eps2('adintern', 'bdintern',raised=True,
dotted=True),'adintern').contract('bdintern').lowerIndex(mu)
    return shifted



def sprodFromLaLat(La_input, Lat_input, a_label='a', ad_label='ad'):
    assert La_input.indices[0].representation == sp6 and Lat_input.indices[0].representation
== sp6
    assert La_input.indices[1].representation == sp and Lat_input.indices[1].representation
== spd
    la = La_input.relabel(La_input.indices[0].label,
'Aintern').relabel(La_input.indices[1].label, a_label)
    lat = Lat_input.relabel(Lat_input.indices[0].label,
'Aintern').relabel(Lat_input.indices[1].label, ad_label)
    return contract(la, lat, 'Aintern')


def eps2(a_up,b_up, raised=True, dotted=False):
```



```
    if dotted:
        return Tensor(sp.tensorToRaise, [Index(spd,a_up,raised), Index(spd,b_up,raised)])
    else:
        return Tensor(sp.tensorToRaise, [Index(sp,a_up,raised), Index(sp,b_up,raised)])


def sprod(mom1, mom2, a_label='a', ad_label='ad'):
    return contract(la6(mom1, 'A', a_label), lat6(mom2, 'A', ad_label), 'A')

def polvec(mom, momq, mu='mu', a_label='a', ad_label='ad'):
    tmp = c(la6(mom, 'Aintern', a_label).lowerIndex(a_label), sigma6(mu,'Aintern',
'Bintern'), 'Aintern')
    tmp = c(tmp, la6(momq,'Bintern','bintern').lowerIndex('bintern'),'Bintern')
    tmp = c(tmp, tinv(sprod(momq, mom, 'bintern',
ad_label)).raiseIndex('bintern').lowerIndex(ad_label),'bintern')
    tmp =  1/np.sqrt(2)*tmp
    return tmp.reorder([Index(Lorenz6, mu, raised=True), Index(sp, a_label, raised=False),
Index(spd, ad_label, raised=False)])

def polvec_v2(mom, momq, mu='mu', a_label='a', ad_label='ad'):
    tmp = c(lat6(momq, 'Aintern', 'bdintern'), sigma6t(mu,'Aintern', 'Bintern'), 'Aintern')
    tmp = c(tmp, lat6(mom,'Bintern', ad_label),'Bintern')
    tmp = c(tmp, tinv(sprod(mom, momq, a_label, 'bdintern')),'bdintern')
    tmp = 1/np.sqrt(2)*tmp
    return tmp.reorder([Index(Lorenz6, mu, raised=True), Index(sp, a_label, raised=False),
Index(spd, ad_label, raised=False)])
```

```
import numpy as np

class Representation:
    '''
    the tensors are the ones to multiply from the right to obtain the transformed index
    '''
    def __init__(self, name, tensorToRaise=None, tensorToLower=None):
        self.name = name
        self.tensorToLower = tensorToLower
        self.tensorToRaise  = tensorToRaise


class Index:
    def __init__(self, representation, label, raised=False):
        self.representation = representation
        self.label = label
        self.raised = raised
    def __repr__(self):
        return f"Index(label={self.label}, representation={self.representation.name},
raised={self.raised})"
    def __str__(self):
        if self.raised:
            return f"{self.label}↑"
        else:
            return f"{self.label}↓"
    def __eq__(self, other):
        # needed to ignore the raised status in equality checks
        if not isinstance(other, Index):
            return False
        return (self.label == other.label and
                self.representation.name == other.representation.name)
    def __hash__(self):
        return hash(tuple([self.representation.name, self.label, self.raised]))


class Tensor:
    def __init__(self, data, indices):
        self.data = np.array(data)
        self.indices = indices
    def __repr__(self):
        return f"Tensor(data={self.data}, indices={[str(index) for index in self.indices]})"


    def getIndexAndPosition(self, index):
        if type(index) == str:
            inds = [i for i in self.indices if i.label == index]
            #assert len(inds) == 1, "Index must be unique in tensor."
            return self.getIndexAndPosition(inds[0])
        if index not in self.indices:
            raise ValueError(f"Index {index.label} not found in tensor indices.")
        pos = self.indices.index(index)
        return self.indices[pos], pos

    def get2IndexPositions(self, index):
        if type(index) == str:
```



```
            inds = [i for i in self.indices if i.label == index]
            #assert len(inds) == 1, "Index must be unique in tensor."
            return self.get2IndexPositions(inds[0])
        indices = [i for i, ind in enumerate(self.indices) if ind == index]
        assert len(indices) == 2, "Index must appear exactly twice in tensor."
        return indices

    def raiseIndex(self, input_index):
        #print('calling raiseIndex')
        index, index_position = self.getIndexAndPosition(input_index)
        if index.raised == True:
            return self.copy()  # If already raised or no tensor to raise, return a copy
        if index.representation.tensorToRaise is None:
            new_indices = self.copy_indices()
            new_indices[index_position] = Index(index.representation, index.label,
raised=True)  # Mark as raised
            return Tensor(self.data, new_indices)
        if type(index.representation.tensorToRaise) == str and
index.representation.tensorToRaise == 'disallowed':
            raise ValueError(f"Raising index {index.label} is not allowed.")
        letters = 'abcdeghjklmnopqrstuvwxy'
        combo_list = [letters[i] for i in range(len(self.indices))]
        combo_list[index_position] = 'i'  # Replace the index position with 'i'
        combo_l = ''.join(combo_list)
        combo_r = 'i'+'z'
        combo_list[index_position] = 'z'  # Replace the index position with 'i'
        combo_res = ''.join(combo_list)
        combo = f'{combo_l},{combo_r}->{combo_res}'
        contracted = np.einsum(combo, self.data, index.representation.tensorToRaise)
        # Raise the index using the tensorToRaise
        #moved_to_the_end = np.moveaxis(self.data, index_position, -1)
        #raised_data = np.dot(moved_to_the_end, index.representation.tensorToRaise)
        #moved_back = np.moveaxis(raised_data, -1, index_position)
        new_indices = self.copy_indices()
        new_indices[index_position] = Index(index.representation, index.label, raised=True)
# Mark as raised
        return Tensor(contracted, new_indices)

    def lowerIndex(self, input_index):
        #print('calling lowerIndex')
        index, index_position = self.getIndexAndPosition(input_index)
        if index.raised == False:
            return self.copy()  # No change needed if already lowered or no lowering tensor
        if index.representation.tensorToLower is None:
            new_indices = self.copy_indices()
            new_indices[index_position] = Index(index.representation, index.label,
raised=False)  # Mark as raised
            return Tensor(self.data, new_indices)
        if type(index.representation.tensorToLower) == str and
index.representation.tensorToLower == 'disallowed':
            raise ValueError(f"Lowering index {index.label} is not allowed.")
        letters = 'abcdeghjklmnopqrstuvwxy'
        combo_list = [letters[i] for i in range(len(self.indices))]
        combo_list[index_position] = 'i'  # Replace the index position with 'i'
        combo_l = ''.join(combo_list)
```

```
        combo_r = 'i'+'z'
        combo_list[index_position] = 'z'  # Replace the index position with 'i'
        combo_res = ''.join(combo_list)
        combo = f'{combo_l},{combo_r}->{combo_res}'
        contracted = np.einsum(combo, self.data, index.representation.tensorToLower)
        # Raise the index using the tensorToRaise
        # moved_to_the_end = np.moveaxis(self.data, index_position, -1)
        # raised_data = np.dot(moved_to_the_end, index.representation.tensorToLower)
        # moved_back = np.moveaxis(raised_data, -1, index_position)

        new_indices = self.copy_indices()
        new_indices[index_position] = Index(index.representation, index.label, raised=False)
# Mark as raised
        return Tensor(contracted, new_indices)
    def __getitem__(self, index):
        if isinstance(index, int):
            return Tensor( self.data[index].copy(), self.indices[1:].copy() )



    def contract(self, index):
        if type(index) == str:
            thelabel = index
        elif isinstance(index, Index):
            thelabel = index.label
        pos = self.get2IndexPositions(index)
        assert self.indices[pos[0]].raised != self.indices[pos[1]].raised, "Indices have to
have opposite raised status to contract!, Got: " \
            f"{self.indices[pos[0]]} and {self.indices[pos[1]]}"
        letters = 'abcdeghjklmnopqrstuvwxyz'
        combo = [ letters[i] for i, _ in enumerate(self.indices)  ]
        combo[pos[0]] = 'i'
        combo[pos[1]] = 'i'
        #print(combo, pos)
        contracted = np.einsum(''.join(combo), self.data)
        new_indices = [idx for p,idx in enumerate(self.copy_indices()) if p not in pos]
        return Tensor(contracted, new_indices)



    def reorder(self, new_order):
        """
        Reorder the tensor indices to match the new order.
        """
        if set(new_order) != set(self.indices):
            raise ValueError("New order must contain the same indices as the tensor. Got: "
                             f"{new_order} and {self.indices}")
        new_indices = [self.indices.index(i) for i in new_order]
        return Tensor(np.transpose(self.data, new_indices), new_order)

    def relabel(self, old_label, new_label):
        """
        Relabel an index in the tensor.
        """
        new_indices = []
        if type(old_label) == int:
```



```
            new_indices = self.copy_indices()
            new_indices[old_label] = Index(new_indices[old_label].representation, new_label,
new_indices[old_label].raised)
            return Tensor(self.data, new_indices)
        for idx in self.copy_indices():
            if idx.label == old_label:
                new_indices.append(Index(idx.representation, new_label, idx.raised))
            else:
                new_indices.append(idx)
        return Tensor(self.data, new_indices)

    def copy_indices(self):
        return [Index(idx.representation, idx.label, idx.raised) for idx in self.indices]

    def copy(self):
        """
        Create a copy of the tensor with the same data and indices.
        """
        return Tensor(self.data.copy(), self.copy_indices())

    def __add__(self, other):
        if not isinstance(other, Tensor):
            raise TypeError("Can only add another Tensor.")
        return sumTensors(self, other)

    def __neg__(self):
        return Tensor(-self.data, self.indices.copy())

    def __sub__(self, other):
        if not isinstance(other, Tensor):
            raise TypeError("Can only subtract another Tensor.")
        return sumTensors(self, -other)

    def __mul__(self, other):
        if isinstance(other, Tensor):
            return cartesianProd(self, other)
        elif isinstance(other, (int, float,complex)):
            return Tensor(self.data * other, self.copy_indices())
        else:
            raise TypeError("Can only multiply by another Tensor or a scalar.")
    def __truediv__(self, other):
        if isinstance(other, (int, float,complex)):
            return Tensor(self.data / other, self.copy_indices())
        else:
            print('At least I tried...')
            raise TypeError("Can only divide by another Tensor or a scalar.")

    def __rmul__(self, other):
        if isinstance(other, Tensor):
            return cartesianProd(other, self)
        elif isinstance(other, (int, float,complex)):
            return Tensor(self.data * other, self.copy_indices())
        else:
            raise TypeError("Can only multiply by another Tensor or a scalar.")
```

```
def contract(t1, t2, index):
    if type(index) == str:
        inds = [i for i in t1.indices if i.label == index]
        assert len(inds) == 1, "Index must be unique in tensor."
        return contract(t1, t2, inds[0])
    assert index in t1.indices and index in t2.indices, "Index must be present in both
tensors. Got: " \
        f"{t1.indices} and {t2.indices} with index {index.label}"
    index_position_1 = t1.indices.index(index)
    index_position_2 = t2.indices.index(index)
    i1 = t1.indices[index_position_1]
    i2 = t2.indices[index_position_2]
    if i1.raised == i2.raised:
        raise ValueError(f"Cannot contract indices {i1} and {i2} with the same raised
status.")
    contracted = np.tensordot(t1.data, t2.data, axes=(index_position_1, index_position_2))
    new_indices = [idx for idx in t1.copy_indices() if idx != index] + [idx for idx in
t2.copy_indices() if idx != index]
    return Tensor(contracted, new_indices)



def cartesianProd(t1, t2):
    letters = 'abcdeghjklmnopqrstuvwxyz'
    combo1 = ''
    combo2 = ''
    currentLetter=0
    for i in range(len(t1.indices)):
        combo1 += letters[currentLetter]
        currentLetter += 1
    for i in range(len(t2.indices)):
        combo2 += letters[currentLetter]
        currentLetter += 1

    combo = f'{combo1},{combo2}->{combo1}{combo2}'
    #print(combo)
    data = np.einsum(combo, t1.data, t2.data)
    return Tensor(data, t1.copy_indices() + t2.copy_indices())

def sumTensors(t1,t2):
    if set(t1.indices) != set(t2.indices):
        raise ValueError("Tensors must have the same indices to be summed. Got: "
                         f"{t1.indices} and {t2.indices}")
    if t1.indices != t2.indices:
        new_t2 = t2.reorder(t1.indices)
    else:
        new_t2 = t2.copy()

    for i1, i2 in zip(t1.indices, new_t2.indices):
        if i1.raised != i2.raised:
            raise ValueError(f"Cannot sum tensors with mismatched raised status for index
{i1.label}. "
                             f"Got {i1.raised} and {i2.raised}.")
    return Tensor(t1.data + new_t2.data, t1.indices)
```



```
eps = np.array([[0,1],[-1,0]])



eps4 = np.empty((4,4,4,4))
for i in range(4):
    for j in range(4):
        for k in range(4):
            for l in range(4):
                a = [i,j,k,l]
                if len(set(a)) < 4:
                    eps4[i,j,k,l] = 0
                    continue
                # count the number of inversions in a
                # an inversion is a pair (ii,jj) such that ii < jj and a[ii] > a[jj]
                # this is the number of pairs that are out of order
                # if the number of inversions is even, eps4[i,j,k,l] = 1
                # if the number of inversions is odd, eps4[i,j,k,l] = -1
                # if there are repeated indices, eps4[i,j,k,l] = 0
                # this is a simple bubble sort algorithm to count inversions
                # it is not the most efficient, but it is simple and works for small arrays
                cnt=0
                n = len(a)
                for ii in range(n-1):
                    for jj in range(ii+1,n):
                        if (a[ii]>a[jj]):
                            cnt+=1
                eps4[i,j,k,l] = (-1)**cnt

gmunu_vals = np.diag([1,-1,-1,-1])  # Example metric tensor, can be replaced with actual
metric
gmunu6_vals = np.diag([1,-1,-1,-1,-1,-1])  # Example metric tensor, can be replaced with
actual metric

Lorenz = Representation("Lorenz", tensorToRaise=gmunu_vals, tensorToLower=gmunu_vals)
Lorenz6 = Representation("Lorenz6", tensorToRaise=gmunu6_vals, tensorToLower=gmunu6_vals)
Euclidean = Representation("Euclidean", tensorToRaise=None, tensorToLower=None)
# does not reconstruc pslash right
# sp = Representation("sp", tensorToRaise=eps, tensorToLower=eps.T)
#spd = Representation("spd", tensorToRaise=eps.T, tensorToLower=eps)
# this reconstruction works
sp = Representation("sp", tensorToRaise=eps, tensorToLower=eps.T)
spd = Representation("spd", tensorToRaise=eps, tensorToLower=eps.T)



sp6 = Representation("sp6", tensorToRaise='disallowed', tensorToLower='disallowed')
sp6t = Representation("sp6t", tensorToRaise='disallowed', tensorToLower='disallowed')


sig0 = np.array([[1, 0], [0, 1]])
sig1 = np.array([[0, 1], [1, 0]])
sig2 = np.array([[0, -1j], [1j, 0]])
```

```
sig3 = np.array([[1, 0], [0, -1]])

def sigmas4(mu_down, ad_down, a_down):
    return Tensor([sig0, sig1, sig2, sig3], [Index(Lorenz, mu_down, False), Index(spd,
ad_down, False), Index(sp, a_down, False)])

def sigmasT4(mu_down, a_up, ad_up):
    return Tensor([sig0, -sig1, -sig2, -sig3], [Index(Lorenz, mu_down, False), Index(sp,
a_up, True), Index(spd, ad_up, True)])


s6_vals = np.stack([
    1j *np.kron(sig1,sig2),
    1j *np.kron(sig2,sig3),
    -1 *np.kron(sig2,sig0),
    -1j *np.kron(sig2,sig1),
    -1 * np.kron(sig3,sig2),
    1j *np.kron(sig0,sig2),
]
    )

s6t_vals = np.stack([
    -1j *np.kron(sig1,sig2),
    1j *np.kron(sig2,sig3),
    1 *np.kron(sig2,sig0),
    -1j *np.kron(sig2,sig1),
    1 * np.kron(sig3,sig2),
    1j *np.kron(sig0,sig2),
    ])

def sigma6(mu,A,B):
    return Tensor(s6_vals, [Index(Lorenz6, mu, True), Index(sp6, A, False), Index(sp6, B,
False)])

def sigma6t(mu,A,B):
    return Tensor(s6t_vals, [Index(Lorenz6, mu, True), Index(sp6, A, True), Index(sp6, B,
True)])


def MP(p1,p2, mu='mu'):
    p1d = p1.lowerIndex(mu)
    p2u = p2.raiseIndex(mu)
    return contract(p1d,p2u, mu).data



c = contract

def eps2(a_up,b_up, raised=True, dotted=False):
    if dotted:
        return Tensor(sp.tensorToRaise, [Index(spd,a_up,raised), Index(spd,b_up,raised)])
    else:
        return Tensor(sp.tensorToRaise, [Index(sp,a_up,raised), Index(sp,b_up,raised)])
```



```
def momFromLa(la, mu='mu'):
    la_L = la.relabel(la.indices[0].label, 'Aintern').relabel(la.indices[1].label,
'aintern').raiseIndex('aintern')
    la_R = la.relabel(la.indices[0].label, 'Bintern').relabel(la.indices[1].label,
'bintern').raiseIndex('bintern')
    shifted = -0.25 * c(c(c(la_L, sigma6(mu, 'Aintern', 'Bintern'), 'Aintern'), la_R,
'Bintern'), eps2('aintern',
'bintern',raised=False),'aintern').contract('bintern').lowerIndex(mu)
    return shifted

def momFromLat(lat, mu='mu'):
    lat_L = lat.relabel(lat.indices[0].label, 'Aintern').relabel(lat.indices[1].label,
'adintern').lowerIndex('adintern')
    lat_R = lat.relabel(lat.indices[0].label, 'Bintern').relabel(lat.indices[1].label,
'bdintern').lowerIndex('bdintern')
    shifted = -0.25 * c(c(c(lat_L, sigma6t(mu, 'Aintern', 'Bintern'), 'Aintern'), lat_R,
'Bintern'), eps2('adintern', 'bdintern',raised=True,
dotted=True),'adintern').contract('bdintern').lowerIndex(mu)
    return shifted



def tinv(t):
    assert len(t.data.shape) == 2
    ind1, ind2 = t.indices
    data = np.linalg.inv(t.data)
    return Tensor(data, [Index(ind2.representation, ind2.label, not ind2.raised),
Index(ind1.representation, ind1.label, not ind1.raised)])


def eps4T(A_label='A', B_label='B', C_label='C', D_label='D', raised=False):
    return Tensor(eps4, [Index(sp6, 'A', raised), Index(sp6, 'B', raised), Index(sp6, 'C',
raised), Index(sp6, 'D', raised)])
```

In [1]:
```
import numpy as np
from tensors import *
from lorentz import makeOnShell
from spinor_products import *
```

In [2]:
```
mom6 = Tensor([np.sqrt(1+4+9+16+25),1,2,3,4,5], [Index(Lorenz6, "mu", False)])
```

In [3]:
```
pslash6 = contract(mom6,sigma6('mu','A', 'B'), 'mu')
pslash6t = contract(mom6,sigma6t('mu','A', 'B'), 'mu')
```

In [4]:
```
mom1 = makeOnShell(13, -3, -7, 2, -2, mu='mu')
mom2 = makeOnShell(-9,7,-5,1,-7, mu='mu')
```

In [5]:
```
def tinv(t):
    assert len(t.data.shape) == 2
    ind1, ind2 = t.indices
    data = np.linalg.inv(t.data)
    return Tensor(data, [Index(ind2.representation, ind2.label, not ind2.raised), Index(ind1.representation, ind1.label, not i
```

In [6]:
```
eps_vec = polvec(mom1,mom2)
eps_vec
```

Out[6]:
```
Tensor(data=[[[-0.23997307+0.02776756j -0.36734733-0.15111862j]
  [-0.36734733+0.15111862j  0.23997307+0.02776756j]]

 [[ 0.10623516+0.07372073j  0.01922527+0.31806546j]
  [ 0.01922527-0.31806546j -0.10623516+0.07372073j]]

 [[-0.13921547+0.09768719j  0.12852807+0.62132007j]
  [ 0.12852807-0.62132007j  0.13921547+0.09768719j]]

 [[-0.05720355-0.03969578j -0.77789146+0.06506783j]
  [-0.77789146-0.06506783j  0.05720355-0.03969578j]]

 [[ 0.01634387-0.69576513j  0.08043586+0.12529714j]
  [ 0.08043586-0.12529714j -0.01634387-0.69576513j]]

 [[-0.72345065-0.01134165j -0.08043586-0.12529714j]
  [-0.08043586+0.12529714j  0.72345065-0.01134165j]]], indices=['mu↑', 'a↓', 'ad↓'])
```

In [7]:
```
pv1 = polvec(mom1,mom2)
pv2 = polvec_v2(mom1,mom2)
(pv1-pv2).data.round(10)
```

Out[7]:
```
array([[[-0.-0.j,  0.-0.j],
        [-0.-0.j, -0.-0.j]],

       [[-0.-0.j,  0.+0.j],
        [ 0.+0.j, -0.-0.j]],

       [[ 0.+0.j, -0.+0.j],
        [-0.+0.j,  0.+0.j]],

       [[-0.-0.j,  0.-0.j],
        [ 0.+0.j, -0.-0.j]],

       [[ 0.-0.j, -0.+0.j],
        [ 0.+0.j,  0.+0.j]],

       [[-0.+0.j,  0.-0.j],
        [ 0.-0.j, -0.-0.j]]])
```

In [8]:
```
# eq 29

eps_vec_a = polvec(mom1,mom2, 'mu', 'a', 'ad')
eps_vec_b = polvec(mom1,mom2, 'mu', 'b', 'bd')

eps_contraction = c(eps_vec_a, eps_vec_b.lowerIndex('mu'), 'mu')
eps_prod = (eps2('a', 'b', raised=False)*eps2('ad', 'bd', raised=False, dotted=True)).reorder(eps_contraction.indices)
assert np.allclose((eps_contraction - eps_prod).data,0)
```

In [9]:
```
eps_vec_mu = polvec(mom1,mom2, 'mu', 'a', 'ad').raiseIndex('a').raiseIndex('ad')
eps_vec_nu = polvec(mom1,mom2, 'nu', 'a', 'ad')
```

In [10]:
```
LHS = c(eps_vec_mu, eps_vec_nu, 'a').contract('ad')
```

In [11]:
```
# eq 33

mom1nu = mom1.relabel('mu', 'nu')
mom2nu = mom2.relabel('mu', 'nu')

gmunu = Tensor(gmunu6_vals, [Index(Lorenz6, 'mu', False), Index(Lorenz6, 'nu', False)])

RHS = gmunu - (mom1*mom2nu + mom2*mom1nu)/float(MP(mom1,mom2))
RHS = RHS.raiseIndex('mu').raiseIndex('nu')
```

In [12]:
```
np.allclose(RHS.data, LHS.data)
```

Out[12]:
```
True
```

shift

In [13]:
```
x = Tensor([1,2j], [Index(sp, 'a', True)])
xt = Tensor([1,3], [Index(spd, 'ad', True)])

y = c(xt, tinv(sprod(mom2, mom1, 'b', 'ad')).lowerIndex('ad'), 'ad')
y
yt = c(x, tinv(sprod(mom1, mom2, 'a', 'bd')).lowerIndex('a'), 'a')
yt
```

Out[13]:
```
Tensor(data=[-0.02651765+0.01257003j -0.03776698+0.07661194j], indices=['bd↑'])
```

In [14]:
```
z = 1
```

```
FUDGE = -1


def shifted(mom1, mom2, z, x, xt):

    l1 = la6(mom1, 'A', 'a')
    l2 = la6(mom2, 'A', 'a')
    lt1 = lat6(mom1, 'A', 'ad')
    lt2 = lat6(mom2, 'A', 'ad')

    lx = c(x, l1.lowerIndex('a'), 'a')
    ly = c(y.relabel('b', 'a'), l2, 'a')
    ltx = c(xt.relabel('bd', 'ad'), lt1, 'ad')
    lty = c(yt.relabel('bd', 'ad'), lt2, 'ad')

    sqrt2 = np.sqrt(2)

    la1hat = l1 + FUDGE * sqrt2 * z * x*ly

    lat1hat = lt1 - sqrt2 * z * lty*xt.lowerIndex('ad')

    la2hat = l2 + sqrt2 * z * y.relabel('b','a').raiseIndex('a')*lx


    lat2hat = lt2 - sqrt2 * z * ltx.relabel('bd','ad')*yt.relabel('bd','ad').lowerIndex('ad')

    r = (momFromLa(la1hat, mu='mu')-mom1)
    return la1hat, lat1hat, la2hat, lat2hat, r
```

In [15]:
```
la1hat, lat1hat, la2hat, lat2hat, r = shifted(mom1, mom2, 1, x, xt)


shifted_1 = momFromLa(la1hat, mu='mu')
shifted_2 = momFromLa(la2hat, mu='mu')
shifted_1t = momFromLat(lat1hat, mu='mu')
shifted_2t = momFromLat(lat2hat, mu='mu')
```

In [16]:
```
(shifted_1-mom1).data.round(10)
```

Out[16]:
```
array([-1.81085763+0.27955545j, -0.35771751-0.42895666j,
       -0.90288576-3.05399637j,  2.02256761+1.05705391j,
       -4.68283652+0.25706522j,  1.14730261-3.79259912j])
```

In [17]:
```
(shifted_2-mom2).data.round(10)
```

Out[17]:
```
array([ 1.81085763-0.27955545j,  0.35771751+0.42895666j,
        0.90288576+3.05399637j, -2.02256761-1.05705391j,
        4.68283652-0.25706522j, -1.14730261+3.79259912j])
```

In [18]:
```
(shifted_1t-mom1).data.round(10)
```

Out[18]:
```
array([-1.81085763+0.27955545j, -0.35771751-0.42895666j,
       -0.90288576-3.05399637j,  2.02256761+1.05705391j,
       -4.68283652+0.25706522j,  1.14730261-3.79259912j])
```

In [19]:
```
(shifted_2t-mom2).data.round(10)
```

Out[19]:
```
array([ 1.81085763-0.27955545j,  0.35771751+0.42895666j,
        0.90288576+3.05399637j, -2.02256761-1.05705391j,
        4.68283652-0.25706522j, -1.14730261+3.79259912j])
```

In [20]:
```
zpart = (shifted_1-mom1)
```

In [21]:
```
MP(zpart,mom1)
```

Out[21]:
```
array(-3.8191672e-14-7.99360578e-15j)
```

In [22]:
```
MP(zpart,eps_vec).round(10)
```

Out[22]:
```
array([[ 0.+6.j,  0.-2.j],
       [-3.-0.j,  1.-0.j]])
```

In [23]:
```
MP(mom2,eps_vec).round(10)
```

Out[23]:
```
array([[ 0.+0.j, -0.+0.j],
       [-0.+0.j, -0.+0.j]])
```

In [24]:
```
xeps = c(c(x,eps_vec.lowerIndex('a').lowerIndex('ad'), 'a'), xt, 'ad').lowerIndex('mu')
xeps
```

Out[24]:
```
Tensor(data=[-1.81085763+0.27955545j -0.35771751-0.42895666j -0.90288576-3.05399637j
  2.02256761+1.05705391j -4.68283652+0.25706522j  1.14730261-3.79259912j], indices=['mu↓'])
```

In [25]:
```
(r.data/ xeps.data).round(10)
```

Out[25]:
```
array([1.-0.j, 1.+0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j])
```

3 point

I take the cyclic order to be 1hat, p ,3

In [26]:
```
mom3 = makeOnShell(1,-2,3,-4,25, mu='mu')
eps_vec = polvec(mom1,mom2)
xeps = c(c(x,eps_vec.lowerIndex('a').lowerIndex('ad'), 'a'), xt, 'ad').lowerIndex('mu')

z = - MP(mom1, mom3)/MP(xeps, mom3)

la1hat, lat1hat, la2hat, lat2hat, r = shifted(mom1, mom2, z, x, xt)

mom1hat = momFromLa(la1hat, mu='mu')

mom13 = -mom3 - mom1hat


mom1hat, mom3, mom13
```

Out[26]:
```
(Tensor(data=[10.36848958-3.36347295j 13.09835385-1.82445387j  1.63884869-9.32793632j
  -4.50852734+7.03734689j -9.79201332-9.81400084j  9.19589168-6.5312287j ], indices=['mu↓']),
 Tensor(data=[25.59296778  1.         -2.          3.         -4.         25.        ], indices=['mu↓']),
 Tensor(data=[-35.96145736+3.36347295j -14.09835385+1.82445387j
    0.36115131+9.32793632j   1.50852734-7.03734689j
   13.79201332+9.81400084j -34.19589168+6.5312287j ], indices=['mu↓']))
```

In [27]:
```
assert np.isclose(MP(mom1hat, mom1hat),0)
assert np.isclose(MP(mom3, mom3),0)
assert np.isclose(MP(mom13, mom13),0)
```

In [28]:
```
# def sprodFromLaLat(La_input, Lat_input, a_label='a', ad_label='ad'):
#     assert La_input.indices[0].representation == sp6 and Lat_input.indices[0].representation == sp6
#     assert La_input.indices[1].representation == sp and Lat_input.indices[1].representation == spd
#     la = La_input.relabel(La_input.indices[0].label, 'Aintern').relabel(La_input.indices[1].label, a_label)
#     lat = Lat_input.relabel(Lat_input.indices[0].label, 'Aintern').relabel(Lat_input.indices[1].label, ad_label)
#     return contract(la, lat, 'Aintern')
```

In [29]:
```
la3 = la6(mom3, 'A', 'a')
lat3 = lat6(mom3, 'A', 'ad')
la13 = la6(mom13, 'A', 'a')
lat13 = lat6(mom13, 'A', 'ad')


s1hp = sprod(mom1hat, mom13)
s3p = sprod(mom3, mom13)
s1h3 = sprod(mom1hat, mom3)

sp1h = sprod(mom13, mom1hat)
sp3 = sprod(mom13, mom3)
s31h = sprod(mom3, mom1hat)


s1hp = sprodFromLaLat(la1hat, lat13)
s3p = sprodFromLaLat(la3, lat13)
s1h3 = sprodFromLaLat(la1hat, lat3)

sp1h = sprodFromLaLat(la13, lat1hat)
sp3 = sprodFromLaLat(la13, lat3)
s31h = sprodFromLaLat(la3, lat1hat)



np.linalg.det(s1hp.data),np.linalg.det(s3p.data),np.linalg.det(s1h3.data)
```

Out[29]:
```
(np.complex128(9.96810815328841e-14+3.017822048365349e-13j),
 np.complex128(2.5906936961455506e-13+1.6872626448751288e-13j),
 np.complex128(-4.583676499560423e-13-3.5721343919827865e-13j))
```

In [30]:
```
def getuutilde(la1, la2, la3, lat1, lat2, lat3):
    '''strategy is to make the spinor product s12 diagonal, from there we can deduct the transformed versions
    of its two constituents as spinors with one zero constituent u1, ut2. Then we extract the other ut3 from s13 adn u3
    from s32 and then we can get ut1 and u2 from s21 and s31 respectively.
    '''
    s12 = sprodFromLaLat(la1, lat2).relabel(la1.indices[0].label, 'a').lowerIndex('a').lowerIndex('ad')
    s23 = sprodFromLaLat(la2, lat3).relabel(la1.indices[0].label, 'a').lowerIndex('a').lowerIndex('ad')
    s31 = sprodFromLaLat(la3, lat1).relabel(la1.indices[0].label, 'a').lowerIndex('a').lowerIndex('ad')

    s21 = sprodFromLaLat(la2, lat1).lowerIndex('a').lowerIndex('ad')
    s13 = sprodFromLaLat(la1, lat3).lowerIndex('a').lowerIndex('ad')
    s32 = sprodFromLaLat(la3, lat2).lowerIndex('a').lowerIndex('ad')



    eig_res = np.linalg.eig(s12.data)
    if np.isclose(eig_res.eigenvalues[1],0) :
        print('swap of eigenvectors needed!')
        # swap the eigenvalues adnd eigenvectors
        S = np.zeros((2,2), dtype=complex)
        orig = np.array(eig_res.eigenvectors)
        S[:,0] = orig[:,1]
        S[:,1] = orig[:,0]

    else:
        S = eig_res.eigenvectors
    Sinv = np.linalg.inv(S)

    s12_trans = (Sinv@s12.data@S)
    s13_trans = (Sinv@s13.data@S)
    s21_trans = (Sinv@s21.data@S)
    s23_trans = (Sinv@s23.data@S)
    s31_trans = (Sinv@s31.data@S)
    s32_trans = (Sinv@s32.data@S)


    val = s12_trans[1,1]
    u1_trans = Tensor([0, np.sqrt(val)], [Index(sp, 'a', False)])
    ut2_trans = Tensor([0, np.sqrt(val)], [Index(spd, 'ad', False)])
    ut3_trans = Tensor( - s13_trans[1]/np.sqrt(val), [Index(spd, 'ad', False)])  # negative because of the revers order
    u3_trans = Tensor( - s32_trans[:,1]/np.sqrt(val), [Index(sp, 'a', False)])
    ut1_trans = Tensor(s31_trans[0]/u3_trans[0].data, [Index(spd, 'ad', False)])
    u2_trans = Tensor(-s21_trans[:,0]/ut1_trans[0].data, [Index(sp, 'a', False)])

    #positive order
    assert np.allclose((u1_trans*ut2_trans).data, s12_trans), ((u1_trans*ut2_trans).data, s12_trans)
    assert np.allclose((u2_trans*ut3_trans).data, s23_trans)
    assert np.allclose((u3_trans*ut1_trans).data, s31_trans)

    # negative order
    assert np.allclose(-(u1_trans*ut3_trans).data, s13_trans)
    assert np.allclose(-(u2_trans*ut1_trans).data, s21_trans)
    assert np.allclose(-(u3_trans*ut2_trans).data, s32_trans)


    u1 = Tensor(S@u1_trans.data, [Index(sp, 'a', False)])
    u2 = Tensor(S@u2_trans.data, [Index(sp, 'a', False)])
    u3 = Tensor(S@u3_trans.data, [Index(sp, 'a', False)])
```

```
    ut1 = Tensor(Sinv.T@ut1_trans.data, [Index(spd, 'ad', False)])
    ut3 = Tensor(Sinv.T@ut3_trans.data, [Index(spd, 'ad', False)])
    ut2 = Tensor(Sinv.T@ut2_trans.data, [Index(spd, 'ad', False)])

    u1_1, u1h_2 = u1.data
    u2_1, up_2 = u2.data
    u3_1, u3_2 = u3.data

    w1_1, w1_2 = 0,1/u1_1
    w1 = Tensor([w1_1, w1_2], [Index(sp, 'a', False)])
    w2_1, w2_2 = 0,1/u2_1
    w2 = Tensor([w2_1, w2_2], [Index(sp, 'a', False)])
    w3_1, w3_2 = 0,1/u3_1
    w3pre = Tensor([w3_1, w3_2], [Index(sp, 'a', False)])
    #print(f'{w1=}, {w2=}, {w3pre=}')
    #print(f'{la1=}, {la2=}, {la3=}')
    momsum = (w1*la1 +w2*la2 + w3pre*la3  ).contract('a')
    factor = (momsum.data/(u1*la1).contract('a').data)[0]  # u1*la1 = u2*la2 = u3*la3
    w3 = w3pre - factor*u3

    ut1_1, ut1h_2 = ut1.data
    ut2_1, utp_2 = ut2.data
    ut3_1, ut3_2 = ut3.data

    wt1_1, wt1_2 = 0,1/ut1_1
    wt1 = Tensor([wt1_1, wt1_2], [Index(spd, 'ad', False)])
    wt2_1, wt2_2 = 0,1/ut2_1
    wt2 = Tensor([wt2_1, wt2_2], [Index(spd, 'ad', False)])
    wt3_1, wt3_2 = 0,1/ut3_1
    wt3pre = Tensor([wt3_1, wt3_2], [Index(spd, 'ad', False)])
    momsum = (wt1*lat1.raiseIndex('ad') + wt2*lat2.raiseIndex('ad') + wt3pre*lat3.raiseIndex('ad')  ).contract('ad')

    factor = (momsum.data/(ut1*lat1.raiseIndex('ad')).contract('ad').data)[0]  # u1*la1 = u2*la2 = u3*la3
    wt3 = wt3pre - factor*ut3



    return u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3
```

In [31]:
```
u1h, up, u3, ut1h, utp, ut3,w1, w2,  w3, wt1, wt2, wt3 = getuutilde(la1hat, la13, la3, lat1hat, lat13, lat3)
```

In [32]:
```
#positive order
assert np.allclose((u1h*utp).raiseIndex('a').data, s1hp.data)
assert np.allclose((up*ut3).raiseIndex('a').data, sp3.data)
assert np.allclose((u3*ut1h).raiseIndex('a').data, s31h.data)

# negative order
assert np.allclose(-(u1h*ut3).raiseIndex('a').data, s1h3.data)
assert np.allclose(-(up*ut1h).raiseIndex('a').data, sp1h.data)
assert np.allclose(-(u3*utp).raiseIndex('a').data, s3p.data)
```

In [33]:
```
assert np.allclose((u1h*w1.relabel('a', 'b')-u1h.relabel('a', 'b') *w1 ).data, eps2('a', 'b', raised=False).data)
assert np.allclose((up*w2.relabel('a', 'b')-up.relabel('a', 'b') *w2 ).data, eps2('a', 'b', raised=False).data)
assert np.allclose((u3*w3.relabel('a', 'b')-u3.relabel('a', 'b') *w3 ).data, eps2('a', 'b', raised=False).data)

assert np.allclose((ut1h*wt1.relabel('ad', 'bd')-ut1h.relabel('ad', 'bd') *wt1 ).data, eps2('ad', 'bd', raised=False, dotted=T
assert np.allclose((utp*wt2.relabel('ad', 'bd')-utp.relabel('ad', 'bd') *wt2 ).data, eps2('ad', 'bd', raised=False, dotted=Tru
assert np.allclose((ut3*wt3.relabel('ad', 'bd')-ut3.relabel('ad', 'bd') *wt3 ).data, eps2('ad', 'bd', raised=False, dotted=Tru
```

In [34]:
```
assert np.allclose((w1*la1hat +w2*la13 + w3*la3  ).contract('a').data, 0 )
assert np.allclose((wt1*lat1hat.raiseIndex('ad') +wt2*lat13.raiseIndex('ad') + wt3*lat3.raiseIndex('ad')  ).contract('ad').dat
```

In [35]:
```
def gammagammabar(la1, la2, la3, lat1, lat2, lat3, a_label='a', b_label='b', c_label='c', ad_label='ad', bd_label='bd', cd_lab

    u1, u2, u3, ut1, ut2, ut3, w1, w2, w3, wt1, wt2, wt3 = getuutilde(la1, la2, la3, lat1, lat2, lat3)
    u1 = u1.relabel(u1.indices[0].label, a_label)
    u2 = u2.relabel(u2.indices[0].label, b_label)
    u3 = u3.relabel(u3.indices[0].label, c_label)
    ut1 = ut1.relabel(ut1.indices[0].label, ad_label)
    ut2 = ut2.relabel(ut2.indices[0].label, bd_label)
    ut3 = ut3.relabel(ut3.indices[0].label, cd_label)

    w1 = w1.relabel(w1.indices[0].label, a_label)
    w2 = w2.relabel(w2.indices[0].label, b_label)
    w3 = w3.relabel(w3.indices[0].label, c_label)
    wt1 = wt1.relabel(wt1.indices[0].label, ad_label)
    wt2 = wt2.relabel(wt2.indices[0].label, bd_label)
    wt3 = wt3.relabel(wt3.indices[0].label, cd_label)

    return u1*u2*w3 + u1*w2*u3 + w1*u2*u3, ut1*ut2*wt3 + ut1*wt2*ut3 + wt1*ut2*ut3
```

In [36]:
```
gammagammabar(la1hat, la13, la3, lat1hat, lat13, lat3)
```

Out[36]:
```
(Tensor(data=[[[ 2.52064746+0.19643971j -4.4257652 +3.77913129j]
   [ 2.69341225-1.04943353j  1.3770117 +0.35908511j]]

  [[-1.75406118+1.39985499j  2.2970235 -3.94018678j]
   [-2.95683046-0.38050009j  0.25001487+0.02025387j]]], indices=['a↓', 'b↓', 'c↓']),
 Tensor(data=[[[-0.20347571+2.58694533j  7.55520942-9.67784805j]
   [-2.23806806+4.17027203j -0.738519  +5.29791853j]]

  [[ 1.64415816-0.32697043j -7.90077432-4.96683052j]
   [ 6.24102187+1.47889157j -0.55874876-0.54731561j]]], indices=['ad↓', 'bd↓', 'cd↓']))
```

4 point reconstruction

In [63]:
```
p1= makeOnShell( 1,1,2,1,1.5, mu='mu', energySign=-1)
p2= makeOnShell(-1,-1,-2,-1,1, mu='mu')
p3= makeOnShell( 2,-1,-1,1,-1, mu='mu', energySign=-1)
p4= makeOnShell(-2,1,1,-1,-1.5, mu='mu')

la1 = 1j*la6(-p1, 'A', 'a')
la2 = la6(p2, 'B', 'b')
la3 = 1j*la6(-p3, 'C', 'c')
la4 = la6(p4, 'D', 'd')
```

```
lat1 = 1j*lat6(-p1, 'A', 'ad')
lat2 = lat6(p2, 'B', 'bd')
lat3 = 1j*lat6(-p3, 'C', 'cd')
lat4 = lat6(p4, 'D', 'dd')
```

In [64]:
```
p1+p2+p3+p4
```

Out[64]:
```
Tensor(data=[0. 0. 0. 0. 0. 0.], indices=['mu↓'])
```

In [ ]:
```
def eps4T(A_label='A', B_label='B', C_label='C', D_label='D', raised=False):
    return Tensor(eps4, [Index(sp6, 'A', raised), Index(sp6, 'B', raised), Index(sp6, 'C', raised), Index(sp6, 'D', raised)])
```

In [65]:
```
f1 = c(c(c(c(eps4T(),la1,'A'),la2, 'B'), la3, 'C'), la4, 'D')
f1.data.round(10)
```

Out[65]:
```
array([[[[  0.         +0.j        ,  -1.61437155-12.73741469j],
         [  1.1660456 +12.10674129j,  -4.83675476 -3.82697997j]],

        [[  2.99439733 +4.97757293j,   5.17464855 +5.80493468j],
         [-11.69895179 -4.33153493j,   5.32809638 -3.69752118j]]],


       [[[ -5.32809638 -3.69752118j, -11.69895179 +4.33153493j],
         [  5.17464855 -5.80493468j,  -2.99439733 +4.97757293j]],

        [[ -4.83675476 +3.82697997j,  -1.1660456 +12.10674129j],
         [  1.61437155-12.73741469j,   0.         +0.j        ]]]])
```

In [66]:
```
f2 = c(c(c(c(eps4T(raised=True),lat1,'A'),lat2, 'B'), lat3, 'C'), lat4, 'D')
f2.data.round(10)
```

Out[66]:
```
array([[[[  0.         +0.j        ,  -2.31929248-12.86435175j],
         [  1.82990707+12.24700077j,  -5.00451188 -3.60251446j]],

        [[  3.30936694 +4.90138032j,   6.68078996 -1.77089997j],
         [-12.28946561 -1.76683215j,   5.61393547 -3.47572705j]]],


       [[[ -5.61393547 -3.47572705j, -12.28946561 +1.76683215j],
         [  6.68078996 +1.77089997j,  -3.30936694 +4.90138032j]],

        [[ -5.00451188 +3.60251446j,  -1.82990707+12.24700077j],
         [  2.31929248-12.86435175j,   0.         +0.j        ]]]])
```

In [67]:
```
t= -2*MP(p1,p3)
s = 2*MP(p1,p2)
prod_target = -1j/(s*t)* (f1*f2).lowerIndex('a').lowerIndex('b').lowerIndex('c').lowerIndex('d')

prod_target.data.round(10)
```

Out[67]:
```
array([[[[[[[[ 0.        -0.j        ,  0.        -0.j        ],
             [ 0.        -0.j        ,  0.        -0.j        ]],

            [[ 0.        -0.j        ,  0.        -0.j        ],
             [ 0.        -0.j        ,  0.        -0.j        ]]],


           [[[ 0.        -0.j        ,  0.        -0.j        ],
             [ 0.        -0.j        ,  0.        -0.j        ]],

            [[ 0.        -0.j        ,  0.        -0.j        ],
             [ 0.        -0.j        ,  0.        -0.j        ]]]],



          [[[[ 0.        +0.j        , -0.06998844-1.33694197j],
             [ 0.02821472+1.26791427j, -0.46208882-0.43047749j]],

            [[ 0.27312832+0.54061862j,  0.70160275-0.09389875j],
             [-1.22591153-0.33777675j,  0.61515969-0.28085525j]]],


           [[[-0.52564167-0.42544324j, -1.27141663+0.02125918j],
             [ 0.65599288+0.26596379j, -0.39936432+0.45538523j]],

            [[-0.55487228+0.30158532j, -0.34363859+1.22078466j],
             [ 0.40131231-1.27720814j,  0.        +0.j        ]]]],



         [[[[[-0.        -0.j        ,  0.10432643+1.26392985j],
             [-0.06280691-1.19975806j,  0.4497956 +0.39445674j]],

            [[-0.27400806-0.50412577j, -0.66165999+0.10888176j],
             [ 1.17040445+0.28493826j, -0.57448719+0.28344632j]]],


           [[[ 0.50982915+0.38788089j,  1.20327236-0.05632056j],
             [-0.6287164 -0.23316274j,  0.36518715-0.44256249j]],

            [[ 0.51681224-0.30135912j,  0.29063465-1.16571678j],
             [-0.34363859+1.22078466j, -0.        -0.j        ]]]],



          [[[[ 0.        +0.j        ,  0.42553158-0.48219621j],
             [-0.4166525 +0.4444687j , -0.0137815 -0.30305919j]],

            [[-0.08807951+0.2773078j ,  0.27227104+0.20369795j],
             [-0.30699543-0.52808939j,  0.30547865+0.11049284j]]],


           [[[-0.03727724-0.32270154j, -0.44333147-0.42021638j],
             [ 0.1356211 +0.31181932j, -0.29013118+0.02194342j]],

            [[-0.29176635-0.08310955j, -0.52837651+0.30326553j],
             [ 0.56713477-0.30323006j,  0.        +0.j        ]]]]],




         [[[[[[-0.        -0.j        , -0.21518771+0.56618179j],
             [ 0.21987297-0.52998027j,  0.11265673+0.26257611j]],

            [[-0.0143259 -0.27365804j, -0.30756252-0.08926218j],
             [ 0.44575517+0.36369705j, -0.30592395+0.0039113j ]]],


           [[[ 0.13988253+0.27209858j,  0.53015975+0.22339169j],
             [-0.22296361-0.22989056j,  0.2484732 -0.11556394j]],

            [[ 0.284755  -0.02350204j,  0.36518715-0.44256249j],
             [-0.39936432+0.45538523j, -0.        -0.j        ]]]],



          [[[[-0.        -0.j        , -0.42361179+0.69141921j],
             [ 0.42079018-0.64263217j,  0.08303196+0.3733879j ]],

            [[ 0.04907566-0.36356084j, -0.38245266-0.19376388j],
             [ 0.49613451+0.58909033j, -0.40342268-0.07078475j]]],


           [[[ 0.11648482+0.39267241j,  0.64199494+0.42546371j],
             [-0.23625641-0.35776722j,  0.35555668-0.09035681j]],

            [[ 0.38043672+0.03975844j,  0.59025818-0.49156461j],
             [-0.63840185+0.49995055j, -0.        -0.j        ]]]],



         [[[[[ 0.        +0.j        ,  1.12037494-0.66092719j],
             [-1.07967252+0.59392666j,  0.16327352-0.59149894j]],

            [[-0.34305532+0.47818513j,  0.39609637+0.56226829j],
             [-0.25974345-1.20790989j,  0.51833018+0.40380343j]]],


           [[[ 0.13038482-0.64399005j, -0.5895077 -1.08581473j],
             [ 0.0655729 +0.68464455j, -0.57174579-0.13948017j]],

            [[-0.50910528-0.3425508j , -1.2061264 +0.25239001j],
             [ 1.28064731-0.2280509j , -0.        +0.j        ]]]],
```

```
   [[[[ 0.        -0.j        , -0.61515969-0.28085525j],
      [ 0.57448719+0.28344632j, -0.30071787+0.10644424j]],

     [[ 0.30592395+0.0039113j ,  0.12178124-0.33617496j],
      [-0.43756532+0.47020775j,  0.01785726-0.34111527j]]],


    [[[-0.31330391+0.13608515j, -0.28737958+0.57443164j],
      [ 0.27231275-0.23171111j,  0.11070727+0.28521686j]],

     [[ 0.00550606+0.31895344j,  0.46654257+0.43899361j],
      [-0.47834665-0.47800168j,  0.        -0.j        ]]]]]]],




[[[[[[[ 0.        +0.j        ,  0.47834665-0.47800168j],
       [-0.46654257+0.43899361j,  0.00550606-0.31895344j]],

      [[-0.11070727+0.28521686j,  0.27231275+0.23171111j],
       [-0.28737958-0.57443164j,  0.31330391+0.13608515j]]],


     [[[-0.01785726-0.34111527j, -0.43756532-0.47020775j],
       [ 0.12178124+0.33617496j, -0.30592395+0.0039113j ]],

      [[-0.30071787-0.10644424j, -0.57448719+0.28344632j],
       [ 0.61515969-0.28085525j,  0.        +0.j        ]]]],


    [[[[-0.        +0.j        ,  1.28064731+0.2280509j ],
       [-1.2061264 -0.25239001j,  0.50910528-0.3425508j ]],

      [[-0.57174579+0.13948017j, -0.0655729 +0.68464455j],
       [ 0.5895077 -1.08581473j,  0.13038482+0.64399005j]]],


     [[[ 0.51833018-0.40380343j,  0.25974345-1.20790989j],
       [-0.39609637+0.56226829j, -0.34305532-0.47818513j]],

      [[-0.16327352-0.59149894j, -1.07967252-0.59392666j],
       [ 1.12037494+0.66092719j, -0.        +0.j        ]]]],



   [[[[[ 0.        -0.j        , -0.63840185-0.49995055j],
       [ 0.59025818+0.49156461j, -0.38043672+0.03975844j]],

      [[ 0.35555668+0.09035681j,  0.23625641-0.35776722j],
       [-0.64199494+0.42546371j,  0.11648482-0.39267241j]]],


     [[[-0.40342268+0.07078475j, -0.49613451+0.58909033j],
       [ 0.38245266-0.19376388j,  0.04907566+0.36356084j]],

      [[-0.08303196+0.3733879j ,  0.42079018+0.64263217j],
       [-0.42361179-0.69141921j,  0.        -0.j        ]]],


    [[[[-0.        +0.j        ,  0.39936432+0.45538523j],
       [-0.36518715-0.44256249j,  0.284755  +0.02350204j]],

      [[-0.2484732 -0.11556394j, -0.22296361+0.22989056j],
       [ 0.53015975-0.22339169j, -0.13988253+0.27209858j]]],


     [[[ 0.30592395+0.0039113j ,  0.44575517-0.36369705j],
       [-0.30756252+0.08926218j,  0.0143259 -0.27365804j]],

      [[ 0.11265673-0.26257611j, -0.21987297-0.52998027j],
       [ 0.21518771+0.56618179j, -0.        +0.j        ]]]]],




 [[[[[[-0.        +0.j        ,  0.56713477+0.30323006j],
       [-0.52837651-0.30326553j,  0.29176635-0.08310955j]],

      [[-0.29013118-0.02194342j, -0.1356211 +0.31181932j],
       [ 0.44333147-0.42021638j, -0.03727724+0.32270154j]]],


     [[[ 0.30547865-0.11049284j,  0.30699543-0.52808939j],
       [-0.27227104+0.20369795j, -0.08807951-0.2773078j ]],

      [[ 0.0137815 -0.30305919j, -0.4166525 -0.4444687j ],
       [ 0.42553158+0.48219621j, -0.        +0.j        ]]]],


    [[[[-0.        +0.j        ,  0.34363859+1.22078466j],
       [-0.29063465-1.16571678j,  0.51681224+0.30135912j]],

      [[-0.36518715-0.44256249j, -0.6287164 +0.23316274j],
       [ 1.20327236+0.05632056j, -0.50982915+0.38788089j]]],


     [[[ 0.57448719+0.28344632j,  1.17040445-0.28493826j],
       [-0.66165999-0.10888176j,  0.27400806-0.50412577j]],

      [[ 0.4497956 -0.39445674j,  0.06280691-1.19975806j],
       [-0.10432643+1.26392985j, -0.        -0.j        ]]]],
```

```
[[[[[ 0.        -0.j        , -0.40131231-1.27720814j],
      [ 0.34363859+1.22078466j, -0.55487228-0.30158532j]],

     [[ 0.39936432+0.45538523j,  0.65599288-0.26596379j],
      [-1.27141663-0.02125918j,  0.52564167-0.42544324j]]],


    [[[-0.61515969-0.28085525j, -1.22591153+0.33777675j],
      [ 0.70160275+0.09389875j, -0.27312832+0.54061862j]],

     [[-0.46208882+0.43047749j, -0.02821472+1.26791427j],
      [ 0.06998844-1.33694197j,  0.        +0.j        ]]]],



   [[[[ 0.        -0.j        , -0.        -0.j        ],
      [ 0.        +0.j        , -0.        +0.j        ]],

     [[ 0.        -0.j        , -0.        -0.j        ],
      [-0.        +0.j        , -0.        -0.j        ]]],


    [[[-0.        +0.j        , -0.        +0.j        ],
      [ 0.        -0.j        ,  0.        +0.j        ]],

     [[ 0.        +0.j        ,  0.        +0.j        ],
      [-0.        -0.j        ,  0.        -0.j        ]]]]])
```

In [71]:
```
def polvecFromLa(la_in, laq_in, lat_in, latq_in, mu='mu', a_label='a', ad_label='ad'):
    la = la_in.relabel(la_in.indices[0].label, 'Aintern').relabel(la_in.indices[1].label, a_label)
    laq = laq_in.relabel(laq_in.indices[0].label, 'Bintern').relabel(laq_in.indices[1].label, 'bintern')
    tmp = c(la.lowerIndex(a_label), sigma6(mu,'Aintern', 'Bintern'), 'Aintern')
    #print(f'{tmp=}')
    tmp = c(tmp, laq.lowerIndex('bintern'),'Bintern')
    #print(tmp)
    tmp = c(tmp, tinv(sprodFromLaLat(laq, lat_in, 'bintern', ad_label)).raiseIndex('bintern').lowerIndex(ad_label),'bintern')
    #print(tmp)
    tmp =  1/np.sqrt(2)*tmp
    return tmp.reorder([Index(Lorenz6, mu, raised=True), Index(sp, a_label, raised=False), Index(spd, ad_label, raised=False)]
```

In [72]:
```
def shiftedFromLa(la1,la2,lat1,lat2, z, x, xt):

    y = c(xt, tinv(sprodFromLaLat(la2, lat1, 'b', 'ad')).lowerIndex('ad'), 'ad')
    yt = c(x, tinv(sprodFromLaLat(la1, lat2, 'a', 'bd')).lowerIndex('a'), 'a')

    l1 = la1.relabel(la1.indices[0].label, 'A').relabel(la1.indices[1].label, 'a')
    l2 = la2.relabel(la2.indices[0].label, 'A').relabel(la2.indices[1].label, 'a')
    lt1 = lat1.relabel(lat1.indices[0].label, 'A').relabel(lat1.indices[1].label, 'ad')
    lt2 = lat2.relabel(lat2.indices[0].label, 'A').relabel(lat2.indices[1].label, 'ad')

    lx = c(x, l1.lowerIndex('a'), 'a')
    ly = c(y.relabel('b', 'a'), l2, 'a')
    ltx = c(xt.relabel('bd', 'ad'), lt1, 'ad')
    lty = c(yt.relabel('bd', 'ad'), lt2, 'ad')

    sqrt2 = np.sqrt(2)

    la1hat = l1 + FUDGE * sqrt2 * z * x*ly

    lat1hat = lt1 - sqrt2 * z * lty*xt.lowerIndex('ad')

    la2hat = l2 + sqrt2 * z * y.relabel('b','a').raiseIndex('a')*lx


    lat2hat = lt2 - sqrt2 * z * ltx.relabel('bd','ad')*yt.relabel('bd','ad').lowerIndex('ad')

    mom1 = momFromLa(la1, mu='mu')

    r = (momFromLa(la1hat, mu='mu')-mom1)
    return la1hat, lat1hat, la2hat, lat2hat, r
```

In [73]:
```
x = Tensor([1,0], [Index(sp, 'a', True)])
xt = Tensor([1,0], [Index(spd, 'ad', True)])

eps_vec = polvecFromLa(la1,la2,lat1, lat2)
#eps_vec = polvec(-p1,p2)

eps_vec.data.round(10)
```

Out[73]:
```
array([[[ 1.10498716-0.41870179j, -0.3480831 +0.8691341j ],
        [-0.3480831 -0.8691341j , -1.10498716-0.41870179j]],

       [[ 0.53415428-0.02377714j, -0.72445607+0.10777809j],
        [-0.72445607-0.10777809j, -0.53415428-0.02377714j]],

       [[ 0.01457462-0.37016358j, -0.3077314 +0.76837931j],
        [-0.3077314 -0.76837931j, -0.01457462-0.37016358j]],

       [[ 1.06830856-0.04755428j,  0.01888818+0.80340159j],
        [ 0.01888818-0.80340159j, -1.06830856-0.04755428j]],

       [[ 0.53415428-0.73088392j, -0.01976686+0.04935619j],
        [-0.01976686-0.04935619j, -0.53415428-0.73088392j]],

       [[ 0.09412464-0.03566571j, -0.02965029+0.07403428j],
        [-0.02965029-0.07403428j, -0.09412464-0.03566571j]]])
```

In [74]:
```
xeps = c(c(x,eps_vec.lowerIndex('a').lowerIndex('ad'), 'a'), xt, 'ad').lowerIndex('mu')

z = - MP(p1, p4)/MP(xeps, p4)

la1hat, lat1hat, la2hat, lat2hat, r = shiftedFromLa(la1, la2, lat1, lat2, z, x, xt)

mom1hat = momFromLa(la1hat, mu='mu')
mom2hat = momFromLa(la2hat, mu='mu')
```

```
momP = -p4 - mom1hat
lap, latp = la6(momP, 'A', 'a'), lat6(momP, 'A', 'ad')
lamp, latmp = 1j*lap, 1j * latp

mom1hat, p4, momP
```

Out[74]:
```
(Tensor(data=[-0.12627773-0.15786318j -0.27534359-0.34421442j  0.68835898+0.86053606j
  -0.55068719-0.68842885j -0.80511228+1.32048762j  1.25168665+0.01344705j], indices=['mu↓']),
 Tensor(data=[ 3.04138127 -2.          1.          1.         -1.         -1.5       ], indices=['mu↓']),
 Tensor(data=[-2.91510354+0.15786318j  2.27534359+0.34421442j -1.68835898-0.86053606j
  -0.44931281+0.68842885j  1.80511228-1.32048762j  0.24831335-0.01344705j], indices=['mu↓']))
```

In [75]:
```
assert np.allclose((mom1hat + p4 + momP).data, 0)
assert(np.isclose(MP(mom1hat, mom1hat),0))
assert(np.isclose(MP(p4, p4),0))
assert(np.isclose(MP(momP, momP),0))
```

In [76]:
```
assert np.allclose((mom2hat + p3 - momP).data, 0)
assert(np.isclose(MP(mom2hat, mom2hat),0))
assert(np.isclose(MP(p3, p3),0))
assert(np.isclose(MP(-momP, -momP),0))
```

In [ ]:
```
gL, gtL = gammagammabar(la1hat, lap, la4.relabel('d', 'a').relabel('D','A'), lat1hat, latp, lat4.relabel('D','A').relabel('dd'
gR, gtR = gammagammabar(la2hat, la3.relabel('c', 'a').relabel('C','A'), lamp, lat2hat, lat3.relabel('C','A').relabel('cd','ad'
prod = -1j/complex(2*MP(p1,p3))*c(gL,gR.raiseIndex('e'),'e')*c(gtL,gtR.raiseIndex('ed'),'ed')
reordered = prod.reorder(prod_target.indices)
contracted = c(c(reordered,x,'a'), xt,'ad')
```

In [ ]:

swap of eigenvectors needed!

In [ ]:

In [ ]:

In [ ]:
```
contracted.data.round(4)
```

Out[ ]:
```
array([[[[[[ 0.    +0.j    ,  0.    +0.j    ],
           [-0.    -0.j    ,  0.    +0.j    ]],

          [[-0.    -0.j    , -0.    +0.j    ],
           [ 0.    +0.j    , -0.    +0.j    ]]],


         [[[-0.    -0.j    , -0.07  -1.3369j],
           [ 0.0282+1.2679j, -0.4621-0.4305j]],

          [[ 0.2731+0.5406j,  0.7016-0.0939j],
           [-1.2259-0.3378j,  0.6152-0.2809j]]]],



        [[[[ 0.    +0.j    ,  0.1043+1.2639j],
           [-0.0628-1.1998j,  0.4498+0.3945j]],

          [[-0.274 -0.5041j, -0.6617+0.1089j],
           [ 1.1704+0.2849j, -0.5745+0.2834j]]],


         [[[ 0.    -0.j    ,  0.4255-0.4822j],
           [-0.4167+0.4445j, -0.0138-0.3031j]],

          [[-0.0881+0.2773j,  0.2723+0.2037j],
           [-0.307 -0.5281j,  0.3055+0.1105j]]]]],



       [[[[[ 0.    +0.j    , -0.2152+0.5662j],
           [ 0.2199-0.53j  ,  0.1127+0.2626j]],

          [[-0.0143-0.2737j, -0.3076-0.0893j],
           [ 0.4458+0.3637j, -0.3059+0.0039j]]],


         [[[-0.    +0.j    , -0.4236+0.6914j],
           [ 0.4208-0.6426j,  0.083 +0.3734j]],

          [[ 0.0491-0.3636j, -0.3825-0.1938j],
           [ 0.4961+0.5891j, -0.4034-0.0708j]]]],



        [[[[ 0.    -0.j    ,  1.1204-0.6609j],
           [-1.0797+0.5939j,  0.1633-0.5915j]],

          [[-0.3431+0.4782j,  0.3961+0.5623j],
           [-0.2597-1.2079j,  0.5183+0.4038j]]],


         [[[-0.    +0.j    , -0.6152-0.2809j],
           [ 0.5745+0.2834j, -0.3007+0.1064j]],

          [[ 0.3059+0.0039j,  0.1218-0.3362j],
           [-0.4376+0.4702j,  0.0179-0.3411j]]]]]])
```

In [99]:
```
selected_target = prod_target.data[0,:,:,:,0,:,:,:]
selected_target.round(4)
```

Out[99]:
```
array([[[[[[ 0.    -0.j    ,  0.    -0.j    ],
           [ 0.    -0.j    ,  0.    -0.j    ]],

          [[ 0.    -0.j    ,  0.    -0.j    ],
           [ 0.    -0.j    ,  0.    -0.j    ]]],


         [[[ 0.    +0.j    , -0.07  -1.3369j],
           [ 0.0282+1.2679j, -0.4621-0.4305j]],

          [[ 0.2731+0.5406j,  0.7016-0.0939j],
           [-1.2259-0.3378j,  0.6152-0.2809j]]]],



        [[[[-0.    -0.j    ,  0.1043+1.2639j],
           [-0.0628-1.1998j,  0.4498+0.3945j]],

          [[-0.274 -0.5041j, -0.6617+0.1089j],
           [ 1.1704+0.2849j, -0.5745+0.2834j]]],


         [[[ 0.    +0.j    ,  0.4255-0.4822j],
           [-0.4167+0.4445j, -0.0138-0.3031j]],

          [[-0.0881+0.2773j,  0.2723+0.2037j],
           [-0.307 -0.5281j,  0.3055+0.1105j]]]]],



       [[[[[-0.    -0.j    , -0.2152+0.5662j],
           [ 0.2199-0.53j  ,  0.1127+0.2626j]],

          [[-0.0143-0.2737j, -0.3076-0.0893j],
           [ 0.4458+0.3637j, -0.3059+0.0039j]]],


         [[[-0.    -0.j    , -0.4236+0.6914j],
           [ 0.4208-0.6426j,  0.083 +0.3734j]],

          [[ 0.0491-0.3636j, -0.3825-0.1938j],
           [ 0.4961+0.5891j, -0.4034-0.0708j]]]],



        [[[[ 0.    +0.j    ,  1.1204-0.6609j],
           [-1.0797+0.5939j,  0.1633-0.5915j]],

          [[-0.3431+0.4782j,  0.3961+0.5623j],
           [-0.2597-1.2079j,  0.5183+0.4038j]]],


         [[[ 0.    -0.j    , -0.6152-0.2809j],
           [ 0.5745+0.2834j, -0.3007+0.1064j]],

          [[ 0.3059+0.0039j,  0.1218-0.3362j],
           [-0.4376+0.4702j,  0.0179-0.3411j]]]]])
```

In [ ]:
```
nonzero = abs(contracted.data)>1e-8
nonzero_target = abs(selected_target)>1e-8

assert np.all(nonzero == nonzero_target)
```

In [100…
```
(contracted.data[nonzero]/selected_target[nonzero]).round(10)
```

Out[100…
```
array([1.+0.j, 1.+0.j, 1.+0.j, 1.-0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
       1.+0.j, 1.+0.j, 1.-0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
       1.+0.j, 1.-0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.-0.j, 1.-0.j, 1.-0.j,
       1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.-0.j,
       1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.-0.j, 1.+0.j,
       1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.-0.j, 1.+0.j, 1.+0.j,
       1.+0.j])
```

In [62]:
```
1
```

Out[62]:
```
1
```

In [110…
```
def BCFW(x,xt):
    eps_vec = polvecFromLa(la1,la2,lat1, lat2)
    xeps = c(c(x,eps_vec.lowerIndex('a').lowerIndex('ad'), 'a'), xt, 'ad').lowerIndex('mu')

    z = - MP(p1, p4)/MP(xeps, p4)

    la1hat, lat1hat, la2hat, lat2hat, r = shiftedFromLa(la1, la2, lat1, lat2, z, x, xt)

    mom1hat = momFromLa(la1hat, mu='mu')
    mom2hat = momFromLa(la2hat, mu='mu')

    momP = -p4 - mom1hat
    lap, latp = la6(momP, 'A', 'a'), lat6(momP, 'A', 'ad')
    lamp, latmp = 1j*lap, 1j * latp

    assert np.allclose((mom1hat + p4 + momP).data, 0)
    assert(np.isclose(MP(mom1hat, mom1hat),0))
    assert(np.isclose(MP(p4, p4),0))
    assert(np.isclose(MP(momP, momP),0))

    assert np.allclose((mom2hat + p3 - momP).data, 0)
    assert(np.isclose(MP(mom2hat, mom2hat),0))
    assert(np.isclose(MP(p3, p3),0))
    assert(np.isclose(MP(-momP, -momP),0))

    gL, gtL = gammagammabar(la1hat, lap, la4.relabel('d', 'a').relabel('D','A'), lat1hat, latp, lat4.relabel('D','A').relabel(
    gR, gtR = gammagammabar(la2hat, la3.relabel('c', 'a').relabel('C','A'), lamp, lat2hat, lat3.relabel('C','A').relabel('cd',
    prod = -1j/complex(2*MP(p1,p3))*c(gL,gR.raiseIndex('e'),'e')*c(gtL,gtR.raiseIndex('ed'),'ed')
    reordered = prod.reorder(prod_target.indices)
    contracted = c(c(reordered,x,'a'), xt,'ad')
```

```
    return contracted
```

In [111…
```
x = Tensor([1,0], [Index(sp, 'a', True)])
xt = Tensor([1,0], [Index(spd, 'ad', True)])

bcfw = BCFW(x,xt)

selected_target = prod_target.data[0,:,:,:,0,:,:,:]

nonzero = abs(contracted.data)>1e-8
nonzero_target = abs(selected_target)>1e-8

assert np.all(nonzero == nonzero_target)

assert np.allclose((bcfw.data[nonzero]/selected_target[nonzero]), 1)
```

```
swap of eigenvectors needed!
```

In [121…
```
x = Tensor([0,1], [Index(sp, 'a', True)])
xt = Tensor([0,1], [Index(spd, 'ad', True)])

bcfw = BCFW(x,xt)

selected_target = prod_target.data[1,:,:,:,1,:,:,:]

nonzero = abs(bcfw.data)>1e-8
nonzero_target = abs(selected_target)>1e-8

assert np.all(nonzero == nonzero_target)

assert np.allclose((bcfw.data[nonzero]/selected_target[nonzero]), 1)
```

```
swap of eigenvectors needed!
```

In [122…
```
x = Tensor([1,0], [Index(sp, 'a', True)])
xt = Tensor([0,1], [Index(spd, 'ad', True)])

bcfw = BCFW(x,xt)

selected_target = prod_target.data[0,:,:,:,1,:,:,:]

nonzero = abs(bcfw.data)>1e-8
nonzero_target = abs(selected_target)>1e-8

assert np.all(nonzero == nonzero_target)

assert np.allclose((bcfw.data[nonzero]/selected_target[nonzero]), 1)
```

In [123…
```
x = Tensor([0,1], [Index(sp, 'a', True)])
xt = Tensor([1,0], [Index(spd, 'ad', True)])

bcfw = BCFW(x,xt)

selected_target = prod_target.data[1,:,:,:,0,:,:,:]

nonzero = abs(bcfw.data)>1e-8
nonzero_target = abs(selected_target)>1e-8

assert np.all(nonzero == nonzero_target)

assert np.allclose((bcfw.data[nonzero]/selected_target[nonzero]), 1)
```

```
swap of eigenvectors needed!
swap of eigenvectors needed!
```

In [ ]:

In [1]:
```
import numpy as np
from tensors import *
from lorentz import makeOnShell
from spinor_products import *
```

3 point

I take the cyclic order to be 1hat, p ,3

In [2]:
```
def getuutilde(la1, la2, la3, lat1, lat2, lat3):
    '''strategy is to make the spinor product s12 diagonal, from there we can deduct the transformed versions
    of its two constituents as spinors with one zero constituent u1, ut2. Then we extract the other ut3 from s13 adn u3
    from s32 and then we can get ut1 and u2 from s21 and s31 respectively.
    '''
    s12 = sprodFromLaLat(la1, lat2).relabel(la1.indices[0].label, 'a').lowerIndex('a').lowerIndex('ad')
    s23 = sprodFromLaLat(la2, lat3).relabel(la1.indices[0].label, 'a').lowerIndex('a').lowerIndex('ad')
    s31 = sprodFromLaLat(la3, lat1).relabel(la1.indices[0].label, 'a').lowerIndex('a').lowerIndex('ad')

    s21 = sprodFromLaLat(la2, lat1).lowerIndex('a').lowerIndex('ad')
    s13 = sprodFromLaLat(la1, lat3).lowerIndex('a').lowerIndex('ad')
    s32 = sprodFromLaLat(la3, lat2).lowerIndex('a').lowerIndex('ad')



    eig_res = np.linalg.eig(s12.data)
    if np.isclose(eig_res.eigenvalues[1],0) :
        #print('swap of eigenvectors needed!')
        # swap the eigenvalues adnd eigenvectors
        S = np.zeros((2,2), dtype=complex)
        orig = np.array(eig_res.eigenvectors)
        S[:,0] = orig[:,1]
        S[:,1] = orig[:,0]

    else:
        S = eig_res.eigenvectors
    Sinv = np.linalg.inv(S)

    s12_trans = (Sinv@s12.data@S)
    assert np.isclose(s12_trans[0,0], 0), f'{s12_trans.round(10)}'
    s13_trans = (Sinv@s13.data@S)
    s21_trans = (Sinv@s21.data@S)
    s23_trans = (Sinv@s23.data@S)
    s31_trans = (Sinv@s31.data@S)
    s32_trans = (Sinv@s32.data@S)


    val = s12_trans[1,1]
    u1_trans = Tensor([0, np.sqrt(val)], [Index(sp, 'a', False)])
    ut2_trans = Tensor([0, np.sqrt(val)], [Index(spd, 'ad', False)])
    ut3_trans = Tensor( - s13_trans[1]/np.sqrt(val), [Index(spd, 'ad', False)])  # negative because of the revers order
    u3_trans = Tensor( - s32_trans[:,1]/np.sqrt(val), [Index(sp, 'a', False)])
    ut1_trans = Tensor(s31_trans[0]/u3_trans[0].data, [Index(spd, 'ad', False)])
    u2_trans = Tensor(-s21_trans[:,0]/ut1_trans[0].data, [Index(sp, 'a', False)])

    #positive order
    assert np.allclose((u1_trans*ut2_trans).data, s12_trans), ((u1_trans*ut2_trans).data.round(10), s12_trans.round(10))
    assert np.allclose((u2_trans*ut3_trans).data, s23_trans), ((u2_trans*ut3_trans).data, s23_trans)
    assert np.allclose((u3_trans*ut1_trans).data, s31_trans)

    # negative order
    assert np.allclose(-(u1_trans*ut3_trans).data, s13_trans), (-(u1_trans*ut3_trans).data, s13_trans)
    assert np.allclose(-(u2_trans*ut1_trans).data, s21_trans)
    assert np.allclose(-(u3_trans*ut2_trans).data, s32_trans)


    u1 = Tensor(S@u1_trans.data, [Index(sp, 'a', False)])
    u2 = Tensor(S@u2_trans.data, [Index(sp, 'a', False)])
    u3 = Tensor(S@u3_trans.data, [Index(sp, 'a', False)])

    ut1 = Tensor(Sinv.T@ut1_trans.data, [Index(spd, 'ad', False)])
    ut3 = Tensor(Sinv.T@ut3_trans.data, [Index(spd, 'ad', False)])
    ut2 = Tensor(Sinv.T@ut2_trans.data, [Index(spd, 'ad', False)])

    u1_1, u1h_2 = u1.data
    u2_1, up_2 = u2.data
    u3_1, u3_2 = u3.data

    w1_1, w1_2 = 0,1/u1_1
    w1 = Tensor([w1_1, w1_2], [Index(sp, 'a', False)])
    w2_1, w2_2 = 0,1/u2_1
    w2 = Tensor([w2_1, w2_2], [Index(sp, 'a', False)])
    w3_1, w3_2 = 0,1/u3_1
    w3pre = Tensor([w3_1, w3_2], [Index(sp, 'a', False)])
    #print(f'{w1=}, {w2=}, {w3pre=}')
    #print(f'{la1=}, {la2=}, {la3=}')
    momsum = (w1*la1 +w2*la2 + w3pre*la3  ).contract('a')
    factor = (momsum.data/(u1*la1).contract('a').data)[0]  # u1*la1 = u2*la2 = u3*la3
    w3 = w3pre - factor*u3

    ut1_1, ut1h_2 = ut1.data
    ut2_1, utp_2 = ut2.data
    ut3_1, ut3_2 = ut3.data

    wt1_1, wt1_2 = 0,1/ut1_1
    wt1 = Tensor([wt1_1, wt1_2], [Index(spd, 'ad', False)])
    wt2_1, wt2_2 = 0,1/ut2_1
    wt2 = Tensor([wt2_1, wt2_2], [Index(spd, 'ad', False)])
    wt3_1, wt3_2 = 0,1/ut3_1
    wt3pre = Tensor([wt3_1, wt3_2], [Index(spd, 'ad', False)])
    momsum = (wt1*lat1.raiseIndex('ad') + wt2*lat2.raiseIndex('ad') + wt3pre*lat3.raiseIndex('ad')  ).contract('ad')

    factor = (momsum.data/(ut1*lat1.raiseIndex('ad')).contract('ad').data)[0]  # u1*la1 = u2*la2 = u3*la3
    wt3 = wt3pre - factor*ut3
```

```
    return u1, u2, u3, ut1, ut2, ut3, w1, w2,  w3, wt1, wt2, wt3
```

In [3]:
```
def gammagammabar(la1, la2, la3, lat1, lat2, lat3, a_label='a', b_label='b', c_label='c', ad_label='ad', bd_label='bd', cd_lab

    u1, u2, u3, ut1, ut2, ut3, w1, w2, w3, wt1, wt2, wt3 = getuutilde(la1, la2, la3, lat1, lat2, lat3)
    u1 = u1.relabel(u1.indices[0].label, a_label)
    u2 = u2.relabel(u2.indices[0].label, b_label)
    u3 = u3.relabel(u3.indices[0].label, c_label)
    ut1 = ut1.relabel(ut1.indices[0].label, ad_label)
    ut2 = ut2.relabel(ut2.indices[0].label, bd_label)
    ut3 = ut3.relabel(ut3.indices[0].label, cd_label)

    w1 = w1.relabel(w1.indices[0].label, a_label)
    w2 = w2.relabel(w2.indices[0].label, b_label)
    w3 = w3.relabel(w3.indices[0].label, c_label)
    wt1 = wt1.relabel(wt1.indices[0].label, ad_label)
    wt2 = wt2.relabel(wt2.indices[0].label, bd_label)
    wt3 = wt3.relabel(wt3.indices[0].label, cd_label)

    return u1*u2*w3 + u1*w2*u3 + w1*u2*u3, ut1*ut2*wt3 + ut1*wt2*ut3 + wt1*ut2*ut3
```

4 point reconstruction

In [40]:
```
momfile = 'sam.npz'
#momfile = 'rr46.npz'
#momfile = 'rr6.npz'

if momfile == 'rr46.npz':
    rr6 = np.load('rr46.npz')
    p1= Tensor( rr6['p1'], [Index(Lorenz6, 'mu', False)])
    p2= Tensor( rr6['p2'], [Index(Lorenz6, 'mu', False)])
    p3= Tensor( rr6['p3'], [Index(Lorenz6, 'mu', False)])
    p4= Tensor( rr6['p4'], [Index(Lorenz6, 'mu', False)])

    la1 = 1j*la6(-p1, 'A', 'a')
    la2 = 1j*la6(-p2, 'B', 'b')
    la3 = la6(p3, 'C', 'c')
    la4 = la6(p4, 'D', 'd')

    lat1 = 1j*lat6(-p1, 'A', 'ad')
    lat2 = 1j*lat6(-p2, 'B', 'bd')
    lat3 = lat6(p3, 'C', 'cd')
    lat4 = lat6(p4, 'D', 'dd')
elif momfile == 'sam.npz':
    rr6 = np.load('sam.npz')

    p1= Tensor( rr6['p1'], [Index(Lorenz6, 'mu', False)])
    p2= Tensor( rr6['p2'], [Index(Lorenz6, 'mu', False)])
    p3= Tensor( rr6['p3'], [Index(Lorenz6, 'mu', False)])
    p4= Tensor( rr6['p4'], [Index(Lorenz6, 'mu', False)])

    la1 = la6(p1, 'A', 'a')
    la2 = la6(p2, 'B', 'b')
    la3 = 1j*la6(-p3, 'C', 'c')
    la4 = 1j*la6(-p4, 'D', 'd')

    lat1 = lat6(p1, 'A', 'ad')
    lat2 = lat6(p2, 'B', 'bd')
    lat3 = 1j*lat6(-p3, 'C', 'cd')
    lat4 = 1j*lat6(-p4, 'D', 'dd')

elif momfile == 'rr6.npz':
    rr6 = np.load('rr6.npz')

    p1= Tensor( rr6['p1'], [Index(Lorenz6, 'mu', False)])
    p2= Tensor( rr6['p2'], [Index(Lorenz6, 'mu', False)])
    p3= Tensor( rr6['p3'], [Index(Lorenz6, 'mu', False)])
    p4= Tensor( rr6['p4'], [Index(Lorenz6, 'mu', False)])

    la1 = 1j*la6(-p1, 'A', 'a')
    la2 = 1j*la6(-p2, 'B', 'b')
    la3 = la6(p3, 'C', 'c')
    la4 = la6(p4, 'D', 'd')

    lat1 = 1j*lat6(-p1, 'A', 'ad')
    lat2 = 1j*lat6(-p2, 'B', 'bd')
    lat3 = lat6(p3, 'C', 'cd')
    lat4 = lat6(p4, 'D', 'dd')
```

```
Using la64D for la6
Using la64D for la6
Using la64D for la6
Using la64D for la6
Using lat64D for lat6
Using lat64D for lat6
Using lat64D for lat6
Using lat64D for lat6
```

In [41]:
```
p1+p2+p3+p4
```

Out[41]:
```
Tensor(data=[ 4.44089210e-16 -1.11022302e-16  0.00000000e+00  1.11022302e-16
  0.00000000e+00  0.00000000e+00], indices=['mu↓'])
```

In [42]:
```
def eps4T(A_label='A', B_label='B', C_label='C', D_label='D', raised=False):
    return Tensor(eps4, [Index(sp6, 'A', raised), Index(sp6, 'B', raised), Index(sp6, 'C', raised), Index(sp6, 'D', raised)])
```

In [43]:
```
f1 = c(c(c(c(eps4T(),la1,'A'),la2, 'B'), la3, 'C'), la4, 'D')
f1.data.round(10)
```

Out[43]:
```
array([[[[ 0.        +0.j        ,  0.        +0.j        ],
         [ 0.        +0.j        , -0.6713311 +6.00751294j]],

        [[ 0.        +0.j        , -1.45080825-3.50613397j],
         [ 2.12213935+0.74906491j,  0.        +0.j        ]]],


       [[[ 0.        +0.j        ,  2.12213935-0.74906491j],
         [-1.45080825+3.50613397j,  0.        +0.j        ]],

        [[-0.6713311 -6.00751294j,  0.        +0.j        ],
         [ 0.        +0.j        ,  0.        +0.j        ]]]])
```

In [44]:
```
f2 = c(c(c(c(eps4T(raised=True),lat1,'A'),lat2, 'B'), lat3, 'C'), lat4, 'D')
f2.data.round(10)
```

Out[44]:
```
array([[[[ 0.        +0.j        ,  0.        +0.j        ],
         [ 0.        +0.j        , -0.6713311 +6.00751294j]],

        [[ 0.        +0.j        , -1.45080825-3.50613397j],
         [ 2.12213935+0.74906491j,  0.        +0.j        ]]],


       [[[ 0.        +0.j        ,  2.12213935-0.74906491j],
         [-1.45080825+3.50613397j,  0.        +0.j        ]],

        [[-0.6713311 -6.00751294j,  0.        +0.j        ],
         [ 0.        +0.j        ,  0.        +0.j        ]]]])
```

In [45]:
```
t= -2*MP(p1,p3)
s = 2*MP(p1,p2)
prod_target = -1j/(s*t)* (f1*f2).lowerIndex('a').lowerIndex('b').lowerIndex('c').lowerIndex('d')

prod_target.data.round(10)
```

Out[45]:
```
array([[[[[[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]],


            [[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]],



           [[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]],


            [[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]]],



          [[[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]],


            [[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]],



           [[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -1.59309338j]],

             [[ 0.        -0.j        ,  0.48260415+0.87583859j],
              [-0.5777394 -0.1340781j ,  0.        -0.j        ]]],


            [[[ 0.        -0.j        , -0.53389156+0.25830129j],
              [ 0.27736638-0.96076422j,  0.        -0.j        ]],

             [[ 0.35166043+1.55379582j,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]]]],




         [[[[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]],


            [[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]],

             [[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]],



           [[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        , -0.48260415+0.87583859j]],

             [[ 0.        -0.j        ,  0.        -0.6277096j ],
              [ 0.27700813+0.24873006j,  0.        -0.j        ]]],


            [[[ 0.        -0.j        ,  0.37176735+0.01972769j],
              [-0.44353771+0.44417749j,  0.        -0.j        ]],

             [[ 0.27736638-0.96076422j,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]],



          [[[[[ 0.        -0.j        ,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.5777394 -0.1340781j ]],

             [[ 0.        -0.j        , -0.27700813+0.24873006j],
              [ 0.        -0.22080297j,  0.        -0.j        ]]],


            [[[ 0.        -0.j        , -0.13860707-0.17187796j],
              [ 0.37176735+0.01972769j,  0.        -0.j        ]],

             [[-0.53389156+0.25830129j,  0.        -0.j        ],
              [ 0.        -0.j        ,  0.        -0.j        ]]]],
```

```
   [[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]],


    [[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]]]]],




[[[[[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]],


    [[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]],


   [[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.53389156+0.25830129j]],

     [[ 0.        -0.j        , -0.37176735+0.01972769j],
      [ 0.13860707-0.17187796j,  0.        -0.j        ]]],


    [[[ 0.        -0.j        ,  0.        -0.22080297j],
      [ 0.27700813+0.24873006j,  0.        -0.j        ]],

     [[-0.5777394 -0.1340781j ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]]],



  [[[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        , -0.27736638-0.96076422j]],

     [[ 0.        -0.j        ,  0.44353771+0.44417749j],
      [-0.37176735+0.01972769j,  0.        -0.j        ]]],


    [[[ 0.        -0.j        , -0.27700813+0.24873006j],
      [ 0.        -0.6277096j ,  0.        -0.j        ]],

     [[ 0.48260415+0.87583859j,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]],


   [[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]],


    [[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]]]],




 [[[[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        , -0.35166043+1.55379582j]],

     [[ 0.        -0.j        , -0.27736638-0.96076422j],
      [ 0.53389156+0.25830129j,  0.        -0.j        ]]],


    [[[ 0.        -0.j        ,  0.5777394 -0.1340781j ],
      [-0.48260415+0.87583859j,  0.        -0.j        ]],

     [[ 0.        -1.59309338j,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]],


   [[[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]],


    [[[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]],

     [[ 0.        -0.j        ,  0.        -0.j        ],
      [ 0.        -0.j        ,  0.        -0.j        ]]]]],
```

```
[[[[[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]],

   [[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]]],


  [[[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]],

   [[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]]]],



 [[[[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]],

   [[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]]],


  [[[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]],

   [[ 0.        -0.j        ,  0.        -0.j        ],
    [ 0.        -0.j        ,  0.        -0.j        ]]]]])
```

In [46]:
```
def polvecFromLa(la_in, laq_in, lat_in, latq_in, mu='mu', a_label='a', ad_label='ad'):
    la = la_in.relabel(la_in.indices[0].label, 'Aintern').relabel(la_in.indices[1].label, a_label)
    laq = laq_in.relabel(laq_in.indices[0].label, 'Bintern').relabel(laq_in.indices[1].label, 'bintern')
    tmp = c(la.lowerIndex(a_label), sigma6(mu,'Aintern', 'Bintern'), 'Aintern')
    #print(f'{tmp=}')
    tmp = c(tmp, laq.lowerIndex('bintern'),'Bintern')
    #print(tmp)
    tmp = c(tmp, tinv(sprodFromLaLat(laq, lat_in, 'bintern', ad_label)).raiseIndex('bintern').lowerIndex(ad_label),'bintern')
    #print(tmp)
    tmp =  1/np.sqrt(2)*tmp
    return tmp.reorder([Index(Lorenz6, mu, raised=True), Index(sp, a_label, raised=False), Index(spd, ad_label, raised=False)]
```

In [47]:
```
FUDGE = -1
def shiftedFromLa(la1,la2,lat1,lat2, z, x, xt):

    y = c(xt, tinv(sprodFromLaLat(la2, lat1, 'b', 'ad')).lowerIndex('ad'), 'ad')
    yt = c(x, tinv(sprodFromLaLat(la1, lat2, 'a', 'bd')).lowerIndex('a'), 'a')

    l1 = la1.relabel(la1.indices[0].label, 'A').relabel(la1.indices[1].label, 'a')
    l2 = la2.relabel(la2.indices[0].label, 'A').relabel(la2.indices[1].label, 'a')
    lt1 = lat1.relabel(lat1.indices[0].label, 'A').relabel(lat1.indices[1].label, 'ad')
    lt2 = lat2.relabel(lat2.indices[0].label, 'A').relabel(lat2.indices[1].label, 'ad')

    lx = c(x, l1.lowerIndex('a'), 'a')
    ly = c(y.relabel('b', 'a'), l2, 'a')
    ltx = c(xt.relabel('bd', 'ad'), lt1, 'ad')
    lty = c(yt.relabel('bd', 'ad'), lt2, 'ad')

    sqrt2 = np.sqrt(2)

    la1hat = l1 + FUDGE * sqrt2 * z * x*ly

    lat1hat = lt1 - sqrt2 * z * lty*xt.lowerIndex('ad')

    la2hat = l2 + sqrt2 * z * y.relabel('b','a').raiseIndex('a')*lx


    lat2hat = lt2 - sqrt2 * z * ltx.relabel('bd','ad')*yt.relabel('bd','ad').lowerIndex('ad')

    mom1 = momFromLa(la1, mu='mu')

    r = (momFromLa(la1hat, mu='mu')-mom1)
    return la1hat, lat1hat, la2hat, lat2hat, r
```

In [48]:
```
x = Tensor([1,1], [Index(sp, 'a', True)])
xt = Tensor([1,1], [Index(spd, 'ad', True)])

eps_vec = polvecFromLa(la1,la2,lat1, lat2)
#eps_vec = polvec(-p1,p2)

eps_vec.data[:,1,1].round(10)
```

Out[48]:
```
array([0.        +0.j        , 0.        +0.j        ,
       0.        +0.j        , 0.        +0.j        ,
       0.        -0.70710678j, 0.70710678+0.j        ])
```

In [49]:
```
xeps = c(c(x,eps_vec.lowerIndex('a').lowerIndex('ad'), 'a'), xt, 'ad').lowerIndex('mu')

z = - MP(p1, p4)/MP(xeps, p4)

la1hat, lat1hat, la2hat, lat2hat, r = shiftedFromLa(la1, la2, lat1, lat2, z, x, xt)

mom1hat = momFromLa(la1hat, mu='mu')
mom2hat = momFromLa(la2hat, mu='mu')

momP = -p4 - mom1hat
lap, latp = la6(momP, 'A', 'a'), lat6(momP, 'A', 'ad')
lamp, latmp = 1j*lap, 1j * latp

mom1hat, p4, momP
```

Out[49]:
```
(Tensor(data=[ 1.49861376e+00-2.22044605e-16j -9.65164405e-01+4.44089210e-16j
   3.19836109e+00-1.06437843e-15j -4.82721806e+00+8.88178420e-16j
   1.77635684e-15+5.67602388e+00j  2.80466360e-16+5.76771277e-16j], indices=['mu↓']),
 Tensor(data=[-1.29064994 -0.56992463  0.65968691  0.95172284  0.          0.        ], indices=['mu↓']),
 Tensor(data=[-2.07963818e-01+2.22044605e-16j  1.53508903e+00-4.44089210e-16j
  -3.85804801e+00+1.06437843e-15j  3.87549522e+00-8.88178420e-16j
  -1.77635684e-15-5.67602388e+00j -2.80466360e-16-5.76771277e-16j], indices=['mu↓']))
```

In [50]:
```
assert np.allclose((mom1hat + p4 + momP).data, 0)
assert(np.isclose(MP(mom1hat, mom1hat),0))
assert(np.isclose(MP(p4, p4),0))
assert(np.isclose(MP(mom1hat, momP),0))
assert(np.isclose(MP(momP, p4),0))
```

In [51]:
```
assert np.allclose((mom2hat + p3 - momP).data, 0)
assert(np.isclose(MP(mom2hat, mom2hat),0))
assert(np.isclose(MP(p3, p3),0))
assert(np.isclose(MP(-momP, -momP),0))
assert(np.isclose(MP(mom2hat, -momP),0))
assert(np.isclose(MP(mom2hat, p3),0))
```

In [52]:
```
np.linalg.det(sprodFromLaLat(la1hat, latp, 'a', 'ad').data).round(10)
```

Out[52]:
```
np.complex128(-0j)
```

In [53]:
```
gL, gtL = gammagammabar(la1hat, lap, la4.relabel('d', 'a').relabel('D','A'), lat1hat, latp, lat4.relabel('D','A').relabel('dd'
gR, gtR = gammagammabar(la2hat, la3.relabel('c', 'a').relabel('C','A'), lamp, lat2hat, lat3.relabel('C','A').relabel('cd','ad'
prod = -1j/complex(2*MP(p1,p3))*c(gL,gR.raiseIndex('e'),'e')*c(gtL,gtR.raiseIndex('ed'),'ed')
reordered = prod.reorder(prod_target.indices)
contracted = c(c(reordered,x,'a'), xt,'ad')
```

In [54]:
```
contracted.data.round(4)
```

Out[54]:
```
array([[[[[[ 0.    +0.j    , -0.    +0.j    ],
           [ 0.    -0.j    , -0.    -0.j    ]],

          [[ 0.    +0.j    ,  0.    +0.j    ],
           [-0.    +0.j    , -0.    +0.j    ]]],


         [[[ 0.    -0.j    , -0.    -0.2208j],
           [ 0.277 +0.2487j,  0.5339+0.2583j]],

          [[-0.5777-0.1341j, -0.3718+0.0197j],
           [ 0.1386-0.1719j,  0.    +0.j    ]]]],



        [[[[-0.    +0.j    , -0.277 +0.2487j],
           [-0.    -0.6277j, -0.2774-0.9608j]],

          [[ 0.4826+0.8758j,  0.4435+0.4442j],
           [-0.3718+0.0197j,  0.    -0.j    ]]],


         [[[-0.    +0.j    , -0.5339+0.2583j],
           [ 0.2774-0.9608j, -0.    -1.5931j]],

          [[ 0.3517+1.5538j,  0.4826+0.8758j],
           [-0.5777-0.1341j,  0.    -0.j    ]]]]],




       [[[[[ 0.    +0.j    ,  0.5777-0.1341j],
           [-0.4826+0.8758j, -0.3517+1.5538j]],

          [[ 0.    -1.5931j, -0.2774-0.9608j],
           [ 0.5339+0.2583j, -0.    +0.j    ]]],


         [[[ 0.    +0.j    ,  0.3718+0.0197j],
           [-0.4435+0.4442j, -0.4826+0.8758j]],

          [[ 0.2774-0.9608j,  0.    -0.6277j],
           [ 0.277 +0.2487j, -0.    +0.j    ]]]],



        [[[[-0.    -0.j    , -0.1386-0.1719j],
           [ 0.3718+0.0197j,  0.5777-0.1341j]],

          [[-0.5339+0.2583j, -0.277 +0.2487j],
           [ 0.    -0.2208j,  0.    +0.j    ]]],


         [[[-0.    -0.j    ,  0.    -0.j    ],
           [-0.    +0.j    , -0.    +0.j    ]],

          [[ 0.    -0.j    , -0.    -0.j    ],
           [ 0.    +0.j    , -0.    -0.j    ]]]]]])
```

In [55]:
```
selected_target = prod_target.data[0,:,:,:,0,:,:,:] + prod_target.data[0,:,:,:,1,:,:,:] +prod_target.data[1,:,:,:,0,:,:,:]+pro
selected_target.round(4)
```

Out[55]:
```
array([[[[[[ 0.    -0.j    ,  0.    -0.j    ],
           [ 0.    -0.j    ,  0.    -0.j    ]],

          [[ 0.    -0.j    ,  0.    -0.j    ],
           [ 0.    -0.j    ,  0.    -0.j    ]]],


         [[[ 0.    -0.j    ,  0.    -0.2208j],
           [ 0.277 +0.2487j,  0.5339+0.2583j]],

          [[-0.5777-0.1341j, -0.3718+0.0197j],
           [ 0.1386-0.1719j,  0.    -0.j    ]]]],



        [[[[ 0.    -0.j    , -0.277 +0.2487j],
           [ 0.    -0.6277j, -0.2774-0.9608j]],

          [[ 0.4826+0.8758j,  0.4435+0.4442j],
           [-0.3718+0.0197j,  0.    -0.j    ]]],


         [[[ 0.    -0.j    , -0.5339+0.2583j],
           [ 0.2774-0.9608j,  0.    -1.5931j]],

          [[ 0.3517+1.5538j,  0.4826+0.8758j],
           [-0.5777-0.1341j,  0.    -0.j    ]]]]],



       [[[[[ 0.    -0.j    ,  0.5777-0.1341j],
           [-0.4826+0.8758j, -0.3517+1.5538j]],

          [[ 0.    -1.5931j, -0.2774-0.9608j],
           [ 0.5339+0.2583j,  0.    -0.j    ]]],


         [[[ 0.    -0.j    ,  0.3718+0.0197j],
           [-0.4435+0.4442j, -0.4826+0.8758j]],

          [[ 0.2774-0.9608j,  0.    -0.6277j],
           [ 0.277 +0.2487j,  0.    -0.j    ]]]],



        [[[[ 0.    -0.j    , -0.1386-0.1719j],
           [ 0.3718+0.0197j,  0.5777-0.1341j]],

          [[-0.5339+0.2583j, -0.277 +0.2487j],
           [ 0.    -0.2208j,  0.    -0.j    ]]],


         [[[ 0.    -0.j    ,  0.    -0.j    ],
           [ 0.    -0.j    ,  0.    -0.j    ]],

          [[ 0.    -0.j    ,  0.    -0.j    ],
           [ 0.    -0.j    ,  0.    -0.j    ]]]]])
```

In [56]:
```
nonzero = abs(contracted.data)>1e-8
nonzero_target = abs(selected_target)>1e-8

assert np.all(nonzero == nonzero_target)
```

In [57]:
```
(contracted.data[nonzero]/selected_target[nonzero]).round(10)
```

Out[57]:
```
array([1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j,
       1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j, 1.-0.j,
       1.-0.j, 1.-0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
       1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
       1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j])
```

In [ ]:

Appendices to Part II

Appendix G

Describing CMB fluctuations

G.1 CMB power spectrum

Treating the CMB radiation as 'noise' from the Big Bang, the amplitude of fluctuations as a function of scale is quantified by the power spectrum

$$P(k) = |a_k|^2 \tag{G.1.1}$$

where a_k are Fourier coefficients which describe the amplitude of the mode and are given by

$$a_k = \int f(x) e^{-ikx} dx \tag{G.1.2}$$

G.2 Spherical harmonics

The Fourier transform is only defined in flat space. In other spaces, the basis wave functions for the expansion are found by solving the Laplace equation

$$\nabla^2 \psi = 0 \tag{G.2.1}$$

Assuming that the CMB is defined on a sphere we wish to solve the Laplace equation in spherical co-ordinates $\psi(\theta, \phi)$, i.e.

$$\left[\frac{1}{\sin^2\phi}\frac{\partial^2}{\partial\theta^2}+\frac{1}{\sin\phi}\frac{\partial}{\partial\phi}\left(\sin\phi\frac{\partial}{\partial\phi}\right)\right]\psi=0 \tag{G.2.2}$$

which has solution

$$\psi=\left[\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}\right]^{\frac{1}{2}}P_{lm}(\cos\theta)d^{im\phi}=Y_{lm}(\theta,\phi) \tag{G.2.3}$$

for $l\geq 0$ and $m=-l,...,l$. Here, rather than the flat space wave number k we have a multipole moment, l, which determines the 'wave length'; and m, the 'shape' of the mode.

Any function defined on the sphere may be expanded into spherical harmonics:

$$T(\hat{n})=\sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}a_{lm}Y_{lm}(\hat{n}) \tag{G.2.4}$$

with expansion coefficients

$$a_{lm}=\int_{4\pi}T(\hat{n})Y^*_{lm}(\hat{n})d\Omega \tag{G.2.5}$$

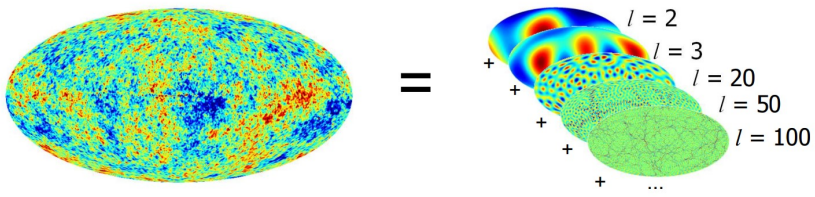


Figure G.2.1: Schematic representation of the CMB spherical harmonics transforms. *Image credit Hans Kristian Eriksen 2015*

The angular power spectrum, defined as an average over m for every l, measures amplitude as a function of wavelength. Given many independent realisations, the averaged spectrum is

$$C_l=\left\langle\frac{1}{2l+1}\sum_{m=-l}^{l}|a_{lm}|^2\right\rangle \tag{G.2.6}$$

Whilst the physics is described by C_l, in the sky we see only one realisation of the spectrum. In other words, we can only observe

$$\hat{C}_l = \frac{1}{2l+1} \sum_{m=-l}^{l} |a_{lm}|^2 \tag{G.2.7}$$

Thus there is an inescapable uncertainty in CMB measurements that is usually described as 'cosmic variance'.

Appendix H

Post-Newtonian approximation and numerical relativity

H.1 Introduction

(Appendix is based almost entirely on semi-review article by Will in 2011 [153]. Need to go back to original sources and check for recent updates.)

Post-Newtonian approach, numerical simulations using 'exact' (???!) restatement of field equations ... unreasonable effectiveness (well, yes, because even in strong-gravity regime the relative potential difference between two massive bodies is reduced as they approach each other ... AND the effectiveness is measured, initally by Will v numerical simulations which are of suspect 'exactness' and, more recently, v measured wave forms ... that have been *selected* for agreement to the theoretical predictions).

H.2 Post-Newtonian approximation

The post-Newtonian approximation is considered valid when the gravitational fields in and around the source are weak and $v \ll c$[1]. Spacetime is treated as flat and small perturbations are introduced to a known exact solution, such as the Schwarzshild or Kerr solution.

The field equations $G - \mu\nu = 8\pi(G/c^4)T_{\mu nu}$ are restated in a 'relaxed' form [153]

$$\Box h^{\alpha\beta} = -16\pi(G/c^4)\tau^{\alpha\beta} \tag{H.2.1}$$

where:

$\Box \equiv -\partial^2/\partial(ct)^2 + \bigtriangledown^2$ is the flat spacetime wave operator;

$h^{\alpha\beta} \equiv \eta^{\alpha\beta} - (-g)^{1/2}g^{\alpha\beta}$, which is small because gravity specified as weak;

g is the determinant of $g_{\alpha\beta}$;

a co-ordinate system has been specified by harmonic gauge condition $\partial h^{\alpha\beta}/\partial x^\beta = 0$; and

the source on the right hand side is an 'effective' energy-momentum *pseudo*tensor $\tau^{\alpha\beta} = (-g)T^{\alpha\beta} + (16\pi)^{-1}F^{\alpha\beta}$ and $F^{\alpha\beta}$ is the nonlinear field contribution given be terms quadratic and higher in $h^{\alpha\beta}$ and its derivatives.

The term 'relaxed' is used because equation H.2.1 can be solved as a functional of source variables without specifying the motion of the source, using an integration over the past flat-spacetime null cone C of a field point $(t, \mathbf{x})$.

$$h^{\alpha\beta}(t, \mathbf{x}) = \frac{4G}{c^4}\int_C \frac{\tau^{\alpha\beta}(t - |\mathbf{x} - \mathbf{x}'|/c, \mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|}d^3x' \tag{H.2.2}$$

The motion of the source can then be determined using a relationship derived from the harmonic gauge condition:$\partial\tau^{\alpha\beta}/\partial x^\beta = 0$.

The first step is to put the approximation $h_0^{\alpha\beta}$ into the source $\tau^{\alpha\beta}$ in equation H.2.2 and solve for $h_1^{\alpha\beta}$. This procedure can be repeated to increase accuracy.

[1] If velocities do not fulfil this slow condition the technique below is described as post-Minkowskian theory

H.3 Numerical relativity

Numerical relativity is applied to the strong-field regime in which the post-Newtonian approximation does not apply. It has only become viable since computing resources have become sufficiently powerful to carry out the considerable computations necesary to handle highly dynamical and asymmetrical situations. Nevertheless, numerical relativity is still not a full solution to the field equations of GR: it, too, is an approximation. The approach is broadly as follows [234, 235]:

- A reference frame is selected with respect to which the system will be described.
- The reference frame is time-foliated, i.e. each 4D object is decomposed into $3 + 1$ components.
- An appropriate form of equations and set of variables must be chosen for the specific system. The appropriate field equations are written in terms of the $3 + 1$ compnents.
- Inital and/or boundary conditions are defined.

I will not attempt to provide specific examples here, but suggest that the interested reader may begin with Palenzuela's brief introduction [235] and references therein.

Appendix I

The mass of a proton

The mass of the proton, $m_p = 938.27\text{MeV} \approx 2000 m_e$, sets the mass scale for all baryonic matter in the universe. The mass of the elementary electron m_e is properly attributed in the Standard Model (SM) to its interaction with the Higgs boson. As a composite particle, however, the proton owes only $\sim 1\%$ of m_p to the constituent valence quark masses supplied by the Higgs mechanism, which total just 9.8 MeV. The majority of m_p remains unaccounted for by the Higgs.

In 1994 Ji published a first QCD analysis of the mass structure of a nucleon [236]. Setting a mass scale of $\mu^2 = 1\text{GeV}^2$ he used the deep-inelastic sum rule and the energy momentum tensor of QCD to calculate that: $\sim 9\%$ of total mass is due to the quark condensate, i.e. masses of the up and down quarks and the sea of virtual pairs of strange quarks; the dynamics of the confined quarks and gluons account for $\sim 32\%$ and $\sim 37\%$ respectively; and the remaining $\sim 23\%$ is a purely quantum effect associated with the QCD mass scale, consisting of contributions from all quark flavors.

We reference Ji's early result because it nicely illustrates that there are contributions to mass from dynamics, i.e. potential and kinetic energy. In other words, the majority of m_p arises from the energy needed to hold quarks together in nucleons.

There have been QCD-connected ab initio computations of the hadron spectrum using non-perturbative methods which have produced a spectrum of ground-state hadrons that is in good agreement with experiment, the first being [237]. There

is hence no suggestion there is anything 'missing' from the SM QCD formalism. However the calculations rely on a Λ_{QCD} scale being specified and are unable to predict m_p.

To explore this further it is valuable to go on to review a recent perspective on emergent hadronic mass, for which we follow Roberts [238] and begin with the QCD Lagrangian for the strong interaction sector of the SM:

$$\begin{aligned} \mathcal{L}_{QCD} &= \sum_{j=u,d,s} \bar{q}_j[\gamma_\mu D_\mu + m_j]q_j + \frac{1}{4}G^a\mu\nu G^{a\mu\nu}. \\ D_\mu &= \partial_\mu + ig\frac{1}{2}\lambda^a A^a_{/mu}, \qquad G^a_{\mu\nu} = \partial_\mu A^a_\nu + \partial_\nu A^a_\mu - gf^{abc}A^b_\mu A^c_\nu \end{aligned} \tag{I.0.1}$$

where $\{q_i\}$ are the quark fields with flavour j and Higgs-generated masses m_j; $\{A^a_\mu, a = 1, ..., 8\}$ are the gluon fields; and the generators of the SU(3) colour gauge group in the fundamental representation are $\{\frac{1}{2}\lambda^a\}$. The Higgs mechanism provides the only obvious energy scale in equation I.0.1, that for the quark masses m_j, but we have already seen that this explains just 1% of m_p.

From a classical perspective, QCD is a local non-Abelian gauge field theory defined in four spacetime dimensions. If the Lagrangian masses for the quarks are omitted, there is no mass scale in $\mathcal{L}_{\mathrm{QCD}}$ - this state defines the chiral limit. When the theory is quantised, a mass scale emerges through the regularisation and renormalisation of ultraviolet divergences and all quantities in $\mathcal{L}_{\mathrm{QCD}}$, including field operators, become dependent on the mass scale. This results in the appearance of a chiral-limit trace anomaly in the QCD energy momentum tensor. Whereas in QED, photon vacuum polarisation has no infrared mass scale and so the trace anomaly is negligible, in QCD the trace anomaly expresses a mass scale that is empirically very significant.

This difference in impact is unsurprising, given that the defining distinction between QCD and QED lies in the existence in equation I.0.1 $\mathcal{L}_{QCD}$ of the gluon *self-interaction* term $gf^{abc}A^b_\mu A^c_\nu$. Roberts comments that, as a result of this self-interaction, 'gluons, although behaving as massless entities in the perturbative domain, actually possess a running mass, with $m_0 \approx m_p/2$ characterising its value at infrared momenta' [238].

The mechanisms behind emerging hadronic mass and hence m_p are summarised

by Roberts [238] as:

- The generation of a running mass for the gluon driven by gluon self-interactions. Once Λ_{QCD} is fixed, the scale of this effect is known. Roberts notes, however, that Λ_{QCD} cannot be predicted from within the SM.
- Dynamical chiral symmetry breaking (DCSB), the effect by which quarks which are massless in the absence of Higgs mechanism also acquire a running mass as a result of gluon mass generation. At infrared momenta, the scale of DCSB is $\sim \frac{1}{3} m_p$.

For our purposes, the main point is that the empirical mass that clearly and unequivocally emerges from dynamics and interaction is two orders of magnitude greater than the Higgs-associated mass of the components.

Appendix J

Fractal dimension

The main tool in fractal analysis is the *fractal dimension*. This brief introduction to the mathematical background to fractal dimension is based largely on Falconer [239].

J.1 Measures and mass distributions

Measure theory is a key part of the mathematics of fractal analysis. Whilst the group theory language of measure theory appears technical, a 'measure' is really just a way of attributing a numerical quantity to sets in such a way that, if one set is decomposed into parts, then the sum of the parts is equal to the numerical value of the whole [239].

In the set of real numbers of n dimensions, $\mathbb{R}^n$, μ is a *measure* on $\mathbb{R}^n$ if it assigns a non-negative number (which might be ∞) to each subset of $\mathbb{R}^n$ such that:

1. $\mu(\emptyset) = 0$, i.e. the null set has zero measure.
2. If $A \subset B$, $\mu(A) \leqslant \mu(B)$: i.e. the larger the set, the larger the measure.
3. For a finite sequence of sets $A_1, A_2, ...$ then $\mu(\cup_{i=1}^{\infty} A_i) \leqslant \sum_{i=1}^{\infty} \mu(A_i)$: if a set is a union of a non-infinite number of parts then the sum of the measure of those parts is at least equal to the measure of the whole.
4. Where the A_i are disjoint Borel sets, $\mu(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$: i.e. if the decomposition of the set is into a finite number of dijoint Borel sets then

the total measure of the parts is equal to the measure of the whole. (The class of Borel sets is defined as the smallest collection of subsets of $\mathbb{R}^n$ with the properties that (i) every open ad every closed set is a Borel set; (ii) the union of every finite collection of Borel sets is a Borel set; and (iii) so, too, is the intersection of every finite collection.)

It is therefore straightforward to think of $\mu(A)$ as the size of A measured in some way. We can treat $\mu(A)$ as the mass of the set A and then a *mass distribution* is a measure on a bounded subset of $\mathbb{R}^n$ for which $0 < \mu(\mathbb{R}^n) < \infty$.

Using a point mass as a simple example, a point a in $\mathbb{R}^n$ may be described as a mass distribution μ where $\mu(A)$ is defined as 1 if A contains a and 0 if it does not.

The Hausdorff measure generalises length, area, volume and so on to any number of dimensions. The *s-dimensional Hausdorff measure*, $\mathscr{H}^s(F)$, of a subset F of $\mathbb{R}^n$ is established as follows[1]:

- Let U be any non-empty subset of $\mathbb{R}^n$.
- The *diameter* of U, being the greatest distance between any pair of points within it, is defined as $|U| = \sup\{|x - y| : x, y \in U\}$.
- $\{U_i\}$ is called a δ-*cover* of F if it is a finite collection of sets of diameter no greater than δ that cover F, so $F \subset \cup_{i=1}^{\infty} U_i$.
- Define $\mathscr{H}^s_\delta(F) = \inf\{\sum_{i=1}^{\infty} |U_i|^s : \{U_i\} \text{ is a } \delta\text{-cover of } F\}$, for any $\delta > 0$.

The s-dimensional Hausdorff measure of F is then given by

$$\mathscr{H}^s(F) = \lim_{\delta \to 0} \mathscr{H}^s_\delta(F) \tag{J.1.1}$$

J.2 Definitions of fractal dimension

Fractals are usually characterised by their dimension. The dimension does not uniquely define the form of the fractal pattern but it is a measure that clearly differentiates fractal from classical homogeneity. Simply, the mass-radius relation is used which, in homogeneous cases, is

[1]The abbreviations 'sup' and 'inf' are for supremum (the smallest upper bound of a set or subset), and infimum (the largest quantity that is less than or equal to each of a given set or subset of quantities) respectively.

$$M(R) = \delta F R^D \tag{J.2.1}$$

where:

- $M(R)$ is the mass within a sphere of radius R whose centre, importantly, is part of the distribution.
- D is the dimension. In classical homogeneity $D = 3$ but in fractal homogeneity $0 < D < 3$.
- In classical homogeneity F is $4\pi/3$ but in the fractal case is a random variable rather than a numerical factor.
- δ is a density. In the fractal case, δ is separated from F by normalising $\langle F \rangle = 1$.

There is more than one definition of fractal dimension, but all utilise the concept of measures at a scale, δ. For each set, irregularities of a scale less than δ are ignored, and the behaviour of the measurements as $\delta \to 0$ are explored.

The oldest definition [239], constructed long before Mandelbrot coined the term 'fractal', is the *Hausdorff dimension*, based on defining measures using covers of sets. Its principal advantages are that it is mathematically defined for any set, and the measures on which it is based are easy to manipulate. However, it can be difficult to calculate and difficult even to estimate numerically in real data sets. We introduce it here because awareness of the Hausdorff dimension is onsidered a good foundation for understanding the nature of fractal analysis.

The s-dimensional Hausdorff measure has been defined in equation J.1.1. The nature of the function is that, if $\mathscr{H}^s(F)$ is plotted against values of s, there is a critical value of s at which $\mathscr{H}^s(F)$ abruptly jumps from ∞ to 0. This value of s is the Hausdorff dimension of F, written $\dim_H F$. Specifically:

$$\dim_H F = \inf\{s : \mathscr{H}^s(F) = 0\} = \sup\{s : \mathscr{H}^s(F) = \infty\} \tag{J.2.2}$$

The Hausdorff dimension has the following principal properties, which are expected to hold for any other 'reasonable' definition of dimension [239]:

- If $F \subset \mathbb{R}^n$ is an open set, F has a positive n-dimensional volume and so $\dim_H F = n$.

- If $F \subset \mathbb{R}^n$ is a continuosly differentiable m-dimensional submanifold of $\mathbb{R}^n$ then $\dim_H F = n$.
- If $E \subset F$ then $\dim_H E \leqslant \dim_H F$.

A simple definition that lends itself well to computational methods is a *power law* approach, where the dimension of F is determined by the power law obeyed by a measurement $M_\delta(F)$ as $\delta \to 0$. For constants dimension s and s-dimensional 'length' c :

$$\begin{aligned} M_\delta(F) &\sim c\delta^{-s} \\ \log M_\delta(F) &\simeq \log c - s\ log\delta \\ s &= \lim_{s\to 0} \frac{\log M_\delta(F)}{-\log \delta} \end{aligned} \tag{J.2.3}$$

Then s can be estimated as the gradient of a log-log graph plotted over a range of δ. If there is no exact power law, lower and uppler limites may be set.

An alternative definition is *box-counting dimension*[2]. The box-counting dimension of F is defined as:

$$\dim_B F = \lim_{\delta \to\ 0} \frac{\log N_\delta(F)}{-\log \delta} \tag{J.2.4}$$

where F is a non-empty bounded subset of $\mathbb{R}^n$ and $N_\delta(F)$ is the smallest number of sets of diameter at most δ which can cover F.

[2]Also known as Kolmogorov entropy, entropy dimension or capacity dimension

Appendix K

Five-body simulation: additional material

K.1 Numerical approximation

Numerical solutions to differential equations are approximations and we need to consider their reliability if we wish to draw inferences about physical phenomena.

K.1.1 Selection of time interval for the simulation

A numerical approach to solving the differential equation 13.2.11 requires an appropriate time-step for the calculation.

We first show that our model converges to a solution of equation 13.2.11 as the calculation time interval, $\Delta T = t$, is reduced. Figure K.1.1 shows that solutions for the position evolution of the system converge as $t \to 0$s, and Figure K.1.2 similarly demonstrates convergence for the mass-energy of the particles.

Clearly the extended time step $t = 100s$ is a poor approximation, as we would expect. While $t = 0.001s$ is the best of the intervals we have presented it is computationally intensive so, for the remaining analysis, we use $t = 0.1s$ as a reasonable degree of approximation.

K.1.2 Numerical stability

Errors can arise in numerical solutions as a result of the methods used in their design and coding. A textbook example of inherent instability in a numerical simulation is the Euler solution to the problem of simple harmonic motion, in which the instability arises because the algorithm fails to conserve energy. An effective correction is to apply the Euler-Cromer method [240], a minor change to the code. In this example the existence of an error in the Euler solution is very easy to spot because an analytic solution to the problem exists, against which the numerical outcome can be compared.

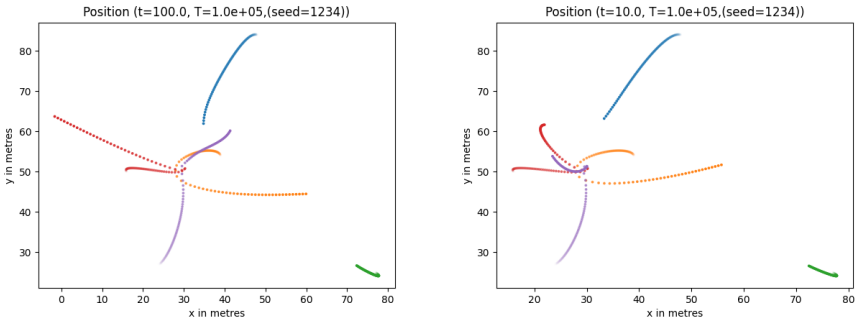
In another example, although there is no analytic solution to the Newtonian three-body problem there is a theoretical expectation: three bodies of equal mass which are arranged initially in an exactly equidistant configuration should persist thereafter in stable orbits. Many simulations of this scenario exist, and it is well known that the numerical quality of the simulation can be improved by various computational methods, for example the Kahan-Babuška-Klein algorithm, which addresses the problem of cumulative loss of accuracy in sums of floating point numbers.

Despite best efforts at precision, simulations of the three-body Newtonian case, after a sufficiently large number of iterations, do become unstable and the simulated system displays chaotic behaviour. From a computing perspective, this is seen as an error and it is desireable to work towards eradicating it. However, there is an underlying physical message in the behaviour of the simulation: a perturbation from (highly non-physical) perfect symmetry in this Newtonian system, however tiny it may be, will result in irregular and unpredictable (although fully-determined) behaviour. The apparent stability of the three-body system in its early phases is not a physical equilibrium. Any three-body system of this kind is inherently, physically, unstable. It is a complex system.

Which leaves us, still, with the question of how to assess the reliability of the numerical simulation of our toy, five-body system. We have no analytic solution and, explicitly, no theoretical expectation other than that the system will demonstrate complex behaviour. In GR theory there is not even energy conservation to help us.

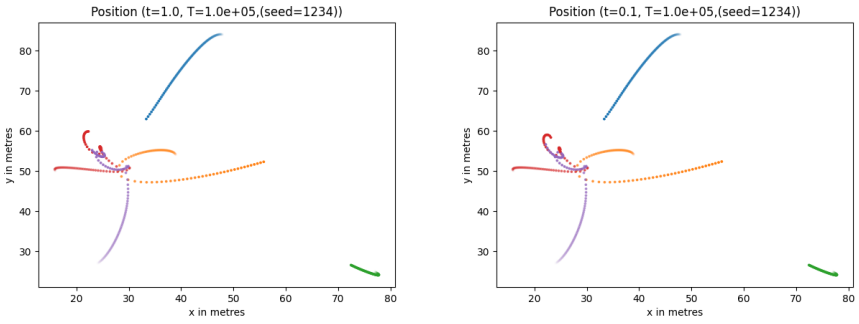
We can check that our simulation converges to a solution as the time interval becomes small, and we have demonstrated in Section K.1.1 that it does. In relation to our equations and the simulation code, we have considered the following:

- We code in Python. The simulation calculations are simple but include multiplication, division, squaring, trigonometric functions and summations, all of floating point numbers. Floating point numbers are approximations. It is no doubt possible to improve the precision of the calculations through judicious use of carefully designed algorithms but, as in the three-body Newtonian case, any perturbations caused by tiny inaccuracies in the floating point numbers in a sense make the system more physically realistic.
- The dynamics and mass variances of the simulation reported in Section **??** and, for alternative random initial conditions in Appendix **??**, tell a consistent story. Irrespective of the numerical values of particular plots, the profile of the outcomes of the various simulations indicate that there is a case to answer. The complex, five-body toy - whether using Newtonian dynamics or a GR model - is not the same as a simple system and the two approaches are not the same as each other.



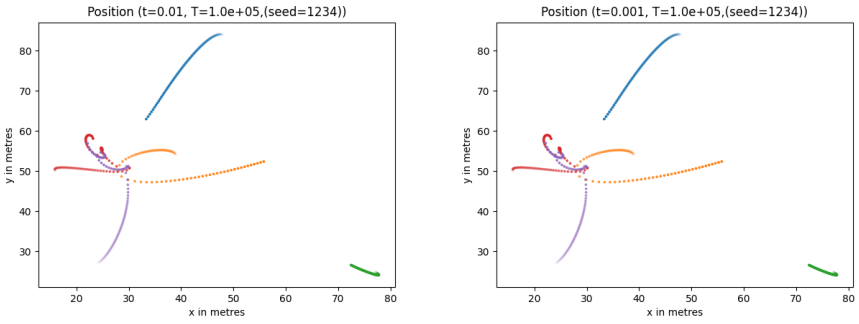
(a) t = 100s

(b) t = 10s



(c) t = 1s

(d) t = 0.1s

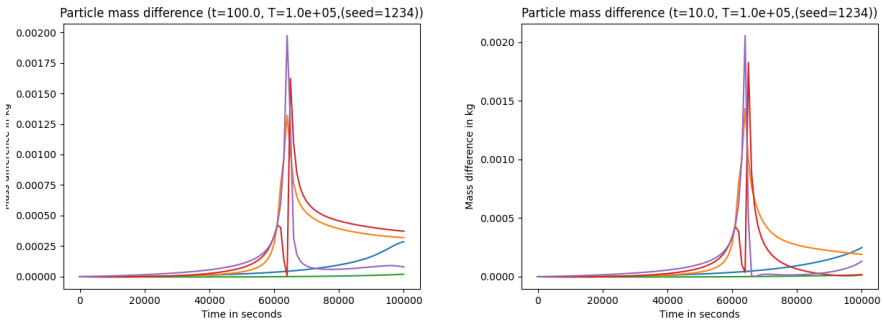


(e) t = 0.01s

(f) t = 0.001s

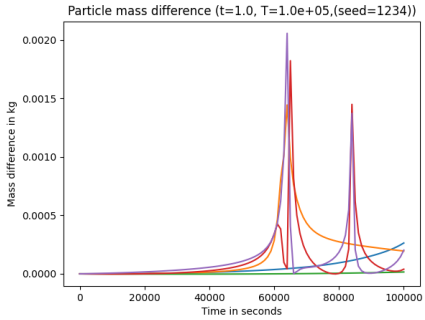
Figure K.1.1: Evolution of position of 5 bodies with initial random configuration in Figure 13.4.1 over elapsed time $T = 10^5 s$: convergence towards the solution of the differential equations as time interval $t = \Delta T \to 0s$

.

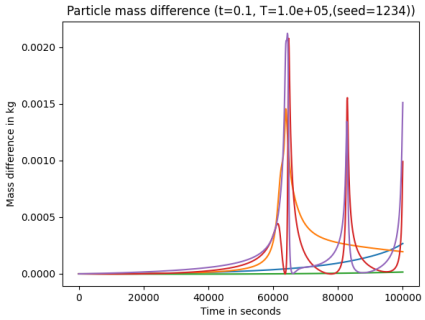


(a) t = 100s

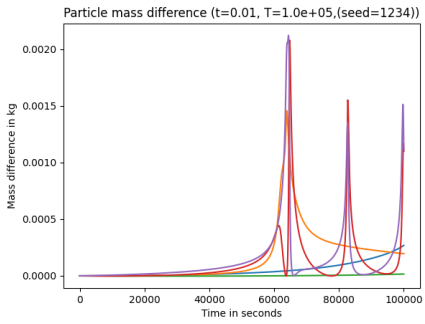
(b) t = 10s



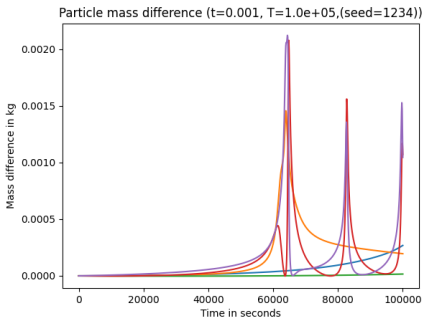
(c) t = 1s



(d) t = 0.1s



(e) t = 0.01s



(f) t = 0.001s

Figure K.1.2: Evolution of particle mass difference for 5 bodies with initial equal intrinsic mass and very small random velocities over elapsed time $T = 10^5 s$: convergence towards the solution of the differential equations as time interval $t = \Delta T \to 0s$

.

K.2 Additional results

In Section 13.4.3 we have presented results for a single set of random initial conditions. In this Appendix we present plots obtained from diffent random start points.

K.2.1 Position evolution: GR approximation v Newton

The following figures show the difference between position evolution in the GR approximation and the Newtonian calculation for five different random initial configurations.

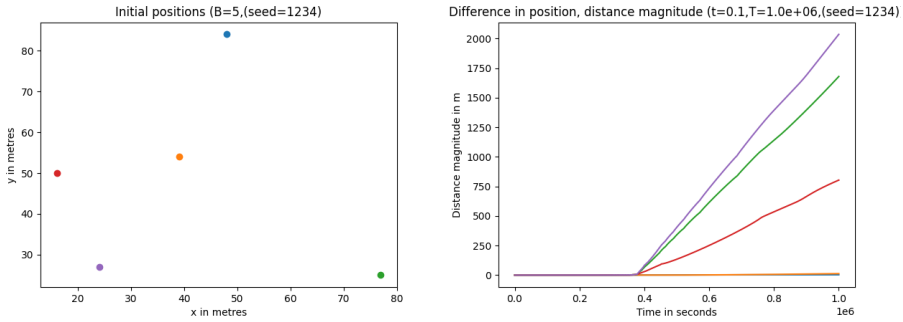


Figure K.2.1: Position comparison GR approx v N, initial random 1234

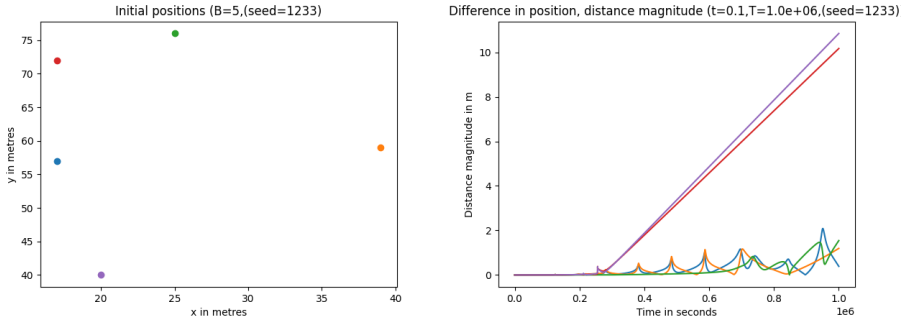


Figure K.2.2: Position comparison GR approx v N, initial random 1233

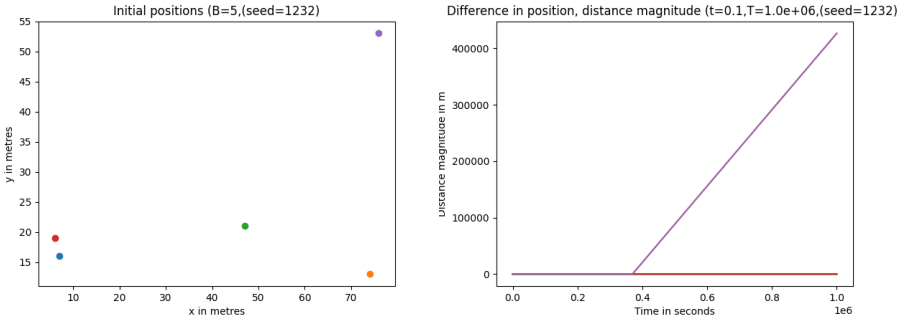


Figure K.2.3: Position comparison GR approx v N, initial random 1232

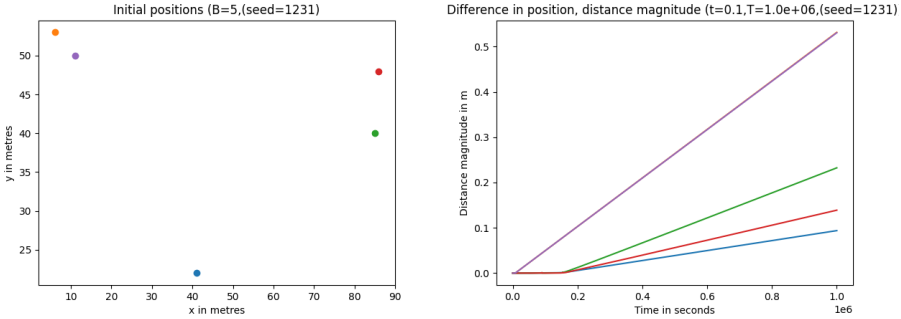


Figure K.2.4: Position comparison GR approx v N, initial random 1231

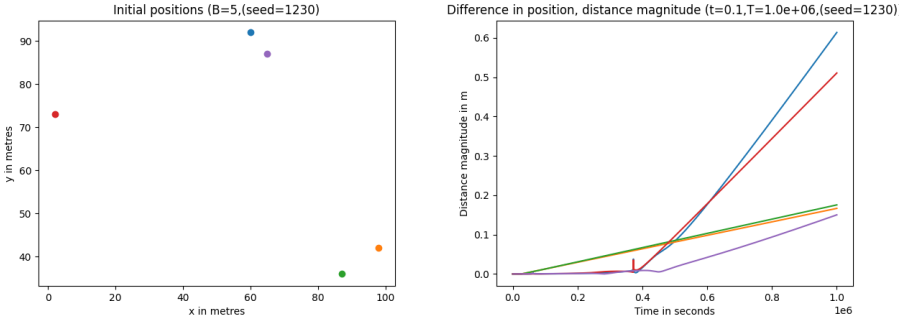
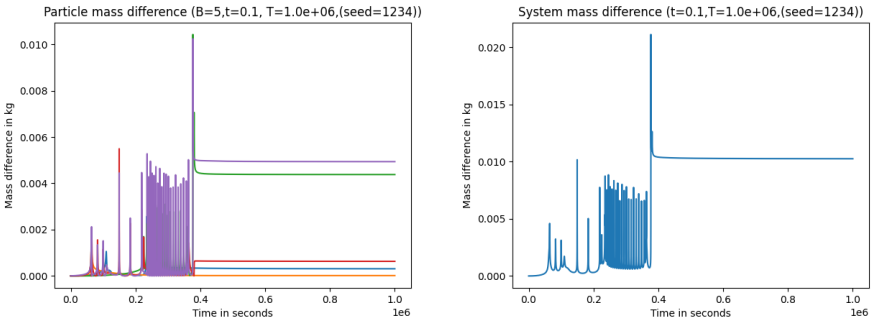


Figure K.2.5: Position comparison GR approx v N, initial random 1230

K.2.2 Mass-energy evolution: GR approximation

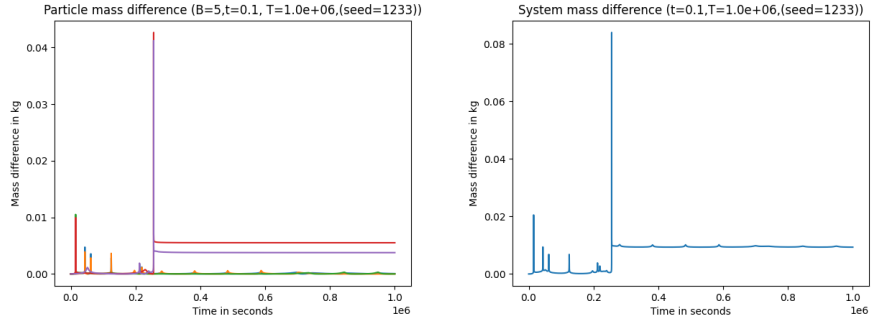
Particle and system mass-energy evolution for the five different random initial configurations in the GR approximation, displayed as variance from initial mass. Note that in each case the equivalent Newtonian variance is zero, so these plots also show the difference between the GR approximation and the Newtonian calculation.



(a) Particle mass difference (b) System mass difference

Figure K.2.6: Mass-energy difference over time for random seed 1234 initial state. Elapsed time $T = 10^6 s$, time interval $t = \Delta T = 0.1 s$.)

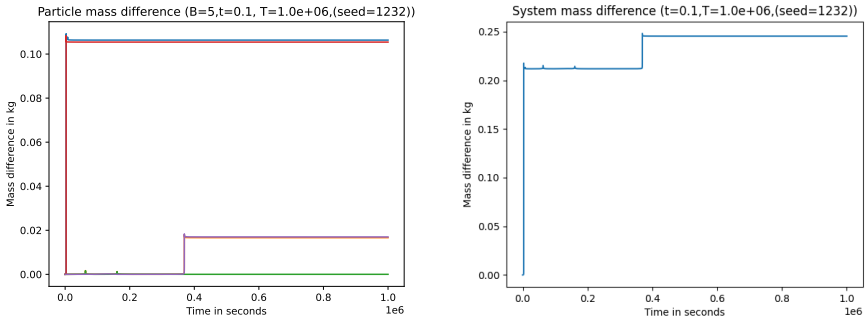
.



(a) Particle mass difference (b) System mass difference

Figure K.2.7: Mass-energy difference over time for random seed 1233 initial state. Elapsed time $T = 10^6 s$, time interval $t = \Delta T = 0.1 s$.)

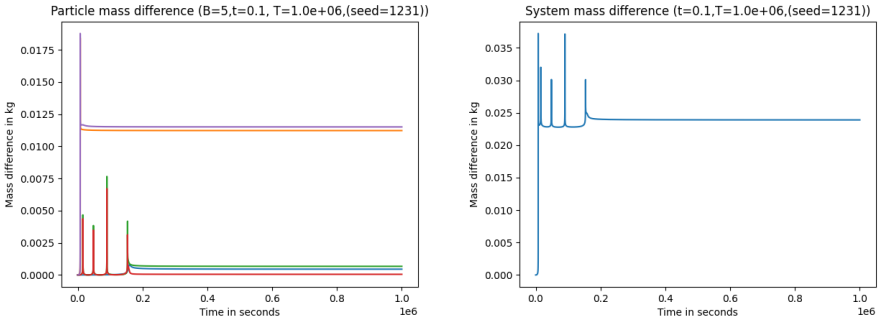
.



(a) Particle mass difference (b) System mass difference

Figure K.2.8: Mass-energy difference over time for random seed 1232 initial state. Elapsed time $T = 10^6 s$, time interval $t = \Delta T = 0.1 s$.)

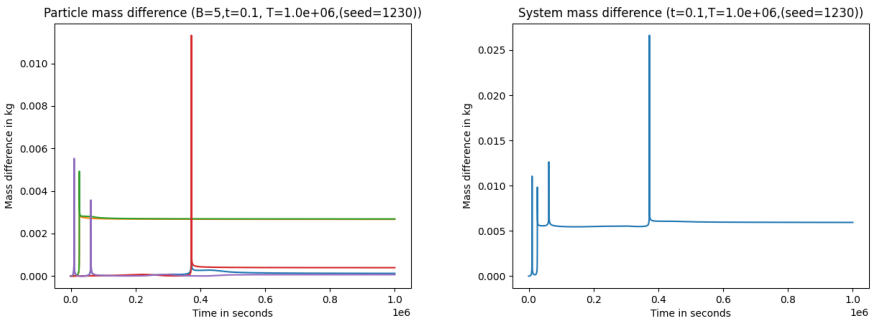
.



(a) Particle mass difference (b) System mass difference

Figure K.2.9: Mass-energy difference over time for random seed 1231 initial state. Elapsed time $T = 10^6 s$, time interval $t = \Delta T = 0.1 s$.)

.



(a) Particle mass difference

(b) System mass difference

Figure K.2.10: Mass-energy difference over time for random seed 1230 initial state. Elapsed time $T = 10^6 s$, time interval $t = \Delta T = 0.1 s$.)

.

Bibliography

[1] Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality: An explanation of the 1/f noise. *Phys. Rev. Lett.*, 59:381–384, Jul 1987.

[2] ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, September 2012.

[3] CMS Collaboration. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012.

[4] Alexander Huss, Joey Huston, Stephen Jones, and Mathieu Pellen. Les houches 2021—physics at tev colliders: report on the standard model precision wishlist. *Journal of Physics G: Nuclear and Particle Physics*, 50(4):043001, March 2023.

[5] Heather E. Logan. Tasi 2013 lectures on higgs physics within and beyond the standard model, 2022.

[6] M.L. Mangano and S.J. Parke. Multiparton amplitudes in gauge theories. *Phys. Rept.*, 200:301–367, 1991.

[7] Jeppe R. Andersen, James D. Cockburn, Marian Heil, Andreas Maier, and Jennifer M. Smillie. Finite quark-mass effects in higgs boson production with dijets at large energies. *Journal of High Energy Physics*, 2019(4), April 2019.

[8] Simon Armstrong. *Next to Leading Order Calculations for Higgs Boson + Jets*. Phd thesis, University of Durham, Institute of Particle Physics Phenomenology, August 2017.

[9] Giuseppe De Laurentis. Analytical amplitudes from numerical solutions of the scattering equations. *JHEP*, 02:194, 2020.

[10] C.F. Berger, Z. Bern, L.J. Dixon, F. Febres Cordero, D. Forde, H. Ita, D.A. Kosower, and D. Maitre. An Automated Implementation of On-Shell Methods for One-Loop Amplitudes. *Phys. Rev.*, D78:036003, 2008.

[11] C. F. Berger, Z. Bern, Lance J. Dixon, F. Febres Cordero, D. Forde, H. Ita, D. A. Kosower, and D. Maitre. One-Loop Calculations with BlackHat. *Nucl. Phys. Proc. Suppl.*, 183:313–319, 2008.

[12] Lance J. Dixon. Calculating scattering amplitudes efficiently. In *QCD and beyond. Proceedings, Theoretical Advanced Study Institute in Elementary Particle Physics, TASI-95, Boulder, USA, June 4-30, 1995*, pages 539–584, 1996.

[13] L.J. Dixon. A brief introduction to modern amplitude methods. In *Proceedings, 2012 European School of High-Energy Physics (ESHEP 2012): La Pommeraye, Anjou, France, June 06-19, 2012*, pages 31–67, 2014.

[14] S.J. Parke and T.R. Taylor. An Amplitude for n Gluon Scattering. *Phys. Rev. Lett.*, 56:2459, 1986.

[15] C. Buttar et al. Les houches physics at tev colliders 2005, standard model and higgs working group: Summary report, 2006.

[16] Gudrun Heinrich. Collider physics at the precision frontier. *Physics Reports*, 922:1–69, 2021. Collider physics at the precision frontier.

[17] Z. Bern, G. Diana, L. J. Dixon, F. Febres Cordero, S. Höche, D. A. Kosower, H. Ita, D. Maître, and K. Ozeren. Four-jet production at the large hadron collider at next-to-leading order in qcd. *Physical Review Letters*, 109(4), July 2012.

[18] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, and K. J. Ozeren. Next-to-Leading Order $W + 5$-Jet Production at the LHC. *Phys. Rev.*, D88(1):014025, 2013.

[19] R. Dendy, P Helander, and Michel Tagger. Self-organised criticality in astrophysical accretion systems. *Physica Scripta*, 1999:133, 11 2006.

[20] Robert V. Harlander, Tobias Neumann, Kemal J. Ozeren, and Marius Wiesemann. Top-mass effects in differential higgs production through gluon fusion at $o\left(\alpha_s^4\right)$. *Journal of High Energy Physics*, 2012(8), August 2012.

[21] Zvi Bern and David A. Kosower. The Computation of loop amplitudes in gauge theories. *Nucl. Phys. B*, 379:451–561, 1992.

[22] Zvi Bern, Lance J. Dixon, David C. Dunbar, and David A. Kosower. One loop n point gauge theory amplitudes, unitarity and collinear limits. *Nucl. Phys.*, B425:217–260, 1994.

[23] Zvi Bern, Lance J. Dixon, David C. Dunbar, and David A. Kosower. Fusing gauge theory tree amplitudes into loop amplitudes. *Nucl. Phys.*, B435:59–101, 1995.

[24] Z. Bern and A.G. Morgan. Massive loop amplitudes from unitarity. *Nuclear Physics B*, 467(3):479–509, May 1996.

[25] B. Feng R. Britto, F. Cachazo and E. Witten. Direct proof of tree-level recursion relation in Yang-Mills theory. *Phys. Rev. Lett.*, 94:181602, 2005.

[26] F. Cachazo R. Britto and B. Feng. New recursion relations for tree amplitudes of gluons. *Nucl. Phys.*, B715:499–522, 2005.

[27] J.M. Henn and J.C. Plefka. Scattering Amplitudes in Gauge Theories. *Lect. Notes Phys.*, 883:pp.1–195, 2014.

[28] Harald Ita. Susy Theories and QCD: Numerical Approaches. *J. Phys.*, A44:454005, 2011.

[29] R. Keith Ellis, Zoltan Kunszt, Kirill Melnikov, and Giulia Zanderighi. One-loop calculations in quantum field theory: from Feynman diagrams to unitarity cuts. *Phys. Rept.*, 518:141–250, 2012.

[30] S. D. Badger. Direct Extraction Of One Loop Rational Terms. *JHEP*, 01:049, 2009.

[31] Giovanni Ossola, Costas G. Papadopoulos, and Roberto Pittau. Reducing full one-loop amplitudes to scalar integrals at the integrand level. *Nucl. Phys.*, B763:147–169, 2007.

[32] Darren Forde. Direct extraction of one-loop integral coefficients. *Phys. Rev.*, D75:125019, 2007.

[33] C. F. Berger, Z. Bern, Lance J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maitre. Precise Predictions

for W + 4 Jet Production at the Large Hadron Collider. *Phys. Rev. Lett.*, 106:092001, 2011.

[34] C. F. Berger, Z. Bern, Lance J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maitre. Vector Boson + Jets with BlackHat and Sherpa. *Nucl. Phys. Proc. Suppl.*, 205-206:92–97, 2010.

[35] C. F. Berger, Z. Bern, Lance J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maitre. Next-to-Leading Order QCD Predictions for Z, γ^* + 3-Jet Distributions at the Tevatron. *Phys. Rev.*, D82:074002, 2010.

[36] L.J. Dixon Z. Bern and D.A. Kosower. On-shell recurrence relations for one-loop QCD amplitudes. *Phys. Rev.*, D71:105013, 2005.

[37] Zvi Bern, Lance J. Dixon, and David A. Kosower. The last of the finite loop amplitudes in QCD. *Phys. Rev.*, D72:125003, 2005.

[38] Zvi Bern, Lance J. Dixon, and David A. Kosower. Bootstrapping multiparton loop amplitudes in QCD. *Phys. Rev.*, D73:065013, 2006.

[39] Darren Forde and David A. Kosower. All-multiplicity one-loop corrections to MHV amplitudes in QCD. *Phys. Rev.*, D73:061701, 2006.

[40] Carola F. Berger, Zvi Bern, Lance J. Dixon, Darren Forde, and David A. Kosower. Bootstrapping One-Loop QCD Amplitudes with General Helicities. *Phys. Rev.*, D74:036009, 2006.

[41] Carola F. Berger, Zvi Bern, Lance J. Dixon, Darren Forde, and David A. Kosower. All One-loop Maximally Helicity Violating Gluonic Amplitudes in QCD. *Phys. Rev.*, D75:016006, 2007.

[42] Carola F. Berger, Vittorio Del Duca, and Lance J. Dixon. Recursive construction of higgs + multiparton loop amplitudes: The last of the "ϕ-nite" loop amplitudes. *Phys. Rev. D*, 74:094021, Nov 2006.

[43] S. D. Badger, E. W. Nigel Glover, and Kasper Risager. One-loop phi-MHV amplitudes using the unitarity bootstrap. *JHEP*, 07:066, 2007.

[44] E. W. Nigel Glover, Pierpaolo Mastrolia, and Ciaran Williams. One-loop phi-MHV amplitudes using the unitarity bootstrap: The General helicity case. *JHEP*, 08:017, 2008.

[45] Zvi Bern, John Joseph Carrasco, Tristan Dennen, Yu-tin Huang, and Harald Ita. Generalized Unitarity and Six-Dimensional Helicity. *Phys. Rev.*, D83:085022, 2011.

[46] Walter T. Giele, Zoltan Kunszt, and Kirill Melnikov. Full one-loop amplitudes from tree amplitudes. *JHEP*, 04:049, 2008.

[47] R. Keith Ellis, Walter T. Giele, Zoltan Kunszt, and Kirill Melnikov. Masses, fermions and generalized d-dimensional unitarity. *Nuclear Physics B*, 822(1–2):270–282, November 2009.

[48] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maître. Next-to-leading order qcd predictions w + 3-jet distributions at hadron colliders. *Physical Review D*, 80(7), October 2009.

[49] Scott Davies. One-Loop QCD and Higgs to Partons Processes Using Six-Dimensional Helicity and Generalized Unitarity. *Phys. Rev.*, D84:094016, 2011.

[50] Z. Bern, A. De Freitas, L. Dixon, and H. L. Wong. Supersymmetric regularization, two-loop qcd amplitudes, and coupling shifts. *Physical Review D*, 66(8), October 2002.

[51] Gerard 't Hooft and M. J. G. Veltman. Regularization and Renormalization of Gauge Fields. *Nucl. Phys. B*, 44:189–213, 1972.

[52] John C. Collins. *Renormalization: An Introduction to Renormalization, the Renormalization Group and the Operator-Product Expansion.* Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1984.

[53] Warren Siegel. Supersymmetric dimensional regularization via dimensional reduction. *Physics Letters B*, 84(2):193–196, 1979.

[54] William B. Kilgore. Regularization schemes and higher order corrections. *Physical Review D*, 83(11), June 2011.

[55] Clifford Cheung and Donal O'Connell. Amplitudes and spinor-helicity in six dimensions. *Journal of High Energy Physics*, 2009(07):075–075, jul 2009.

[56] Simon Badger, Christian Bronnum-Hansen, Francesco Buciuni, and Donal O'Connell. A unitarity compatible approach to one-loop amplitudes with massive fermions. *Journal of high energy physics.*, 2017(6):141, June 2017.

[57] Lance J. Dixon, E. W. Nigel Glover, and Valentin V. Khoze. MHV rules for Higgs plus multi-gluon amplitudes. *JHEP*, 12:015, 2004.

[58] Simon Badger, E. W. Nigel Glover, Pierpaolo Mastrolia, and Ciaran Williams. One-loop Higgs plus four gluon amplitudes: Full analytic results. *JHEP*, 01:036, 2010.

[59] J. R. Andersen et al. Les Houches 2015: Physics at TeV Colliders Standard Model Working Group Report. In *9th Les Houches Workshop on Physics at TeV Colliders (PhysTeV 2015) Les Houches, France, June 1-19, 2015*, 2016.

[60] S. Dawson and R. P. Kauffman. Production rates for higgs boson plus multiple jets at the superconducting super collider. *Phys. Rev. Lett.*, 68:2273–2276, Apr 1992.

[61] Russel P. Kauffman, Satish V. Desai, and Dipesh Risal. Production of a Higgs boson plus two jets in hadronic collisions. *Phys. Rev.*, D55:4005–4015, 1997. [Erratum: Phys. Rev.D58,119901(1998)].

[62] V. Del Duca, W. Kilgore, C. Oleari, C. Schmidt, and D. Zeppenfeld. Higgs + 2 jets via gluon fusion. *Phys. Rev. Lett.*, 87:122001, 2001.

[63] V. Del Duca, W. Kilgore, C. Oleari, C. Schmidt, and D. Zeppenfeld. Gluon fusion contributions to H + 2 jet production. *Nucl. Phys.*, B616:367–399, 2001.

[64] Vittorio Del Duca, Alberto Frizzo, and Fabio Maltoni. Higgs boson production in association with three jets. *JHEP*, 05:064, 2004.

[65] E.W.N. Glover S.D Badger and V. Khoze. Mhv rules for higgs plus multi-parton amplitudes. *Journal of High Energy Physics*, 2005(03):023, 2005.

[66] Michelangelo L. Mangano, Mauro Moretti, Fulvio Piccinini, Roberto Pittau, and Antonio D. Polosa. ALPGEN, a generator for hard multiparton processes in hadronic collisions. *JHEP*, 07:001, 2003.

[67] Fabio Maltoni and Tim Stelzer. MadEvent: Automatic event generation with MadGraph. *JHEP*, 02:027, 2003.

[68] Carl R. Schmidt. H —> g g g (g q anti-q) at two loops in the large M(t) limit. *Phys. Lett.*, B413:391–395, 1997.

[69] G. Zanderighi. Semi-numerical evaluation of one-loop corrections. In *Proceedings, 41st Rencontres de Moriond, 2006 QCD and High Energy Hadronic Interactions: La Thuile, Val d'Aoste, Italy, Mar 18-25, 2006*, pages 93–96, 2006.

[70] R. K. Ellis, W. T. Giele, and G. Zanderighi. Virtual qcd corrections to higgs boson plus four parton processes. *Phys. Rev. D*, 72:054018, Sep 2005.

[71] Simon Badger, John M. Campbell, R. Keith Ellis, and Ciaran Williams. Analytic results for the one-loop NMHV Hqqgg amplitude. *JHEP*, 12:035, 2009.

[72] Lance J Dixon and Yorgos Sofianatos. Analytic one-loop amplitudes for a higgs boson plus four partons. *Journal of High Energy Physics*, 2009(08):058–058, Aug 2009.

[73] Lucy Budge, John M. Campbell, Giuseppe De Laurentis, R. Keith Ellis, and Satyajit Seth. The one-loop amplitudes for higgs + 4 partons with full mass effects. *Journal of High Energy Physics*, 2020(5), May 2020.

[74] G. Cullen et al. NLO QCD Production of Higgs plus jets with GoSam. *PoS*, EPS-HEP2013:446, 2013.

[75] Nicolas Greiner, Stefan Höche, Gionata Luisoni, Marek Schönherr, Jan-Christopher Winter, and Valery Yundin. Phenomenological analysis of higgs boson production through gluon fusion in association with jets. *Journal of High Energy Physics*, 2016(1):169, Jan 2016.

[76] G. Luisoni, N. Greiner, S. Höche, M. Schönherr, J. Winter, and V. Yundin. Higgs Boson production in association with jets in gluon-gluon fusion. *J. Phys. Conf. Ser.*, 762(1):012069, 2016.

[77] X. Chen, T. Gehrmann, E.W.N. Glover, and M. Jaquier. Precise qcd predictions for the production of higgs + jet final states. *Physics Letters B*, 740:147 – 150, 2015.

[78] Bernhard Mistlberger. Higgs boson production at hadron colliders at n3lo in qcd. *Journal of High Energy Physics*, 2018(5), May 2018.

[79] BlackHat Project. `http://blackhat.hepforge.org/` [Accessed: 20 May 2025].

[80] Z. Bern et al. The BlackHat Library for One-Loop Amplitudes. *J. Phys. Conf. Ser.*, 523:012051, 2014.

[81] Simon Badger, Christian Brønnum-Hansen, Francesco Buciuni, and Donal O'Connell. A unitarity compatible approach to one-loop amplitudes with massive fermions. *Journal of High Energy Physics*, 2017(6):141, Jun 2017.

[82] R. Keith Ellis and Giulia Zanderighi. Scalar one-loop integrals for QCD. *JHEP*, 02:002, 2008.

[83] R. K. Ellis and G. Zanderighi. Qcdloop: A repository for one-loop scalar integrals, 2013. Available at: http://qcdloop.fnal.gov/ [Accessed 14 May 2019].

[84] I. Prigogine and I. Stengers. *The End of Certainty*. Free Press, 1997.

[85] A. Gabrielli, F.S. Labini, M. Joyce, and L. Pietronero. *Statistical Physics for Cosmic Structures*. Lecture Notes in Physics. Springer, 2005.

[86] C.W. Misner, K.S. Thorne, K.S. Thorne, J.A. Wheeler, W.H. Freeman, and Company. *Gravitation*. Number pt. 3 in Gravitation. W. H. Freeman, 1973.

[87] R.J. Lambourne. *Relativity, Gravitation and Cosmology*. Relativity, Gravitation and Cosmology. Cambridge University Press, 2010.

[88] David L Wiltshire. What is dust?—physical foundations of the averaging problem in cosmology. *Classical and Quantum Gravity*, 28(16):164006, aug 2011.

[89] Andrea Gabrielli, Michael Joyce, and Francesco Sylos Labini. Glass-like universe: Real-space correlation properties of standard cosmological models. *Phys. Rev. D*, 65:083523, Apr 2002.

[90] S. Navas et al. Review of particle physics. *Phys. Rev. D*, 110(3):030001, 2024.

[91] Harald P Pfeiffer. Numerical simulations of compact object binaries. *Classical and Quantum Gravity*, 29(12):124004, jun 2012.

[92] Abdalla et al. Cosmology intertwined: A review of the particle physics, astrophysics, and cosmology associated with the cosmological tensions and anomalies, 2022.

[93] A.A. Penzias and R.W. Wilson. A Measurement of Excess Antenna Temperature at 4080 Mc/s. *Astrophysical Journal*, 142:419–421, July 1965.

[94] R.H. Dicke, P.J.E. Peebles, P.G. Roll, and D.T. Wilkinson. Cosmic Black-Body Radiation. *Astrophysical Journal*, 142:414–419, July 1965.

[95] G. Gamow. Expanding universe and the origin of elements. *Phys. Rev.*, 70:572–573, Oct 1946.

[96] R. A. Alpher, H. Bethe, and G. Gamow. The origin of chemical elements. *Phys. Rev.*, 73:803–804, Apr 1948.

[97] Ralph A. Alpher and Robert C. Herman. On the relative abundance of the elements. *Phys. Rev.*, 74:1737–1742, Dec 1948.

[98] Ralph A. Alpher and Robert C. Herman. Remarks on the evolution of the expanding universe. *Phys. Rev.*, 75:1089–1095, Apr 1949.

[99] Victor Alpher. Ralph a. alpher, robert c. herman, and the cosmic microwave background radiation. *Physics in Perspective*, 14, 09 2012.

[100] N. Aghanim, Y. Akrami, F. Arroja, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, and et al. Planck2018 results. *Astronomy and Astrophysics*, 641:A1, Sep 2020.

[101] R. Penrose. *Cycles of Time: An Extraordinary New View of the Universe*. Random House, 2013.

[102] Alan H. Guth. Inflationary universe: A possible solution to the horizon and flatness problems. *Phys. Rev. D*, 23:347–356, Jan 1981.

[103] Ana Achúarro et al. Inflation: Theory and observations, 2022. and Matteo Biagetti and Matteo Braglia and Giovanni Cabass and Robert Caldwell and Emanuele Castorina and Xingang Chen and William Coulton and Raphael Flauger and Jacopo Fumagalli and Mikhail M. Ivanov and Hayden Lee and

Azadeh Maleknejad and P. Daniel Meerburg and Azadeh Moradinezhad Dizgah and Gonzalo A. Palma and Guilherme L. Pimentel and Sébastien Renaux-Petel and Benjamin Wallisch and Benjamin D. Wandelt and Lukas T. Witkowski and W. L. Kimmy Wu.

[104] D. J. Fixsen. The temperature of the cosmic microwave background. *The Astrophysical Journal*, 707(2):916, nov 2009.

[105] Planck Collaboration. Planck 2018 results - v. cmb power spectra and likelihoods. *Astronomy and Astrophysics*, 641:A5, 2020.

[106] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, N. Bartolo, and et al. Planck 2018 results. *Astronomy and Astrophysics*, 641:A6, Sep 2020.

[107] P.A. Zyla et al. Review of Particle Physics. *PTEP*, 2020(8):083C01, 2020.

[108] Thomas Buchert, Martin J. France, and Frank Steiner. Model-independent analyses of non-Gaussianity in Planck CMB maps using Minkowski functionals. *Classical and Quantum Gravity*, 34(9):094002, May 2017.

[109] Nathan J. Secrest, Sebastian von Hausegger, Mohamed Rameez, Roya Mohayaee, and Subir Sarkar. A challenge to the standard cosmological model. *The Astrophysical Journal Letters*, 937(2):L31, sep 2022.

[110] Lawrence Dam, Geraint F Lewis, and Brendon J Brewer. Testing the cosmological principle with catwise quasars: a bayesian analysis of the number-count dipole. *Monthly Notices of the Royal Astronomical Society*, 525(1):231–245, 08 2023.

[111] Dominik J Schwarz, Craig J Copi, Dragan Huterer, and Glenn D Starkman. Cmb anomalies after planck. *Classical and Quantum Gravity*, 33(18):184001, aug 2016.

[112] Pranav, Pratyush and Buchert, Thomas. Homology reveals significant anisotropy in the cosmic microwave background. *Astronomy and Astrophysics*, 695:A35, 2025.

[113] P Bak and M Paczuski. Complexity, contingency, and criticality. *Proceedings of the National Academy of Sciences*, 92(15):6689–6696, 1995.

[114] Edwin Hubble. A Relation between Distance and Radial Velocity among Extra-Galactic Nebulae. *Proceedings of the National Academy of Science*, 15(3):168–173, March 1929.

[115] Adam G. Riess, Alexei V. Filippenko, Peter Challis, Alejandro Clocchiatti, Alan Diercks, Peter M. Garnavich, Ron L. Gilliland, Craig J. Hogan, Saurabh Jha, Robert P. Kirshner, B. Leibundgut, M. M. Phillips, David Reiss, Brian P. Schmidt, Robert A. Schommer, R. Chris Smith, J. Spyromilio, Christopher Stubbs, Nicholas B. Suntzeff, and John Tonry. Observational evidence from supernovae for an accelerating universe and a cosmological constant. *The Astronomical Journal*, 116(3):1009–1038, sep 1998.

[116] S. Perlmutter, G. Aldering, G. Goldhaber, R. A. Knop, P. Nugent, P. G. Castro, S. Deustua, S. Fabbro, A. Goobar, D. E. Groom, I. M. Hook, A. G. Kim, M. Y. Kim, J. C. Lee, N. J. Nunes, R. Pain, C. R. Pennypacker, R. Quimby, C. Lidman, R. S. Ellis, M. Irwin, R. G. McMahon, P. Ruiz-Lapuente, N. Walton, B. Schaefer, B. J. Boyle, A. V. Filippenko, T. Matheson, A. S. Fruchter, N. Panagia, H. J. M. Newberg, W. J. Couch, and The Supernova Cosmology Project. Measurements of omega and lambda from 42 high-redshift supernovae. *The Astrophysical Journal*, 517(2):565–586, jun 1999.

[117] Steven Weinberg. The cosmological constant problem. *Rev. Mod. Phys.*, 61:1–23, Jan 1989.

[118] Thomas Buchert. Dark energy from structure: a status report. *General Relativity and Gravitation*, 40(2-3):467–527, dec 2007.

[119] Subir Sarkar. Is the evidence for dark energy secure? *General Relativity and Gravitation*, 40(2-3):269–284, dec 2007.

[120] Richard Watkins, Trey Allen, Collin James Bradford, Jr Ramon, Albert, Alexandra Walker, Hume A Feldman, Rachel Cionitti, Yara Al-Shorman, Ehsan Kourkchi, and R Brent Tully. Analysing the large-scale bulk flow using cosmicflows4: increasing tension with the standard cosmological model. *Monthly Notices of the Royal Astronomical Society*, 524(2):1885–1892, 07 2023.

[121] Sebastian von Hausegger. The expected kinematic matter dipole is robust against source evolution, 2024.

[122] Kimberly K. Boddy, Mariangela Lisanti, Samuel D. McDermott, Nicholas L. Rodd, Christoph Weniger, Yacine Ali-Haïmoud, Malte Buschmann, Ilias Cholis, Djuna Croon, Adrienne L. Erickcek, Vera Gluscevic, Rebecca K. Leane, Siddharth Mishra-Sharma, Julian B. Muñoz, Ethan O. Nadler, Priyamvada Natarajan, Adrian Price-Whelan, Simona Vegetti, and Samuel J. Witte. Astrophysical and cosmological probes of dark matter, 2022.

[123] K.C. Freeman. On the disks of spiral and s0 galaxies. *The Astrophysical Journal*, 160:811, jun 1970.

[124] V. C. Rubin, Jr. Ford, W. K., and N. Thonnard. Extended rotation curves of high-luminosity spiral galaxies. IV. Systematic dynamical properties, Sa -> Sc. *The Astrohysical Journal*, 225:L107–L111, November 1978.

[125] F. Zwicky. Republication of: The redshift of extragalactic nebulae. *Gen Relativ Gravit*, 41(1):207–224, 2008.

[126] Jürgen Ehlers. Editorial note to: F. Zwicky The redshift of extragalactic nebulae. *Gen. Rel. Grav.*, 41(1):203–206, 2008.

[127] Houjun Mo, Frank van den Bosch, and Simon White. *Galaxy Formation and Evolution*. Cambridge University Press, 2010.

[128] G. W. Pratt, M. Arnaud, A. Biviano, D. Eckert, S. Ettori, D. Nagai, N. Okabe, and T. H. Reiprich. The galaxy cluster mass scale and its impact on cosmological constraints from the cluster population. *Space Science Reviews*, 215(2), Feb 2019.

[129] F. I. Cooperstock and S. Tieu. Galactic dynamics via general relativity: A compilation and new developments. *International Journal of Modern Physics A*, 22(13):2293–2325, may 2007.

[130] H. BALASIN and D. GRUMILLER. Non-newtonian behavior in weak field general relativity for extended rotating sources. *International Journal of Modern Physics D*, 17(03n04):475–488, March 2008.

[131] G. O. Ludwig. Galactic rotation curve and dark matter according to gravitomagnetism. *European Physical Journal C*, 81(2):186, February 2021.

[132] A N Lasenby, M P Hobson, and W E V Barker. Gravitomagnetism and galaxy rotation curves: a cautionary tale. *Classical and Quantum Gravity*, 40(21):215014, oct 2023.

[133] Douglas Clowe, Maruša Bradač, Anthony H. Gonzalez, Maxim Markevitch, Scott W. Randall, Christine Jones, and Dennis Zaritsky. A direct empirical proof of the existence of dark matter. *The Astrophysical Journal*, 648(2):L109–L113, Aug 2006.

[134] Joop Schaye, Robert A. Crain, Richard G. Bower, Michelle Furlong, Matthieu Schaller, Tom Theuns, Claudio Dalla Vecchia, Carlos S. Frenk, I. G. McCarthy, John C. Helly, and et al. The eagle project: simulating the evolution and assembly of galaxies and their environments. *Monthly Notices of the Royal Astronomical Society*, 446(1):521–554, Nov 2014.

[135] Annalisa Pillepich, Dylan Nelson, Volker Springel, Rüdiger Pakmor, Paul Torrey, Rainer Weinberger, Mark Vogelsberger, Federico Marinacci, Shy Genel, Arjen van der Wel, and Lars Hernquist. First results from the TNG50 simulation: the evolution of stellar and gaseous discs across cosmic time. *Monthly Notices of the Royal Astronomical Society*, 490(3):3196–3233, 09 2019.

[136] Mark Vogelsberger, Federico Marinacci, Paul Torrey, and Ewald Puchwein. Cosmological simulations of galaxy formation. *Nature Reviews Physics*, 2(1):42–66, January 2020.

[137] Eleonora Di Valentino, Olga Mena, Supriya Pan, Luca Visinelli, Weiqiang Yang, Alessandro Melchiorri, David F Mota, Adam G Riess, and Joseph Silk. In the realm of the hubble tension - a review of solutions. *Classical and Quantum Gravity*, 38(15):153001, jul 2021.

[138] G. F. R. Ellis. Relativistic Cosmology: Its Nature, Aims and Problems. *Fundam. Theor. Phys.*, 9:215–288, 1984.

[139] M. F. Shirokov and I. Z. Fisher. Isotropic Space with Discrete Gravitational-Field Sources. On the Theory of a Nonhomogeneous Isotropic Universe. *General Relativity and Gravitation*, 30:1411–1427, 1998.

[140] Thomas Buchert and Syksy Räsänen. Backreaction in late-time cosmology. *Ann. Rev. Nucl. Part. Sci.*, 62:57–79, 2012.

[141] DAVID L. WILTSHIRE. From time to timescape - einstein's unfinished revolution. *International Journal of Modern Physics D*, 18(14):2121–2134, dec 2009.

[142] James Binney, Roya Mohayaee, John Peacock, and Subir Sarkar. Manifesto: challenging the standard cosmological model. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 383(2290):20240036, 2025.

[143] Thomas Buchert and Syksy Räsänen. Backreaction in late-time cosmology. *Annual Review of Nuclear and Particle Science*, 62(1):57–79, November 2012.

[144] Angela Adamo et al. The first billion years, according to jwst, 2024.

[145] P. W. Anderson. More is different. *Science*, 177(4047):393–396, 1972.

[146] G. Nicolis, I. Prigogine, W. H. Freeman, and Company. *Exploring Complexity: An Introduction*. W.H. Freeman, 1989.

[147] The Nobel Prize in Physics 2021. `https://www.nobelprize.org/prizes/physics/2021/summary/`. Accessed: 2022-03-13.

[148] James Ladyman, James Lambert, and Karoline Wiesner. What is a complex system? *European Journal for Philosophy of Science*, 3(1):33–67, 2013.

[149] G. Nicolis and I. Prigogine. *Self-Organization in Nonequilibrium Systems: From Dissipative Structures to Order Through Fluctuations*. A Wiley-Interscience publication. Wiley, 1977.

[150] Stanisław Sieniutycz. Chapter 2 - selforganization and complexity. In Stanisław Sieniutycz, editor, *Complexity and Complex Thermo-Economic Systems*, pages 9–24. Elsevier, 2020.

[151] J. Ladyman and K. Wiesner. *What Is a Complex System?* Yale University Press, 2020.

[152] S. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2024.

[153] Clifford M. Will. On the unreasonable effectiveness of the post-newtonian approximation in gravitational physics. *Proceedings of the National Academy of Sciences*, 108(15):5938–5945, 2011.

[154] LIGO Scientific Collaboration and VIRGO Collaboration. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102, Feb 2016. Abbott, B. P. et al.

[155] Bogdan Povh, Klaus Rith, Christoph Scholz, Frank Zetsche, and Martin Lavelle. *Many-Body Systems in the Strong Interaction*, pages 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[156] Chao Tang, Kurt Wiesenfeld, Per Bak, Susan Coppersmith, and Peter Littlewood. Phase organization. *Phys. Rev. Lett.*, 58:1161–1164, Mar 1987.

[157] F.Sylos Labini, M. Montuori, and L. Pietronero. Scale-invariance of galaxy clustering. *Physics Reports*, 293(2):61–226, 1998.

[158] D.J. Amit and V. Martin-mayor. *Field Theory, The Renormalization Group, And Critical Phenomena: Graphs To Computers (3rd Edition)*. World Scientific Publishing Company, 2005.

[159] A. Erzan, L. Pietronero, and A. Vespignani. The fixed-scale transformation approach to fractal growth. *Rev. Mod. Phys.*, 67:545–604, Jul 1995.

[160] A Gabrielli, R Cafiero, M Marsili, and L Pietronero. Theory of self-organized criticality for problems with extremal dynamics. *Europhysics Letters (EPL)*, 38(7):491–496, jun 1997.

[161] R. Durrer and F. Sylos Labini. A fractal galaxy distribution in a homogeneous universe? *aap*, 339:L85–L88, nov 1998.

[162] Francesco Sylos Labini and Thierry Baertschiger. Gravitational structure formation. *arXiv: Astrophysics*, 2005.

[163] L. Pietronero and F Sylos Labini. The problem of cosmological dark matter and statistical physics. *Eur. Phys. J. Spec. Top.*, 143:223–230, 2007.

[164] F. Sylos Labini and L. Pietronero. Statistical physics for cosmic structures. *The European Physical Journal B*, 64(3-4):615–623, Jan 2008.

[165] F. Sylos Labini, N. L. Vasilyev, L. Pietronero, and Y. V. Baryshev. Absence of self-averaging and of homogeneity in the large-scale galaxy distribution. *EPL (Europhysics Letters)*, 86(4):49001, May 2009.

[166] Francesco Sylos Labini and Luciano Pietronero. The complex universe: recent observations and theoretical challenges. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(11):P11029, Nov 2010.

[167] Giordano De Marzo, Francesco Sylos Labini, and Luciano Pietronero. Zipf's law for cosmic structures: How large are the greatest structures in the universe? *Astronomy and Astrophysics*, 651:A114, jul 2021.

[168] Luciano Pietronero and Francesco Sylos Labini. *Statistical Physics for Cosmic Structures*, pages 91–101. World Scientific, January 2005.

[169] Boris Alexeev. To the theory of galaxies rotation and the hubble expansion in the frame of non-local physics. *Journal of Modern Physics*, 3:1103–1122, 01 2012.

[170] B.V. Alexeev. *Nonlocal Astrophysics: Dark Matter, Dark Energy and Physical Vacuum*. Elsevier Science, 2017.

[171] P. Bak. *How Nature Works: The Science of Self-organized Criticality*. Copernicus Series. Springer, 1996.

[172] Philip W Anderson. *More and Different*. WORLD SCIENTIFIC, 2011.

[173] B.B. Mandelbrot, W. H. Freeman, and Company. *The Fractal Geometry of Nature*. Einaudi paperbacks. Henry Holt and Company, 1983.

[174] W. H. Press. Flicker noises in astronomy and elsewhere. *Comments on Astrophysics*, 7(4):103–119, January 1978.

[175] Benoit B Mandelbrot. *Les objets fractals : forme, hasard et dimension*. Paris : Flammarion, 1975.

[176] H.J. Jensen. *Self-Organized Criticality: Emergent Complex Behavior in Physical and Biological Systems*. Cambridge Lecture Notes in Physics. Cambridge University Press, 1998.

[177] Gunnar Pruessner. *Self-Organised Criticality: Theory, Models and Characterisation*. Cambridge University Press, 2012.

[178] M. Aschwanden. *Self-Organized Criticality in Astrophysics: The Statistics of Nonlinear Processes in the Universe*. Springer Praxis Books. Springer Berlin Heidelberg, 2011.

[179] Nicholas W. Watkins, Gunnar Pruessner, Sandra C. Chapman, Norma B. Crosby, and Henrik J. Jensen. 25 years of self-organized criticality: Concepts and controversies. *Space Science Reviews*, 198(1-4):3–44, May 2015.

[180] Markus Aschwanden, Norma Crosby, M. Dimitropoulou, Manolis Georgoulis, Stefan Hergarten, James Mcateer, Alexander Milovanov, Shin Mineshige, Laura Morales, Naoto Nishizuka, Gunnar Pruessner, Raul Sanchez, Surja Sharma, Antoine Strugarek, and V. Uritsky. 25 years of self-organized criticality: Solar and astrophysics. *Space Science Reviews*, 198, 03 2014.

[181] A. Sharma, Markus Aschwanden, Norma Crosby, Alexander Klimas, Alexander Milovanov, Raul Sanchez, and V. Uritsky. 25 years of self-organized criticality: Space and laboratory plasmas. *Space Science Reviews*, 198, 01 2016.

[182] Gian F. Giudice, Matthew McCullough, and Tevong You. Self-organised localisation. *Journal of High Energy Physics*, 2021(10), oct 2021.

[183] Lee Smolin. Cosmology as a problem in critical phenomena. In Ramón López-Peña, Henri Waelbroeck, Riccardo Capovilla, Ricardo García-Pelayo, and Federico Zertuche, editors, *Complex Systems and Binary Networks*, pages 184–223, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[184] Thomas Schweigler, Marek Gluza, Mohammadamin Tajik, Spyros Sotiriadis, Federica Cataldini, Si-Cong Ji, Frederik S. Møller, João Sabino, Bernhard Rauer, Jens Eisert, and et al. Decay and recurrence of non-gaussian correlations in a quantum many-body system. *Nature Physics*, 17(5):559–563, Jan 2021.

[185] R. O. Dendy, P. Helander, and M. Tagger. Self-organised criticality in astrophysical accretion systems. *Physica Scripta*, T82(1):133, 1999.

[186] Robert A. Crain, Joop Schaye, Richard G. Bower, Michelle Furlong, Matthieu Schaller, Tom Theuns, Claudio Dalla Vecchia, Carlos S. Frenk, Ian G. McCarthy, John C. Helly, and et al. The eagle simulations of galaxy formation: calibration of subgrid physics and model variations. *Monthly Notices of the Royal Astronomical Society*, 450(2):1937–1961, Apr 2015.

[187] Benoit B Mandelbrot. Galaxy distributions and fractals. *Astrophysical Letters and Communications*, 36:1–5, 1997.

[188] P.J.E. Peebles. The fractal galaxy distribution. *Physica D: Nonlinear Phenomena*, 38(1):273–278, 1989.

[189] T. Antal, F. Sylos Labini, N. L. Vasilyev, and Y. V. Baryshev. Galaxy distribution and extreme-value statistics. *EPL (Europhysics Letters)*, 88(5):59001, dec 2009.

[190] J. Richard Gott III, Mario Jurić , David Schlegel, Fiona Hoyle, Michael Vogeley, Max Tegmark, Neta Bahcall, and Jon Brinkmann. A map of the universe. *The Astrophysical Journal*, 624(2):463–484, may 2005.

[191] Alexia Lopez, Roger Clowes, and G. Williger. A big ring on the sky. *Journal of Cosmology and Astroparticle Physics*, 2024, 07 2024.

[192] Hans Boehringer, Gayoung Chon, Joachim Truemper, Renee C. Kraan-Korteweg, and Norbert Schartel. Unveiling the largest structures in the nearby universe: Discovery of the quipu superstructure, 2025.

[193] Morag I. Scrimgeour, Tamara Davis, Chris Blake, J. Berian James, Gregory B. Poole, Lister Staveley-Smith, Sarah Brough, Matthew Colless, Carlos Contreras, Warrick Couch, Scott Croom, Darren Croton, Michael J. Drinkwater, Karl Forster, David Gilbank, Mike Gladders, Karl Glazebrook, Ben Jelliffe, Russell J. Jurek, I-hui Li, Barry Madore, D. Christopher Martin, Kevin Pimbblet, Michael Pracy, Rob Sharp, Emily Wisnioski, David Woods, Ted K. Wyder, and H. K. C. Yee. The wigglez dark energy survey: the transition to large-scale cosmic homogeneity: Cosmic homogeneity in the wigglez survey. *Monthly Notices of the Royal Astronomical Society*, 425(1):116–134, July 2012.

[194] R. T. James McAteer, Markus J. Aschwanden, Michaila Dimitropoulou, Manolis K. Georgoulis, Gunnar Pruessner, Laura F. Morales, Jack Ireland, and Valentyna Abramenko. 25 years of self-organized criticality: Numerical detection methods. *Space Science Reviews*, 198:217–266, 2015.

[195] J.A. Wheeler and K. Ford. *Geons, Black Holes, and Quantum Foam: A Life in Physics*. W. W. Norton, 2010.

[196] Clifford M. Will. The confrontation between general relativity and experiment. *Living Reviews in Relativity*, 17(1), Jun 2014.

[197] M. Jammer. *Concepts of Mass in Classical and Modern Physics*. Dover classics of science and mathematics. Dover Publications, 1997.

[198] M. Jammer. *Concepts of Mass in Contemporary Physics and Philosophy*. Princeton University Press, 2009.

[199] A. Einstein. Does the intertia of a body depend upon its energy content? In A. Beck and P Havas, editors, *The collected papers of Albert Einstein: English translation*, chapter 24, pages 172–173. Princeton University Press, Princeton, N.J, 1987.

[200] H. Weyl. *Space, Time, Matter*. Dover Books on Advanced Mathematics. Dover Publications, 1952.

[201] Jaime Salcido, Richard G Bower, Luke A Barnes, Geraint F Lewis, Pascal J Elahi, Tom Theuns, Matthieu Schaller, Robert A Crain, and Joop Schaye. The impact of dark energy on galaxy formation. what does the future of our universe hold? *Monthly Notices of the Royal Astronomical Society*, 477(3):3744–3759, Apr 2018.

[202] A. Einstein, L. Infeld, and B. Hoffmann. The gravitational equations and the problem of motion. *Annals of Mathematics*, 39(1):65–100, 1938.

[203] S. Carlip. *General Relativity: A Concise Introduction*. Oxford University Press, 2019.

[204] Stephen Hawking. The conservation of matter in general relativity. *Communications in Mathematical Physics*, 18(4):301–306, December 1970.

[205] L.B. Szabados. Quasi-local energy-momentum and angular momentum in general relativity. *Living Rev. Relativ.*, 12(4), 2009.

[206] A. Einstein. *The Meaning of Relativity 5th Edition Including the Relativistic Theory of the Nonsymmetric Field*. Princeton paperbacks. Princeton University Press, 1956.

[207] B. Mashhoon, F. W. Hehl, and D. S. Theiss. On the gravitational effects of rotating masses: the Thirring-Lense papers. *General Relativity and Gravitation*, 16(8):711–750, August 1984.

[208] V. Venkatraman Krishnan, M. Bailes, W. van Straten, N. Wex, P. C. C. Freire, E. F. Keane, T. M. Tauris, P. A. Rosado, N. D. R. Bhat, C. Flynn, A. Jameson, and S. Osłowski. Lense–thirring frame dragging induced by a fast-rotating white dwarf in a binary pulsar system. *Science*, 367(6477):577–580, January 2020.

[209] Yehuda Hoffman, Daniel Pomarède, R. Brent Tully, and Hélène M. Courtois. The dipole repeller. *Nature Astronomy*, 1(2), jan 2017.

[210] H. Stephani, D. Kramer, M. MacCallum, C. Hoenselaers, and E. Herlt. *Exact Solutions of Einstein's Field Equations*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, 2009.

[211] Roy P. Kerr. Gravitational Field of a Spinning Mass as an Example of Algebraically Special Metrics. *Physical Review Letters*, 11(5):237–238, September 1963.

[212] Eric Poisson and Clifford M. Will. *Gravity: Newtonian, Post-Newtonian, Relativistic*. Cambridge University Press, 2014.

[213] Carles Simó. New families of solutions in n-body problems. In Carles Casacuberta, Rosa Maria Miró-Roig, Joan Verdera, and Sebastià Xambó-Descamps, editors, *European Congress of Mathematics*, pages 101–115, Basel, 2001. Birkhäuser Basel.

[214] Clifford M. Will. *Theory and Experiment in Gravitational Physics*. Cambridge University Press, 9 2018.

[215] Andrew M. Abrahams, Stuart L. Shapiro, and Saul A. Teukolsky. Disk collapse in general relativity. *Physical Review D*, 50(12):7282–7291, December 1994.

[216] George F.R. Ellis and Thomas Buchert. The universe seen at different scales. *Physics Letters A*, 347(1):38–46, 2005. Einstein Special Issue.

[217] Chris Clarkson, George Ellis, Julien Larena, and Obinna Umeh. Does the growth of structure affect our dynamical models of the universe? the averaging, backreaction, and fitting problems in cosmology. *Reports on Progress in Physics*, 74(11):112901, oct 2011.

[218] G. de Vaucouleurs. The case for a hierarchical cosmology. *Science*, 167(3922):1203–1213, 1970.

[219] A. Einstein. On a stationary system with spherical symmetry consisting of many gravitating masses. *Annals of Mathematics*, 40(4):922–936, 1939.

[220] Paolo Salucci. The distribution of dark matter in galaxies. *The Astronomy and Astrophysics Review*, 27(1), feb 2019.

[221] Yimeng Tang, Aaron J. Romanowsky, Jonah S. Gannon, Steven R. Janssens, Jean P. Brodie, Kevin A. Bundy, Maria Luisa Buzzo, Enrique A. Cabrera, Shany Danieli, Anna Ferré-Mateu, Duncan A. Forbes, and Pieter G. van Dokkum. An unexplained origin for the unusual globular cluster system in the ultradiffuse galaxy fcc 224. *The Astrophysical Journal*, 982(1):1, mar 2025.

[222] Buzzo, Maria Luisa, Forbes, Duncan A., Romanowsky, Aaron J., Haacke, Lydia, Gannon, Jonah S., Tang, Yimeng, Hilker, Michael, Ferré-Mateu, Anna, Janssens, Steven R., Brodie, Jean P., and Valenzuela, Lucas M. A new class of dark matter-free dwarf galaxies? - i. clues from fcc 224, ngc 1052-df2, and ngc 1052-df4. *Astronomy and Astrophysics*, 695:A124, 2025.

[223] N Jeffrey and et al. Dark Energy Survey Year 3 results: Curved-sky weak lensing mass map reconstruction. *Monthly Notices of the Royal Astronomical Society*, 05 2021. stab1495.

[224] T. M. C. Abbott et al. Dark energy survey year 3 results: Cosmological constraints from galaxy clustering and weak lensing. *Physical Review D*, 105(2), jan 2022.

[225] Jonathan HW Wong, T Shanks, N Metcalfe, and JR Whitbourn. The local hole: a galaxy underdensity covering 90 per cent of the sky to approx 200 mpc. *Monthly Notices of the Royal Astronomical Society*, 511:5742–5755, 2022.

[226] DES Collaboration. Dark energy survey year 3 results: Cosmological constraints from galaxy clustering and weak lensing, 2021.

[227] B.B. Mandelbrot. *Fractals: Form, Chance, and Dimension*. Mathematics Series. W. H. Freeman, 1977.

[228] C. L. Bennett, D. Larson, J. L. Weiland, N. Jarosik, G. Hinshaw, N. Odegard, K. M. Smith, R. S. Hill, B. Gold, M. Halpern, E. Komatsu, M. R. Nolta, L. Page, D. N. Spergel, E. Wollack, J. Dunkley, A. Kogut, M. Limon, S. S. Meyer, G. S. Tucker, and E. L. Wright. Nine-year wilkinson microwave anisotropy probe (wmap) observations: Final maps and results. *The Astrophysical Journal Supplement Series*, 208(2):20, September 2013.

[229] Naoki Kobayashi, Yoshihiro Yamazaki, Hiroto Kuninaka, Makoto Katori, Mitsugu Matsushita, Satoki Matsushita, and Lung-Yih Chiang. Fractal

structure of isothermal lines and loops on the cosmic microwave background. *Journal of the Physical Society of Japan*, 80(7):074003, Jul 2011.

[230] P. Tarakanov, M. Yezhkov, and M. Kostina. Fractal dimension of the cosmic microwave background as a test of "planck" spacecraft data. *Astrophysics*, 63, 06 2020.

[231] T.S. Kuhn and I. Hacking. *The Structure of Scientific Revolutions*. University of Chicago Press, 2012.

[232] H.F. Jones, Institute of Physics, and Physical Society (London). *Groups, Representations, and Physics*. Insitute of Physics Pub., 1998.

[233] H. Georgi and S.L. Glashow. *Lie Algebras In Particle Physics: From Isospin To Unified Theories*. Advanced Book Program. Basic Books, 1982.

[234] Luis Lehner. Numerical relativity: a review. *Classical and Quantum Gravity*, 18(17):R25–R86, August 2001.

[235] Carlos Palenzuela. Introduction to numerical relativity. *Frontiers in Astronomy and Space Sciences*, 7, September 2020.

[236] Xiangdong Ji. Qcd analysis of the mass structure of the nucleon. *Physical Review Letters*, 74(7):1071–1074, Feb 1995.

[237] S. Durr, Z. Fodor, J. Frison, C. Hoelbling, R. Hoffmann, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, and et al. Ab initio determination of light hadron masses. *Science*, 322(5905):1224–1227, Nov 2008.

[238] Craig D. Roberts. Empirical consequences of emergent mass. *Symmetry*, 12(9), 2020.

[239] K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, 2013.

[240] N.J. Giordano and H. Nakanishi. *Computational Physics*. Pearson/Prentice Hall, 2006.