The automated calculation of Feynman diagrams in affine Toda field theory

Paul Bayton

The Automated Calculation of Feynman Diagrams in Affine Toda Field Theory

by
Paul Bayton

A thesis Presented for the degree
of Doctor of Philosophy at the
University of Durham

Department of Mathematical Sciences
University of Durham
England
April 1994

Abstract

Paul Bayton, 1994

The purpose of this thesis is to calculate the residues of the poles of S matrix elements in the d_6, e_6 e_7 and e_8 Affine Toda Field theories. We will use the standard method of Feynman graphs in perturbation theory to calculate these results. Since there can be very many graphs, we will automate both the generation of these graphs and their subsequent calculation by the use of computer programs. This is the first ever machine calculation of residues in Toda Field theory. Affine Toda Field Theory has been studied in great depth for its integrability properties, and comparison will be made between the results we have generated, and the so called "exact" S matrices obtained by bootstrap and other properties. The results for the second and third order poles at positions predicted by the exact S matrix are tabulated for all of the algebras studied. In addition, for e_6 all of the second and third order poles are tabulated regardless of the exact S matrix. The Feynman diagrams for the fourth order poles in d_6 are generated, and the problem of calculating them is discussed.

Chapters

Chapter 1

Contains an introduction to integrable systems and a discusion of "spring models". In particular simple harmonic systems are analysed both classically and quantum mechanically.

Chapter 2

"Spring models" for Toda chains, molecules and Toda fields are discussed. There then follows a brief discussion of the many developments in solving Toda systems including the Lax method, classical soliton solutions for the A_n Toda field, conserved charges and the bootstrap method. Finally the conventional derivation of the exact S-matrices is presented.

Chapter 3

Contains a more detailed analysis of perturbation theory, including the development of the method of singularity analysis. The derivation of the masses and couplings used in the programs is also presented.

Chapter 4

We begin with an introduction to logic and logic programming and then show how these techniques can be used to solve the problem of how to generate graphs. The specific graph generation algorithm is discussed along with the two programs that actually did the work.

Chapter 5

The calculation algorithm and program are discussed. Various sections of the program are tested. We then discuss the final output- the results of the S matrix calculation.

Preface

This thesis is the result of work carried out in the Department of Mathematical Sciences at the University of Durham between October 1990 and September 1993, under the supervision of Dr D.B. Fairlie. No part of it has been previously submitted for any other degree, either in this or at any other university. The whole or any part of this thesis may be freely copied under the British Library access agreement at any time after the degree has been awarded. All software in this thesis is copyright under the GNU General Public License and may be freely redistributed and/or modified provided the copyright notice remains intact.

As this is a "Calculational" project essentially no new mathematics has been generated, however, the programs listed in the appendix together with the results form the bulk of the work. Much of this work has gone into producing programs that are error free, and work up to, in general fourth order on current computing machines. Given a more powerful computer, fifth and higher order poles could be checked, however since the algorithm is NP complete, the direct calculation of very high order poles (they go up to twelfth order) may be impossible on classical computing engines. Empirically, (for d_6) the CPU time increases 100 fold every order so that the twelfth order poles would need a machine 10^{16} times more powerful than todays machines (we assume that P $\neq$ NP, which is another problem entirely). So even with MPA calculating the 12th order poles would require a great deal of effort. Sets and predicates are generally given in the Z notation [50].

Acknowledgments

First of all I would like to thank my supervisor, Dr D.B.Fairlie. I would also like to thank Dr E.F.Corrigan for introducing me to Affine Toda Field theory in the first place, and to this problem in particular. I would like to thank all of the students in the Department of Mathematical Sciences for much interesting discussion, in particular R.Hall and D.Hind with whom I shared a room. Many thanks also to the artificial intelligence group of Durham University Department of Computing, and to the computing centre for the use of their machines. I would finally like to thank the Science and Engineering Research Council, for funding my three years of study.

Contents

1 Introduction

The aim of this study is to completely automate both the production and computation of Feynman diagrams for the calculation of S-matrix elements. Even using automation, the complete S-matrix elements are too complicated, and so only numerical factors for the residuals at certain poles in the complex "s" plane are calculated. When these are compared with the result expected from the exact S-matrices, there is agreement but only up to a point. The d_6 group was the first to be studied because it is the simplest group that has fourth order poles, and the lower order poles in all the groups were thought to be solved. In d_6 the second and third order poles were as expected, but the fourth order poles diverged. Also on calculating the lower order poles in the exceptional groups, virtually none of the poles were in agreement with the exact S matrices. It is still uncertain whether or not this is as a result of errors in the very complex and error prone programs, or whether it is an actual divergence, in which case new S-matrices will have to be proposed.

Exact solutions for the S (scattering) matrices have been proposed by many people (see for example [8]). Some of these have been calculated exactly by the inverse scattering method. Toda field theory is a particularly interesting field theory to which soliton solutions and conserved currents have been found [30], and exact S matrices have conjectured. It is these S matrices which we are going to check using standard perturbative techniques.

After the work of Poincare [56], it was thought that integrable systems were very special, and that most systems were not integrable. At the turn of the century, only a very few systems had been completely integrated. The examples of the hydrogen atom, Onsagers solution of the ising model, and the harmonic oscillator just about exhaust the list. Even today the number of integrable systems is few.

Fortunately most physical systems of interest, at least in certain, ie free approximations are also integrable. Modern physics has concentrated on systems that are not immediately integrable, eg many body bound systems, quark confinement in nuclei, Hartree Fock calculations of many electron atoms etc. Although in many cases a approximation scheme can be developed, the systems of interest are so complex that progress down this path has been slow (eg lattices requiring thousands of hours of computation).

Recently with the discovery of the inverse scattering method, and solitons a new class of integrable systems has been found. Solitons are not a new phenomenon. Probably the earliest example of an observation of a soliton comes from from Scott-Russell, when in 1834 he observed waves in water that were solitary (hence solitary waves, or solitons). In the early 1970's, solutions to many more non-linear equations were found, such as the Sine Gordon, the Boussinesq equation, the Hirota equation, the non-linear Schrödinger equation, the Born-Infeld equation, and the Kadomtsev-Petviashvili equation. A number of new techniques were also formed to solve non-linear equations. Firstly the inverse method, where by finding a suitable pair of operators the equation can be written in a Lax form. Hirota's method involves transforming the equation into a bilinear form. Also methods such as the Bäcklund transformation have been developed that transform one set of solutions into another.

A particular class of problems formed by potentials that are exponential in the configuration coordinates were studied, known by the general title of Toda systems have recently received much attention. In particular their quantum field theories were studied, and a set of conjectures were formed about the masses of the physical particles, and their scattering matrices. However, these S-matrices were only conjectures, guides by principles of crossing symmetry, and bootstrap.

The traditional, and rigorous way in which to calculate S-matrices is by Feynman path integration. S-matrix elements are written down as a polynomial in the coupling constant. The higher the order of the term, the more accurate the result. For anything less than very simple, low order coupling constants, the calculation gets very complicated, so that even two loop diagrams require the use of algebraic manipulation packages. For three loop amplitudes, the very process of producing the diagrams is difficult, and to some extent uncertain as some diagrams may be overlooked if done by hand.

1.1 Introduction to quantum field theory; the free field

We begin by stating a few standard forms for the Toda Lagrangian in Toda field theory. Later I will discuss in more detail the derivation of this

Lagrangian, and the other forms that it can take.

$$\mathcal{L}_a = \frac{1}{2}\partial_\mu \Phi^a \partial^\mu \Phi^a + \sum_i \lambda_i e^{\alpha_i . \Phi^a} \tag{1}$$

or sometimes,

$$\mathcal{L}_a = \frac{1}{2}\partial_\mu \Phi^a \partial^\mu \Phi^a + \frac{m^2}{\beta^2}\sum_i n_i e^{\alpha_i . \Phi^a} \tag{2}$$

Expressed with the potential given in terms of β and m and where the dot product

$$\alpha_i . \Phi = \sum \alpha_i^a \Phi_a \tag{3}$$

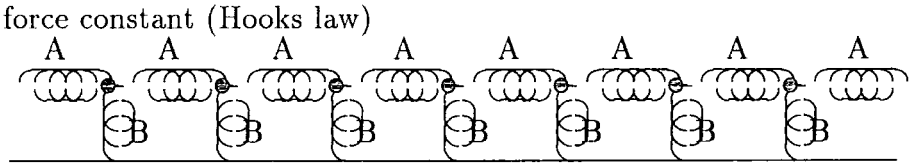
over a set of fields $\{\Phi\}$ indexed by a has been used.

1.2 Theory of Free Fields

The study of free fields goes back a long way. Classically they model many physical phenomena, electromagnetic waves, non dispersive water waves etc, and they are the first really complex systems to be solved exactly. The first, and simplest example of a free particle is the Klein-Gordon equation

$$(\partial_t^2 - \nabla^2 - m^2)\phi = 0 \tag{4}$$

This can be modelled by a simple system of springs, this time with a linear force constant (Hooks law)



The equations of motion for this lattice are

$$m\ddot{y}_i = \frac{1}{2}K_B y_i^2 + \frac{1}{2}K_A(y_{i+1} - y_i) + \frac{1}{2}K_A(y_{i-1} - y_i) \tag{5}$$

Where m is a "node mass", (the mass of an atom if we consider this system to model a solid, and a fictitious quantity in the case of a continuum field). K_A and K_B are the Hooke constants for the two types of springs.

Associating the continuum field with lattice points of spacing d: $\phi(id) \leftrightarrow y_i$ gives

$$(\partial_t^2 - \frac{K_A d^2}{2m}\nabla^2 - \frac{K_B}{2m})\phi(x) = 0 \tag{6}$$

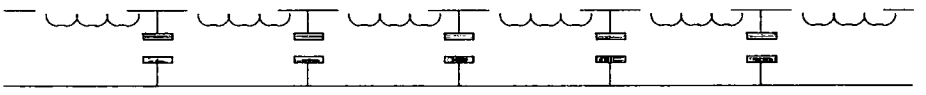
Thus the springs of type A give the wave phenomena $c^2 \leftrightarrow \frac{K_A d^2}{2m}$, and those of type B are responsible for the mass, $M \leftrightarrow \frac{K_B}{2m}$ so as to give the dispersion relationship

$$E = \pm\sqrt{m^2 + p^2} \tag{7}$$

The problem with models of this type, (and in general with models that try to discretise space-time) is they are not really Lorentz invariant, although of course the continuum limit equations which they represent are.

1.3 Applications

Transmission lines are collections of inductors and capacitors as shown below. They are used for carrying information, radio signals etc, and are in general a continuum approximation to the network shown (eg two wires placed a fixed distance apart). The separating medium is in general a dielectric material with a high relative permitivity. The permitivity is directly related to the molecular forces on the polarizable molecules in this material.



If the capacitance were linear, this would represent a "normal" transmission line, and we would once again recover the Klein Gordon equation. In reality most capacitors do have a certain (although very slight) non linearity. The precise non linearity depends on a combination of factors, but from what we know about intermolecular forces, it is very reasonable to suppose they are exponential together with some other polynomial (eg 6th order in the case of the Lenard-Jones self induced dipole-dipole interaction). Such lattices have been investigated by Hirota, Suzuki [29] and others.

An extreme example is the case of the two plates being made out of superconducting material (eg Indium), and being placed very close together, then due to a the Josephson effect of tunneling electrons the transmission

line produced has very non linear behavior, and "fluxons" have been observed.

There are possible applications here in fibre optics. Conventional fiber optic cables have dispersive properties, so pulses of information spread out, eventually mergeing and destroying the information. If a non-linear optical cable could be used to carry information in the form of solitons the pulses would could go great distances without dispersing.

Other examples of standard real world integrable fields are the Dirac equation

$$(\not{\partial} - m)\psi = 0; \quad \not{\partial} = \gamma^\mu \partial_m u \tag{8}$$

and the Maxwell equation

$$\partial_\mu F^{\mu\nu} = j^\nu \quad \partial_\mu \tilde{F}^{\mu\nu} = j^\nu_{magnetic} = 0^1 \quad F^{\mu\nu} = [\partial^\mu, A^\nu] \tag{9}$$

Both of which are first or second order partial differential equations, and have classical normal mode solutions:

$$\phi(x) = e^{i(k.x-\omega t)} \quad k^2 = m^2 \tag{10}$$

for the Klein Gordon equation

$$\psi(x) = e^{\pm ik.x} u_\pm(k) \tag{11}$$

for the Dirac equation ($u_\pm$ is a spinor)

$$A^\mu(x) = e^{ik.x} A^\mu_0 \tag{12}$$

for the zero current Maxwell equation

1.4 Quantisation

I shall now briefly discuss the standard quantisation of the Klein-Gordon field. For quantisation of the other field theories, one must choose a suitable set of coordinates which are analogous to position, together with their conjugate fields, analogous to momentum. To quantise, we take the field at

[1] assuming there are no magnetic monopoles

each point on the lattice as an independent variable, which is subject to the usual rules of canonical quantisation

$$[p_i, q_j] = -i\hbar\delta_{ij}; \quad [p_i, p_j] = [q_i, q_j] = 0 \tag{13}$$

where q_i represents the field at the i'th point of the lattice and p_i represents the conjugate momenta. In the continuum limit this is replaced by the more usual form

$$[\pi(x), \phi(y)] = -i\hbar Z(\epsilon)\delta(x-y); \quad [\pi(x), \pi(y)] = [\phi(x), \phi(y)] = 0 \tag{14}$$

$$\phi(x) = \phi(n\epsilon) \leftrightarrow \frac{1}{\epsilon}q_n \quad \pi(x) = \pi(n\epsilon) \leftrightarrow \frac{1}{\epsilon}p_n \tag{15}$$

I have introduced a parameter $Z(\epsilon)$ into the commutator above. By comparison with the lattice model, this essentially represents the number of degrees of freedom per unit length. (This could, if one likes be regarded as a pre-renormalisation (prenormalisation) parameter. As a function of the lattice spacing it is also a function of Λ, the UV cut off frequency ($\epsilon = 1/\Lambda$). With a suitable choice of Z, one may get $M_0 = M_{phys}$, the bare and renormalised masses to be equal. This would be a very satisfactory result from an ontological perspective. Note that whether we pre- or re- normalise our fields, we still end up with the same result which is all that counts in the long run)

The equation of motion is now the Schrödinger equation,

$$(i\hbar\frac{\partial}{\partial t} - H)\Psi = 0 \tag{16}$$

where the wave function is a function of all the fields in configuration space.

$$\Psi(\{q\}) \text{or} \Psi(\phi) \tag{17}$$

$$\{q\} = \{...., q_0, q_1, ..., q_i, ...\} \leftrightarrow \phi = \{...., \phi(\epsilon), \phi(2\epsilon), ...\} \tag{18}$$

Since there is an infinite number of degrees of freedom this makes solving equation (16) a very difficult problem. It is a problem however that is greatly simplified by the use of normal modes. To obtain the necessary Hamiltonian,

since the field theory is local, we can make use of the Lagrangian density, which for the Klein-Gordon equation is:

$$\mathcal{L} = \partial_\mu \phi \partial^\mu \phi - \frac{1}{2} m^2 \phi^2 \tag{19}$$

which, by the use of the field Lagrange equations

$$\partial_\mu \frac{\partial \mathcal{L}}{\partial \partial_\mu \phi} = \frac{\partial \mathcal{L}}{\partial \phi} \tag{20}$$

gives the equations of motion. The Hamiltonian is simply

$$\mathcal{H} = \frac{1}{2} \partial_\mu \phi \partial^\mu \phi + \frac{1}{2} m^2 \phi^2 \tag{21}$$

By definition the normal modes must be able to be added without disrupting the rest of the system. The Hamiltonian must be broken down as a sum of sub Hamiltonians

$$H = H_1 + H_2 + H_3 + ... \rightarrow \int dk H(k) \tag{22}$$

where each H_i is a solvable quantum system. If we consider the variables

$$\phi(x) = \int \frac{d^D x}{\sqrt{(2\pi)^D \sqrt{2i}}} \phi(k)[e^{ikx} + ie^{-ikx}] \quad \pi(x) = \int \frac{d^D x}{(2\pi)^D) \sqrt{2i}} \pi(k)[e^{ikx} + ie^{-ikx}] \tag{23}$$

so that

$$\phi(k) = \int \frac{d^D x}{\sqrt{(2\pi)^D \sqrt{2i}}} \phi(k)[e^{ikx} - ie^{-ikx}] \quad \pi(x) = \int \frac{d^D x}{\sqrt{(2\pi)^D \sqrt{2i}}} \pi(k)[e^{ikx} - ie^{-ikx}] \tag{24}$$

The fields $\phi(k)$, $\pi(k)$ obey the same commutation relations as the original fields, and the Hamiltonian is written as

$$H = \int d^D x [\frac{1}{2} \pi(x)^2 - \frac{1}{2} (\partial \phi(x))^2 + m^2 \phi(x)^2] = \int d^D k [i\pi(k)^2 - ik^2 \phi(k)^2 + 2im^2 \phi(k)^2] \tag{25}$$

Thus is written as an integral of sub Hamiltonian's, each of which is identified as a simple harmonic oscillator, and is solvable in the usual way, ie the wave function in this basis is a product of terms of the form

$$\Psi_k^{(n)}(\phi(k)) = H_n(\phi) e^{-\frac{\omega \phi^2}{2}} = (a^\dagger)^n e^{-\frac{\phi^2}{2}} \tag{26}$$

$$H_k \Psi_p^{(n)} = \frac{1}{2}\hbar\omega n \Psi_p \delta(k-p) \tag{27}$$

$$a_k^\dagger = \frac{1}{\sqrt{2}}[\omega^{\frac{1}{2}}\phi - i\omega^{\frac{1}{2}}\pi] \tag{28}$$

$$\omega = m^2 - k^2 \tag{29}$$

Thus we have solved the eigenvalue problem for this particular Hamiltonian. If the number of states is small, we can further simplify things by just writing down the integers corresponding to the n'th state for each mode. An eigenstate wavefunction in the "coordinate-field" basis is

$$\Phi^{\{n_k\}}(\{\phi_k\}) = \prod_k \Phi^{n_k}(\phi_k) \tag{30}$$

and in the number basis is

$$\Phi^{\{n_i\}}(\{\phi_k\}) = (a_{k_1}^\dagger)^{n_1}(a_{2_1}^\dagger)^{n_2}(a_{k_3}^\dagger)^{n_3}...|0> \tag{31}$$

This only represents one vector (Hamiltonian eigenvector) of the system. The total wavefunction for any system is a linear combination of these vectors. The usual basis for the total wavefunction is in terms of Wightman functions [4]. Unfortunately this situation is rarely observed due to interactions. If we think of our Klein Gordon field as a model of phonons, then there is no change in entropy, no heat conduction etc. Similarly in other integrable systems, eg the hydrogen atom, there is no decay, and excitation or stimulated emission. One of the most important phenomena today, stimulated emission comes from external field interaction.

We can assume the fields to be hermitian (in most physical systems, where the fields are measurable quantities this is true). We note that

$$\phi^\dagger(k) = \phi^*(k) = \phi(-k) \tag{32}$$

In our discussion that follows we use the fact that the antiparticles are represented by vectors drawn in the opposite direction to the particles.

1.5 Perturbative expansion

This appendix will briefly discus the standard method of calculating S-matrix elements in any quantum field theory. This information can be found in any good book on quantum field theory, for example Itzykson and Zuber [33], but is included here for completeness. This is by the perturbation theory method first invented by Feynman. It is convenient to work in the interaction picture where the Hamiltonian is split into two parts: the free Hamiltonian H_0 and the perturbation V.

$$H = H_0 + V \tag{33}$$

Operators evolve under unitary evolution under H_0, but the wave function evolves by V, ie

$$\dot{\hat{O}} = -i[H_0, \hat{O}] \Rightarrow O(t) = e^{-iH_0 t}\hat{O}(0)e^{iH_0 t} \tag{34}$$

$$-i\partial_t \mid \psi >= V(t) \mid \psi > \tag{35}$$

By cutting the time evolution into n intervals of duration ϵ ($t = n\epsilon$) and by using the infinitesimal generator

$$(1 + i\epsilon V_n), \tag{36}$$

the whole evolution of the wave function is made by successive applications of this generator

$$U(t,0) = (1 + i\epsilon V_n)(1 + i\epsilon V_{n-1})(1 + i\epsilon V_{n-2}).....(1 + i\epsilon V_1)(1 + i\epsilon V_0) \tag{37}$$

$$1 + in\epsilon \sum_i V_i + i(n\epsilon)^2 \sum_{ij} V_i V_j + ... \tag{38}$$

$$1 + it \sum_i V_i + i\frac{t^2}{2!} \sum_{ij} T(V_i V_j) + ... \tag{39}$$

or in the integral representation, for continuous fields and Hamiltonian density

$$U(t, t_0) = 1 - i \int d^2x T(\mathcal{V}(x)) - \int d^2x d^2y T(\mathcal{V}(x)\mathcal{V}(y)) + ... \tag{40}$$

$$S = U(\infty, -\infty) = \sum_{n=0}^{\infty} \frac{(-i)^n}{n!} \int (\prod_{i=1}^{n} d^2 x_i) T(\prod_{i=1}^{n} \mathcal{V}(x_i)) \tag{41}$$

where we have introduced T, the time ordering operator, which simply reorders its arguments in terms of time (latest first), ie

$$\begin{aligned}
T(V_i V_j) &= V_i V_j \quad \text{if} \quad i > j \\
&= V_j V_i \quad \text{if} \quad j > i \\
T(V_i V_j V_k) &= V_i V_j V_k \quad \text{if} \quad i > j > k \\
&= V_i V_k V_j \quad \text{if} \quad i > k > j \\
&= V_k V_i V_j \quad \text{if} \quad k > i > j \quad etc
\end{aligned}$$

or in general,

$$T(V_{i_1} V_{i_2} ... V_{i_n}) = V_{p_1(\{i\})} V_{p_2(\{i\})} ..... V_{p_{n-1}(\{i\})} V_{p_n(\{i\})} \tag{42}$$

where p is the permutation operator

$$(\forall j \bullet p_j(\{i\}) \in \{i\}) \tag{43}$$

$$\forall j, k \bullet j > k \Rightarrow p_j(\{i\}) > p_k(\{i\}) \tag{44}$$

In our model, V consists of the product of three, four, five etc fields (called three, four, five etc point couplings) together with 1st, second, third etc powers of the coupling constant, which can be chosen arbitrarily.

For perturbation theory to work well, ie for the above to converge quickly, this coupling parameter should approach zero (this method in fact breaks down when β approaches or exceeds one). The next step is to construct Feynman graphs. The quantity we are interested in calculating are the S-matrix elements:

$$S_{ab} = < 0 \mid a_k^a a_{k'}^b U(\infty, -\infty) a_k^{\dagger a} a_{k'}^{\dagger b} \mid 0 > \tag{45}$$

Remember that the creation/annihilation operators can be written as a linear combinations of the scalar fields

$$a^{\dagger}(k) = \int dx e^{ik_x x} (k_t \phi(x) + i\pi(x)) \tag{46}$$

$$a(k) = \int dx e^{-ik_x x}(k_t \phi(x) - i\pi(x)) \tag{47}$$

Since we are calculating vacuum expectation values of finite, short strings of operators we can expand the time ordered product of the terms in the integrals of (41), and using Wicks theorem (a proof of which is given in [2]) for general operators,

$$\begin{aligned} < A_1 A_2 A_3 .... A_n > = & <: A_1 A_2 ... A_n :> + \\ & < A_i A_j ><: A_1 ... A_i ... A_j ... A_n :> + \\ < A_i A_j >< A_k A_l ><: A_1 ... A_i ... A_j ... A_k ... A_l ... A_n :> & + ... + \\ & < A_i A_j >< A_k A_l > .... \end{aligned}$$

Where "::" represents normal ordering (placing creation operators before annihilation operators). Since the vacuum expectation value of a normal ordered expression is zero, we are only left to consider the (very many) expressions like the last line of the above. Each of these contractions represents a Feynman diagram. we will use the standard notation of contraction, namely

$$< \phi(\underbrace{x_1}\phi(\underbrace{x_2})\phi(\underbrace{x_3}\phi(\underbrace{x_4}) >= < \phi(x_1)\phi(x_2) >< \phi(x_3)\phi(x_4) > \tag{48}$$

Also, in particular, the contraction of two adjacent fields gives rise to a propagator:

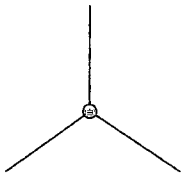
$$T(< \phi(\underbrace{x}\phi(\underbrace{y}) >) = \phi(x)\phi(y) = \Delta_F^+(x-y) = \int dp \frac{e^{p(x-y)}}{p^2 - m^2 + i\epsilon} \tag{49}$$

On examination of 41 we see that each integral can be written down as a sum of contracted fields, and therefore as sum of integrals of the form

$$\int \prod_i \frac{e^{ip_i x_j}}{p_i^2 - m^2 + i\epsilon} \tag{50}$$

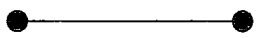
The product of all the terms C_{abc} (called coupling constants) are the same. In our model, this coupling constant is always proportional to β, the overall coupling parameter. It is not difficult to see therefore that we can represent these integrals by diagrams (known as Feynman diagrams), constructed by the following rules:

- For each vertex, there exists a coupling constant C_{abc}



- For each line, a propagator

$$\Delta_F(x-y) = \int \frac{d^D p e^{ip(x-y)}}{p^2 - m^2 + i\epsilon} \tag{51}$$



- For the external lines, with momenta $p_1..p_4$, an overall term

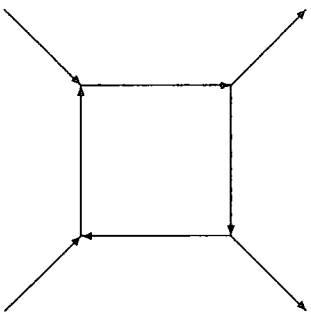
$$\delta(p_1 + p_2 + p_3 + p_4) \tag{52}$$

  which represents the initial and final states over which we are contracting, and implies conservation of 2-momenta.

- Each internal line (momenta) is integrated.

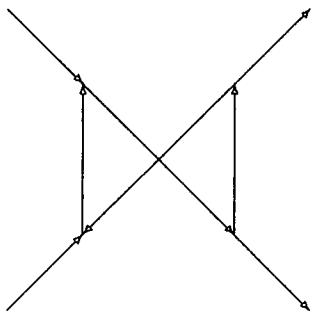
As an example, the S-matrix elements corresponding to the set of four 3 point couplings are

- The box diagram



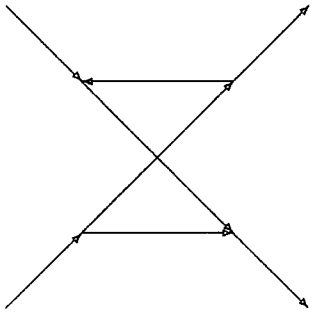
$$< a_{p_a} a_{p_b} \phi_a \phi_e \phi_h(x_1) \phi_b \phi_e \phi_f(x_2) \phi_c \phi_f \phi_g(x_3) \phi_d \phi_g \phi_h(x_4) a^\dagger_{p_c} a^\dagger_{p_d} > \tag{53}$$

- The s channel crossed box diagram



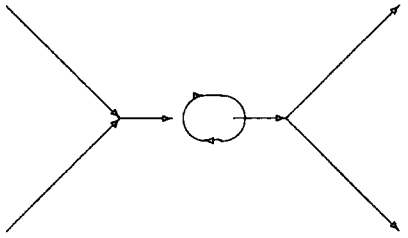
$$< a_{p_a} a_{p_b} \phi_a \phi_e \phi_h(x_1) \phi_b \phi_e \phi_f(x_2) \phi_c \phi_h \phi_g(x_3) \phi_d \phi_f \phi_g(x_4) a^\dagger_{p_c} a^\dagger_{p_d} > \qquad (54)$$

- The t channel crossed box diagram



$$< a_{p_a} a_{p_b} \phi_a \phi_e \phi_h(x_1) \phi_b \phi_g \phi_f(x_2) \phi_f \phi_h \phi_c(x_3) \phi_d \phi_h \phi_g(x_4) a^\dagger_{p_c} a^\dagger_{p_d} > \qquad (55)$$

- The one loop diagram



$$< a_{p_a} a_{p_b} \phi_a \phi_b \phi_e(x_1) \phi_e \phi_f \phi_g(x_2) \phi_g \phi_f \phi_h(x_3) \phi_h \phi_d \phi_c(x_4) a^\dagger_{p_c} a^\dagger_{p_d} > \qquad (56)$$

Notice that this diagram has a certain topological symmetry. This has important consequences for the S matrix elements. If the indexes f and g are dissimilar, then there is only one possible connection scheme. there are $(3!)^4$ other such elements caused by the permutation of the field indexes at each point, which cancels the 6^4 factor in the coupling constants. If however, $f = g$ then there are $6^2 3^2$ similar elements and only two ways of connecting the diagram, ie

$$< a_{p_a} a_{p_b} \phi_a \phi_b \phi_e(x_1) \phi_e \phi_f \phi_f(x_2) \phi_f \phi_f \phi_h(x_3) \phi_h \phi_d \phi_c(x_4) a^\dagger_{p_c} a^\dagger_{p_d} > \qquad (57)$$

and

$$< a_{p_a} a_{p_b} \phi_a \phi_b \phi_e(x_1) \phi_e \phi_f \phi_f(x_2) \phi_f \phi_f \phi_h(x_3) \phi_h \phi_d \phi_c(x_4) a^\dagger_{p_c} a^\dagger_{p_d} > \qquad (58)$$

This leads to an overall "symmetry" factor $\frac{1}{2}$ that has to be taken into account.

1.6 Approximation methods

The following approximation scheme is due to Landau [37]. As we have stated above, to calculate the Feynman integral exactly is not an easy problem. For a two dimensional field theory, however, the calculation of the integral near a pole is very much easier, and in fact can be done automatically. By a pole contribution, we mean the S-matrix element when all the momenta are on mass shell. ie

$$p^2 = m^2 \qquad (59)$$

where p is any momenta (internal or external), and m is the mass of the particle. Let p_a, p_b be the incoming momenta. A small change in $p_a \rightarrow p_a^{(0)} + \delta p_a$ and $p_b \rightarrow p_a^{(0)} + \delta p_a$ such that $p_a^2 = const = m_a^2, p_b^2 = const = m_b^2$ will affect all the other on-shell momenta in the diagram. The internal momenta are generally given by

$$p_i = p_0 + l_j \qquad (60)$$

Let

$$p_i = a_i p_a + b_i p_b \qquad (61)$$

then

$$\begin{aligned} p_i^2 - m_i^2 &= ab(p_a.\delta p_b + p_b.\delta p_a) && (62)\\ &= ab((p_a + p_b)^2 - (p_a^{(0)} + p_b^{(0)})^2 && (63)\\ &= ab(s - s_0) && (64) \end{aligned}$$

where

$$s = (p_a + p_b)^2 \tag{65}$$

is the standard Mandelstam variable. The momenta have only two components and are gigen as

$$\begin{aligned} p &= m(p_x, p_t) && (66)\\ &= m(\cosh\theta, \sinh\theta) && (67)\\ &= m(\cos\phi, -i\sin\phi) && (68) \end{aligned}$$

where $\phi = i\theta$ Minkowski space $\mathcal{M}_2, g = diag(1, -1)$ rapidity θ

$$(s - s_0) = 2m_a m_b \sinh(\theta)(\theta - i\theta_0) \tag{69}$$

the loop momenta are:

$$P_i \rightarrow p_i + \sum \lambda_{ij} l_i \tag{70}$$

this contains a slow moving part (the $p_i(\theta)$) and a fast moving part (the loop integration variables l_j). Let $\sum \lambda_{ij} l_j = k_i$, then

$$\begin{aligned} P^2 - m^2 &\rightarrow (s - s_0)(a_i b_i + 2p_i.k_i + k_i.k_i) && (71)\\ &\approx (s - s_0)(a_i b_i + 2p_i.k') && (72) \end{aligned}$$

where

$$k' = (s - s_0)k = \sum \lambda_{ij} l'_j \tag{73}$$

To calculate the Feynman integral, we use

$$\int \prod_{loops} d^2 l \prod \frac{1}{P_i^2 - m_i^2 + i\epsilon} \approx \frac{1}{(s - s_0)^{p-2l}} \int \prod_{loops} d^2 l' \prod \frac{1}{(2p_i^{(0)}.k'_i + a_i b_i + i\epsilon)} \tag{74}$$

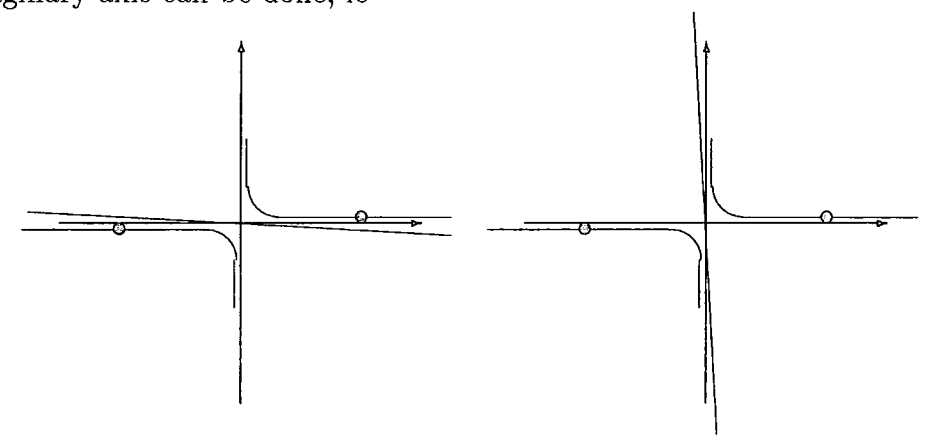
The dot product in the above, using $p_i = (\cos(\theta), \sin(\theta))$ is euclidean, ie

$$p_i^{(0)}.k' = k'_t \cos(\theta_i) + k'_x \sin(\theta_i) \tag{75}$$

This is acceptable because of the fact that a rotation can be done on the contour of integration of k'_x. Note that this is not the usual Wick rotation however. From the pole condition

$$p_x = \sqrt{p_l^2 - m^2 + i\epsilon} \tag{76}$$

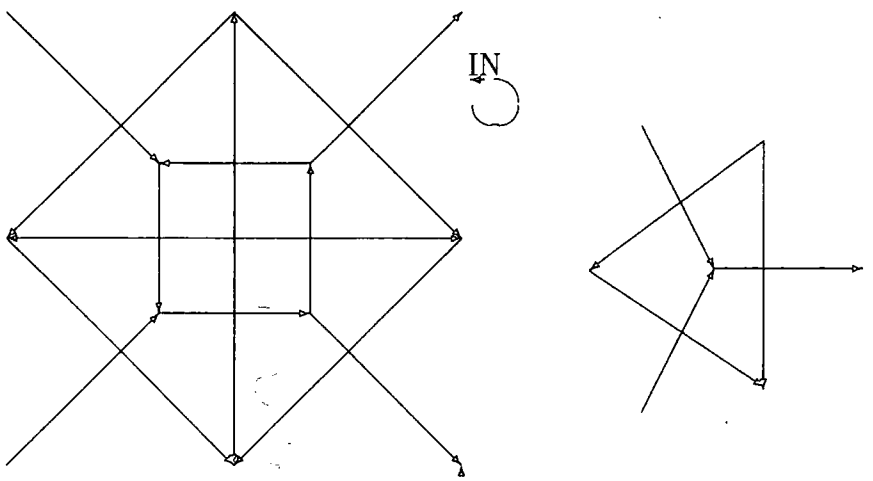
The pole, when p_x is integrated traces a hyperbola in the complex p_x plane so that a rotation of the p_x contour of integration from the real to the imaginary axis can be done, ie



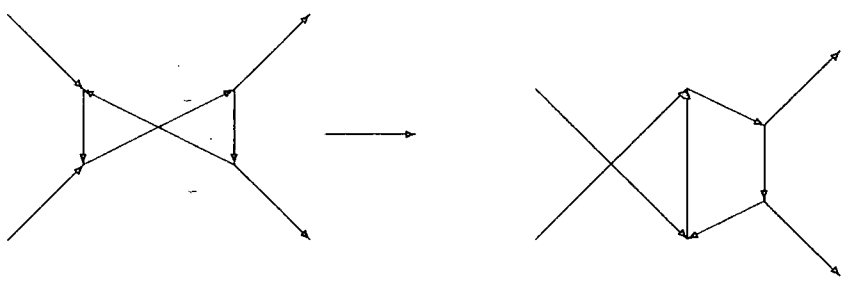
1.7 Dual Diagrams

A useful tool for considering these integrals is the use of dual diagrams, also described by Landau [37]. These are formed by replacing three, four, five... n point coupling etc with their respective triangles, squares, pentagons...n-gons etc. Their on shell momenta is represented as a vector, the sum of which is zero, (if they are all incoming or all outgoing) and hence the polygon is closed. For a mixture of incoming and outgoing momenta, their direction is reversed so that they all lie in the same direction.

Some examples are:



It is in fact the dual diagrams that accurately represent the on shell momentum vectors, and they could be plotted directly from such information, as was attempted (not entirely successfully) in one of the program's. For diagrams that do not involve the crossing of internal momenta, the dual diagram "tiles" the trapezium formed by the external momenta. In some cases however, (for example, the crossed diagrams) the diagram can not be topologically disentangled. The diagram involves a crossing of momenta at some point. Of course, practically all the diagrams can have crosses in them, but many of them can be completely disentangled. In the cases of the diagrams that can not be disentangled, there may be several possible ways of drawing it, for example



We call these different diagrams interpretations. After much analysis of such diagrams the heuristic that for any diagram there is a dual diagram that contains no crossings, only "holes" has been developed. We call such interpretations planar diagrams. the dual diagram contains no crossings, only "holes". The use of such geometrical analysis has however been rejected

as a basis of automated calculation, as proof of the heuristic is lacking, and it does not greatly simplify the detailed analysis, that we are embarking upon.

2 Toda Systems

2.1 Generalised Toda field theory in N dimensions

Early in the development of Toda field theory, Toda [52] wrote down the equations of motion for a 1 dimensional lattice (ie a one dimensional lattice which evolves in time) with an exponential potential or force[2].

$$m\ddot{y} = V'^{3}(y_{n+1} - y_n) - V'(y_n - y_{n-1}) = V'(r_n) - V'(r_{n-1}); \quad r_n = y_n - y_{n-1} \tag{77}$$

where the potential for an exponential lattice is

$$V(r) = \frac{a}{b}e^{-br} + ar - \frac{a}{b} \Rightarrow V'(r) = a(1 - e^{-br}) \tag{78}$$

Toda also wrote down a two dimensional version of this equation,

$$\frac{\partial^2 y_n}{\partial t^2} - \frac{\partial^2 y_n}{\partial x^2} = V'(r_n) - V'(r_{n-1}) \tag{79}$$

Although this equation can be generalised to higher dimensions, it ceases to be solvable in these higher dimensions. The original lattice was assumed to be of infinite length, and for this reason is called a Toda lattice, or Toda chain. The mechanical model it represents is a series of springs that have non-linear coupling

For a lattice of finite size, and especially for small size, the resulting lattice resembles a molecule, and is called a Toda molecule. For example, the simplest molecule has the model:

[2]exponential potential $\Leftrightarrow$ exponential force

[3]we use V as the potential here, replacing ϕ in [52] to avoid confusion with our later use of fields

with the equations

$$m\ddot{y}_0 = V'(r_0) - V'(r_1) \quad (80)$$
$$m\ddot{y}_{-1} = V'(r_1) \quad (81)$$
$$m\ddot{y}_1 = V'(r_0) \quad (82)$$
$$(83)$$

In the continuum limit of the infinite Toda lattice in N dimensions, we can produce a field equation which although different from that of (79) has similar soliton solutions:

$$m\ddot{\phi}(x,t) = \epsilon^2 \frac{\partial^2 V'(\phi)}{\partial x^2}; r_n(t) \rightarrow \phi(n\epsilon, t) \quad (84)$$

We shall see later that a system is completely integrable if we can produce a Lax [38] representation for it. The Lax representation involves using a pair of operators (called the Lax pair) which has a commutative structure. It is natural therefore to consider objects that have well defined commutation relationships- ie Lie algebras, and their infinite generalisations (Kac-Moody algebras). If one takes a classical Lie algebra, with basis h_i, e_{α_i}

$$[e_\alpha, e_\beta] = N_{\alpha\beta} e_{\alpha+\beta} \quad N_{\alpha\beta} = -N_{\beta\alpha} \quad (85)$$
$$[e_\alpha, e_{-\alpha}] = h_\alpha = \alpha^i h_i \quad [h_i, e_\alpha] = \alpha^i e_\alpha \quad (86)$$
$$\alpha^i = (h_i, e_\alpha) = \mathrm{trace}(\mathrm{Ad}h_i \mathrm{Ad}e_\alpha) \quad (h_i, h_j) = \delta_{ij} \quad (87)$$

Let us consider the following Toda like Hamiltonian:

$$H = \sum_j \frac{1}{2} p_i^2 + l_j^2 \quad l_j = b_j e^{\alpha_j^k q_k} \quad (88)$$

The Lax pair for this Hamiltonian is simply

$$L = \sum_j l_j (e_{\alpha_j} + e_{-\alpha_j}) + p_k h_k \quad (89)$$

$$A = l_j (e_{\alpha_j} - e_{-\alpha_j}) \quad (90)$$

Then working through the Lax equation gives

$$\begin{aligned} \dot{L} &= \dot{l}_j(e_{\alpha_j} + e_{-\alpha_j} + \dot{p}_k h_k \\ = [L, A] &= l_i l_j([e_{\alpha_i}, e_{\alpha_j}] - [e_{-\alpha_i}, e_{-\alpha_j}] \\ &+ [e_{\alpha_i}, e_{-\alpha_j}] - [e_{-\alpha_i}, e_{\alpha_j}]) \\ &+ l_i p_j([h_i, e_{\alpha_j}] + [h_i, e_{\alpha_j}]) \\ &= l_i^2 2\alpha_i^j h_j + l_i p_j 2\alpha_i^j e_{\alpha_j} \end{aligned}$$

Since the terms in $l_i l_j$ are summed over all i and j, all terms in $l_i l_j \quad i \neq j$ cancel because of the antisymmetry of the commutators. Also the terms in $[e_{\alpha_i}, e_{\alpha_i}]$ cancel because of the antisymmetry of $N_{\alpha\beta}$). Equating the coefficients of the group elements gives the equation of motion, ie

$$\dot{p}_j h_j = 2\alpha_i^j l_i^2 h_j \tag{91}$$

$$\dot{l}_i(e_{\alpha_i} + e_{-\alpha_i}) = l_i p_j \alpha_i^j (e_{\alpha_i} + e_{-\alpha_i}) \supset \dot{q} = p \tag{92}$$

Which is the equation of motion for the Hamiltonian (H). The form of the potential depends upon the roots α_i^j, and some potentials for the classical, non affine Lie algebras $A_n, B_n, C_n, D_n, E_6, E_7, E_8, F_4, G_2$ are listed below:

$$V_{A_n} = V_n + \exp(q_{n+1} - q_1) \tag{93}$$

$$V_{B_n} = V_{n-1} + \exp(q_n) + \exp(-q_1 - q_2) \tag{94}$$

$$V_{C_n} = V_{n-1} + \exp(2q_n) + \exp(-2q_n) \tag{95}$$

$$V_{D_n} = V_{n-1} + \exp(q_n + q_{n-1}) + \exp(-q_1 - q_2) \tag{96}$$

$$V_{E_6} = V_5 + \exp(\frac{1}{2}(-q_1 - q_2 - q_3 + q_4 + q_5 + q_6) + \frac{q_7}{\sqrt{2}}) + \exp(-\sqrt{2}q_7) \tag{97}$$

$$V_{E_7} = V_5 + \exp(\frac{1}{2}(-q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7 - q_8)) + \exp(-q_1 - q_2) + \exp(q_8 - q_7) \tag{98}$$

$$V_{E_8} = V_6 + \exp(\frac{1}{2}(-q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7 - q_8)) + \exp(-q_1 - q_2) + \exp(q_7 + q_8) \tag{99}$$

$$V_{F_4} = \exp(q_1 - q_2) + \exp(q_2 - q_3) + \exp(q_3) + \exp(\frac{1}{2}(-q_1 - q_2 - q_3 + q_4) + \exp(-q_1 - q_4) \tag{100}$$

$$V_{G_2} = \exp(q_1 - q_2) + \exp(-2q_1 - q_2 - q_3) + \exp(q_1 + q_2 - 2q_3) \quad (101)$$

where

$$V_n = \sum e^{(q_i - q_{i+1})} \quad (102)$$

We can rewrite equations (EM) in its general form (using the notation ϕ in place of q)

$$\partial^2 \phi = \sum_i \lambda_i e^{\alpha.\phi} \quad (103)$$

Where we have used multiple fields $\{\phi_1, \phi_2, ..., [\phi_0]_{opt}\}$ where $[\phi_0]_{opt}$ indicates an optional field for the affine algebras. We have also used the dot product

$$\alpha.\phi = \alpha^a \phi^a \quad (104)$$

The α's are roots of a generalised Lie algebra.

2.2 Lie Algebras

We consider a generalised Lie algebra (not just the classical Lie algebras but also Kac-Moody algebras), then the only restriction on this generalised Lie algebra is that its Cartan [14] matrix

$$C_{ij} = \frac{1}{2} \frac{\alpha_i.\alpha_j}{\alpha_i^2} \quad (105)$$

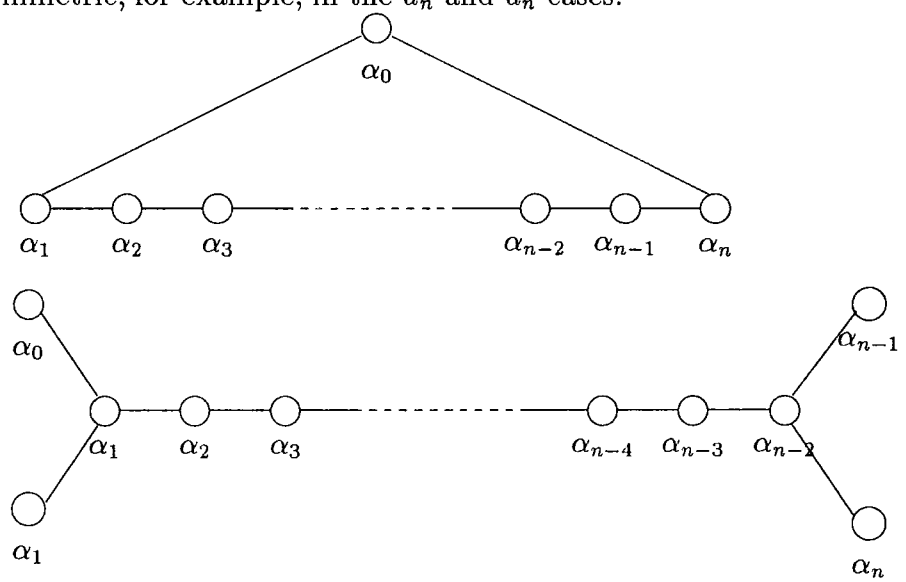
must satisfies the following predicates:

- $$(\forall i, j \in \Xi \bullet C_{ij} \subset Z) \quad (106)$$
- $$(\forall i \in \Xi \bullet C_{ii} = 2) \quad (107)$$
- $$(\forall i, j \mid i \neq j \bullet C_{ij} < 0) \quad (108)$$
- $$(\forall i, j \in \Xi \bullet C_{ij} = 0 \Rightarrow C_{ji} = 0) \quad (109)$$

Where $\Xi = \{[0], 1, 2, ... \dim(G)\}$ We denote a particular algebra $g^{(\tau)}$, where τ is known as the twist. for $\tau = 0$ we get the classical untwisted Lie algebras. These potentials have a classical minima when all the fields are at $-\infty$. When $\tau = 1$ we have the extended Lie algebra. The extension involves a new root denoted by

$$\alpha_0 = \sum_{i=1}^{n} n_i \alpha_i \tag{110}$$

The associated potential is now a minima at a certain finite position in ϕ space. The addition of the new roots in general makes the Dynkin diagrams more symmetric, for example, in the a_n and d_n cases:



For $\tau > 1$ we also have the twisted Lie algebras. These have been studied in the context of Affine Toda field theory, but are of no further interest to us. The roots are formed when a Cartan-Weyl basis is used to describe the algebra. The root space, denoted by $R \subset H^*$ is produced by the eigenvalues of the generators of the Cartan subalgebra h_i. The whole algebra

$g(A)$ is spanned by the generators $(\forall i \in \Xi \circ \{h_i, e_i^+, e_i^-\})$ or alternatively $(\forall i \in \Xi \forall \alpha \in \mathbb{N} \circ \{h_i, e_\alpha\})$

$$[h_i, h_j] = 0 \tag{111}$$

$$[h_{\alpha_i}, e_\alpha^\pm] = \pm C_{ij} e_j^\pm, or \quad [h_{\alpha_i}, e_{\alpha_j}] = \alpha(h_i) e_\alpha \tag{112}$$

$$[e_i^+, e_j^-] = \delta_{ij} h_j \tag{113}$$

The root system corresponding to $\tau = 1$ extended untwisted simply laced Lie algebra is our basis for the rest of this thesis. The extension to the algebra puts the Affine into Affine Toda field theory.

2.3 Canonical Toda Field

The form of the Toda Lagrangian we will use from now on involves replacing λ by n, m, β:

$$\mathcal{L} = \partial_\mu \phi \partial^\mu \phi - \frac{m^2}{\beta^2} \sum_i n_i e^{\beta \alpha . \phi} \tag{114}$$

The equations of motion being

$$\partial^2 \phi^a = \partial_+ \partial_- \phi^a = \frac{m^2}{\beta} \sum_i n_i \alpha_i^a e^{\beta \alpha . \phi} \tag{115}$$

2.4 Soliton Solutions of Toda Field Theory: The a_n Case

Soliton solutions with imaginary coupling constants for the a_n algebras have been found (see [30]). Soliton solutions for other Lie algebras have also been calculated, and are a continuing area for research. The a_n case however is particularly interesing since the solution for an arbitrary number of solitons has been found. For the many of the other Lie algebras, only 1 or 2, and in some cases 3 soliton solutions have been found (see [28]). First of all, let us look at some classical soliton solutions. A one soliton solution for the sine Gordon equation is well known and is given by

$$\frac{4}{\beta} \tan^{-1}(e^\xi) \tag{116}$$

where ξ is the comoving coordinate

$$\xi = \sigma(x - vt - \xi_0) \tag{117}$$

and $\sigma^2(1 - v^2) = 4m^2$. for a general Toda Field equation, let us represent the n fields ϕ^a by n+1 fields τ_i,

$$\phi^a = \sum_{i=0}^{n} \frac{-1}{\beta} \alpha_i^a \log \tau_i \tag{118}$$

then, the equations of motion,

$$\forall a \quad \partial_\mu^2 \phi^a = \frac{m^2}{\beta} \sum_i n_i \alpha_i^a e^{\beta \alpha_i . \phi} \tag{119}$$

in terms of τ are

$$\forall i \alpha_i (\ddot{\tau}\tau + \dot{\tau}^2 + \tau''\tau + (\tau')^2) = m^2 \tau_i^2 (\prod_{j=0}^{n} n_i \alpha_i \tau_k^{-\frac{1}{2}C_{ij} + \delta_{ij}} + 1) \tag{120}$$

For the a_n series, we can write this down in a bilinear form

$$\ddot{\tau}_i \tau_i - \dot{\tau}_i^2 - \tau_i'' \tau_i + \tau_j'^2 = m^2 (\tau_{i-1} \tau_{i+1} \tau_i^2) \tag{121}$$

identifying the above as a (Hirota [29]) bilinear form, we have the one soliton solutions as

$$\tau_i = e^{\sigma \xi} \omega_a^i \tag{122}$$

or,

$$\phi = \frac{-1}{\beta} \sum_j \alpha_j \log(1 + e^{\sigma \xi} \omega_a^j) \tag{123}$$

for multiple soliton solutions, we introduce

$$\Phi_j^p(\sigma_p, v_p, \xi_p^0, a_p) = \sigma_p(x - v_p - \xi_p^0) + \frac{2\pi i j a}{h} \tag{124}$$

so that

$$\tau_j^{2soliton} = 1 + e^{\Phi_j^1} + e^{\Phi_j^2} + A e^{\Phi_j^1 + \Phi_j^2} \tag{125}$$

and the three soliton solution is

$$\tau_j = 1 + e^{\Phi_j^1} + e^{\Phi_j^1} + e^{\Phi_j^1} + A^{12} e^{\Phi_j^1 + \Phi_j^2} + A^{13} e^{\Phi_j^1 + \Phi_j^3} + A^{23} e^{\Phi_j^2 + \Phi_j^3} + A^{12} A^{13} A^{23} e^{\Phi_j^1 + \Phi_j^2 + \Phi_j^3} \tag{126}$$

which can be extended in general. A is given by

$$A^{pq} = -\frac{(\sigma_p - \sigma_q)^2 - (\sigma_p v_p - \sigma_q v_q)^2 - 4m^2 \sin^2(\frac{\pi(a_p - a_q)}{h})}{(\sigma_p - \sigma_q)^2 - (\sigma_p v_p - \sigma_q v_q)^2 - 4m^2 \sin^2(\frac{\pi(a_p + a_q)}{h})} \tag{127}$$

this is a similar form to the Bethe anzatz. It is interesting to note that, on comparing the energy and momentum, that the soliton has a constant classical mass given by

$$M_a^{cl} = \frac{2h}{\beta^2} m_a \tag{128}$$

where m_a is the bare quantum mass. In fact it has been thought that, to all orders of perturbation, the renormalised mass is also finite, and the ratios are the same as for the bare mass. It has been done in [8] to one loop order. The result is that the renormalised mass counterterms are

$$m_c = m_{0c} + \delta m_c, \quad \frac{\delta m_c^2}{m_c^2} = \frac{\beta^2}{2h} \cot \frac{\pi}{h} \tag{129}$$

This is true for all the simply laced Lie algebras. [4]

2.5 Conservation laws, the Lax method and exact S matrices

We begin with the classical solution of the field equations, and then proceed later to the quantum case (which is what we are really interested in)

In 1968 Lax [38] developed a method for integrating non-linear differential equations, essentially linearising them. His method involves constructing two operators (usually in the form of a matrix representation) of the generalised coordinates p,q. Starting with a Hamiltonian H, the equations of motion are

$$\dot{q} = \frac{\partial H}{\partial p} \quad \dot{p} = -\frac{\partial H}{\partial q} \tag{130}$$

If we could write these equations down in a matrix form using a antisymmetric operator $B(\{p\},\{q\})$ and a unitary matrix $L(\{p\},\{q\})$

$$\dot{L} = [B, L] \tag{131}$$

[4] Recent results [55] indicate that this does not hold for some groups

For example, for the simple Toda chain, we could use the matrix representation developed by [19]

$$L = \begin{pmatrix} b_1 & a_1 & & & & a_N \\ a_1 & b_2 & a_2 & & & \\ & a_2 & b_3 & a_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{N-2} & b_{N-1} & a_{N-1} \\ a_N & & & & a_{N-1} & b_N \end{pmatrix} \tag{132}$$

$$L = \begin{pmatrix} 0 & -a_1 & & & & a_N \\ a_1 & 0 & a_2 & & & \\ & a_2 & 0 & -a_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{N-2} & 0 & -a_{N-1} \\ -a_N & & & & a_{N-1} & 0 \end{pmatrix} \tag{133}$$

This gives the equations of motion

$$\dot{a}_n = a_n(b_{n+1} - b_n) \quad \dot{b}_n = 2(a_n^2 - a_{n-1}^2) \tag{134}$$

where, as before the a's are identified with an exponential of the configuration coordinates

$$a_n = \frac{1}{2}e^{(q_{n-1}-q_n)/2} \quad b_n = \frac{1}{2}p_{n-1} \tag{135}$$

This is known as the Lax form of the equations of motion. This is a similar equation to the Heisenberg equation of motion, except that it determines the classical time evolution. If we consider the eigenvalues of L:

$$L\Phi = \lambda\Phi \tag{136}$$

on differentiating, this gives:

$$[B, L]\phi + L\dot{\phi} - \lambda\dot{\phi} - \dot{\lambda}\phi = (\lambda - L)B\phi - (\lambda - L)\dot{\phi} - \dot{\lambda}\phi = 0 \tag{137}$$

Since B is antisymmetric, the eigenvalues λ are constants of motion (by connection with the Schrödinger and Heisenberg equations) The solution

$$\dot{\lambda} = 0 \quad \dot{\Phi} = B\phi \tag{138}$$

Since the trace of a matrix is the sum of its eigenvalues, the trace of the square of a matrix the sum of the squares of its eigenvalues etc, ie

$$\text{trace}(L) = const = I_1, \text{trace}(L^2) = const = I_2, \tag{139}$$

$$\text{trace}(L^3) = const = I_3....etc \tag{140}$$

then these quantities $I_1, I_2, ...$ are also constants of motion. Thus we have a mapping

$$\{\forall i \in \mathbb{N} \circ \lambda_i\} \leftrightarrow \{\forall i \in \mathbb{N} \circ I_i\} \tag{141}$$

between two sets of conserved quantities. Using the Jacobi identity, the commutator of two of these conserved quantities is also a conserved quantity.

$$[H, [I_i, I_j]] = [I_i, [H, I_j]] + [I_j, [H, I_i]] = 0 \tag{142}$$

hence

$$(\forall I, J \in \{I\} \circ [I, J] \in \{I\}) \tag{143}$$

If this commutator $[I_i, I_j] = 0$ for all i and j, then the constants of motion are said to be in involution. This is a necessary condition for the constants of motion to survive quantisation. This is because in the quantum case, the integrals of motion represent operators acting on some Hilbert space. The involution condition then implies that they have simultaneous eigenvalues.

Using action angle variables [12] (esp [36]), elements of phase space can be reparametrised by a canonical transformation

$$\{p\}, \{q\} \leftrightarrow \{\theta\}, \{I\} \tag{144}$$

where the θ variable is 2π periodic. In this space, the motion flows on a n-torus of constant I (the action variables). For example, for a standing wave in the classical Klein Gordon model discussed earlier, the variables $\phi(k), \pi(k)$ trace an ellipse, and can be reparametrised by $R_k = , \theta_k$ such that $\phi_k = R_k \cos\theta_k, \pi_k = R_k \sin\theta_k$. On semiclassical quantisation, the flow around the theta variables is roughly speaking a de-Broglie wave, and the I variables are quantised by the relation

$$I_n \approx i\hbar\omega \tag{145}$$

thus the wave function can be written as

$$\Psi(\{\theta\}) = \prod e^{iks\theta} \tag{146}$$

Zamolodchikov [57] suggested that this picture could be used to construct the scattering matrices

$$S_{in,out} \mid in >= \mid out > \tag{147}$$

Since we do not expect the conserved quantities to depend on position, we expect them therefore to be polynomials in the momenta variables p_0 and p_1. In general, if we have a large number of such polynomials, then all the powers, of momenta are conserved. Also then any function of momenta would be conserved. In particular, we choose the set of conserved quantities to be powers of $p^+ = p_0 + p_1 = e^{\theta}, p^- = p_0 - p_1 = e^{-\theta}$. Thus $e^{s\theta}, e^{-s\theta}$ for some integers s are conserved.

Because of the conserved quantities, which roughly speaking are powers of the momenta, the particles retain their identity. For example there can be no long lived bound states since the conditions

$$p_c = p_a + p_b, \quad p_c^{s_1} = p_a^{s_1} + p_b^{s_1}, ... \tag{148}$$

cannot all be simultaneously satisfied. For dissimilar particles, the reflection coefficient is also zero assuming there are odd powers of momenta in the conserved quantities.

Thus the S matrix is assumed to be elastic, and all particle collisions are of the form

$$A + B \rightarrow B + A \tag{149}$$

From one of the conservation laws: the conservation of energy-momenta the 2-momentum of the particles in the initial states are the same as their outgoing states. The S-matrix can therefore "only" be a simple phase shift:

$$S_{ij} \rightarrow e^{i\Omega(P_i,P_j)} \tag{150}$$

Hear P_i and P_j are the momenta of the two (free) incoming particles. Since we are dealing with a two dimensional theory, we can write $\Omega(P_a, P_b)$ in terms of just one variable, the "rapidity" difference of the two particles denoted by θ.

$$\Omega(P_a, P_b) \to \Omega(\theta) \tag{151}$$

where

$$\theta = \theta_a - \theta_b \tag{152}$$

$$P_a = m_a(\cosh(\theta_a), \sinh(\theta_a)) \tag{153}$$

$$P_b = m_b(\cosh(\theta_b), \sinh(\theta_b)) \tag{154}$$

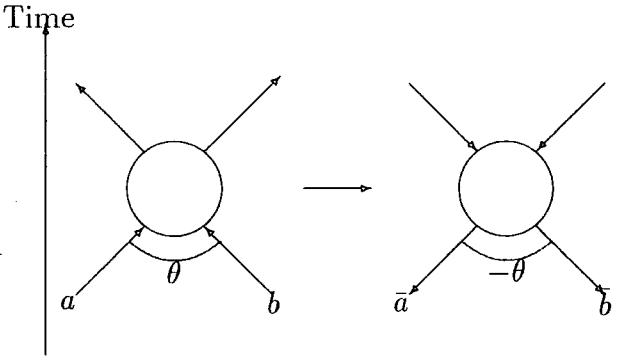
The exact S-matrix is built up as follows: for elastic scattering in 1+1 dimensions we have the following axioms:

- Unitarity As in most theories, the S-matrix is a unitary operator (The Hamiltonian is Hermitian). It would otherwise violate conservation of probability.

$$SS^\dagger = 1 \tag{155}$$

By replacing particles by antiparticles, and reflecting in time, the unitarity condition may be re-written in a more suitable form

$$S(\theta)S(-\theta) = 1 \tag{156}$$

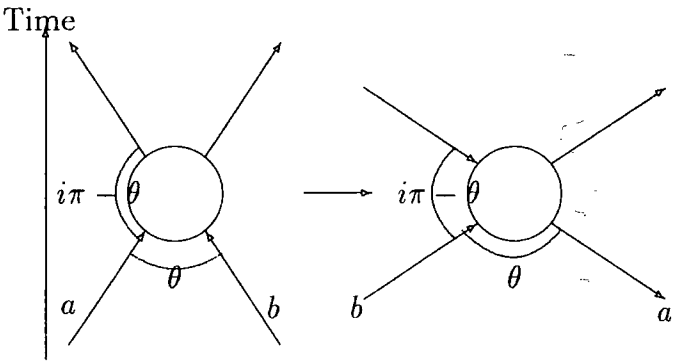


The figure above shows how this is obtained.

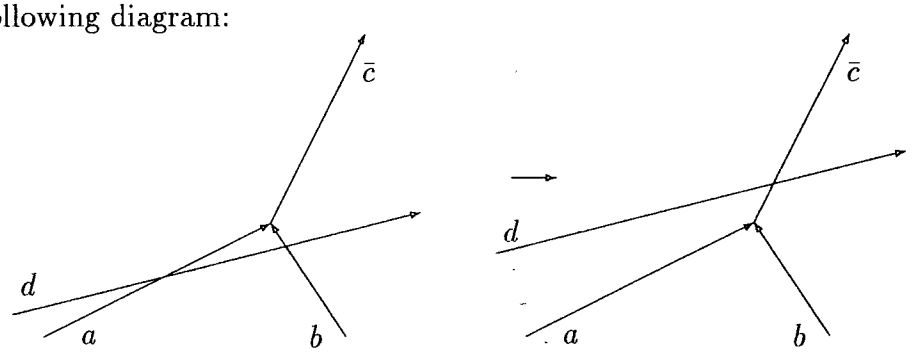
- Crossing The crossing condition is written as

$$S_{ab}(\theta) = S_{b\bar{a}}(i\pi - \theta) \tag{157}$$

which is derived by a reflection of the space and time coordinates, or by swapping the Mandelstam variables s and t.



- Bootstrap The bootstrap condition is generated by considering the following diagram:



as the S-matrix is elastic, we deduce that

$$S_{d\bar{c}}(\theta) = S_{da}(\theta - i\bar{\theta}^{b}_{ac})S_{db}(\theta + i\bar{\theta}^{a}_{bc}) \tag{158}$$

This shows the equivalence of the S matrix under such a transformation. either, first particle d and a scatter followed by the scattering of d and b, or alternatively particles a and b could fuse to form particle $\bar{c}$, and then particles d and $\bar{c}$ could scatter. The lines in the diagrams

represent classically well separated jets. Their points of arrival would be determined by the asymptotic states (ie when a,b and d were well separated at $t = -\infty$). The asymptotic states for the two processes are almost indistinguishable from one another, and so it is reasonable that the S matrices for the two processes are the same.

This type of equivalence is a very common structure in modern mathematics. It is related to the familiar Yang-Baxter equation, whereby the equivalence of the S matrix without any fusion is considered. If the three particles were in a loop space (ie the fields at $+\infty$ and $-\infty$ were the same) then the Yang Baxter diagram could form a single "knotted loop" which was invariant under the transformations. In knot theory, this is equivalent to the third Reidermeister move.

The conserved quantities can be deformed to a set of conserved charges that speaking roughly are powers of the momentum, and operate on multi-particle states by

$$Q_n \mid P_1, P_2, P_3, .. >= (\omega_n(p_1) + \omega_n(p_2) + \omega_n(p_3) + ...) \mid P_1, P_2, P_3, .. > \quad (159)$$

$$\omega_s(p_a) = q_s^a e^{s\theta_a} \quad (160)$$

By considering the conserved charges before and after the fusion of a and b, we get

$$\begin{aligned} \omega_s(p_{\bar{c}}) &= \omega_s(p_a) + \omega_s(p_b) \\ q_s^{\bar{c}} &= q_s^a e^{-is\theta_{ac}^b} + q_s^b e^{is\theta_{bc}^a} \end{aligned}$$

since

$$\theta_c = \theta_a + \theta_{ac}^b \quad \theta_b = \theta_c - \theta_{bc}^a \quad (161)$$

The set of spins s must be consistent with all scattering's. Not all integer spins are consistent with this set. For example for the fusion of $1 + 1 \rightarrow 2$ in all the d_n cases, we have the conditions

$$q^1 = q^2 e^{-\frac{2\pi i}{h}} + q^2 e^{\frac{3\pi i}{h}} \quad q^2 = q^1 e^{-\frac{3\pi i}{h}} + q^1 e^{\frac{3\pi i}{h}} \quad (162)$$

hence

$$e^{\frac{-3\pi i s}{h}} + e^{\frac{(h-3)\pi i s}{h}} = e^{\frac{-3\pi i s}{h}}(1 + e^{i\pi s}) = 0 \supset s \in 1, 3, 5... \quad (163)$$

The values of s for all the simply laced algebras have been calculated by Zamolodchikov and others [57] and are listed below:

$$\begin{array}{ll} a_n & s = 1..n \bmod (n+1) \\ d_n & s = 1,3,5..n-1 \bmod (2n-2) \\ e_6 & s = 1,4,5,7,8,11 \bmod (12) \\ e_7 & s = 1,5,7,,9,13,17 \bmod (18) \\ e_8 & s = 1,7,11,13,17,19,23,29 \end{array}$$

We now turn our attention to the S matrices themselves. By considering the fusing, we would expect simple poles at imaginary rapidity values. These poles would repeat every $2\pi in$. There is no reason to suppose that there would be any cuts, and so the S matrix form is a product of meromophic functions. The "minimal" elements are first derived, and then multiplied by similar factors involving the coupling constant To construct the exact S-matrices, the following building blocks have been used (from [8])

$$(x) = \frac{\sinh(\frac{\theta}{2} + \frac{i\pi x}{2h})}{\sinh(\frac{\theta}{2} - \frac{i\pi x}{2h})} \tag{164}$$

$$\{x\} = \frac{(x+1)(x-1)}{(x+1-B)(x-1+B)} \tag{165}$$

$$[x] = \{x\}\{h-x\} \tag{166}$$

Where B is a function of the coupling constant, usually taken to be

$$B = \frac{\beta^2}{(2\pi)(1+\frac{\beta^2}{4\pi})} \tag{167}$$

For the simply laced algebras, the S matrices are listed below. For the tables, the S matrix is a product of the building blocks for a particular element.

2.6 The a_n S matrix

$$S_{ab} = \prod_{p=|a-b|+1 step 2}^{a+b-1} \{p\} \tag{168}$$

2.7 The d_n S matrix

$$S_{ab} = \prod_{p=|a-b|+1 step 2}^{a+b-1} \{p\}\{h-p\} \tag{169}$$

For n even:

$$S_{ss} = S_{s's'} = \prod_{p=1 step 4}^{h-1} \{p\} \tag{170}$$

$$S_{ss'} = S_{s's} = \prod_{p=3 step 4}^{h-3} \{p\} \tag{171}$$

$$S_{sa} = S_{s'a} = \prod_{p=0 step 2}^{2a-2} \{n-a+p\} \tag{172}$$

and for n odd:

$$S_{ss} = S_{s's'} = \prod_{p=1 step 4}^{h-3} \{p\} \tag{173}$$

$$S_{ss'} = S_{s's} = \prod_{p=3 step 4}^{h-1} \{p\} \tag{174}$$

$$S_{sa} = S_{s'a} = \prod_{p=0 step 2}^{2a-2} \{n-a+p\} \tag{175}$$

In particular, we have for the d_6 case:

2.8 The d_6 S matrix

Table 1: The d_6 S Matrix

	1	2	3	4	s	s'
1	[1]	[2]	[2]	[4]	{5}	{5}
2	[2]	[1][3]	[2][4]	[3][5]	[4]	[4]
3	[2]	[2][4]	[1][3] [5]	[2][4] [6]	[3]{5}	[3]{5}
4	[4]	[3][5]	[2][4] [6]	[1][3] [5][7]	[2][4]	[2][4]
s	{5}	[4]	[3]{5}	[2][4]	[1]{5}	[3]
s'	{5}	[4]	[3]{5}	[2][4]	[3]	[1]{5}

2.9 The e_6 S matrix

Table 2: The e_6 S Matrix

	l	$\bar{l}$	L	h	$\bar{h}$	H
l	{1}{7}	{5}{11}	{4}	{4}{6}{10}	{2}{6}{8}	{3}{5}
$\bar{l}$	{5}{11}	{1}{7}	{4}	{2}{6}{8}	{4}{6}{10}	{3}{5}
L	{4}	{4}	{1}{5}	{3}{5}	{3}{5}	{2}{4}{6}
h	{4}{6}{10}	{2}{6}{8}	{3}{5}	{1}{3}{5}{7}^{2} {9}	{3}{5}^{2} {7}{9}{11}	{2}{4}^{2} {6}
$\bar{h}$	{2}{6}{8}	{4}{6}{10}	{3}{5}	{3}{5}^{2} {7}{9}{11}	{1}{3}{5}{7}^{2} {9}	{2}{4}^{2} {6}
H	{3}{5}	{3}{5}	{2}{4}{6}	{2}{4}^{2} {6}	{2}{4}^{2} {6}	{1}{3}^{2} {5}^{3}

2.10 The e_7 S matrix

Table 3: The e_7 S Matrix

	1	2	3	4	5	6	7
1	$[1]\{9\}$	$[6]$	$[5]\{9\}$	$[2][8]$	$[5][7]$	$[3][7]$ $\{9\}$	$[4][6]$ $[8]$
2	$[6]$	$[1][7]$	$[4][8]$	$[5][7]$	$[2][6]$ $[8]$	$[4][6]$ $[8]$	$[3][5]$ $[7][9]$
3	$[5]\{9\}$	$[4][8]$	$[1][5]$ $[7]\{9\}$	$[4][6]$ $[8]$	$[3][5]$ $[7][9]$	$[3][5]$ $[7]^2\{9\}$	$[2][4]$ $[6]^2[8]^2$
4	$[2][8]$	$[5][7]$	$[4][6]$ $[8]$	$[1][3]$ $[7][9]$	$[4][6]^2$ $[8]$	$[2][4]$ $[6][8]$	$[3][5]^2$ $[7]^2[9]$
5	$[5][7]$	$[2][6]$ $[8]$	$[3][5]$ $[7][9]$	$[4][6]^2$ $[8]$	$[1][3]$ $[5][7]^2$ $[9]$	$[3][5]^2$ $[7]^2[9]$	$[2][4]^2$ $[6]^2[8]^3$
6	$[3][7]$ $\{9\}$	$[4][6]$ $[8]$	$[3][5]$ $[7]^2\{9\}$	$[2][4]$ $[6][8]$	$[3][5]^2$ $[7]^2[9]$	$[1][3]$ $[5]^2[7]^2$ $\{9\}^3$	$[2][4]^2$ $[6]^3[8]^3$
7	$[4][6]$ $[8]$	$[3][5]$ $[7][9]$	$[2][4]$ $[6]^2[8]^2$	$[3][5]^2$ $[7]^2[9]$	$[2][4]^2$ $[6]^2[8]^3$	$[2][4]^2$ $[6]^3[8]^3$	$[1][3]^2$ $[5]^3[7]^4$ $[9]^2$

2.11 The e_8 S matrix

Table 4: The e_8 S Matrix

	1	2	3	4	5	6	7	8
1	[1][11]	[7][13]	[2][10] [12]	[6][10] [14]	[3][9] [11][13]	[6][8] [12][14]	[4][8] [10][12] [14]	[5][7] [9][11] [13][15]
2	[7][13]	[1][7] [11][13]	[6][8] [12][14]	[4][8] [10][12] [14]	[5][7] [9][11] [13][15]	[2][6] [8][10] $[12]^2[14]$	[4][6] $[8][10]^2$ $[12][14]^2$	[3][7] $[5][9]^2$ $[11]^2[13]^2$ [15]
3	[2][10] [12]	[6][8] [12][14]	[1][3] $[9][11]^2$ [13]	[5][7] [9][11] [13][15]	[2][4] $[8][10]^2$ $[12]^2[14]$	$[5][7]^2$ [9][11] $[13]^2[15]$	[3][5] $[7][9]^2$ $[11]^2[13]$ [15]	[4][6]^2 $[8]^2[10]^2$ $[12]^2[14]^3$
4	[6][10] [14]	[4][8] [10][12] [14]	[5][7] [9][11] [13][15]	[1][5] [7][9] $[11]^2[13]$ [15]	[4][6] [8][10] $[12]^2[14]^2$	[3][5] $[7][9]^2$ $[11]^2[13]^2$ [15]	[3][5] $[7]^2[9]^2$ $[11]^2[13]^3$ [15]	[2][4] $[6]^2[8^2]$ $[10]^3[12]^3$ $[14]^3$
5	[3][9] [11][13]	[5][7] [9][11] [13][15]	[2][4] $[8][10]^2$ $[12]^2[14]$	[4][6] [8][10] $[12]^2[14]^2$	[1][3] [5][7] $[9]^2[11]^3$ $[13]^2[15]$	$[4][6]^2$ $[8]^2[10]^2$ $[12]^2[14]^3$	[2][4] $[6]^2[8]^2$ $[10]^3[12]^3$ $[14]^3$	$[3][5]^2$ $[7]^3[9]^3$ $[11]^3[13]^4$ $[15]^2$
6	[6][8] [12][14]	[2][6] [8][10] $[12]^2[14]$	$[5][7]^2$ [9][11] $[13]^2[15]$	[3][5] $[7][9]^2$ $[11]^2[13]^2$ [15]	$[4][6]^2$ $[8]^2[10]^2$ $[12]^2[14]^3$	[1][3] $[5][7]^2$ $[9]^2[11]^3$ $[13]^3[15]$	$[3][5]^2$ $[7]^2[9]^3$ $[11]^3[13]^3$ $[15]^2$	$[2][4]^2$ $[6]^2[8]^3$ $[10]^4[12]^4$ $[14]^4$
7	[4][8] [10][12] [14]	[4][6] $[8][10]^2$ $[12][14]^2$	[3][5] $[7][9]^2$ $[11]^2[13]$ [15]	[3][5] $[7]^2[9]^2$ $[11]^2[13]^3$ [15]	[2][4] $[6]^2[8]^2$ $[10]^3[12]^3$ $[14]^3$	$[3][5]^2$ $[7]^2[9]^3$ $[11]^3[13]^3$ $[15]^2$	[1][3] $[5]^2[7]^3$ $[9]^3[11]^4$ $[13]^4[15]^2$	$[2][4]^3$ $[6]^3[8]^4$ $[10]^4[12]^5$ $[14]^5$
7	[5][7] [9][11] [13][15]	[3][7] $[5][9]^2$ $[11]^2[13]^2$ [15]	$[4][6]^2$ $[8]^2[10]^2$ $[12]^2[14]^3$	[2][4] $[6]^2[8^2]$ $[10]^3[12]^3$ $[14]^3$	$[3][5]^2$ $[7]^3[9]^3$ $[11]^3[13]^4$ $[15]^2$	$[2][4]^2$ $[6]^2[8]^3$ $[10]^4[12]^4$ $[14]^4$	$[2][4]^3$ $[6]^3[8]^4$ $[10]^4[12]^5$ $[14]^5$	$[1][3]^2$ $[5]^3[7]^4$ $[9]^5[11]^6$ $[13]^6[15]^3$

3 Toda Purturbation Theory

In order to calculate the S matrices in Affine Toda Field theory, we need to find the mass matrix and the coupling constants. We first of all expand the potential in equation (114)

$$\mathcal{V}(\phi) = \frac{m^2}{\beta^2}\sum_i n_i e^{\beta\alpha.\phi} = \frac{m^2}{\beta^2}\sum_i n_i(1 + \beta\alpha.\phi + \frac{\beta^2(\alpha.\phi)^2}{2!} + \frac{\beta^3(\alpha.\phi)^3}{3!} + ...) \tag{176}$$

If we are dealing with a simply laced affine Lie algebra, then the linear term in ϕ vanishes (since $\alpha_0 = \sum \alpha_i$). The second term is

$$M^{ab}\phi^a\phi^b \tag{177}$$

where

$$M^{ab} = m^2\sum_i n_i\alpha_i^a\alpha_i^b \tag{178}$$

is known as the mass matrix. It is dependent on the particular choice of a basis for the root system, and can be diagonalised

$$M^{ab} = M^a\delta^{ab} \tag{179}$$

this is particularly useful, as we than have, to lowest order in β a n field perturbed Klein Gordon field. We will now construct such suitable basis for the a, d, e series.

Firstly the A_n series, A representation for this is given in [25],

$$\underline{e}_i - \underline{e}_j \tag{180}$$

and a suitable basis is

$$\alpha_i = \underline{e}_{i+1} - \underline{e}_i \tag{181}$$

However, simply using unit vectors for $\underline{e}_i$ gives an n+1 dimensional space for a root lattice of only n dimensions. This can be remedied by making

the last root

$$\alpha_n = \frac{\sqrt{h}-1}{n}\sum_{i=1}^{n-1}\underline{e}_i + (1+\frac{\sqrt{h}-1}{n})\underline{e}_n \tag{182}$$

which with $h = n+1$ has a length squared equal to 2. The eigenvectors are related by a unitary transformation

$$\underline{e}_i = (e_i^1, e_i^2, ...) \tag{183}$$

$$\alpha_i^a = \alpha_i^j e_j^a \tag{184}$$

$$\underline{e}^a.\underline{e}^b = e_i^a e_i^b \tag{185}$$

$$X^{ij} = \alpha_k^i \alpha_k^j \tag{186}$$

$$M^{ab} = e_i^a X^{ij} e_j^b \tag{187}$$

thus the eigenvalues of X are the masses. X is related to M by a unitary transformation.

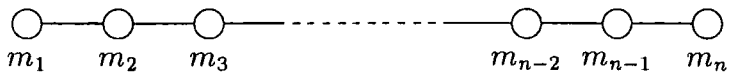
The masses can also be found by using the Perron-Frobenius [21] eigenvector technique, ie The vector

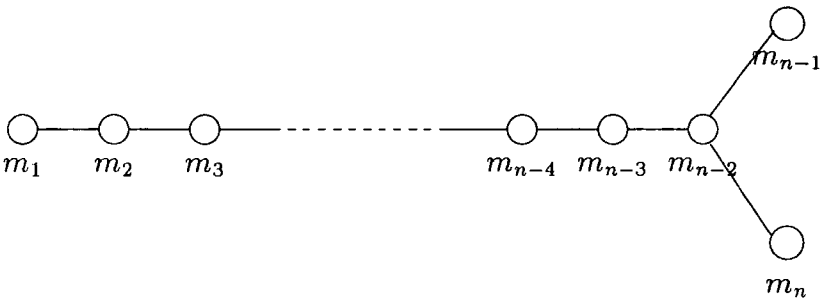
$$\begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{pmatrix} \tag{188}$$

is an eigenvector of the Cartan matrix with its eigenvalue being that of the smallest mass

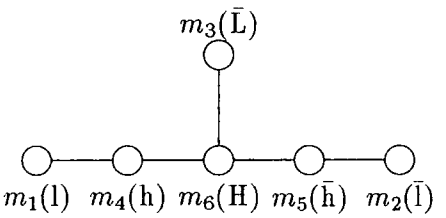
$$4\sin^2(\frac{\pi}{h}) \tag{189}$$

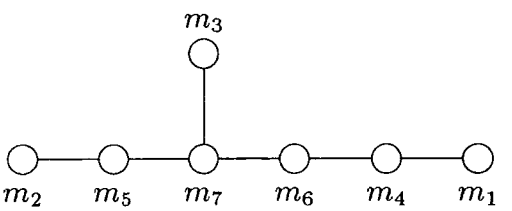
This is also the only eigenvector that has all of its components positive. This allows us to associate the masses with the roots in the Dynkin diagram. eg for the a_n and d_n series:

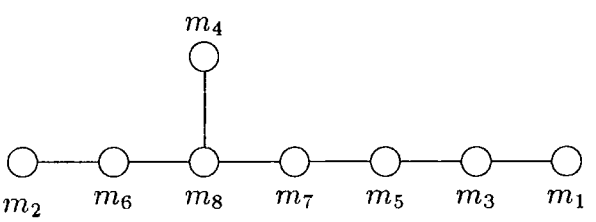




For the e_6, e_7 and e_8 algebras:







This technique works for all the simply laced algebras, but not for the non-simply laced ones.

There is a simple complex representation for the roots devised by [8]. This essentially represents the $\underline{e}_i$ vectors by (partial) complex fourier modes.

$$\alpha_j^a = \frac{1}{\sqrt{n+1}}(e^{\frac{-2\pi i(j+1)a}{n+1}} - e^{\frac{-2\pi ija}{n+1}}) \tag{190}$$

so that

$$C_{ij} = \alpha_i^* \alpha_j \tag{191}$$

is the Cartan matrix. conditioning the n complex (2n real) fields by

$$(\phi^a)^* = \phi^{h-a} \tag{192}$$

then the mass matrix is

$$M^{ab}(\phi^a)^2 = m^2(\alpha_i^a)^*\alpha_i^b(\phi^a)^*\phi^b = m^2 \sin^2(\frac{n\pi a}{h})\delta^{ab} = m_a\delta^{ab} \tag{193}$$

The masses are twofold degenerate (apart from the central one if n is odd), ie $M^{aa} = M^{n-a,n-a}$. By "folding" the group (using the automorphism $\alpha_i \rightarrow \alpha_{n+1-i}$) we have the nonsimply laced algebra, c_n. I have also constructed an equivalent real representation for this group. The vectors are

$$e_i = \begin{pmatrix} \cos\frac{2\pi i}{h} \\ \cos\frac{4\pi i}{h} \\ \vdots \\ \sin\frac{4\pi i}{h} \\ \sin\frac{4\pi i}{h} \\ \vdots \end{pmatrix} \tag{194}$$

with $\alpha_i = e_{i+1} - e_i$ being the same as above. We can show this gives diagonal mass matrix, and also gives the correct Cartan matrix by a small BASIC program with the results listed below:

3.1 Coupling constants

For this algebra the coupling constants are

$$C^{abc} = m^2\beta\sum_i(\alpha_i^a)^*(\alpha_i^b)^*(\alpha_i^c)^* \tag{195}$$

$$= m^2\beta(\omega^a-1)(\omega^b-1)(\omega^c-1) \quad (196)$$

$$= m^2\beta 2im_am_b\sin(\theta^c_{ab}),\ a+b+c \bmod h = 0 \quad (197)$$

$$(198)$$

where

$$\theta^c_{ab} = \frac{(a+b)\pi}{h} \quad (199)$$

There is a simple geometric analogy to this coupling constant, in that its magnitude is proportional to the area of a triangle with sides m_a, m_b and m_c. In fact this is an almost universal for all the affine algebras.

$$\mid C^{abc} \mid = \frac{2\beta}{\sqrt{h}} m_am_b\sin(\theta^c_{ab}) \quad (200)$$

We can construct the d_n roots from the following vectors.

$$\alpha_0 = -(e_1+e_2), \alpha_i = e_i - e_{i+1} \quad (201)$$

$$\alpha_n = e_n + e_{n-1} \quad (202)$$

$$e_1 = (0...0,1,0), e_n = (0....0,1) \quad (203)$$

$$e_i = (l^1_{i-1}, l^2_{i-1}, ..., l^n_{i-1}, 0, 0) \quad (204)$$

$$l^a_i = \sqrt{\frac{2}{n-1}}\sin(\frac{ai\pi}{n-1}) \quad (205)$$

The coupling constants and mass matrices were first calculated by [8] and checked by [28]. The masses and couplings are needed for the calculation, and are too important to be left to human beings. I have therefore written a small BASIC program to check that the above vectors give the correct Cartan matrix, mass matrix, and coupling constants.

The mass matrices are

$$m^2_i = 8m^2\sin^2(\frac{i\pi}{h}) \quad (206)$$

$$m^2_n = m^2_{n-1} = 2m^2 \quad (207)$$

The coupling constants are

$$C^{abc} = \frac{2\beta}{2(n-1)} m_am_b\sin\theta^c_{ab}, a \pm b \pm c = 0 \quad (208)$$

$$C^{abc} = \frac{-2\beta}{2(n-1)} m_a m_b \sin\theta^c_{ab}, a+b+c=h \tag{209}$$

$$C^{abc} = 0 \quad \text{otherwise} \tag{210}$$

For the exceptional algebras:

- e_6

$$m_l^2 = m_{\bar{l}}^2 = (3-\sqrt{3})m^2 \tag{211}$$

$$m_h^2 = m_{\bar{h}}^2 = (3+\sqrt{3})m^2 \tag{212}$$

$$m_L^2 = 2(3-\sqrt{3})m^2 \tag{213}$$

$$m_H^2 = 2(3+\sqrt{3})m^2 \tag{214}$$

- e_7

$$m_1 = 8m^2 \sin^2(\frac{\pi}{9}) \tag{215}$$

$$m_2 = 8\sqrt{3}m^2 \sin^2(\frac{\pi}{18}) \sin^2(\frac{2\pi}{9}) \tag{216}$$

$$m_3 = 8m^2 \sin^2(\frac{2\pi}{9}) \tag{217}$$

$$m_4 = 8\sqrt{3}m^2 \sin^2(\frac{5\pi}{18}) \sin^2(\frac{\pi}{9}) \tag{218}$$

$$m_5 = 8m^2 \sin^2(\frac{3\pi}{9}) \tag{219}$$

$$m_6 = 8m^2 \sin^2(\frac{4\pi}{9}) \tag{220}$$

$$m_7 = 8\sqrt{3}m^2 \sin^2(\frac{7\pi}{18}) \sin^2(\frac{4\pi}{9}) \tag{221}$$

- e_8

$$m_1 = 4\sqrt{3}m^2 \sin(\frac{\pi}{30}) \sin(\frac{\pi}{5}) \tag{222}$$

$$m_2 = 16\sqrt{3}m^2 \sin(\frac{\pi}{30}) \sin(\frac{\pi}{5}) \cos^2(\frac{\pi}{5}) \tag{223}$$

$$m_3 = 16\sqrt{3}m^2 \sin(\frac{\pi}{30}) \sin(\frac{\pi}{5}) \cos^2(\frac{\pi}{30}) \tag{224}$$

$$m_4 = 64\sqrt{3}m^2 \sin(\frac{\pi}{30}) \sin(\frac{\pi}{5}) \cos^2(\frac{\pi}{5}) \cos^2(\frac{7\pi}{30}) \tag{225}$$

$$m_5 = 4\sqrt{3}m^2 \sin(\frac{11\pi}{30}) \sin(\frac{\pi}{5}) \tag{226}$$

$$m_6 = 4\sqrt{3}m^2 \sin(\frac{7\pi}{30}) \sin(\frac{2\pi}{5}) \tag{227}$$

$$m_7 = 4\sqrt{3}m^2 \sin(\frac{13\pi}{30}) \sin(\frac{2\pi}{5}) \tag{228}$$

$$m_8 = 256\sqrt{3}m^2 \sin(\frac{\pi}{30}) \sin(\frac{\pi}{5}) \cos^2(\frac{2\pi}{15}) \cos^4(\frac{\pi}{5}) \tag{229}$$

3.1.1 Coupling constants for e_6

$c^{ll\bar{l}} = -i\frac{2\beta}{\sqrt{h}} m_l m_l \sin\frac{8\pi}{h}$ $c^{llh} = -i\frac{2\beta}{\sqrt{h}} m_l m_l \sin\frac{2\pi}{h}$ $c^{l\bar{l}L} = \frac{2\beta}{\sqrt{h}} m_l m_{\bar{l}} \sin\frac{6\pi}{h}$

$c^{lLl} = \frac{2\beta}{\sqrt{h}} m_l m_L \sin\frac{9\pi}{h}$ $c^{lLh} = \frac{2\beta}{\sqrt{h}} m_l m_L \sin\frac{5\pi}{h}$ $c^{lh\bar{l}} = -i\frac{2\beta}{\sqrt{h}} m_l m_h \sin\frac{11\pi}{h}$

$c^{lh\bar{h}} = -i\frac{2\beta}{\sqrt{h}} m_l m_h \sin\frac{7\pi}{h}$ $c^{l\bar{h}L} = \frac{2\beta}{\sqrt{h}} m_l m_{\bar{h}} \sin\frac{9\pi}{h}$ $c^{l\bar{h}H} = \frac{2\beta}{\sqrt{h}} m_l m_{\bar{h}} \sin\frac{3\pi}{h}$

$c^{lHh} = \frac{2\beta}{\sqrt{h}} m_l m_H \sin\frac{10\pi}{h}$ $c^{l\bar{l}l} = i\frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_{\bar{l}} \sin\frac{8\pi}{h}$ $c^{l\bar{l}h} = i\frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_{\bar{l}} \sin\frac{2\pi}{h}$

$c^{\bar{l}Ll} = \frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_L \sin\frac{9\pi}{h}$ $c^{\bar{l}L\bar{h}} = \frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_L \sin\frac{5\pi}{h}$ $c^{\bar{l}hL} = \frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_h \sin\frac{9\pi}{h}$

$c^{\bar{l}hH} = \frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_h \sin\frac{3\pi}{h}$ $c^{\bar{l}\bar{h}l} = i\frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_{\bar{h}} \sin\frac{7\pi}{h}$ $c^{\bar{l}\bar{h}h} = i\frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_{\bar{h}} \sin\frac{3\pi}{h}$

$c^{\bar{l}Hbarh} = \frac{2\beta}{\sqrt{h}} m_{\bar{l}} m_H \sin\frac{10\pi}{h}$ $c^{LLL} = \frac{2\beta}{\sqrt{h}} m_L m_L \sin\frac{8\pi}{h}$ $c^{LLH} = \frac{2\beta}{\sqrt{h}} m_L m_L \sin\frac{2\pi}{h}$

$c^{Lhl} = \frac{2\beta}{\sqrt{h}} m_L m_h \sin\frac{10\pi}{h}$ $c^{L\bar{h}\bar{l}} = \frac{2\beta}{\sqrt{h}} m_L m_{\bar{h}} \sin\frac{10\pi}{h}$ $c^{LHH} = \frac{2\beta}{\sqrt{h}} m_L m_H \sin\frac{7\pi}{h}$

$c^{hh\bar{l}} = -i\frac{2\beta}{\sqrt{h}} m_h m_h \sin\frac{10\pi}{h}$ $c^{hh\bar{h}} = i\frac{2\beta}{\sqrt{h}} m_h m_h \sin\frac{8\pi}{h}$ $c^{h\bar{h}H} = -\frac{2\beta}{\sqrt{h}} m_h m_{\bar{h}} \sin\frac{6\pi}{h}$

$c^{hHl} = \frac{2\beta}{\sqrt{h}} m_h m_H \sin\frac{11\pi}{h}$ $c^{\bar{h}\bar{h}l} = -i\frac{2\beta}{\sqrt{h}} m_{\bar{h}} m_{\bar{h}} \sin\frac{10\pi}{h}$ $c^{\bar{h}\bar{h}h} = i\frac{2\beta}{\sqrt{h}} m_{\bar{h}} m_{\bar{h}} \sin\frac{8\pi}{h}$

$c^{\bar{h}H\bar{l}} = \frac{2\beta}{\sqrt{h}} m_{\bar{h}} m_H \sin\frac{11\pi}{h}$ $c^{\bar{h}H\bar{h}} = -\frac{2\beta}{\sqrt{h}} m_{\bar{h}} m_H \sin\frac{9\pi}{h}$ $c^{HHL} = -\frac{2\beta}{\sqrt{h}} m_H m_H \sin\frac{10\pi}{h}$

$c^{HHH} = \frac{2\beta}{\sqrt{h}} m_H m_H \sin\frac{8\pi}{h}$

3.1.2 Coupling constants for e_7

$$
\begin{array}{lll}
c^{112} = -1\frac{2\beta}{\sqrt{h}}m_1m_1\sin\frac{10\pi}{h} & c^{114} = -1\frac{2\beta}{\sqrt{h}}m_1m_1\sin\frac{2\pi}{h} & c^{121} = -1\frac{2\beta}{\sqrt{h}}m_1m_2\sin\frac{13\pi}{h} \\
c^{123} = \frac{2\beta}{\sqrt{h}}m_1m_2\sin\frac{7\pi}{h} & c^{132} = \frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{14\pi}{h} & c^{134} = \frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{10\pi}{h} \\
c^{135} = \frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{6\pi}{h} & c^{141} = -1\frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{17\pi}{h} & c^{143} = \frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{11\pi}{h} \\
c^{146} = \frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{3\pi}{h} & c^{153} = \frac{2\beta}{\sqrt{h}}m_1m_5\sin\frac{14\pi}{h} & c^{156} = \frac{2\beta}{\sqrt{h}}m_1m_5\sin\frac{8\pi}{h} \\
c^{164} = \frac{2\beta}{\sqrt{h}}m_1m_6\sin\frac{16\pi}{h} & c^{165} = \frac{2\beta}{\sqrt{h}}m_1m_6\sin\frac{12\pi}{h} & c^{167} = \frac{2\beta}{\sqrt{h}}m_1m_6\sin\frac{4\pi}{h} \\
c^{176} = \frac{2\beta}{\sqrt{h}}m_1m_7\sin\frac{15\pi}{h} & c^{222} = -1\frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{12\pi}{h} & c^{224} = -1\frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{8\pi}{h} \\
c^{225} = -1\frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{2\pi}{h} & c^{231} = \frac{2\beta}{\sqrt{h}}m_2m_3\sin\frac{15\pi}{h} & c^{233} = -1\frac{2\beta}{\sqrt{h}}m_2m_3\sin\frac{11\pi}{h} \\
c^{236} = -1\frac{2\beta}{\sqrt{h}}m_2m_3\sin\frac{5\pi}{h} & c^{242} = -1\frac{2\beta}{\sqrt{h}}m_2m_4\sin\frac{14\pi}{h} & c^{245} = -1\frac{2\beta}{\sqrt{h}}m_2m_4\sin\frac{8\pi}{h} \\
c^{252} = -1\frac{2\beta}{\sqrt{h}}m_2m_5\sin\frac{17\pi}{h} & c^{254} = -1\frac{2\beta}{\sqrt{h}}m_2m_5\sin\frac{13\pi}{h} & c^{257} = -1\frac{2\beta}{\sqrt{h}}m_2m_5\sin\frac{3\pi}{h} \\
c^{263} = -1\frac{2\beta}{\sqrt{h}}m_2m_6\sin\frac{15\pi}{h} & c^{275} = -1\frac{2\beta}{\sqrt{h}}m_2m_7\sin\frac{16\pi}{h} & c^{277} = \frac{2\beta}{\sqrt{h}}m_2m_7\sin\frac{10\pi}{h} \\
c^{332} = -1\frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{14\pi}{h} & c^{337} = -1\frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{2\pi}{h} & c^{341} = \frac{2\beta}{\sqrt{h}}m_3m_4\sin\frac{15\pi}{h} \\
c^{351} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{16\pi}{h} & c^{356} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{10\pi}{h} & c^{362} = -1\frac{2\beta}{\sqrt{h}}m_3m_6\sin\frac{16\pi}{h} \\
c^{365} = \frac{2\beta}{\sqrt{h}}m_3m_6\sin\frac{12\pi}{h} & c^{367} = \frac{2\beta}{\sqrt{h}}m_3m_6\sin\frac{8\pi}{h} & c^{373} = -1\frac{2\beta}{\sqrt{h}}m_3m_7\sin\frac{17\pi}{h} \\
c^{376} = \frac{2\beta}{\sqrt{h}}m_3m_7\sin\frac{13\pi}{h} & c^{444} = -1\frac{2\beta}{\sqrt{h}}m_4m_4\sin\frac{12\pi}{h} & c^{445} = \frac{2\beta}{\sqrt{h}}m_4m_4\sin\frac{10\pi}{h} \\
c^{447} = -1\frac{2\beta}{\sqrt{h}}m_4m_4\sin\frac{4\pi}{h} & c^{452} = -1\frac{2\beta}{\sqrt{h}}m_4m_5\sin\frac{15\pi}{h} & c^{454} = \frac{2\beta}{\sqrt{h}}m_4m_5\sin\frac{13\pi}{h} \\
c^{457} = \frac{2\beta}{\sqrt{h}}m_4m_5\sin\frac{7\pi}{h} & c^{461} = \frac{2\beta}{\sqrt{h}}m_4m_6\sin\frac{17\pi}{h} & c^{466} = \frac{2\beta}{\sqrt{h}}m_4m_6\sin\frac{11\pi}{h} \\
c^{474} = -1\frac{2\beta}{\sqrt{h}}m_4m_7\sin\frac{16\pi}{h} & c^{475} = \frac{2\beta}{\sqrt{h}}m_4m_7\sin\frac{14\pi}{h} & c^{555} = \frac{2\beta}{\sqrt{h}}m_5m_5\sin\frac{12\pi}{h} \\
c^{561} = \frac{2\beta}{\sqrt{h}}m_5m_6\sin\frac{16\pi}{h} & c^{563} = \frac{2\beta}{\sqrt{h}}m_5m_6\sin\frac{14\pi}{h} & c^{572} = -1\frac{2\beta}{\sqrt{h}}m_5m_7\sin\frac{17\pi}{h} \\
c^{574} = \frac{2\beta}{\sqrt{h}}m_5m_7\sin\frac{15\pi}{h} & c^{577} = -1\frac{2\beta}{\sqrt{h}}m_5m_7\sin\frac{11\pi}{h} & c^{664} = \frac{2\beta}{\sqrt{h}}m_6m_6\sin\frac{14\pi}{h} \\
c^{667} = -1\frac{2\beta}{\sqrt{h}}m_6m_6\sin\frac{10\pi}{h} & c^{671} = \frac{2\beta}{\sqrt{h}}m_6m_7\sin\frac{17\pi}{h} & c^{673} = \frac{2\beta}{\sqrt{h}}m_6m_7\sin\frac{15\pi}{h} \\
c^{676} = -1\frac{2\beta}{\sqrt{h}}m_6m_7\sin\frac{13\pi}{h} & c^{772} = \frac{2\beta}{\sqrt{h}}m_7m_7\sin\frac{16\pi}{h} & c^{775} = -1\frac{2\beta}{\sqrt{h}}m_7m_7\sin\frac{14\pi}{h} \\
c^{777} = \frac{2\beta}{\sqrt{h}}m_7m_7\sin\frac{12\pi}{h} &&
\end{array}
$$

3.1.3 Coupling constants for e_8

$$
\begin{array}{lll}
c^{111} = -1\frac{2\beta}{\sqrt{h}}m_1m_1\sin\frac{20\pi}{h} & c^{112} = \frac{2\beta}{\sqrt{h}}m_1m_1\sin\frac{12\pi}{h} & c^{113} = -1\frac{2\beta}{\sqrt{h}}m_1m_1\sin\frac{2\pi}{h} \\
c^{121} = \frac{2\beta}{\sqrt{h}}m_1m_2\sin\frac{24\pi}{h} & c^{122} = -1\frac{2\beta}{\sqrt{h}}m_1m_2\sin\frac{18\pi}{h} & c^{123} = \frac{2\beta}{\sqrt{h}}m_1m_2\sin\frac{14\pi}{h} \\
c^{124} = -1\frac{2\beta}{\sqrt{h}}m_1m_2\sin\frac{8\pi}{h} & c^{131} = -1\frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{29\pi}{h} & c^{132} = \frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{21\pi}{h} \\
c^{134} = \frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{13\pi}{h} & c^{135} = \frac{2\beta}{\sqrt{h}}m_1m_3\sin\frac{3\pi}{h} & c^{142} = -1\frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{25\pi}{h} \\
c^{143} = \frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{21\pi}{h} & c^{144} = -1\frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{17\pi}{h} & c^{145} = -1\frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{11\pi}{h} \\
c^{146} = \frac{2\beta}{\sqrt{h}}m_1m_4\sin\frac{7\pi}{h} & c^{153} = \frac{2\beta}{\sqrt{h}}m_1m_5\sin\frac{28\pi}{h} & c^{154} = -1\frac{2\beta}{\sqrt{h}}m_1m_5\sin\frac{22\pi}{h} \\
c^{156} = \frac{2\beta}{\sqrt{h}}m_1m_5\sin\frac{14\pi}{h} & c^{157} = -1\frac{2\beta}{\sqrt{h}}m_1m_5\sin\frac{4\pi}{h} & c^{164} = \frac{2\beta}{\sqrt{h}}m_1m_6\sin\frac{25\pi}{h} \\
c^{165} = \frac{2\beta}{\sqrt{h}}m_1m_6\sin\frac{19\pi}{h} & c^{167} = \frac{2\beta}{\sqrt{h}}m_1m_6\sin\frac{9\pi}{h} & c^{175} = -1\frac{2\beta}{\sqrt{h}}m_1m_7\sin\frac{27\pi}{h} \\
c^{176} = \frac{2\beta}{\sqrt{h}}m_1m_7\sin\frac{23\pi}{h} & c^{178} = \frac{2\beta}{\sqrt{h}}m_1m_7\sin\frac{5\pi}{h} & c^{187} = \frac{2\beta}{\sqrt{h}}m_1m_8\sin\frac{26\pi}{h} \\
c^{188} = \frac{2\beta}{\sqrt{h}}m_1m_8\sin\frac{16\pi}{h} & c^{221} = -1\frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{24\pi}{h} & c^{222} = \frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{20\pi}{h} \\
c^{224} = \frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{14\pi}{h} & c^{225} = \frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{8\pi}{h} & c^{226} = -1\frac{2\beta}{\sqrt{h}}m_2m_2\sin\frac{2\pi}{h} \\
c^{231} = \frac{2\beta}{\sqrt{h}}m_2m_3\sin\frac{25\pi}{h} & c^{233} = \frac{2\beta}{\sqrt{h}}m_2m_3\sin\frac{19\pi}{h} & c^{236} = \frac{2\beta}{\sqrt{h}}m_2m_3\sin\frac{9\pi}{h} \\
c^{241} = -1\frac{2\beta}{\sqrt{h}}m_2m_4\sin\frac{27\pi}{h} & c^{242} = \frac{2\beta}{\sqrt{h}}m_2m_4\sin\frac{23\pi}{h} & c^{247} = \frac{2\beta}{\sqrt{h}}m_2m_4\sin\frac{5\pi}{h} \\
c^{252} = \frac{2\beta}{\sqrt{h}}m_2m_5\sin\frac{26\pi}{h} & c^{256} = \frac{2\beta}{\sqrt{h}}m_2m_5\sin\frac{16\pi}{h} & c^{262} = -1\frac{2\beta}{\sqrt{h}}m_2m_6\sin\frac{29\pi}{h} \\
c^{263} = \frac{2\beta}{\sqrt{h}}m_2m_6\sin\frac{25\pi}{h} & c^{265} = \frac{2\beta}{\sqrt{h}}m_2m_6\sin\frac{29\pi}{h} & c^{267} = \frac{2\beta}{\sqrt{h}}m_2m_6\sin\frac{13\pi}{h} \\
c^{268} = \frac{2\beta}{\sqrt{h}}m_2m_6\sin\frac{3\pi}{h} & c^{274} = \frac{2\beta}{\sqrt{h}}m_2m_7\sin\frac{27\pi}{h} & c^{276} = \frac{2\beta}{\sqrt{h}}m_2m_7\sin\frac{21\pi}{h} \\
c^{277} = -1\frac{2\beta}{\sqrt{h}}m_2m_7\sin\frac{17\pi}{h} & c^{278} = \frac{2\beta}{\sqrt{h}}m_2m_7\sin\frac{11\pi}{h} & c^{286} = \frac{2\beta}{\sqrt{h}}m_2m_8\sin\frac{28\pi}{h} \\
c^{287} = \frac{2\beta}{\sqrt{h}}m_2m_8\sin\frac{22\pi}{h} & c^{332} = \frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{22\pi}{h} & c^{333} = \frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{20\pi}{h} \\
c^{335} = \frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{14\pi}{h} & c^{336} = \frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{12\pi}{h} & c^{337} = \frac{2\beta}{\sqrt{h}}m_3m_3\sin\frac{4\pi}{h} \\
c^{341} = \frac{2\beta}{\sqrt{h}}m_3m_4\sin\frac{26\pi}{h} & c^{345} = \frac{2\beta}{\sqrt{h}}m_3m_4\sin\frac{16\pi}{h} & c^{351} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{29\pi}{h} \\
c^{353} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{23\pi}{h} & c^{354} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{21\pi}{h} & c^{357} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{13\pi}{h} \\
c^{358} = \frac{2\beta}{\sqrt{h}}m_3m_5\sin\frac{5\pi}{h} & c^{362} = \frac{2\beta}{\sqrt{h}}m_3m_6\sin\frac{26\pi}{h} & c^{363} = \frac{2\beta}{\sqrt{h}}m_3m_6\sin\frac{24\pi}{h}
\end{array}
$$

$$
\begin{array}{lll}
c^{366} = \frac{2\beta}{\sqrt{h}} m_3 m_6 \sin\frac{18\pi}{h} & c^{368} = \frac{2\beta}{\sqrt{h}} m_3 m_6 \sin\frac{8\pi}{h} & c^{373} = \frac{2\beta}{\sqrt{h}} m_3 m_7 \sin\frac{28\pi}{h} \\
c^{375} = \frac{2\beta}{\sqrt{h}} m_3 m_7 \sin\frac{22\pi}{h} & c^{385} = \frac{2\beta}{\sqrt{h}} m_3 m_8 \sin\frac{27\pi}{h} & c^{386} = \frac{2\beta}{\sqrt{h}} m_3 m_8 \sin\frac{25\pi}{h} \\
c^{388} = -1\frac{2\beta}{\sqrt{h}} m_3 m_8 \sin\frac{17\pi}{h} & c^{441} = -1\frac{2\beta}{\sqrt{h}} m_4 m_4 \sin\frac{26\pi}{h} & c^{444} = -1\frac{2\beta}{\sqrt{h}} m_4 m_4 \sin\frac{20\pi}{h} \\
c^{446} = \frac{2\beta}{\sqrt{h}} m_4 m_4 \sin\frac{16\pi}{h} & c^{447} = -1\frac{2\beta}{\sqrt{h}} m_4 m_4 \sin\frac{12\pi}{h} & c^{448} = -1\frac{2\beta}{\sqrt{h}} m_4 m_4 \sin\frac{2\pi}{h} \\
c^{451} = -1\frac{2\beta}{\sqrt{h}} m_4 m_5 \sin\frac{27\pi}{h} & c^{453} = \frac{2\beta}{\sqrt{h}} m_4 m_5 \sin\frac{23\pi}{h} & c^{455} = -1\frac{2\beta}{\sqrt{h}} m_4 m_5 \sin\frac{19\pi}{h} \\
c^{458} = \frac{2\beta}{\sqrt{h}} m_4 m_5 \sin\frac{9\pi}{h} & c^{461} = \frac{2\beta}{\sqrt{h}} m_4 m_6 \sin\frac{28\pi}{h} & c^{464} = \frac{2\beta}{\sqrt{h}} m_4 m_6 \sin\frac{22\pi}{h} \\
c^{472} = \frac{2\beta}{\sqrt{h}} m_4 m_7 \sin\frac{28\pi}{h} & c^{474} = -1\frac{2\beta}{\sqrt{h}} m_4 m_7 \sin\frac{24\pi}{h} & c^{477} = \frac{2\beta}{\sqrt{h}} m_4 m_7 \sin\frac{18\pi}{h} \\
c^{478} = -1\frac{2\beta}{\sqrt{h}} m_4 m_7 \sin\frac{14\pi}{h} & c^{484} = -1\frac{2\beta}{\sqrt{h}} m_4 m_8 \sin\frac{29\pi}{h} & c^{485} = \frac{2\beta}{\sqrt{h}} m_4 m_8 \sin\frac{25\pi}{h} \\
c^{487} = -1\frac{2\beta}{\sqrt{h}} m_4 m_8 \sin\frac{21\pi}{h} & c^{554} = -1\frac{2\beta}{\sqrt{h}} m_5 m_5 \sin\frac{22\pi}{h} & c^{555} = \frac{2\beta}{\sqrt{h}} m_5 m_5 \sin\frac{20\pi}{h} \\
c^{558} = -1\frac{2\beta}{\sqrt{h}} m_5 m_5 \sin\frac{12\pi}{h} & c^{561} = \frac{2\beta}{\sqrt{h}} m_5 m_6 \sin\frac{27\pi}{h} & c^{562} = \frac{2\beta}{\sqrt{h}} m_5 m_6 \sin\frac{25\pi}{h} \\
c^{567} = -1\frac{2\beta}{\sqrt{h}} m_5 m_6 \sin\frac{17\pi}{h} & c^{571} = -1\frac{2\beta}{\sqrt{h}} m_5 m_7 \sin\frac{29\pi}{h} & c^{573} = \frac{2\beta}{\sqrt{h}} m_5 m_7 \sin\frac{25\pi}{h} \\
c^{576} = -1\frac{2\beta}{\sqrt{h}} m_5 m_7 \sin\frac{21\pi}{h} & c^{583} = \frac{2\beta}{\sqrt{h}} m_5 m_8 \sin\frac{28\pi}{h} & c^{584} = \frac{2\beta}{\sqrt{h}} m_5 m_8 \sin\frac{26\pi}{h} \\
c^{585} = -1\frac{2\beta}{\sqrt{h}} m_5 m_8 \sin\frac{24\pi}{h} & c^{588} = -1\frac{2\beta}{\sqrt{h}} m_5 m_8 \sin\frac{18\pi}{h} & c^{663} = \frac{2\beta}{\sqrt{h}} m_6 m_6 \sin\frac{24\pi}{h} \\
c^{666} = -1\frac{2\beta}{\sqrt{h}} m_6 m_6 \sin\frac{20\pi}{h} & c^{668} = -1\frac{2\beta}{\sqrt{h}} m_6 m_6 \sin\frac{14\pi}{h} & c^{671} = \frac{2\beta}{\sqrt{h}} m_6 m_7 \sin\frac{28\pi}{h} \\
c^{672} = \frac{2\beta}{\sqrt{h}} m_6 m_7 \sin\frac{26\pi}{h} & c^{675} = -1\frac{2\beta}{\sqrt{h}} m_6 m_7 \sin\frac{22\pi}{h} & c^{678} = -1\frac{2\beta}{\sqrt{h}} m_6 m_7 \sin\frac{16\pi}{h} \\
c^{682} = \frac{2\beta}{\sqrt{h}} m_6 m_8 \sin\frac{29\pi}{h} & c^{683} = \frac{2\beta}{\sqrt{h}} m_6 m_8 \sin\frac{27\pi}{h} & c^{686} = -1\frac{2\beta}{\sqrt{h}} m_6 m_8 \sin\frac{23\pi}{h} \\
c^{687} = -1\frac{2\beta}{\sqrt{h}} m_6 m_8 \sin\frac{21\pi}{h} & c^{772} = -1\frac{2\beta}{\sqrt{h}} m_7 m_7 \sin\frac{26\pi}{h} & c^{774} = \frac{2\beta}{\sqrt{h}} m_7 m_7 \sin\frac{24\pi}{h} \\
c^{777} = -1\frac{2\beta}{\sqrt{h}} m_7 m_7 \sin\frac{20\pi}{h} & c^{781} = \frac{2\beta}{\sqrt{h}} m_7 m_8 \sin\frac{29\pi}{h} & c^{782} = \frac{2\beta}{\sqrt{h}} m_7 m_8 \sin\frac{27\pi}{h} \\
c^{784} = -1\frac{2\beta}{\sqrt{h}} m_7 m_8 \sin\frac{25\pi}{h} & c^{786} = -1\frac{2\beta}{\sqrt{h}} m_7 m_8 \sin\frac{23\pi}{h} & c^{788} = \frac{2\beta}{\sqrt{h}} m_7 m_8 \sin\frac{19\pi}{h} \\
c^{881} = \frac{2\beta}{\sqrt{h}} m_8 m_8 \sin\frac{28\pi}{h} & c^{883} = -1\frac{2\beta}{\sqrt{h}} m_8 m_8 \sin\frac{26\pi}{h} & c^{885} = -1\frac{2\beta}{\sqrt{h}} m_8 m_8 \sin\frac{24\pi}{h} \\
c^{887} = \frac{2\beta}{\sqrt{h}} m_8 m_8 \sin\frac{22\pi}{h} & c^{888} = \frac{2\beta}{\sqrt{h}} m_8 m_8 \sin\frac{20\pi}{h} &
\end{array}
$$

3.2 Feynman Rules

We will now derive the exact Feynman rules for diagrams in affine Toda Field theory.

3.3 Vertex

Firstly, let us look at the vertex part. The Lagrangian has the form

$$\mathcal{L} = ... - \frac{\beta}{3!}\sum_{abc}\sum_{i}\alpha_i^a\alpha_i^b\alpha_i^c\phi^a\phi^b\phi^c + ... = -\sum_{abc}c_{abc}\phi^a\phi^b\phi^c + ... \qquad (230)$$

In the interaction Hamiltonian, this part is negative what it is in the Lagrangian, ie

$$\mathcal{V}^{(3)} = \frac{1}{3!}\sum_{abc}c_{abc}\phi^a\phi^b\phi^c + ... \qquad (231)$$

$$c_{abc} = m^2\beta\sum_i n_i\alpha_i^a\alpha_i^b\alpha_i^b = \frac{4\beta}{\sqrt{h}}\sigma_{abc} \qquad (232)$$

where σ_{abc} is a phase factor.

In addition, we absorb the $-i$ from the $\frac{(-i)^n}{n!}$ part of the S matrix expansion, and the integration of the exponentials

$$\int d^2x e^{i\sum(k).x} = (2\pi)^2\delta^2(\sum(k)) \qquad (233)$$

If the fields a,b and c are all different, then we have

$$\frac{\beta}{3!}(c_{abc}\phi_a\phi_b\phi_c + c_{acb}\phi_a\phi_c\phi_b + c_{bac}\phi_b\phi_a\phi_c + ...) \qquad (234)$$

since permuting the indices in the coupling constant makes no difference to its numerical value, there are 3! of these terms which cancel the 3! in the denominator. Similarly, if two of the field are the same, the overall factor is $\frac{1}{2}$. If all the fields are the same, then the overall factor is $\frac{1}{6}$. In this case there are 6 ways of contracting the fields, and hence 6 similar Feynman diagrams, which in general cancel. In certain cases, there is a symmetry of the diagram, for example, in the one loop case. The general theory of Feynman rules says that the contribution of such diagrams have to be divided by their symmetry factors. We will come back to this point

later. for now, it is sufficient to state that the vertex is represented by a term

$$C_{abc} = -ic_{abc}(2\pi)^2\delta^2(\sum(k)) = -i\frac{4\beta}{\sqrt{h}}\Delta_{abc}\sigma_{abc} \tag{235}$$

3.4 Propagator

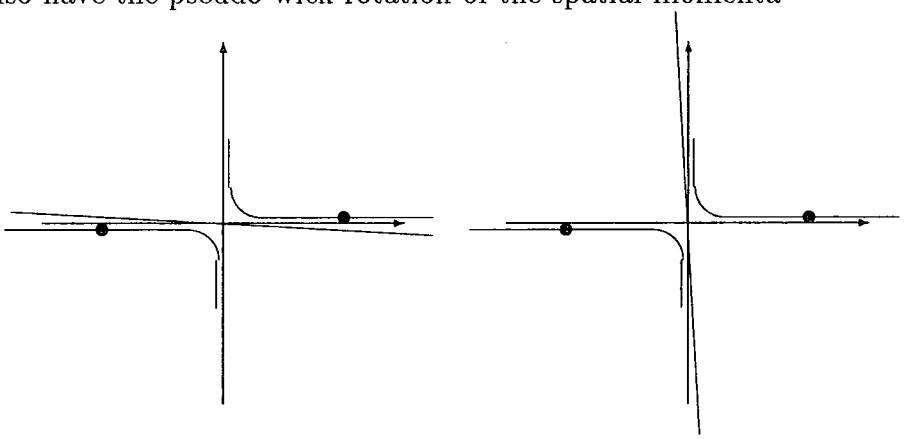
Recall the the contracted fields

$$< T(\phi(x)\phi(y)) >= \int_{-\infty}^{\infty} \frac{d^2k}{(2\pi)^2}\frac{ie^{ik.x}}{(k^2 - m^2 + i\epsilon)} \tag{236}$$

The propagator is represented by

$$\frac{i}{k^2 - m^2 + i\epsilon} \tag{237}$$

we also have the pseudo-wick rotation of the spatial momenta



thus

$$\int_{-\infty}^{+\infty} d^2k \rightarrow \int_{i\infty}^{-i\infty} d^2k = -i\int_{-\infty}^{\infty} d^2k', \quad (k' = ik) \tag{238}$$

giving an extra -i to each propagator. Note that this is a clockwise rotation, as opposed to an anticlockwise rotation for a normal Wick rotation.

This also brings the metric into the Euclidean plane:

$$k.A = k_t\cosh(i\theta) - k_x\sinh(i\theta) = k'_t\cos(\theta) + k'_x\sin(\theta) \tag{239}$$

where $A = (\cosh(i\theta), \sinh(i\theta))$ and $k = (k_t, k_x)$.

3.5 External lines

The contractions of the external creation/annihilation operators

$$< a_k \phi(x) >= e^{-ik.x}, < \phi(x) a_k^\dagger >= e^{ik.x} \tag{240}$$

giving just the usual delta function when integrated over x,

$$(2\pi)^2 \delta^2(\sum(k)) \tag{241}$$

3.6 Loops and β order

Let there be n_3 3 point couplings, n_4 4 point couplings etc, Let there also be P internal lines, and l loops. There are 4 external lines. for each pair of internal lines, we join them by a propagator. Thus there are

$$P = \frac{1}{2}(3n_3 + 4n_4 + 5n_5 + ... - 4) \tag{242}$$

internal lines. Also the order of beta is

$$n = n_3 + 2n_4 + 3n_5 + ... \tag{243}$$

Thus we have

$$S \sim \frac{\beta^n}{(\theta - i\theta_0)^m} \tag{244}$$

Let l be the number if loops, which is given by

$$l = P - (n_3 + n_4 + n_5 + ...) + 1 \tag{245}$$

since each vertex contributes a delta function, which reduces the number of independent variables by one (plus one for the initial loop). Thus

$$m = P - 2l = \frac{1}{2}(n_3 - n_5 - 2n_6 - ....) \tag{246}$$

$$n = n_3 + 2n_4 + 3n_5 + ... \tag{247}$$

Conclusions:

- n_4 does not contribute to m. Adding four point couplings increases the order of the pole without affecting the order of β.
- For a given order of pole, the lowest order of β is made purely out of three point couplings. It is this second observation that we will use in checking the exact S matrices.

3.7 Overall factors

Given that there are only three point couplings, we will further simplify the integration, by multiplying the integral by an overall factor. We also absorb the factor of β into this factor. The actual coupling we use is just

$$m_a m_b \sin\theta^c_{ab} \tag{248}$$

with the factor

$$\frac{2\beta}{\sqrt{h}} \tag{249}$$

taken to the overall factor. The overall factor is

$$s - s_0 = 2m_a m_b(\cosh\theta - \cosh\theta_0) \tag{250}$$

$$= 2m_a m_b \sinh\theta|_{\theta\to i\theta_0}(\theta - i\theta_0) \tag{251}$$

$$= 2m_a m_b i \sin(\theta_0)(\theta - i\theta_0) \tag{252}$$

$$= 4i\Delta(\theta - i\theta_0) \tag{253}$$

3.8 Comparisons with exact S matrix

As we have seen the S matrices are given by the blocks:

$$(x) = \frac{\sinh(\frac{\theta}{2} + \frac{i\pi x}{2h})}{\sinh(\frac{\theta}{2} - \frac{i\pi x}{2h})} \tag{254}$$

and

$$\{x\} = \frac{(x+1)(x-1)}{(x+1-B)(x-1+B)} \tag{255}$$

With the coupling constant dependence given by

$$B = \frac{1}{2\pi} \frac{\beta^2}{1 + \frac{\beta^2}{4\pi}} \tag{256}$$

so that

$$S_{ij} = \prod_i \{i\}^{g_i} \tag{257}$$

where g_i are integers. This leads to the expression above:

$$S_{ij}(\theta)_{leading} = (\frac{\beta}{\sqrt{2h}})^{2p} \frac{\kappa_p}{(\theta - i\theta_0)^p} \tag{258}$$

where $\kappa = \pm i, 1$. This means that there exist poles at purely immaginary values of θ. In general, the product over i is done in steps of 2. If we assume the coupling constant β, and hence B to be small, then poles only occur when an expression in the denominator of S approaches zero, and hence when any of the terms in the sinh's approaches zero.

Let us define

$$\theta_0(x) = \frac{i\pi x}{h} \tag{259}$$

$$\gamma = \frac{i\pi B}{2h} \approx \frac{i\beta^2}{4h} \tag{260}$$

for small β. Then in general

$$\frac{(x+1)}{(x+1-B)} = \frac{\sinh(\frac{\theta + i\theta_0(x+1)}{2}) \sinh(\frac{\theta - i\theta_0(x+1)}{2} + \gamma)}{\sinh(\frac{\theta - i\theta_0(x+1)}{2}) \sinh(\frac{\theta + i\theta_0(x+1)}{2} - \gamma)} \tag{261}$$

Thus we have two poles at $\theta = i\theta_0(x+1)$ and $\theta = -i\theta_0(x+1) + 2\gamma$. Notice that the poles are not symmetric. The pole below the real axis is shifted

by an amount 2γ with respect to the pole above the real axis. We will only however, be considering the positive imaginary poles from now on.

The above term reduces to

$$\frac{\sinh(\frac{\theta-i\theta_0(x+1)}{2}+\gamma)}{\sinh(\frac{\theta-i\theta_0(x+1)}{2})} \approx 1+\frac{\gamma}{\frac{\theta-i\theta_0}{2}} \approx 1+\frac{i\beta^2}{2h(\theta-i\theta_0)} \tag{262}$$

for $\gamma >> \theta - i\theta_0$ and

$$\frac{\sinh(\frac{\theta+i\theta_0}{2}+\gamma)}{\sinh(\frac{\theta+i\theta_0}{2})} \approx 1 \tag{263}$$

for $\gamma << \theta - i\theta_0$ the other two sinh's are 1 from the above.

Thus the block x has two positive imaginary poles (4 poles in total) at $\theta = i\theta_0(x+1)$ and $\theta = i\theta_0(x-1)$, the contribution from the latter being

$$\frac{-i\beta^2}{2h(\theta-i\theta_0)} \tag{264}$$

The full S matrix is a product of the blocks $\{x\}$, and therefore has a pole structure

$$S(\theta \to i\theta_0) = (1+\frac{i\beta^2}{2h(\theta-i\theta_0)})^n \tag{265}$$

for some n. There are also terms like $(\theta - i\theta_0)\beta^2$ in the expansion of the blocks causing higher polynomials in β for a given pole. The form for the S matrix at a pole is

$$S(\theta) = 1+\frac{\beta^2 P^1}{(\theta-i\theta_0)}+\frac{\beta^4 P^2}{(\theta-i\theta_0)^2}+.... \tag{266}$$

$$P^i(\beta) = \sum_{j=0}^{\infty} P^i_j \beta^j \tag{267}$$

where the P^i_j's are arbitrary constants, the index on P is just to distinguish them, and all the θ_0's are constants of the form $\frac{n\pi}{h}$ where n is an integer.. The S matrix is expanded in the poles derived above, up to some maximum order, which is dependent on the S matrix element. This maximum in perturbation theory comes from the three point couplings only and listed in

the tables below[8]. The notation used is $\theta^{o,r}$ where θ is the angle at which the pole occurs, o is the order of the poles, and r is the residue.

Table 1: Poles in d_6

	1	2	3	4	s	s'
1	$2^{1,i}8^{1,-i}$	$1^{1,-i}3^{1,i}$ $7^{1,-i}9^{1,i}$	$1^{1,-i}3^{1,i}$ $7^{1,-i}9^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}$	$4^{1,-i}6^{1,i}$	$4^{1,-i}6^{1,i}$
2	$1^{1,-i}3^{1,i}$ $7^{1,-i}9^{1,i}$	$2^{2,1}4^{1,i}$ $6^{1,-i}8^{2,1}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{2,1}9^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{3,i}8^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}$
3	$1^{1,-i}3^{1,i}$ $7^{1,-i}9^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{2,1}9^{1,i}$	$2^{2,1}4^{3,-i}$ $6^{3,i}8^{2,1}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{3,i}9^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{1,i}$
4	$3^{1,-i}5^{2,1}$ $7^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{3,i}8^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{3,i}9^{1,i}$	$2^{3,-i}4^{4,1}$ $6^{4,1}8^{3,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{2,1}9^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{2,1}9^{1,i}$
5	$4^{1,-i}6^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{2,1}9^{1,i}$	$2^{1,i}4^{1,-i}$ $6^{1,i}8^{1,-i}$	$2^{1,-i}4^{1,i}$ $6^{1,-i}8^{1,i}$
6	$4^{1,-i}6^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{2,1}9^{1,i}$	$2^{1,-i}4^{1,i}$ $6^{1,-i}8^{1,i}$	$2^{1,i}4^{1,-i}$ $6^{1,i}8^{1,-i}$

Table 2: Poles in e_6

	l	$\bar{l}$	L	h	$\bar{h}$	H
l	$2^{1,i}6^{1,-i}$ $8^{1,i}$	$4^{1,-i}6^{1,i}$ $10^{1,-i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}9^{1,-i}$ $11^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$
$\bar{l}$	$4^{1,-i}6^{1,i}$ $10^{1,-i}$	$2^{1,i}6^{1,-i}$ $8^{1,i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}9^{1,-i}$ $11^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$
L	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{1,i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{1,i}$	$2^{1,i}4^{1,-i}$ $6^{2,1}8^{1,i}$ $10^{1,-i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{3,i}$ $9^{2,1}11^{1,i}$
h	$3^{1,-i}5^{2,1}$ $7^{1,i}9^{1,-i}$ $11^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$	$2^{2,1}4^{2,1}$ $6^{3,-i}8^{3,i}$ $10^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{3,i}8^{2,1}$ $10^{2,1}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{4,1}$ $9^{3,i}11^{1,i}$
$\bar{h}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{1,i}9^{1,-i}$ $11^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{3,i}8^{2,1}$ $10^{2,1}$	$2^{2,1}4^{2,1}$ $6^{3,-i}8^{3,i}$ $10^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{4,1}$ $9^{3,i}11^{1,i}$
H	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{2,1}$ $10^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{3,i}$ $9^{2,1}11^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{4,1}$ $9^{3,i}11^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{4,1}$ $9^{3,i}11^{1,i}$	$2^{3,-i}4^{5,-i}$ $6^{6,1}8^{5,i}$ $10^{3,i}$

Table 3: Poles in e_7

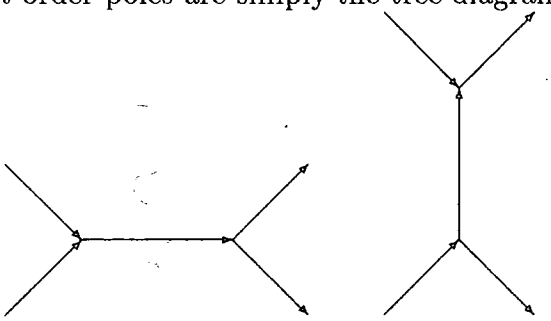
	1	2	3	4	5	6	7
1	$2^{1,i}8^{1,-i}$ $10^{1,i}16^{1,-i}$	$5^{1,-i}7^{1,i}$ $11^{1,-i}13^{1,i}$	$4^{1,-i}8^{1,i}$ $8^{1,-i}10^{1,i}$ $12^{1,-i}14^{1,i}$	$1^{1,-i}3^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{1,i}15^{1,-i}$ $17^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{1,i}10^{1,-i}$ $12^{2,1}14^{1,i}$	$2^{1,-i}4^{1,i}$ $6^{1,-i}8^{2,1}$ $10^{2,1}12^{1,i}$ $14^{1,-i}16^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{1,i}$
2	$5^{1,-i}7^{1,i}$ $11^{1,-i}13^{1,i}$	$2^{1,i}6^{1,-i}$ $8^{1,i}10^{1,-i}$ $12^{1,i}16^{1,-i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{1,i}13^{1,-i}$ $15^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{1,i}10^{1,-i}$ $12^{2,1}14^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{2,1}11^{2,1}$ $13^{1,i}15^{1,-i}$ $17^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{3,i}12^{2,1}$ $14^{2,1}16^{1,i}$
3	$4^{1,-i}6^{1,i}$ $8^{1,-i}10^{1,i}$ $12^{1,-i}14^{1,i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{1,i}13^{1,-i}$ $15^{1,i}$	$2^{1,i}4^{1,-i}$ $6^{2,1}8^{2,1}$ $10^{2,1}12^{2,1}$ $14^{1,i}16^{1,-i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{3,i}12^{2,1}$ $14^{2,1}16^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{3,-i}8^{3,i}$ $10^{3,-i}12^{3,i}$ $14^{2,1}16^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{4,1}$ $9^{4,1}11^{4,1}$ $13^{3,i}15^{2,1}$ $17^{1,i}$
4	$1^{1,-i}3^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{1,i}15^{1,-i}$ $17^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{1,i}10^{1,-i}$ $12^{2,1}14^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{1,i}$	$2^{2,1}4^{1,i}$ $6^{1,-i}8^{3,-i}$ $10^{3,i}12^{1,i}$ $14^{1,-i}16^{2,1}$	$3^{1,-i}5^{3,-i}$ $7^{3,i}9^{2,1}$ $11^{3,-i}13^{3,i}$ $15^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{3,-i}$ $9^{4,1}11^{3,i}$ $13^{2,1}15^{2,1}$ $17^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{4,1}8^{4,1}$ $10^{4,1}12^{4,1}$ $14^{3,i}16^{1,i}$
5	$4^{1,-i}6^{2,1}$ $8^{1,i}10^{1,-i}$ $12^{2,1}14^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{2,1}11^{2,1}$ $13^{1,i}15^{1,-i}$ $17^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{3,i}12^{2,1}$ $14^{2,1}16^{1,i}$	$3^{1,-i}5^{3,-i}$ $7^{3,i}9^{2,1}$ $11^{3,-i}13^{3,i}$ $15^{1,i}$	$2^{2,1}4^{2,1}$ $6^{3,-i}8^{4,1}$ $10^{4,1}12^{3,i}$ $14^{2,1}16^{2,1}$	$2^{1,-i}4^{3,-i}$ $6^{4,1}8^{4,1}$ $10^{4,1}12^{4,1}$ $14^{3,i}16^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{5,-i}$ $9^{6,1}11^{5,i}$ $13^{4,1}15^{3,i}$ $17^{1,i}$
6	$2^{1,-i}4^{1,i}$ $6^{1,-i}8^{2,1}$ $10^{2,1}12^{1,i}$ $14^{1,-i}16^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{3,-i}8^{3,i}$ $10^{3,-i}12^{3,i}$ $14^{2,1}16^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{2,1}7^{3,-i}$ $9^{4,1}11^{3,i}$ $13^{2,1}15^{2,1}$ $17^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{4,1}8^{4,1}$ $10^{4,1}12^{4,1}$ $14^{3,i}16^{1,i}$	$2^{2,1}4^{3,-i}$ $6^{4,1}8^{5,-i}$ $10^{5,i}12^{4,1}$ $14^{3,i}16^{2,1}$	$1^{1,-i}3^{3,-i}$ $5^{5,-i}7^{6,1}$ $9^{6,1}11^{6,1}$ $13^{5,i}15^{3,i}$ $17^{1,i}$
7	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{3,i}12^{2,1}$ $14^{2,1}16^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{4,1}$ $9^{4,1}11^{4,1}$ $13^{3,i}15^{2,1}$ $17^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{4,1}8^{4,1}$ $10^{4,1}12^{4,1}$ $14^{3,i}16^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{5,-i}$ $9^{6,1}11^{5,i}$ $13^{4,1}15^{3,i}$ $17^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{5,-i}7^{6,1}$ $9^{6,1}11^{6,1}$ $13^{5,i}15^{3,i}$ $17^{1,i}$	$2^{3,-i}4^{5,-i}$ $6^{7,-i}8^{8,1}$ $10^{8,1}12^{7,i}$ $14^{5,i}16^{3,i}$

Table 4: Poles in e_8

	1	2	3	4	5	6	7	8
1	$2^{1,i}10^{1,-i}$ $12^{1,i}18^{1,-i}$ $20^{1,i}28^{1,-i}$	$6^{1,-i}8^{1,i}$ $12^{1,-i}14^{1,i}$ $16^{1,-i}18^{1,i}$ $22^{1,-i}24^{1,i}$	$1^{1,-i}3^{1,i}$ $9^{1,-i}11^{2,1}$ $13^{1,i}17^{1,-i}$ $19^{2,1}21^{1,i}$ $27^{1,-i}29^{1,i}$	$5^{1,-i}7^{1,i}$ $9^{1,-i}11^{1,i}$ $13^{1,-i}15^{2,1}$ $17^{1,i}19^{1,-i}$ $21^{1,i}23^{1,-i}$ $25^{1,i}$	$2^{1,-i}4^{1,i}$ $8^{1,-i}10^{2,1}$ $12^{2,1}14^{1,i}$ $16^{1,-i}18^{2,1}$ $20^{2,1}22^{1,i}$ $26^{1,-i}28^{1,i}$	$5^{1,-i}7^{2,1}$ $9^{1,i}11^{1,-i}$ $13^{2,1}15^{2,1}$ $17^{2,1}19^{1,i}$ $21^{1,-i}23^{2,1}$ $25^{1,i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{2,1}17^{2,1}$ $19^{2,1}21^{2,1}$ $23^{1,i}25^{1,-i}$ $27^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{2,1}10^{2,1}$ $12^{2,1}14^{3,-i}$ $16^{3,i}18^{2,1}$ $20^{2,1}22^{2,1}$ $24^{2,1}26^{1,i}$
2	$6^{1,-i}8^{1,i}$ $12^{1,-i}14^{1,i}$ $16^{1,-i}18^{1,i}$ $22^{1,-i}24^{1,i}$	$2^{1,i}6^{1,-i}$ $8^{1,i}10^{1,-i}$ $12^{2,1}14^{1,i}$ $16^{1,-i}18^{2,1}$ $20^{1,i}22^{1,-i}$ $24^{1,i}28^{1,-i}$	$5^{1,-i}7^{2,1}$ $9^{1,i}11^{1,-i}$ $13^{2,1}15^{2,1}$ $17^{2,1}19^{1,i}$ $21^{1,-i}23^{2,1}$ $25^{1,i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{2,1}17^{2,1}$ $19^{2,1}21^{2,1}$ $23^{1,i}25^{1,-i}$ $27^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{2,1}10^{2,1}$ $12^{2,1}14^{3,-i}$ $16^{3,i}18^{2,1}$ $20^{2,1}22^{2,1}$ $24^{2,1}26^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{2,1}11^{3,-i}$ $13^{3,i}15^{2,1}$ $17^{3,-i}19^{3,i}$ $21^{2,1}23^{2,1}$ $25^{1,i}27^{1,-i}$ $29^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{3,-i}$ $11^{3,i}13^{3,-i}$ $15^{4,1}17^{3,i}$ $19^{3,-i}21^{3,i}$ $23^{2,1}25^{2,1}$ $27^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{4,1}12^{4,1}$ $14^{4,1}16^{4,1}$ $18^{4,1}20^{4,1}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{1,i}$
3	$1^{1,-i}3^{1,i}$ $9^{1,-i}11^{2,1}$ $13^{1,i}17^{1,-i}$ $19^{2,1}21^{1,i}$ $27^{1,-i}29^{1,i}$	$5^{1,-i}7^{2,1}$ $9^{1,i}11^{1,-i}$ $13^{2,1}15^{2,1}$ $17^{2,1}19^{1,i}$ $21^{1,-i}23^{2,1}$ $25^{1,i}$	$2^{2,1}4^{1,i}$ $8^{1,-i}10^{3,-i}$ $12^{3,i}14^{1,i}$ $16^{1,-i}18^{3,-i}$ $20^{3,i}22^{1,i}$ $26^{1,-i}28^{2,1}$	$4^{1,-i}6^{2,1}$ $8^{2,1}10^{2,1}$ $12^{2,1}14^{3,-i}$ $16^{3,i}18^{2,1}$ $20^{2,1}22^{2,1}$ $24^{2,1}26^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{1,i}7^{1,-i}$ $9^{3,-i}11^{4,1}$ $13^{3,i}15^{2,1}$ $17^{3,-i}19^{4,1}$ $21^{3,i}23^{1,i}$ $25^{1,-i}27^{2,1}$ $29^{1,i}$	$4^{1,-i}6^{3,-i}$ $8^{3,i}10^{2,1}$ $12^{3,-i}14^{4,1}$ $16^{4,1}18^{3,i}$ $20^{2,1}22^{3,-i}$ $24^{3,i}26^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{4,1}12^{4,1}$ $14^{4,1}16^{4,1}$ $18^{4,1}20^{4,1}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{1,i}$	$3^{1,-i}5^{3,-i}$ $7^{4,1}9^{4,1}$ $11^{4,1}13^{5,-i}$ $15^{6,1}17^{5,i}$ $19^{4,1}21^{4,1}$ $23^{4,1}25^{3,i}$ $27^{1,i}$
4	$5^{1,-i}7^{1,i}$ $9^{1,-i}11^{1,i}$ $13^{1,-i}15^{2,1}$ $17^{1,i}19^{1,-i}$ $21^{1,i}23^{1,-i}$ $25^{1,i}$	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{2,1}17^{2,1}$ $19^{2,1}21^{2,1}$ $23^{1,i}25^{1,-i}$ $27^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{2,1}10^{2,1}$ $12^{2,1}14^{3,-i}$ $16^{3,i}18^{2,1}$ $20^{2,1}22^{2,1}$ $24^{2,1}26^{1,i}$	$2^{1,i}4^{1,-i}$ $6^{2,1}8^{2,1}$ $10^{3,-i}12^{3,i}$ $14^{3,-i}16^{3,i}$ $18^{3,-i}20^{3,i}$ $22^{2,1}24^{2,1}$ $26^{1,i}28^{1,-i}$	$3^{1,-i}5^{2,1}$ $7^{3,-i}9^{3,i}$ $11^{3,-i}13^{4,1}$ $15^{4,1}17^{4,1}$ $19^{3,i}21^{3,-i}$ $23^{3,i}25^{2,1}$ $27^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{4,1}12^{4,1}$ $14^{4,1}16^{4,1}$ $18^{4,1}20^{4,1}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{3,-i}8^{4,1}$ $10^{4,1}12^{5,-i}$ $14^{5,i}16^{5,-i}$ $18^{5,i}20^{4,1}$ $22^{4,1}24^{3,i}$ $26^{2,1}28^{1,i}$ $29^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{4,1}$ $9^{5,-i}11^{6,1}$ $13^{6,1}15^{6,1}$ $17^{6,1}19^{6,1}$ $21^{5,i}23^{4,1}$ $25^{3,i}27^{2,1}$ $29^{1,i}$
5	$2^{1,-i}4^{1,i}$ $8^{1,-i}10^{2,1}$ $12^{2,1}14^{1,i}$ $16^{1,-i}18^{2,1}$ $20^{2,1}22^{1,i}$ $26^{1,-i}28^{1,i}$	$4^{1,-i}6^{2,1}$ $8^{2,1}10^{2,1}$ $12^{2,1}14^{3,-i}$ $16^{3,i}18^{2,1}$ $20^{2,1}22^{2,1}$ $24^{2,1}26^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{1,i}7^{1,-i}$ $9^{3,-i}11^{4,1}$ $13^{3,i}15^{2,1}$ $17^{3,-i}19^{4,1}$ $21^{3,i}23^{1,i}$ $25^{1,-i}27^{2,1}$ $29^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{3,-i}9^{3,i}$ $11^{3,-i}13^{4,1}$ $15^{4,1}17^{4,1}$ $19^{3,i}21^{3,-i}$ $23^{3,i}25^{2,1}$ $27^{1,i}$	$2^{2,1}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{5,-i}12^{5,i}$ $14^{4,1}16^{4,1}$ $18^{5,-i}20^{5,i}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{2,1}$	$3^{1,-i}5^{3,-i}$ $7^{4,1}9^{4,1}$ $11^{4,1}13^{5,-i}$ $15^{6,1}17^{5,i}$ $19^{4,1}21^{4,1}$ $23^{4,1}25^{3,i}$ $27^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{4,1}$ $9^{5,-i}11^{6,1}$ $13^{6,1}15^{6,1}$ $17^{6,1}19^{6,1}$ $21^{5,i}23^{4,1}$ $25^{3,i}27^{2,1}$ $29^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{5,-i}8^{6,1}$ $10^{6,1}12^{7,-i}$ $14^{8,1}16^{8,1}$ $18^{7,i}20^{6,1}$ $22^{6,1}24^{5,i}$ $26^{3,i}28^{1,i}$
6	$5^{1,-i}7^{2,1}$ $9^{1,i}11^{1,-i}$ $13^{2,1}15^{2,1}$ $17^{2,1}19^{1,i}$ $21^{1,-i}23^{2,1}$ $25^{1,i}$	$1^{1,-i}3^{1,i}$ $5^{1,-i}7^{2,1}$ $9^{2,1}11^{3,-i}$ $13^{3,i}15^{2,1}$ $17^{3,-i}19^{3,i}$ $21^{2,1}23^{2,1}$ $25^{1,i}27^{1,-i}$ $29^{1,i}$	$4^{1,-i}6^{3,-i}$ $8^{3,i}10^{2,1}$ $12^{3,-i}14^{4,1}$ $16^{4,1}18^{3,i}$ $20^{2,1}22^{3,-i}$ $24^{3,i}26^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{4,1}12^{4,1}$ $14^{4,1}16^{4,1}$ $18^{4,1}20^{4,1}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{1,i}$	$3^{1,-i}5^{3,-i}$ $7^{4,1}9^{4,1}$ $11^{4,1}13^{5,-i}$ $15^{6,1}17^{5,i}$ $19^{4,1}21^{4,1}$ $23^{4,1}25^{3,i}$ $27^{1,i}$	$2^{2,1}4^{2,1}$ $6^{3,-i}8^{4,1}$ $10^{5,-i}12^{6,1}$ $14^{5,i}16^{5,-i}$ $18^{6,1}20^{5,i}$ $22^{4,1}24^{3,i}$ $26^{2,1}28^{2,1}$	$2^{1,-i}4^{3,-i}$ $6^{4,1}8^{5,-i}$ $10^{6,1}12^{6,1}$ $14^{7,-i}16^{7,i}$ $18^{6,1}20^{6,1}$ $22^{5,i}24^{4,1}$ $26^{3,i}28^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{5,-i}$ $9^{7,-i}11^{8,1}$ $13^{8,1}15^{8,1}$ $17^{8,1}19^{8,1}$ $21^{7,i}23^{5,i}$ $25^{4,1}27^{3,i}$ $29^{1,i}$
7	$3^{1,-i}5^{1,i}$ $7^{1,-i}9^{2,1}$ $11^{2,1}13^{2,1}$ $15^{2,1}17^{2,1}$ $19^{2,1}21^{2,1}$ $23^{1,i}25^{1,-i}$ $27^{1,i}$	$3^{1,-i}5^{2,1}$ $7^{2,1}9^{3,-i}$ $11^{3,i}13^{3,-i}$ $15^{4,1}17^{3,i}$ $19^{3,-i}21^{3,i}$ $23^{2,1}25^{2,1}$ $27^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{4,1}12^{4,1}$ $14^{4,1}16^{4,1}$ $18^{4,1}20^{4,1}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{3,-i}8^{4,1}$ $10^{4,1}12^{5,-i}$ $14^{5,i}16^{5,-i}$ $18^{5,i}20^{4,1}$ $22^{4,1}24^{3,i}$ $26^{2,1}28^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{4,1}$ $9^{5,-i}11^{6,1}$ $13^{6,1}15^{6,1}$ $17^{6,1}19^{6,1}$ $21^{5,i}23^{4,1}$ $25^{3,i}27^{2,1}$ $29^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{4,1}8^{5,-i}$ $10^{6,1}12^{6,1}$ $14^{7,-i}16^{7,i}$ $18^{6,1}20^{6,1}$ $22^{5,i}24^{4,1}$ $26^{3,i}28^{1,i}$	$2^{2,1}4^{3,-i}$ $6^{5,-i}8^{6,1}$ $10^{7,-i}12^{8,1}$ $14^{8,1}16^{8,1}$ $18^{8,1}20^{7,i}$ $22^{6,1}24^{5,i}$ $26^{3,i}28^{2,1}$	$1^{1,-i}3^{4,-1}$ $5^{6,1}7^{7,-i}$ $9^{8,1}11^{9,-i}$ $13^{10,1}15^{10,1}$ $17^{10,1}19^{9,i}$ $21^{8,1}23^{7,i}$ $25^{6,1}27^{4,-1}$ $29^{1,i}$
8	$4^{1,-i}6^{2,1}$ $8^{2,1}10^{2,1}$ $12^{2,1}14^{3,-i}$ $16^{3,i}18^{2,1}$ $20^{2,1}22^{2,1}$ $24^{2,1}26^{1,i}$	$2^{1,-i}4^{2,1}$ $6^{2,1}8^{3,-i}$ $10^{4,1}12^{4,1}$ $14^{4,1}16^{4,1}$ $18^{4,1}20^{4,1}$ $22^{3,i}24^{2,1}$ $26^{2,1}28^{1,i}$	$3^{1,-i}5^{3,-i}$ $7^{4,1}9^{4,1}$ $11^{4,1}13^{5,-i}$ $15^{6,1}17^{5,i}$ $19^{4,1}21^{4,1}$ $23^{4,1}25^{3,i}$ $27^{1,i}$	$1^{1,-i}3^{2,1}$ $5^{3,-i}7^{4,1}$ $9^{5,-i}11^{6,1}$ $13^{6,1}15^{6,1}$ $17^{6,1}19^{6,1}$ $21^{5,i}23^{4,1}$ $25^{3,i}27^{2,1}$ $29^{1,i}$	$2^{1,-i}4^{3,-i}$ $6^{5,-i}8^{6,1}$ $10^{6,1}12^{7,-i}$ $14^{8,1}16^{8,1}$ $18^{7,i}20^{6,1}$ $22^{6,1}24^{5,i}$ $26^{3,i}28^{1,i}$	$1^{1,-i}3^{3,-i}$ $5^{4,1}7^{5,-i}$ $9^{7,-i}11^{8,1}$ $13^{8,1}15^{8,1}$ $17^{8,1}19^{8,1}$ $21^{7,i}23^{5,i}$ $25^{4,1}27^{3,i}$ $29^{1,i}$	$1^{1,-i}3^{4,-1}$ $5^{6,1}7^{7,-i}$ $9^{8,1}11^{9,-i}$ $13^{10,1}15^{10,1}$ $17^{10,1}19^{9,i}$ $21^{8,1}23^{7,i}$ $25^{6,1}27^{4,-1}$ $29^{1,i}$	$2^{3,-i}4^{5,-i}$ $6^{7,-i}8^{9,-i}$ $10^{11,-i}12^{12,1}$ $14^{12,1}16^{12,1}$ $18^{12,1}20^{11,i}$ $22^{9,i}24^{7,i}$ $26^{5,i}28^{3,i}$

3.9 By hand calculation

The 1st and 2nd order poles are simple enough to be calculated directly by hand [9] and are almost a direct result of the fusing rules given earlier. The first order poles are simply the tree diagrams



Depending on the initial particles and angles, only one of these diagrams contribute at any particular time. Examining the dual diagram, one can see that essentially if one diagram contributes to a pole at θ, then the other contributes at $\pi - \theta$.

For each of these trees, the contribution to the S matrix at the pole is easily calculated. For the s channel have the integral

$$\int d^2k \frac{\delta^2(p_a + p_b - k)\delta^2(k - p_c - p_d)}{k^2 - m^2} C_{abc}^2 \tag{268}$$

Where $p_c \rightsquigarrow p_a$ and $p_d \rightsquigarrow p_b$ Integrating the delta function, and using $s = (p_a + p_b)^2$ gives

$$\frac{\delta^2(p_a + p_b - p_c - p_d)}{s - s_0} C_{abc}^2 = \frac{\delta^2(p_a + p_b - p_c - p_d)}{4\Delta(\theta - i\theta_0)} \Delta^2 \frac{\beta^2}{2h} \sigma_{abc} \tag{269}$$

Removing the delta function (to find the Feynman amplitude) and dividing by Δ, for flux compensation gives

$$S = \frac{\beta^2}{2h(s - s_0)} = \frac{i}{\theta - i\theta_0} (\frac{\beta^2}{2h}) \tag{270}$$

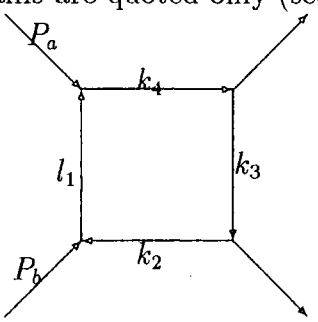
For the t channel pole, we note that

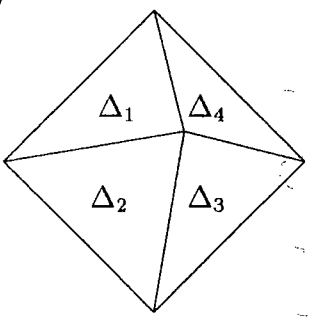
$$t = (p_a - p_b)^2 = m_a^2 + m_b^2 - 2m_a m_b \cos(\theta) \tag{271}$$

$$t - t_0 = -(s - s_0) = -4\Delta(\theta - i\theta_0) \tag{272}$$

The calculation is almost exactly the same, only with $t - t_0$ pole, and hence is minus the result above. Also if the s pole occurs at θ_0, then the t pole occurs at $2\pi - \theta$, as would be expected from the exact S matrices.

We will now calculate by hand the box diagram only. The crossed box diagrams are quoted only (see [9])





Impressively, we can write the expression purely in terms of the area's of the triangles in the dual diagram. The particular route to take depends on the relative sizes of these areas, which is why we have not implemented this method in the automated version. using the notation that

$$\Delta_1 = \Delta_{41a}, \Delta_2 = \Delta_{12b}, \Delta_3 = \Delta_{23a}, \Delta_4 = \Delta_{34b} \tag{273}$$

At the pole configuration, the momenta are

$$k_1 = \frac{\Delta_2 p_a - \Delta_1 p_b}{\Delta} \tag{274}$$

$$k_2 = \frac{\Delta_2 p_a + \Delta_3 p_b}{\Delta} \tag{275}$$

$$k_3 = \frac{-\Delta_4 p_a + \Delta_3 p_b}{\Delta} \tag{276}$$

$$k_4 = \frac{-\Delta_4 p_a - \Delta_1 p_b}{\Delta} \tag{277}$$

and the terms $\epsilon_i = a_i b_i$ are

$$\epsilon_1 = -\frac{\Delta_2 \Delta_1}{\Delta^2} \tag{278}$$

$$\epsilon_2 = \frac{\Delta_2 \Delta_3}{\Delta^2} \tag{279}$$

$$\epsilon_3 = -\frac{\Delta_4\Delta_3}{\Delta^2} \tag{280}$$

$$\epsilon_4 = \frac{\Delta_4\Delta_1}{\Delta^2} \tag{281}$$

It is convenient to integrate over the variables $l_a = l.p_a$ and $l_b = l.p_b$ with jacobian $\frac{1}{2\Delta}$. However, with a further change of variables

$$u = 2l.k_4, v = 2l.k_1 \tag{282}$$

then the jacobian is

$$\frac{4}{\Delta^2}\begin{Vmatrix} -\Delta_4 & -\Delta_1 \\ \Delta_2 & -\Delta_1 \end{Vmatrix} = \frac{4\Delta_1}{\Delta} \tag{283}$$

and the integral reads

$$\int \frac{dudv}{\left[\begin{array}{c}(u+\epsilon_1+i\epsilon)(v+\epsilon_1+i\epsilon) \\ (\frac{-\Delta_2}{\Delta_1}u + \frac{\Delta_1\Delta_2-\Delta_3\Delta_4}{\Delta_1\Delta}v + \epsilon_1 + i\epsilon)(\frac{\Delta_1\Delta_4-\Delta_2\Delta_3}{\Delta 1\Delta}u - \frac{\Delta_4}{\Delta_1}v + \epsilon_1 + i\epsilon)\end{array}\right]} \tag{284}$$

Providing that both the coefficient of u in the second term, and the coefficient of v in the third term are less then zero, ie

$$\Delta_1\Delta_2 < \Delta_3\Delta_4 \qquad \Delta_1\Delta_4 < \Delta_2\Delta_3 \tag{285}$$

The original integral is simply the residuals at the poles ϵ_4 and ϵ_1,

$$I = \frac{-(2\pi i)^2\Delta^2}{8\Delta_1\Delta_2\Delta_4} \tag{286}$$

If the conditions (285) are not met, then two of the poles are on the same side of the real axis, and the integral is zero. Both of the above situations can occur. The program generates exact numerical factors for the areas of the triangles, and the situation is resolved. The crossed box diagrams have the form

$$R_2 = \frac{\Delta_1\Delta_2 - \Delta_3\Delta_4}{\Delta^2}\sigma_1^2\sigma_2^2 \tag{287}$$

$$R_2 = \frac{\Delta_1\Delta_4 - \Delta_2\Delta_3}{\Delta^2}\sigma_1^2\sigma_4^2 \tag{288}$$

It is interesting to note, that the sum of these, providing that all the phase factors are 1 gives the correct value for the S matrix.

This can also be checked by the program. We have to give all the data to do a particular calculation, ie the group, the incoming momenta, and the scattering angle. The above method can be similarly applied to the third order poles, although the calculation requires some effort. This has been done in [9] and the results again agree with the exact S matrices. All of these calculations make assumptions like (285) and the tiling hypothosis, so only a detailed examination would yield an exact result.

4 The Automated Generation of Graphs

In this chapter we shall discus how the Feynman graphs at singularity were automatically generated. In the first section we will describe how the theory of automated theorem proving can be applied to this problem. This will include some background on classical logic and the aim of many people to produce some automated way of solving problems of all sorts in general. We will then go on to discus the programs that used this method to produce graphs.

4.1 Generating graphs by logic

We will first describe the procedure for generating graphs by the technique of logic programming. The first program to do this is called *pattj3*, and is listed in full in appendix A. It is written in the logic programming language PROLOG, which stands for PROgramming in LOgic. For reasons of speed this program was later translated into a modern imperative language (one that is procedural in nature), ie C++. A logic programming scheme was thought the best way to start a graph generation procedure, since the couplings and conservation laws place restrictions on the total set of allowable diagrams and these restrictions can be easily written down in such a language. To illustrate this compare the sizes of the two relevant programs *pattj3* in appendix A and *cth3.c* in appendix B.

As an introduction to the techniques of logic programming I will now review the standard form of logic and some of its historical and philosophical background.

4.2 The Predicate Calculus

There are many different types of logic. The first type of logic is called propositional logic. This has the fundamental clauses of disjunction, conjunction, subjunction or implication, and negation, and any term in the logic can only be true or false.

Propositional logic is generally very limited in what it can describe. The universe of discourse is finite, and there are no quantifiers. For this reason predicate calculus was invented.

One could take the view that as propositional logic is a symbolic Boolean logic, describing practical computing machines, then it is very general in what it can describe. Unfortunately practical machines (eg Meanly-Moore machines) are dynamic systems whereas propositional logic is a system in which static sentences are proven or disproved and the connection between the two is tenuous. The connection between computability theory and logical theorem proving is greater in the more descriptive logics such a predicate calculus.

In first order predicate calculus, the universe of discourse is made up from n-place functions that act on themselves to produce function sentences. This includes 0 place functions which are called names. In second order predicate calculus, the domain of discourse is extended to include all well formed formulas or wff's, and we can quantify over them just as we quantify over functional formulas in the first order predicate calculus.

4.3 Interpretations and Models

In logic, there is a very clear formalism that relates the logic to its interpretation. In any logic we speak of a material value, or valuation of a wff. Logic is meant to represent the real world, and the connection between the logic and the real world is a model (A model is thus an interpretation in which all the axioms are valid). The model is not necessarily unique, for instance the zero and one place functions; 0 and S in Peano's arithmetic are meant to describe the set of natural numbers. Any axiom in Peano's arithmetic, or extension should be true in the model that represents what we understand by arithmetic. This is however a matter of judgment because our model is subjective. We may for example allow terms to commute, which would be correct for the arithmetic of natural numbers. However, the same formal axiom would be invalid in the interpretation describing Lie algebras. We indicate that the wff $\boldsymbol{A}$ is valid in the model $\mathcal{M}$ by using the notation $\mathcal{M} \models \boldsymbol{A}$. The law of excluded middle holds for most, classical logics, that is either $\models \boldsymbol{A}$ or $\not\models \boldsymbol{A}$. This is important when we consider the soundness and completeness of a system.

4.4 Other Logics

There has been a plethora of new logics in recent years, mainly as a result of the use of logic in artificial intelligence. Previously logic was only a philosophical domain. The new logics try and emulate the real world as closely as possible. A small set of the more important logics is presented below

- Bayesian logic models the uncertainty of information present in real world data sources. The logic, as its name would suggest deals with probability. Rules of inference have attached to them a likelihood of that inference being true, stored in general as the log of its probability. A deduction chain that leads to a conclusion therefore also has an attached probability of being true. Generally one likes to see the first few most probable inferences to make a decision. Examples of inference engines that use Bayesian logic include the successful Mycin [13] program for diagnosing diseases.

- Fuzzy logic also models the uncertainty of information that exists in real situations. The rules of fuzzy logic are predictable, but the information that they work on is fuzzy sets. In conventional (finite) crisp sets the set is made up of a sequence of elements, and each element is either in or out of the set. For each element x of a fuzzy set there is associated with it a set membership function $\mu(x)$ which indicates the extent to which it is in the set. Fuzzy logic in particular has become extremely useful in dealing with the real world to the extent that new microprocessors such as the NLX230 have been invented to manipulate fuzzy rules, and wide ranging applications in consumer electronics have been found.

- Quantum logic, first invented by Von-neumann and Birkhoff [23], is the logic of quantum systems. Quantum systems can exist in a superposition of states, and measurement operators can be non-commuting with different sets of eigenvalue states. Predicates represent the truth value of measurements, and conjunction and disjunction of predicates are non-commutative, as one would expect from the non-commutivity of measurement operators. For example if A is true if an electron is spin up when measured by σ_z, and B is true if an electron is spin up

when measured by σ_y. If A and B are operators acting on the Hilbert space of electron spin, then

$$\text{trace}(R(A \wedge B)) \neq \text{trace}(R(B \wedge A)) \tag{289}$$

where R is a suitable preparation operator.

- Model logic [49] [32] introduces the additional symbols necessity $\Box$ and possibility $\Diamond$. Model logic makes a close association with the model of the logic, hence its name. A Kripke model [35] represents many different subsets of valuations. Each of these, in model logic represents a different "world". Subsets of valuations represent specialisations of worlds, and supersets generalisations. Thus a proposition that is possibly true in one sub-model can be made necessarily true in its specialisation.

4.5 Clausal Logic

We now turn our attention to a very important logic that has widely been used for reasoning, that of clausal logic. The system of clausal logic presented here is call the Horn logic, developed by Horn [31]. The actual logic of deductive reasoning used by most logic based inference systems is clausal logic. PROLOG is a language that is based on this logic, and is one implimentation of an abstract theorem prooving machine called the Warren Abstract Machine[53] or WAM for short. There is a distinction to be made between forward and backward inference. The usual forward, or direct deductive process in predicate calculus is one of forward inference whereby to prove a formula $\vdash A$, one takes instances of the hypothesis and axiom schemas and use the deduction rules in order to try and obtain $\vdash A$. The deduction rules generally consist only of *generalisation* (GEN) and *modus ponens* (MP). This is a very difficult thing to do even by hand since there are many ways of deriving new formulas from the axioms and hypothesis, with no guidance on whether or not we are getting close to our goal of proving A. Clausal logic, and in fact most "by hand" methods work by backward inference. By the deduction theorem, the goal formula is unified with the axioms or hypothesis to generate a new goal states which one then tries to prove. That way we are in some way guided in the inferences that are made.

In clausal logic formulas are written down as the universal closure of all of their free variables. If a goal predicate contains a free variable, then we want to know all of the instances of that predicate for which it is valid. The equivalent in classical logic would be to enumerate all instances of a predicate and to find the set for which they are true.

In our case for the generation of all possible diagrams we define a set of predicates such that the instances of some base predicate contain information on the diagram (ie number of nodes, links, masses etc). That is $\boldsymbol{D}(\boldsymbol{X})$ is satisfied if and only if $\boldsymbol{X}$ is one of the on shell Feynman diagrams allowed by the coupling and conservation rules.

The process of generating instances of predicates is called instantiation and the way in which formulas are written down lends itself naturally to backward inference, for example

$$p(X, Y, c1) \subset q(X) \wedge (Y = c2) \tag{290}$$

in clausal notation, or in PROLOG notation

$$p(X, Y, c1) : -q(X), Y = c2 \tag{291}$$

The above sentences mean that $p(X, Y, c1)$ is implied by the conjunction of $q(X)$ and $Y = c2$ where c1 and c2 are both constant names.

The predicate $p(X, Y, Z)$ is valid iff Z is bound to c1, Y is bound to c2, and X is bound to some constant such that q(X) is also valid. Note that variables are always in upper case in PROLOG, and the equals sign is itself a predicate of two variables.

To prove $p(X, Y, Z)$, one has to try and simultaneously satisfy $q(X)$, $Y = c2$ and $Z = c1$. The last two conditions are easily satisfied, and the theorem prover would now try and satisfy $\vdash q(x)$ for some x.

All clauses in Horn logic are written down in this consequent-antecedent form:

$$A(X) \vee B(X) \vee C(X).... \subset P(X) \wedge Q(X) \wedge .... \tag{292}$$

Which states that either A or B etc is true if P,Q,R etc are all true simultaniously. In PROLOG the terms in the anticeedent have to be written on new lines. To prove A or B, we need to prove that P and Q etc can be simultaneously satisfied. It can be shown that there is an axiomatic way of proving any wff in this logic (as there is in predicate calculus). This was

first proved by A.Church [15] and is known as Church's theorem, and the method is often known as canonical derivation In essence **the problem of solving problems has been solved** .

4.6 Completeness of Resolution

The counter argument to the claim that theorem proving can solve any problem is the well known, and often cited (albeit misleadingly) Gödels first and second theorems. This states that any formal system (and arithmetic in particular) is incomplete. That is there are wff that are true but are not theorems. By true we mean $\boldsymbol{N} \models \boldsymbol{P}$, and by a theorem we mean $\vdash \boldsymbol{P}$. In essence this amounts to saying there are wff that are neither true nor false, but undecidable. We shall explain this in more detail later. The system that we use to prove Gödels theorems is recursive arithmetic, or RA for short. This is an extension to first order predicate calculus with additional axioms that deal with numbers and functions. Virtually all mathematical functions can be defined in RA and in addition, because it includes recursion and composition we can define functions that could do any calculation that a Turing machine can do. In other words it can be viewed as a sort of general programming language. In particular we could write a theorem prover. Let us define this theorem prover as $\boldsymbol{Th}(\text{"}\boldsymbol{A}\text{"})$ which would be true iff $\boldsymbol{A}$ is true. [5] Here we denote the Gödel number of a wff by quoting it, ie "$\boldsymbol{A}$" is the Gödel number of the wff $\boldsymbol{A}$. To be more specific it is the representation of that number in RA, but we need not concern ourselves with the difference. By Gödel numbering we have in essence a way of doing our metalogical manipulation in the logic itself

We can also define a wff that takes (the Gödel number of) a function of one variable, and produces a (Gödel number of) a function of the Gödel number of that function. Let this be ***sub*** (for substitution). Let

$$\boldsymbol{I} \rightsquigarrow \boldsymbol{sub}(\text{"}\boldsymbol{x}\text{"}, \boldsymbol{x}, \boldsymbol{x}) \tag{293}$$

where $\boldsymbol{sub}(\text{"}\boldsymbol{x}\text{"}, \boldsymbol{r}, \text{"}\boldsymbol{A}\text{"}) = \text{"}\boldsymbol{S}_r^x\boldsymbol{A}|\text{"}$, and $\boldsymbol{S}_r^x\boldsymbol{A}|$ means replacing the free occurrence of $\boldsymbol{x}$ for $\boldsymbol{r}$ in $\boldsymbol{A}$. In other words, I is a wff of one variable that replaces the free variable $\boldsymbol{x}$ in any wff $\boldsymbol{x} = \text{"}\boldsymbol{A}\text{"}$ by its own Gödel number

[5] In essence PROLOG is a theorem prover. In other words PROLOG **is** $\boldsymbol{Th}()$, yet we now prove it can not exist!

"A". If $J \rightsquigarrow I(\text{"}I\text{"})$, then we can prove a contradiction. If $\vdash J$ then $\neg \boldsymbol{Th}(\boldsymbol{sub}(\text{"}x\text{"}, \text{"}I\text{"}, \text{"}I\text{"}))$, but this means $\vdash \neg \boldsymbol{Th}(J) \quad \vdash \neg J$, which is a contradiction. In other words $J \rightsquigarrow \neg \boldsymbol{Th}(\text{"}J\text{"}^*)$ where $\text{"}J\text{"}^* = \text{"}J\text{"}$ except that it is calculated from I rather that quoted explicitly. This is obviously required by considering its length, so that $\text{"}J\text{"} > \text{"}J^*\text{"}$. This is the proof of Gödels first theorem. His second states that

$$\neg \boldsymbol{Th}(\text{"}\bot\text{"}) \tag{294}$$

($\bot$ means false) is not provable in RA. Neither is its converse of course. The above sentence intuitively means that RA is consistent and therefore ought to be, and in the RA metalanguage, is, true. In actual fact Gödels first theorem itself can be proved in RA, that is:

$$\vdash \neg \boldsymbol{Th}(\text{"}\bot\text{"}) \supset \neg \boldsymbol{Th}(\text{"}J\text{"}) \tag{295}$$

In essence Gödels second theorem is similar to Epimenides statement that "All Cretans are liars", and is a paradox of minor importance. A theorem prover (like PROLOG) could attempt to prove this, and end up in a vicious circle, and would eventually run out of memory.

```
| ?- consult(user).
p(X) :- not p(X).
yes
| ?- p(Y).
Heap overflow
```

The above program demonstrates how a contradictory predicate cannot be resolved, but PROLOG has a good go a trying to prove it before running out of system memory [6].

Gödels two theorems of incompleteness are summarised in a more general way by Tarski's theorem [51], which states that "No consistent formal language can be used to adequately formulate a theory of truth which applies

[6] Actually, we have added a axiom that makes our rule base inconsistent, but it demonstrates the point.

to itself", or more simply **"No consistent formal language is adequate for its own semantics"**.

One common misinterpretation [45] of Gödel and Tarski's theorems is that theorem proving "doesn't work", and that real theorem proving can only be accomplished by human intuition. There are a number of objections to this.

The first proof of Gödels theorem relied on a weaker definition of consistency known as ω consistency. This was subsequently removed by Rossers extension, but it would be useful to look at the definition of ω consistency anyway. It states that a system is ω inconsistent iff $\vdash \neg\exists x A$ and $\vdash S_x^t A$ for some A and each $t \in \{0, 1, 2, 3, 4...\}$ From this we deduce that the inconsistency only manifests itself when t is essentially infinite. In fact probably the strongest objection is that all proofs are finite. To produce an infinite proof would require an infinite amount of paper. Secondly, a corollary to tarski's theorem is that for finite n we can produce a predicate that defines the semantics of any wff with at most n quantifiers. This predicate itself has more than n quantifiers though. Thirdly, we can extend the logic itself. By for example producing a second order logic for RA (RA2) Tarski's function can be produced. The main problem with stating that there are true sentences which are not theorems lie in converting the meta language to the object language, and embedding both in an extended logic circumvents all of the above problems.

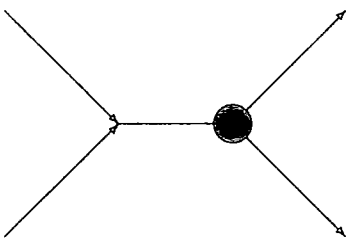
I hope the above arguments have illustrated the importance and universal applicability of the science of theorem proving. There just 2 problems left. Firstly formalising the system, ie converting some non-rigorous language such as english into a formal set of axioms. This is precisely what we have done in writing our programs. Secondly running the theorem prover. Even using backward inference the problem is still NP. Although (classical) computers are exponentially increasing in power, there is a limit to there capabilities. One possible solution may lie with quantum computing. Deutch [16] has shown that a quantum computer can takle this kind of problem much faster than a classical computer. Consider a classical computer that determines the path that it branches by a quantum event. In Everett's interpretation at the end of a calculation there exist many different universes, each one with a computer that has gone down a different branch of its non deterministic algorithm. If one could "deamplify" the total system and find out all the eventual states one has solved the problem in a very short time.

This "deamplification" is certainly theoretically possible [1], but unfortunately the operator needed is very non-classical, ie one who's eigenvectors are in strange states (relative to standard operators such as position, spin etc).

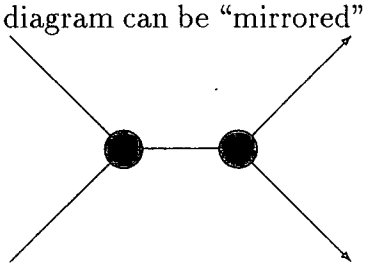
One final point. It must be pointed out though that because of memory constraints most versions of PROLOG work on a depth first search strategy. This can lead down infinite recursive paths. The satisfactory method of inference is by breadth first search.

4.7 The Generation Algorithm

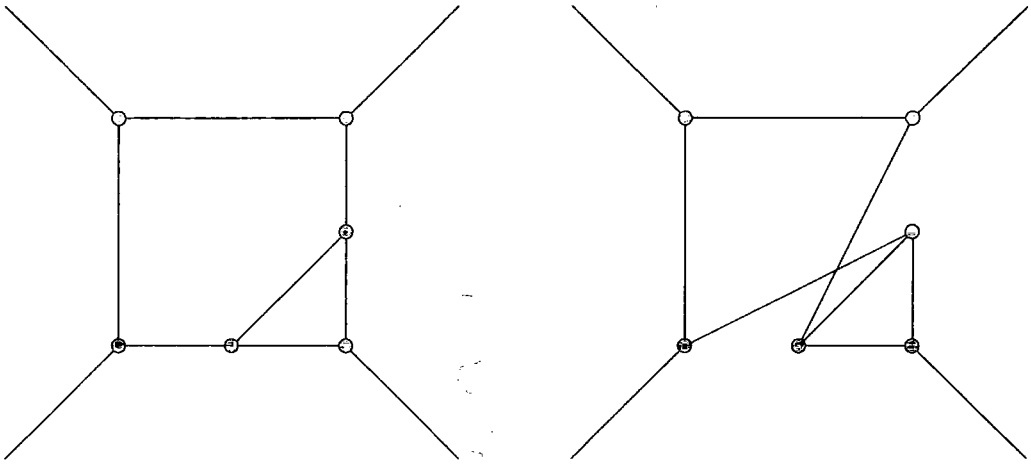
Our problem can be broken down into 4 main parts. Firstly the production of diagrams. These are simply geometric objects made up of nodes, and links linking them. For second order poles there are 4 nodes and 4 internal links, for 3rd order there are 6 nodes and 7 links, for 4th there are 8nodes and 10 links etc. The first four nodes are linked to the external lines. This means we eliminate diagrams whereby two external lines connect to one node:



The reason for this exclusion is that the blob, representing a more complicated diagram can be "mirrored", thus producing a pole of a higher order.



The program then places links from the first available node (one that has less than 3 links associated with it) to one of the other nodes, providing this is less than the "end node". If we link to the end node, the end node pointer is incremented by one. This partially eliminates the overproduction of diagrams that are similar under a permutation of the nodes. It does not completely eliminate them though. For example, the following two diagrams would be independently produced:



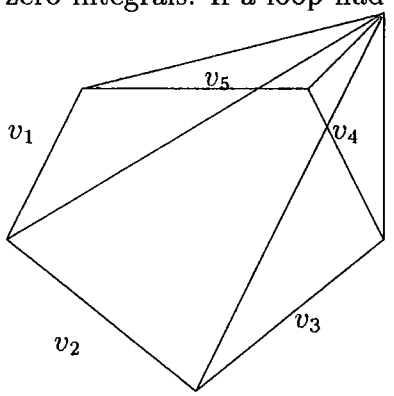
To make sure that there is no overproduction, the diagram, together with diagram's produced by permuting the nodes is stored.

4.8 Association of masses

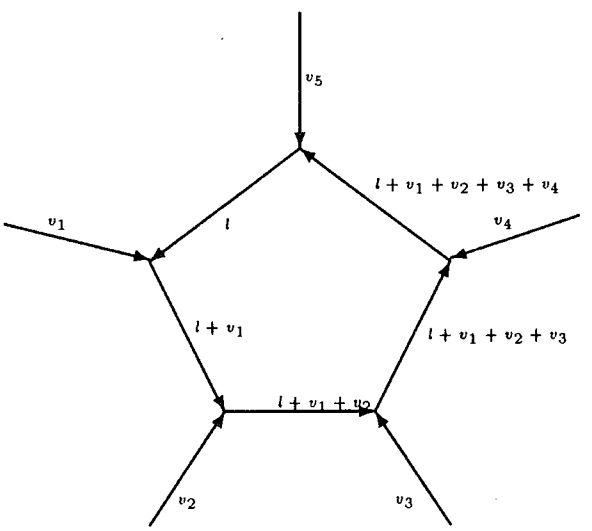
The next part is for each diagram to associate masses, or rather the particular fields to each link. This is where resolution is particularly appropriate, as the masses at each vertex must satisfy the coupling constant conditions (ie only the diagrams with non zero coupling constants at each vertex need be calculated). Finally, in a similar way angles are associated to each link. Branching of the Gentenzen tree only occurs for one instantiated variable in tree point coupling. If two variables are instantiated, then by momentum conservation, there is only one possibility for the mass. If the mass is degenerate, there may be more, but in the theories we study there is at most only one possibility (there could be none at all, in which case the predicate fails and the branch is truncated). The case of all three variables being uninstantiated does not occur (for connected diagrams. The number of branches for the masses is large, and for the angles is infinite, allthough since our representation expressed angles in units of $\frac{n\pi}{h}$ there are at most $(2h)^3$ possible instantaitions. The new C++ program does a prescan on the linklist, and finds a (perhaps not optimised) route to keep the number of instantiated variables in predicates to a maximum.

4.9 Integrability

We finally check for an integrability condition (the integral is zero if this test fails, and non-zero if it succeeds). It was noticed that only "tiled" diagrams had non-zero integrals. If a loop had a dual diagram like:



then this integral would be zero. The proof is simple. For any sub diagram, we can choose a set of integration variables as shown:



where l is the loop integration variable, and

$$\sum v_i = 0 \tag{296}$$

Considering the l part of the integral, we have

$$\int \prod \frac{dl}{2(l + \sum v_i).C_j + \epsilon_i + m_i - i\epsilon} \tag{297}$$

If we can then choose a basis for $l = (l_0, l_1)$ such that $\forall j : \text{sign}(l_0.C_j) = \text{const}$ then the integral is 0.

4.10 Graph Generation: The PROLOG Version

The main predicate that produces the diagrams is called *main3/0.* This in turn calls em main2/4, as many times as is necessary with the last predicate *trig2/1* in the sequent failing until a complete cycle has been done. The predicate *main2/4* returns with its arguments as the node,link,mass and angle lists. The node list, and external momenta (first four terms of the mass and angle lists) are instantiated in this predicate. It subsequently calls the following predicates:

- *linkfrom/4*, to instantiate the diagrams topology
- *assocmass/3*, to instantiate the internal masses
- *checkang/4* to instantiate the internal angles.

We next investigate each of these predicates in turn:

Linkfrom is the predicate that generates the topology of the diagram. Its four arguments are a list containing the nodes to be linked, a list of the number of links that can be made to each node (2 for external nodes, 3 for the rest), an initially uninstantiated list of links, and a number of links to be completed. We shall demonstrate the operation of this predicate on second and third order diagrams:

```
| ?- linkfrom([e1,e2,e3,e4],[2,2,2,2],R,4).

R = [[e1,e3],[e1,e4],[e2,e3],[e2,e4]];
```

```
R = [[e1,e2],[e1,e4],[e2,e3],[e3,e4]];

R = [[e1,e2],[e1,e3],[e2,e4],[e3,e4]];
no
| ?-

| ?- linkfrom([e1,e2,e3,e4,1,2],[2,2,2,2,3,3],R,7).

R = [[e1,1],[e1,2],[e2,1],[e2,2],[e3,e4],[e3,2],[e4,1]];

R = [[e1,1],[e1,2],[e2,1],[e2,2],[e3,e4],[e3,1],[e4,2]];

R = [[e1,1],[e1,2],[e2,e4],[e2,2],[e3,1],[e3,2],[e4,1]];

R = [[e1,1],[e1,2],[e2,e4],[e2,2],[e3,e4],[e3,1],[1,2]];

R = [[e1,1],[e1,2],[e2,e4],[e2,1],[e3,1],[e3,2],[e4,2]];

R = [[e1,1],[e1,2],[e2,e4],[e2,1],[e3,e4],[e3,2],[1,2]];

R = [[e1,1],[e1,2],[e2,e3],[e2,2],[e3,1],[e4,1],[e4,2]];

R = [[e1,1],[e1,2],[e2,e3],[e2,2],[e3,e4],[e4,1],[1,2]];

R = [[e1,1],[e1,2],[e2,e3],[e2,1],[e3,2],[e4,1],[e4,2]];

etc
```

A total of 54 diagrams [7] (all of which will be shown later)

Assocmass has three arguments, R,N and M. R is an instantiated list containing the topology of the diagram, that is a list of links, including this time external links (this is of course needed since the external momenta can only be unified with the internal momenta if the coupling constant is

[7] 27 actually, topologically similar diagrams were not removed at this stage

non-zero). N is the list of nodes, and M is a list of the masses of the links. We shall again check some of the second and third order diagrams.

```
| ?- assocmass([[ef1,e1],[ef2,e2],[ef3,e3],[ef4,e4],[e1,e3],[e1,e4]
,[e2,e3],[e2,e4]],[e1,e2,e3,e4],M).

M = [1,1,4,2,2,1,2,1];

M = [1,1,4,2,2,1,2,3];

M = [1,1,4,4,2,1,2,3];

M = [1,1,1,3,2,1,3,2];

M = [1,1,1,3,2,1,3,4];

M = [1,1,2,2,2,1,4,3];

M = [1,1,2,4,2,1,4,3];

M = [1,1,s1,s1,2,1,s1,s2];
```

So for example, in the first diagram, we have the couplings

Node	external link1	link2	link3	m1	m2	m3
e1	1	5	6	1	2	1
e2	2	7	8	1	2	1
e3	3	5	7	4	2	2
e4	4	6	8	2	1	1

Which is correct. By not instantiating the masslist, all instances of the mass list are generated for all external momenta for the given topology.

The external legs of the diagram representing the incoming and outgoing particles do not have a source or destination node. For this reason we introduce the nodes "ef" in the above lists which are meant to stand for "external far". They are nodes that do not actually exist but are put in for diagramatic consistency.

Checkang is a predicate of 4 variables: R, the link list which is already instantiated above, N, the list of nodes, M, the list of masses, and Th, the list of angles. In general, the first four terms of Th are instantiated to the angels of the external momenta. The predicate then generates instances of the internal angles that satisfy energy-momentum conservation. A simple test (using the previous masses) of this is as follows:

```
| ?- linkfrom([e1,e2,e3,e4],[2,2,2,2],[[e1,e3],[e1,e4],[e2,e3]
,[e2,e4]],4).
yes
| ?- assocmass([[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e3],
[e1,e4],[e2,e3],[e2,e4]],[e1,e2,e3,e4],[2,2,2,2,1,1,1,3]).
yes
checkang([[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e3],[e1,e4],
[e2,e3],[e2,e4]],[e1,e2,e3,e4],[2,2,2,2,1,1,1,3],[72,0,72,0|Th]).

Th = [66,78,78,6];
no
| ?-
```

The angles are given in units of $\pi/60$ so that each unit is a multiple of 6 (this is for historical reasons). The angleshave to be added or subtracted depending on whether they are incoming or outgoing to or from a node. We have denoted outgoing momenta by an asterix in the table below. This table is a representation of the above diagram in a more convenient form.

Node	e1	e2	e3	e4
External link1	1	2	3	4
link2	5	7	5	6
link3	6	8	7	8
m_1	2	2	2	2
m_2	1	1	1	1
m_3	1	3	1	3
$\theta 1$	$\frac{12\pi}{10}$	$\frac{0\pi}{10}$	$*\frac{12\pi}{10}$	$*\frac{0\pi}{10}$
$\theta 2$	$*\frac{11\pi}{10}$	$*\frac{13\pi}{10}$	$\frac{11\pi}{10}$	$\frac{13\pi}{10}$
$\theta 3$	$*\frac{13\pi}{10}$	$*\frac{1\pi}{10}$	$\frac{13\pi}{10}$	$\frac{1\pi}{10}$
$m_1 \cos\theta 1$	-0.476	0.588	0.476	-0.588
$m_2 \cos\theta 2$	0.294	0.182	-0.294	-0.182
$m_3 \cos\theta 3$	0.182	-0.769	-0.182	0.769
Total	0	0	0	0
$m_1 \sin\theta 1$	-0.345	0	0.345	0
$m_2 \sin\theta 2$	0.095	0.25	-0.095	-0.25
$m_3 \sin\theta 3$	0.25	-0.25	-0.25	0.25
Total	0	0	0	0

From the table above, the zeros in each total demonstrates that energy momentum is conserved

Finally, an example of the output from *main3/0*. This is for the second order poles in d6, for scattering of m_2 off m_2 at $\theta = \frac{2\pi}{10}$:

```
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e3],[e1,e4],[e2,e3],[e2,e4]]
masslst=[2,2,2,2,1,1,1,3]
thetalst=[72,0,72,0,66,78,78,6]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e3],[e1,e4],[e2,e3],[e2,e4]]
masslst=[2,2,2,2,3,1,1,1]
thetalst=[72,0,72,0,66,114,114,6]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e3],[e1,e4],[e2,e3],[e2,e4]]
masslst=[2,2,2,2,4,2,2,4]
thetalst=[72,0,72,0,84,36,36,108]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e2],[e1,e4],[e2,e3],[e3,e4]]
masslst=[2,2,2,2,1,1,1,3]
```

```
thetalst=[72,0,72,0,66,78,114,6]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e2],[e1,e4],[e2,e3],[e3,e4]]
masslst=[2,2,2,2,3,1,1,1]
thetalst=[72,0,72,0,66,114,78,6]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e2],[e1,e3],[e2,e4],[e3,e4]]
masslst=[2,2,2,2,1,1,1,1]
thetalst=[72,0,72,0,66,78,114,6]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e2],[e1,e3],[e2,e4],[e3,e4]]
masslst=[2,2,2,2,3,1,1,3]
thetalst=[72,0,72,0,66,114,78,6]
nodelst=[e1,e2,e3,e4]
linklst=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4],[e1,e3],[e1,e4],[e2,e3],[e2,e4]]
masslst=[2,2,2,2,1,1,1,3]
thetalst=[72,0,72,0,66,78,78,6]
yes
| ?-
```

This corresponds (with a slight change in the bracket notation) to the output from the C++ program as will be demonstrated later.

4.11 Graph Generation: The C++ Version

We aim in this section to demonstrate the validity of the production algorithm. As has been stated before, the production process is split into 4 main sections:

- 1 The production of topological graphs
- 2 The instantiation of masses to the propagators
- 3 The instantiation of angles to the propagators
- 4 The elimination of non-integrable diagrams

We first demonstrate that the diagram production process produces one and only one graph of each type. Any production of non-integrable diagrams (that is with a zero integral) does not matter, but the over or under production of diagrams does. We can modify the program to list out all the diagrams for a low order graphs. For the second order poles (1 loop, 4 propagators) there are only 3 such diagrams, and for the 3rd order diagrams (2 loops, 7 propagators) there are 34-7=27 graphs. These are listed below. For the fourth order poles, there are 494-208=268 graphs which can be easily generated, but were thought too many to be listed explicitly.

```
List of 2nd order graphs:

linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,2},{1,3},{2,3}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,3},{1,2},{2,3}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,3},{1,2},{1,3}}
```

We have changed the notation slightly from *patti3* so that the nodes begin at 0 and we have ommited the "e" in the four external nodes. The above is just the box diagram, and the two crossed box diagrams. List of 3rd order graphs:

```
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,2},{1,4},{2,5},{3,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,3},{1,4},{2,4},{2,5},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,2},{2,5},{3,4},{3,5},{4,5}}
```

```
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,3},{2,4},{2,5},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,4},{2,3},{2,5},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,5},{2,3},{2,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,5},{2,3},{2,5},{3,4},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,5},{2,4},{2,5},{3,4},{3,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,3},{1,4},{1,5},{2,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,3},{1,4},{1,5},{2,5},{3,4},{4,5}}*
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,2},{1,5},{3,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,3},{1,4},{2,5},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,3},{1,5},{2,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,3},{1,5},{2,5},{3,4},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,4},{1,5},{2,3},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,4},{1,5},{2,5},{3,4},{3,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,3},{1,5},{2,4},{2,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,4},{1,5},{2,3},{2,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,4},{1,5},{2,4},{2,5},{3,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,3},{2,4},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,3},{2,5},{3,4},{4,5}}*
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,4},{2,3},{3,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,4},{2,5},{3,4},{3,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,5},{2,3},{3,4},{4,5}}*
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,5},{2,4},{3,4},{3,5}}*
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,3},{1,4},{2,3},{2,5},{4,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,3},{1,4},{2,4},{2,5},{3,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,3},{1,5},{2,3},{2,4},{4,5}}*
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,3},{1,5},{2,4},{2,5},{3,4}}*
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,4},{1,5},{2,3},{2,4},{3,5}}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,4},{1,5},{2,3},{2,5},{3,4}}*
```

The asterisk above indicate that a previous diagram has the same topology as a previous diagram. For example, the first "starred" diagram is equivalent to the one before it. The nodes 4 and 5 are just swapped.

The graphs, together with their permutations of their nodes are stored in an array. By running the program twice, we can demonstrate that there is no over-production of diagrams due to interchange of nodes, since all the diagrams on the second run will be eliminated. The array where the list of diagrams is stored is preset to some limit. If this is exceeded an error message will be printed out (if it is vastly exceeded then a segmentation fault will also result).

There is no direct test that the mass instantiation procedure works correctly, however, we can easily show that each diagram satisfies the coupling

conditions, and also display some debugging information

I will just list two of the masslsts (for d6) for the second and fourth order poles:

```
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,2},{1,3},{2,3}}
masslst={ 2,3,2,3,1,1,4,1}

linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,5},{2,3},
{2,6},{3,7},{4,5},{4,6},{5,7},{6,7}}
masslst={ 3,4,3,4,1,2,3,1,2,3,1,1,4,1}
```

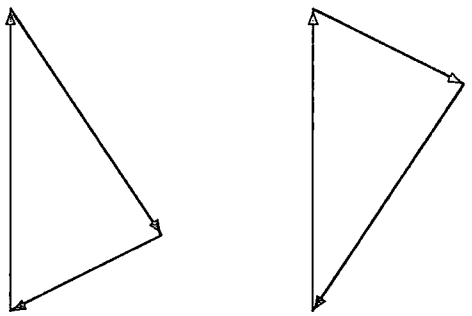
The above masses can be shown to satisfy the coupling conditions by the use of the following table:

node	0	1	2	3
$link_1$	ef0,0	ef1,1	ef2,2	ef3,3
$link_2$	0,1	0,1	0,2	1,3
$link_3$	0,2	1,3	2,3	2,3
m_1	2	3	2	3
m_2	1	1	1	4
m_3	1	4	1	1

node	0	1	2	3	4	5	6	7
$link_1$	ef0,0	ef1,1	ef2,2	ef3,3	0,4	1,5	2,6	3,7
$link_2$	0,1	0,1	2,3	2,3	4,5	4,5	4,6	5,7
$link_3$	0,4	1,5	2,6	3,7	4,6	5,7	6,7	6,7
m_1	3	4	3	4	2	3	2	3
m_2	1	1	1	1	1	1	1	4
m_3	2	3	2	3	1	4	1	1

Similarly with the instantiation of angles, we can demonstrate that that all the angles at nodes satisfy the conservation of momenta condition. The

construction of this procedure was one of the more fiddley aspects of the program, since angles are 2π periodic (or in our units, modulo some integer). For any triangle there are two possibilities or orientations that can take place, ie



This in general gives a 2^n tree. The direction of the links is critical to the direction of the momenta. All momenta is taken to be incoming, and is reversed if it is outgoing. As an example, we examine the first second order pole, and one of the fourth order poles.

2nd order pole(d6):

```
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,2},{1,3},{2,3}}
masslst={ 2,3,2,3,1,1,4,1}
thetalst={ 18,0,78,60,24,12,6,84}
```

4th order pole(d6):

```
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,4},{1,5},{2,3},
{2,6},{3,7},{4,5},{4,6},{5,7},{6,7}}
masslst={ 3,4,3,4,1,2,3,1,2,3,1,1,4,1}
```

```
thetalst={ 30,0,90,60,42,24,6,102,84,66,30,18,12,90}
```

giving for the second order pole

node	0	1	2	3
m_1	2	3	2	3
θ_1	$\frac{3\pi}{10}$	0	$\frac{13\pi}{10}$	$\frac{10\pi}{10}$
m_2	1	1	1	4
θ_1	$*\frac{4\pi}{10}$	$\frac{4\pi}{10}$	$\frac{2\pi}{10}$	$\frac{1\pi}{10}$
m_3	1	4	1	1
θ_3	$*\frac{2\pi}{10}$	$*\frac{1\pi}{10}$	$*\frac{14\pi}{10}$	$\frac{14\pi}{10}$
$p_t^1 = m_1 \cos\theta_1$	0.345	0.809	-0.345	-0.809
$p_t^2 = m_2 \cos\theta_2$	-0.095	0.0954	0.25	0.904
$p_t^3 = m_3 \cos\theta_3$	-0.25	-0.904	0.095	-0.095
total	0	0	0	0
$p_x^1 = m_1 \sin\theta_1$	0.476	0	-0.476	0
$p_x^2 = m_2 \sin\theta_2$	-0.294	0.294	0.182	0.294
$p_x^3 = m_3 \sin\theta_3$	-0.182	-0.294	0.294	00.294
total	0	0	0	0

and for the fourth order pole

node	0	1	2	3	4	5	6	7
m_1	3	4	3	4	2	3	2	3
θ_1	$\frac{5\pi}{10}$	$\frac{0\pi}{10}$	$\frac{15\pi}{10}$	$\frac{10\pi}{10}$	$\frac{4\pi}{10}$	$\frac{1\pi}{10}$	$\frac{14\pi}{10}$	$\frac{11\pi}{10}$
m_2	1	1	1	1	1	1	1	4
θ_1	$*\frac{7\pi}{10}$	$\frac{7\pi}{10}$	$*\frac{17\pi}{10}$	$\frac{17\pi}{10}$	$*\frac{5\pi}{10}$	$\frac{5\pi}{10}$	$\frac{3\pi}{10}$	$\frac{2\pi}{10}$
m_3	2	3	2	3	1	4	1	1
θ_3	$*\frac{4\pi}{10}$	$*\frac{1\pi}{10}$	$*\frac{14\pi}{10}$	$*\frac{11\pi}{10}$	$*\frac{3\pi}{10}$	$*\frac{2\pi}{10}$	$*\frac{15\pi}{10}$	$\frac{15\pi}{10}$
$p_t^1 = m_1 \cos\theta_1$	0	0.951	0	-0.951	0.182	0.769	-0.182	-0.769
$p_t^2 = m_2 \cos\theta_2$	0.18	-0.182	-0.182	0.182	0	0	0.182	0.769
$p_t^3 = m_3 \cos\theta_3$	-0.18	-0.769	0.182	0.769	-0.182	-0.769	0	0
total	0	0	0	0	0	0	0	0
$p_x^1 = m_1 \sin\theta_1$	0.809	0	-0.809	0	0.559	0.25	-0.559	-0.25
$p_x^2 = m_2 \sin\theta_2$	-0.25	0.25	0.25	-0.25	-0.309	0.309	0.25	0.559
$p_x^3 = m_3 \sin\theta_3$	-0.559	-0.25	0.559	0.25	-0.25	-0.559	0.309	-0.309
total	0	0	0	0	0	0	0	0

Our final procedure is the integrability check. It does not matter if this fails to eliminate a diagram, as it will have a zero residue anyway, but it does matter if it falsely eliminates a diagram that is non zero. Although not listed, it has been checked for the second and third order diagrams. The procedure was turned off and the results compared to the case when it had been turned on. The result was that the procedure correctly eliminated all the zero residual diagrams. One of the reasons this procedure was written in the first place was that the number of diagrams printed to file was growing rapidly. For the third order diagrams they took up over a megabyte, and the fourth order diagrams could not be printed because of file size restrictions. The first few diagrams were checked by the same method as above, but if there is just one diagram that is incorrectly eliminated, finding it would be like finding a needle in a haystack. One further check is that (like in the angle instantiation procedure) the result should be invariant under rotation and crossing. Also the number of diagrams eliminated should also be the same, which can be checked by printing out the *dino* debugging variables, eg

for the $S_{34}(\frac{4\pi}{10})$ S matrix, there are hundreds of diagrams eliminated. This has been tested for several different angles.

Finally the output is written to a file in canonical form. The process of diagram production can be restarted at any stage, and it is possible to output several S matrix elements in one go without re-compilation, and they are included from a file that has been generated directly from the table of S matrix elements (via a little Awk program).

5 The Calculation Algorithm

We now discuss the algorithm to calculate the Feynman diagrams produced by the generation program. The data given to this program, generated by the generation algorithm is a set of lists, specifying:

1. A set of nodes
2. The link list giving the topology of the diagram,
3. A mass for each link
4. An angle (normally in units of $\frac{\pi}{h}$) for each link

Basically, the algorithm contains five parts

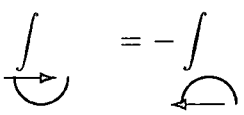
1. The data is taken in, and the expression to be integrated is formed
2. The integrand is integrated over the first variable
3. 2 is repeated for each subsequent variable
4. The final expression is evaluated as $\epsilon \rightarrow 0$
5. The result is multiplied by the overall factor, and is printed.

The integrand is simply a product of (not necessarily simple) poles, and as such, the integration is

$$\int_{-\infty}^{\infty} \frac{F(x)dx}{(x+b+ci\epsilon)^n} = \frac{2\pi i}{(n-1)!} \frac{\partial^{(n-1)} F(x)}{\partial x} \theta(c) \tag{298}$$

where $\theta(c) = 1$ is $c > 0$; $1/2$ if $c = 0$; (important); 0 if $c < 0$

The contour taken can be a semicircle either above or below the axis. Using:



(299)

It is best to take the contour that has the least number of poles. The calculation algorithm does just that.

5.1 The limit

The last part is to calculate the limit as $\epsilon \to 0$. The integral at this point is a sum of terms of the form

$$\frac{.1}{\prod(a_i + b_i\epsilon)} \tag{300}$$

in some cases $a_i = 0$, in which case the term in the integral is infinite. It is, however, normally counterbalanced by another term in the sum which is also infinite (or rather $-\infty$). The algorithm checks, firstly whether or not the whole sum is finite and if it is, calculates its true value. This is done by taking a power expansion in terms of ϵ, of the numerator of terms that are infinite, ie

$$I = \frac{V(\epsilon)}{\epsilon} = \frac{A + B\epsilon + ...}{\epsilon} \tag{301}$$

where

$$B = \frac{\partial}{\partial\epsilon}V \tag{302}$$

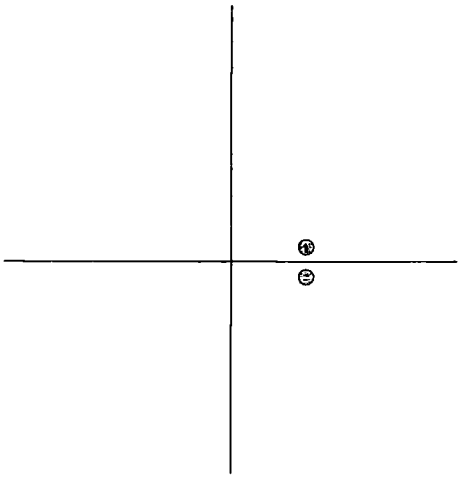
providing that all the A's cancel out

$$\sum A_i = 0 \tag{303}$$

then the value of the integral is

$$I = \sum_i B_i \tag{304}$$

The reason for these singularities is because of pinches:



whereby a pole a $(b + i\epsilon)$ and $(b - i\epsilon)$ occurs, generating an expression

$$\frac{1}{(2i\epsilon)(....)} \tag{305}$$

Another common occurrence is integrands of the type

$$\frac{1}{(au + bv + ... + i\epsilon)(au + cv + ... + i\epsilon)(....} \tag{306}$$

which can lead to terms in the denominator with no $i\epsilon$ part (ie leaving poles on the real line). There is two ways of dealing with this situation. Either one could take the expression as is, and use the fact that $\theta(0) = \frac{1}{2}$, or alternatively, if instead of a constant ϵ, each propagator has its own small imaginary part, $\epsilon^{(i)}$ say (not to be confused with $\epsilon_i = a_i b_i$) then the integral should not be dependent on the path taken of $\forall i \epsilon^{(i)} \rightarrow 0$. In the program this is done by setting $\epsilon^{(i)}$ equal to ϵ multiplied by a random positive number ("1+Random[]", in *setprop*). This in general simplifies the number of terms produced.

5.2 Simple test of the integrator

There are two procedures relating to integration, *necsimp4*, which reduces all of the coefficients of the integration variable in the integrand to one.

The actual integration is done be *int4*, and the integrand must be in the simplified form in order for it to work. There are two things we must ensure that *necsimp4* does.

- It must work for an arbitrary number of terms
- It must work for an arbitrary number of propagators
- It must group propagator terms that are the same, ie $\frac{1}{AA...} \rightarrow \frac{1}{A^2...}$ where A is a propagator
- The simplified expression should be the same as the original, taking into account terms in the denominator of arbitrary power.

To demonstrate how the simplifier works, let us consider a very simple expression:

$$\int dx \frac{A}{(2x+b+i\epsilon)(3x+c-i\epsilon)} \tag{307}$$

This is a single term expression, so the canonical form for it is

$$\{\{\{A,1\},\{2x+b+i\epsilon,1\},\{3x+c-i\epsilon,1\}\}\} \tag{308}$$

We now execute the following in Mathematica:

```
In[3]:= intg={{{A,1},{2*x+b+ie,1},{3*x+c-ie,1}}}

Out[3]= {{{A, 1}, {b + ie + 2 x, 1}, {c - ie + 3 x, 1}}}

In[4]:= simpintg=necsimp4[intg,x]

            A        b + ie           c - ie
Out[4]= {{{-, 1}, {------ + x, 1}, {------ + x, 1}}}
            6          2              3
```

The canonical form translates into

$$\frac{\frac{A}{6}}{(x+\frac{1}{2}b+\frac{1}{2}i\epsilon)(x+\frac{1}{3}b-\frac{1}{3}i\epsilon)} \tag{309}$$

as would be expected. Note that we use e in place of ϵ in the Mathematica code. It is obvious that there are poles at $x=-\frac{1}{2}b-\frac{1}{2}i\epsilon$ and $x=-\frac{1}{3}c+\frac{1}{3}i\epsilon$. Integrating over the bottom contour gives a residue

$$\frac{\frac{A}{6}}{(\frac{1}{2}b-\frac{1}{3}c+\frac{5}{6}i\epsilon)} \tag{310}$$

simpintg is now the simplified integrand, and can now be integrated by *int4*. There are various things *int4* must do. They are

- It must work for an arbitrary number of terms
- It must work for an arbitrary number of propagators
- It must give the correct result for propagator terms of arbitrary powers
- It should treat poles on the real line correctly (with factor $\frac{1}{2}$)

For an initial demonstration we can use the simplified integrand generated above. We execute

```
result=int4[simpintg,x]
```

```
In[5]:= result=int4[simpintg,x]

            -A        b + ie   -c + ie
Out[5]= {{{--, 1}, {------ + -------, 1}}}
            6          2        3
```

The canonical form of the output translates into

$$2\pi i\frac{\frac{-A}{6}}{\frac{1}{2}b-\frac{1}{3}c+\frac{5}{6}i\epsilon} \tag{311}$$

which is the correct result.

The finial stage is computing the limit as $\epsilon \rightarrow 0$. This is done be a procedure *die2* which itself calls a procedure *ldec*. *ldec* stands for list-decoder, and decodes our canonical form into a more standard Mathematica form, in general giving a fraction with a denominator a product of linear terms in ϵ. *die2* must be careful in its computation of the limits, as there are occasions where one of the terms is infinite (cased be one of the terms in the denominator having the form $bi\epsilon$ as opposed to $a + bi\epsilon$). The method for resolving this problem is described above. The other conditions that *die2* must satisfy are

- It must work for an arbitrary number of terms
- It must work for an arbitrary number of propagators
- It must give the correct result for propagator terms of arbitrary powers
- It must give the correct result for summing of infinite expression
- If the result is really infinite, it should say so by giving an error message

For a first test we put in the integrated result we have generated above

```
In[6]:= uint=die2

            -A
Out[6]= ---------
             b   c
          6 (- - -)
             2   3
```

We now check that it deals correctly with seemingly infinite expressions. For this it is helpful to turn on debugging information.

```
Firstly a subtraction of two infinities:
```

```
In[24]:= result={{{1,1},{a+ie,2},{ie,1}},{{-1,1},{a+3*ie,2},{ie,1}}}

Out[24]= {{{1, 1}, {a + ie, 2}, {ie, 1}}, {{-1, 1}, {a + 3 ie, 2}, {ie, 1}}}

In[25]:= die2

                                    1
Power::infy: Infinite expression - encountered.
                                    0

                                    1
Power::infy: Infinite expression - encountered.
                                    0
ordinary term: 0
           4
prepole A1: --
            3
           a
prepole A0: 0

         4
Out[25]= --
          3
         a
```

Thus although there were two infinite expressions, they both cancel, leaving a result

$$\frac{1}{ie(a+ie)^2} - \frac{1}{ie(a+3ie)^2} = \frac{1}{ie}\left(\frac{1}{a^2} - \frac{2ie}{a^3} - \frac{1}{a^2} + \frac{6ie}{a+3ie}\right) = \frac{4}{a^3} \qquad (312)$$

If the poles do not cancel, then *die2* says so, eg

```
In[26]:= result={{{1,1},{a+ie,2},{ie,1}},{{1,1},{a+3*ie,2},{ie,1}}}

Out[26]= {{{1, 1}, {a + ie, 2}, {ie, 1}}, {{1, 1}, {a + 3 ie, 2}, {ie, 1}}}

In[27]:= die2

                                    1
Power::infy: Infinite expression - encountered.
                                    0

                                    1
Power::infy: Infinite expression - encountered.
                                    0
ordinary term: 0
             -8
prepole A1: --
              3
             a
             2
prepole A0: --
              2
             a
************possible infinite expression in die2

          -8
Out[27]= --
           3
          a
```

We now consider a real calculation. The information is read in of the form of the four lists:

- nodelst: a list of nodes a,b,c,d...
- linklst: a list of links a,b indicating that a is linked to b

- masslst: a list of masses which have a 1 to 1 correspondence with the links in the linklst
- thetalst: a list of angles in arbitrary units

The procedure *setprop* takes in this data, and outputs a canonical list that is representative of this data.

We can demonstrate this with a very simple example of one of the first 2nd order poles:

```
In[60]:= nodelst

Out[60]= {0, 1, 2, 3}

In[61]:= linklst

Out[61]= {{ef0, 0}, {ef1, 1}, {ef2, 2}, {ef3, 3}, {0, 1}, {0, 3}, {1, 2},

>    {2, 3}}

In[62]:= masslst

Out[62]= {2, 3, 2, 3, 1, 1, 2, 4}

In[63]:= thetalst

Out[63]= {42, 0, 102, 60, 48, 36, 6, 114}

In[66]:= Simplify[N[den]]

Out[66]= {{{1., 1.}, {-0.0557281 + ie - 0.5 I8x1 + 0.363271 I8x2, 1.},

>     {0.0901699 + ie + 0.190983 I8x1 - 0.587785 I8x2, 1.},

>     {0.326238 + ie + 1.11803 I8x1 + 0.363271 I8x2, 1.},

>     {-0.527864 + ie + 1.80902 I8x1 - 0.587785 I8x2, 1.}}}
```

The masses angles and t and x components of the momenta are given in the following table (note that the mass is normalised by $m^2 = 8$. Any normalisation does not affect the overall result, since we have m^{2v} for the couplings, m^{-2l} for the loops, m^{-2o} for each order, and m^{-2} for the flux factor)

link	m	θ	$2m\cos\theta$	$2m\sin\theta$	a	b	ab
P_a	$m_2 = 0.588$	$\frac{7\pi}{10}$	-0.691	0.951	1	0	0
P_b	$m_3 = 0.809$	$\frac{0\pi}{10}$	1.618	0	0	1	0
0,1	$m_1 = 0.309$	$\frac{8\pi}{10}$	-0.5	0.363	0.382	-0.146	-0.055
0,3	$m_1 = 0.309$	$\frac{6\pi}{10}$	0.191	-0.588	0.618	0.146	0.090
1,2	$m_2 = 0.588$	$\frac{\pi}{10}$	1.118	0.363	0.381	0.854	0.326
2,3	$m_4 = 0.951$	$\frac{-\pi}{10}$	1.809	-0.588	-0.618	0.854	-0.528

These are the values given in the denominator when *setprop* is executed.

Finally we list the masses and coupling constants, and demonstrate that they are in agreement with those listed in the tables. (actually we list 8 times the squares of the masses and

```
setgroupd6;
name[1]=1; name[2]=2; name[3]=3; name[4]=4; name[5]=s1; name[6]=s2;

For[i=1,i<=n,i++,Print["mass ",name[i]," is ",N[8*m[i]*m[i]]]];

In[2]:= setgroupd6

In[3]:= name[1]=1; name[2]=2; name[3]=3; name[4]=4; name[5]=s1; name[6]=s2;

In[4]:= For[i=1,i<=n,i++,Print["mass ",name[i]," is ",N[8*m[i]*m[i]]]];
mass 1 is 0.763932
mass 2 is 2.76393
mass 3 is 5.23607
mass 4 is 7.23607
mass s1 is 2.
```

```
mass s2 is 2.

In[5]:= For[i=1,i<=n,i++,
        For[j=i,j<=n,j++,
        For[k=j,k<=n,k++,
        If[!TrueQ[(cc=c2[Sort[{name[i],name[j],name[k]}]]) == Null],
        Print["coupling ",i,j,k," is ",N[8*cc]]]]]]

coupling 112 is 1.27004
coupling 123 is 3.32502
coupling 134 is 5.37999
coupling 156 is 3.32502
coupling 224 is 7.43496
coupling 244 is -12.03
coupling 255 is 5.37999
coupling 266 is 5.37999
coupling 334 is -14.085
coupling 356 is 5.37999
coupling 455 is 3.32502
coupling 466 is 3.32502



For e6:

In[4]:= ?m
Global`m

m[1] = (3 - 3^(1/2))^(1/2)

m[2] = (3 - 3^(1/2))^(1/2)

m[3] = 2^(1/2)*(3 - 3^(1/2))^(1/2)

m[4] = (3 + 3^(1/2))^(1/2)

m[5] = (3 + 3^(1/2))^(1/2)
```

```
m[6] = 2^(1/2)*(3 + 3^(1/2))^(1/2)


In[7]:= For[i=1,i<=n,i++,
        For[j=i,j<=n,j++,
        For[k=j,k<=n,k++,
        If[NumberQ[cc=N[c[{i,j,k}]]],
        Print["coupling ",i,j,k," is ",N[cc]]]]]]
coupling 112 is -1.09808 I
coupling 113 is 1.26795
coupling 114 is -0.633975 I
coupling 122 is 1.09808 I
coupling 123 is 1.26795
coupling 124 is -0.633975 I
coupling 125 is 0.633975 I
coupling 134 is 1.73205
coupling 135 is 1.73205
coupling 145 is -2.36603 I
coupling 146 is 1.73205
coupling 155 is -2.36603 I
coupling 156 is 1.73205
coupling 223 is 1.26795
coupling 224 is 0.633975 I
coupling 234 is 1.73205
coupling 235 is 1.73205
coupling 244 is -2.36603 I
coupling 245 is 2.36603 I
coupling 246 is 1.73205
coupling 256 is 1.73205
coupling 333 is 2.19615
coupling 336 is 1.26795
coupling 366 is -4.73205
coupling 445 is 4.09808 I
coupling 446 is -4.73205
coupling 455 is 4.09808 I
coupling 456 is -4.73205
```

```
coupling 556 is -4.73205
coupling 666 is 8.19615

In[9]:= setgroupe7

In[10]:= ?m
Global`m

m[1] = 2*2^(1/2)*Sin[Pi/9]

m[2] = 2*2^(1/2)*(3^(1/2)*Sin[Pi/18]*Sin[(2*Pi)/9])^(1/2)

m[3] = 2*2^(1/2)*Sin[(2*Pi)/9]

m[4] = 2*2^(1/2)*(3^(1/2)*Sin[Pi/9]*Sin[(5*Pi)/18])^(1/2)

m[5] = 6^(1/2)

m[6] = 2*2^(1/2)*Sin[(4*Pi)/9]

m[7] = 2*2^(1/2)*(3^(1/2)*Sin[(7*Pi)/18]*Sin[(4*Pi)/9])^(1/2)
```

5.3 Comprehensive tests

In the above examples we have just demonstrated that the procedures work for very simple cases. We now provide a much more rigorous test to see that the program works for all cases.

We demonstrate that necsimp works for multiple powers of propagators, and generates multiple powers by the following test

```
test that higher powers integrate:
```

```
In[21]:= intg={{{A,1},{x+a1+ie,7},{x+a2-ie,5}}}

Out[21]= {{{A, 1}, {a1 + ie + x, 7}, {a2 - ie + x, 5}}}

In[22]:= int4[intg,x]

Out[22]= {{{-210 A, 1}, {a1 - a2 + 2 ie, 11}}}
```

since

$$\oint dx \frac{A}{(x+a1+i\epsilon)^7(x+a2-i\epsilon)^5} = \tag{313}$$

$$[2\pi i]\frac{1}{4!}\frac{\partial^4}{\partial x^4}\frac{A}{(x+a1+i\epsilon)^7}\,|_{x=-a2+i\epsilon} = \tag{314}$$

$$[2\pi i]\frac{-7.-8.-9.-10}{4!}\frac{A}{(x+a1+i\epsilon)^{11}} \tag{315}$$

test that necsimp simplifies similar poles:

```
In[27]:= intg={{{A,1},{2*x+2*a1+2*ie,3},{3*x+3*a1+3*ie,4}}}

Out[27]= {{{A, 1}, {2 a1 + 2 ie + 2 x, 3}, {3 a1 + 3 ie + 3 x, 4}}}

In[28]:=  necsimp4[intg,x]

                A
Out[28]= {{{---, 1}, {a1 + ie + x, 7}}}
               648
```

ie,

$$\frac{A}{(2x+2a1+2i\epsilon)^3(3x+3a1+3i\epsilon)^4} = \frac{A}{2^3 3^4}\frac{1}{(x+a1+i\epsilon)^7} \tag{316}$$

putting it all together:

```
In[8]:=  intg={{{A,1},{x+a1+8*ie,7},{x+a2-9*ie,5}},
         {{B,1},{2*x+2*a2+2*ie,3},{3*x+3*a2+3*ie,4}},
{{C,1},{x+a1+ie,3},{x+a2-3*ie,2},{2*x+2*a3+5*ie,2},{3*x+3*a4+7*ie,3}}}

Out[8]= {{{A, 1}, {a1 + 8 ie + x, 7}, {a2 - 9 ie + x, 5}},

>    {{B, 1}, {2 a2 + 2 ie + 2 x, 3}, {3 a2 + 3 ie + 3 x, 4}},

>    {{C, 1}, {a1 + ie + x, 3}, {a2 - 3 ie + x, 2}, {2 a3 + 5 ie + 2 x, 2},

>     {3 a4 + 7 ie + 3 x, 3}}}

In[9]:=  simpintg=necsimp4[intg,x]

Out[9]= {{{A, 1}, {a1 + 8 ie + x, 7}, {a2 - 9 ie + x, 5}},

        B
>    {{---, 1}, {a2 + ie + x, 7}},
       648

        C                                                      5 ie
>    {{---, 1}, {a1 + ie + x, 3}, {a2 - 3 ie + x, 2}, {a3 + ---- + x, 2},
       108                                                     2

            7 ie
>     {a4 + ---- + x, 3}}}
            3

In[10]:= result=int4[simpintg,x]

Out[10]= {{{-210 A, 1}, {a1 - a2 + 17 ie, 11}},
```

```
        C                                                    11 ie
>     {{--, 1}, {a1 - a2 + 4 ie, 4}, {-a2 + a3 + -----, 2},
        36                                                    2

                    16 ie          C
>      {-a2 + a4 + -----, 3}}, {{--, 1}, {a1 - a2 + 4 ie, 3},
                      3            54

                    11 ie                      16 ie
>      {-a2 + a3 + -----, 3}, {-a2 + a4 + -----, 3}},
                      2                        3

        C                                                    11 ie
>     {{--, 1}, {a1 - a2 + 4 ie, 3}, {-a2 + a3 + -----, 2},
        36                                                    2

                    16 ie
>      {-a2 + a4 + -----, 4}}}
                      3

In[11]:= die2

            -210 A                          C
Out[11]= ----------- + ------------------------------------ +
                   11                3            2            4
         (a1 - a2)     36 (a1 - a2)  (-a2 + a3)  (-a2 + a4)

                       C
>     ------------------------------------ +
                    3           3           3
      54 (a1 - a2)  (-a2 + a3)  (-a2 + a4)

                       C
>     ------------------------------------
                    4           2           3
      36 (a1 - a2)  (-a2 + a3)  (-a2 + a4)
```


```
Test that {\em int4} can deal with poles on the real line:

In[175]:= intg={{{A,1},{x+a,1},{x+b-ie,1}}}

Out[175]= {{{A, 1}, {a + x, 1}, {b - ie + x, 1}}}

In[176]:= int4[intg,x]
no ie <---------------------------- indicates pole is on real axis

                A
Out[176]= {{{-, 1}, {-a + b - ie, 1}}}
                2

If necessary of arbitrary powers:

In[177]:= intg={{{A,1},{x+a,2},{x+b-ie,3}}}

Out[177]= {{{A, 1}, {a + x, 2}, {b - ie + x, 3}}}

In[178]:= int4[intg,x]
no ie

                -3 A
Out[178]= {{{----, 1}, {-a + b - ie, 4}}}
                 2
```

5.4 Overall factors

The overall factor is a product of the coupling constants, the vertex and propagator factors, the s to θ conversion factors and the overall flux factor.

- $-(2\pi)^2 c[\{a, b, c\}]$ for each vertex
- $\frac{1}{(2\pi)^2}$ for each propagator

- $\frac{-1}{(2\pi)^2 8\Delta}$ for the flux and normalisation factors
- $4i\Delta$ for the $s \rightarrow \theta$ conversion
- $2\pi i$ for each loop integration variable (which we left out in the above integrals)

We do not test this procedure, but refer simply to the expression in the program, and note that if any couplings are not valid, then the overall factor will have a "Null" in it.

5.5 Truncation errors

Because most of the calculation is done semi-numerically, we can on occasion have an integral like

$$\oint dx \frac{1}{(x+ay+b+i\epsilon)(x+ay-c-i\epsilon)} \tag{317}$$

$$= \frac{2\pi i}{(a-a)y+b+c+2i\epsilon} \tag{318}$$

which involves a term $(a-a)y = 0$. Because of numerical rounding errors however, "$a-a$" may, in the computers memory not quite be zero, but instead a very small number. This will be of the order of the numerical accuracy to which we are working, which is generally speaking around 10^{-30}. Because we are using contour integration, which is critically dependent on whether poles are above, below, or on the real axis truncation errors could radically alter the end result. I have included a routine called *grsn*, or get rid of small numbers which eliminates all numbers that have an absolute value below a predetermined level, set by default to 10^{-15}. This level must be sufficiently large to capture all truncation errors, but sufficiently small so that it does not eliminate valid arguments. The sort of numbers that appear in the denominator at all stages of integration are generally greater than 0.001. Basically they are due to sums and differences of momenta, ie $\sum k_i m_i \cos\theta + k_i' m_i \sin\theta$ where $\theta \in \{\frac{n\pi}{h}\}$, m_i is one of the masses (of the order 1) and k_i, k_i' varies from -1 to 1. The use of *grsn* is demonstrated below:

```
In[3]:=  intg={{{2,1},
{3*x+2+0.000000000000001ie,1},
{4*x+3-0.000000000000001ie,1},
{5*x+4-0.0000000000000001ie,1},
{6*x+5-0.00000000000000001ie,1}
}}

                                  -14                                -15
Out[3]= {{{2, 1}, {2 + 1. 10    ie + 3 x, 1}, {3 - 1. 10    ie + 4 x, 1},

                 -16                                -17
>     {4 - 1. 10    ie + 5 x, 1}, {5 - 1. 10    ie + 6 x, 1}}}

In[4]:= grsn3[intg]
*********small number (zero?) eliminated
*********small number (zero?) eliminated

                                  -14                                -15
Out[4]= {{{2, 1}, {2 + 1. 10    ie + 3 x, 1}, {3 - 1. 10    ie + 4 x, 1},

>     {4 + 5 x, 1}, {5 + 6 x, 1}}}
```

5.6 Final output

The final output of the program is a script containing 6 fields: The bare value of the integral, the overall factors due to flux and normalisation, and the overall result, which is the product of the two. In addition, since a batch file contains many S matrix calculations, and many integrals for each element the output also lists the calculation and integral numbers. A "result so far" field indicates the sum of the integrals so far for a particular calculation. At the start of a calculation, the angles and masses are listed to indicate which S matrix element is being calculated. For example, the second order pole in d_6 of S_{22} at an angle of $\frac{2\pi}{10}$ is

```
Comment:%start
**************start of S matrix calculation*****************
Comment:%ma=2
Comment:%ta=12
Comment:%mb=2
Comment:%tb=0
Comment:%start
**************start of S matrix calculation*****************
A.calculation is 4 B.integral no is 1
C.integral is -22.2703 D.overall factor is -0.032492
E.overall result is 0.723607
F.result so far is 0.7236067977499785
-------------------------------------------------------------
A.calculation is 4 B.integral no is 2
C.integral is -22.2703 D.overall factor is -0.032492
E.overall result is 0.723607
F.result so far is 1.447213595499957
-------------------------------------------------------------
A.calculation is 4 B.integral no is 3
C.integral is 94.3386 D.overall factor is -0.00474051
E.overall result is -0.447214
F.result so far is 1.
-------------------------------------------------------------
```

The last 'F' field indicates that this has a pole of

$$1 \times \frac{\beta^4}{(2h)^2(\theta - i\theta_0)^2} \tag{319}$$

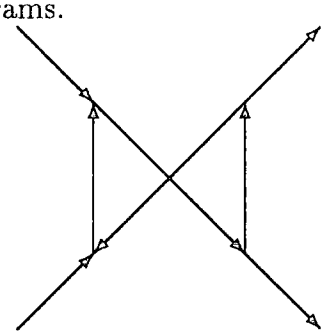
$$\theta_0 = \frac{2\pi}{h} \tag{320}$$

$$h = 10 \tag{321}$$

The data for this calculation comes from the generation program. Here is a listing of the part responsible for that particular calculation:

```
%start
%ma=2
%ta=12
%mb=2
%tb=0
%start
nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,3},{1,2},{2,3}}
masslst={ 2,2,2,2,1,1,3,1}
thetalst={ 12,0,72,60,18,6,6,114}
nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,3},{1,2},{2,3}}
masslst={ 2,2,2,2,1,3,1,1}
thetalst={ 12,0,72,60,54,6,6,78}
nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,3},{1,2},{1,3}}
masslst={ 2,2,2,2,1,1,1,1}
thetalst={ 12,0,72,60,18,6,6,114}
```

It is interesting to note that although there are three diagrams listed, the first two have the same topology. They are in fact both crossed box diagrams.



This means that the other crossed box diagram is missing. This shows that the naive assumption of one and only one of each type of diagram is incorrect, but that all the second and third order poles agree with that of the exact S matrix.

5.7 Discussion of Results

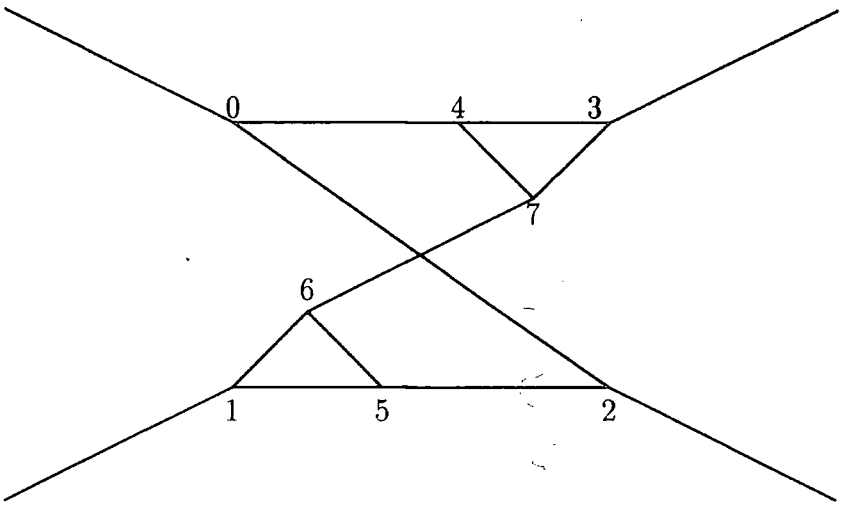
The second and third order poles, after correcting for input data agreed completely with [9] in d_6 as well as the exceptional algebras e_6, e_7 and e_8 in the examples calculated. There was additional, and as yet uncertain difficulty in calculating the fourth order poles, which has only been tried for d_6 so far.

5.7.1 The poles in d_6

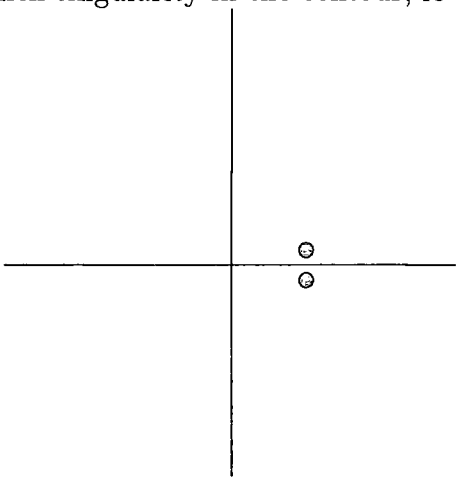
All of the second and third order poles agreed exactly with that of the exact S matrix and the problem came with calculation of the fourth order poles. The first problem that arose was the complexity of producing all the diagrams. Before the integrability predicates were put in the number of diagrams ran into millions. This number in now of the order of 300. Although all the diagrams that contribute to 4th order poles can now be listed, their calculation is still unsolved. The reason for this is that some of the diagrams are not amenable to calculation by the Landau method. For example, one of the diagrams produced was:

```
nodelst={0,1,2,3,4,5,6,7}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,5},{1,6},
{2,5},{3,4},{3,7},{4,7},{5,6},{6,7}
masslst={ 3,4,3,4,1,2,s1,s1,2,s1,s1,s1,s1,4}
thetalst={ 30,0,90,60,42,24,6,114,84,66,54,42,102,108}
```

which corresponds to



When calculating the residue, terms in the denominator cancel so as to give a pinch singularity in the contour, ie



The usual was of dealing with infinite integrals in perturbation theory is to regularise, either by taking the limits of the k integration to some upper UV cutoff, or by adding the dimensionality of space as an extra parameter. Unfortunately both of these are unapplicable in the case of Landau method. The pinches occur at certain points, so one could replace the integral with

$$\int_{-\infty}^{\infty} \rightarrow \lim_{\delta \rightarrow 0} \int_{\infty}^{p-\delta} + \int_{p+\delta}^{\infty} \tag{322}$$

However, this does not solve the problem completely, since the residue is still dependent on ϵ, the small imaginary part of the propagators that place the S matrix in the physical region. There is a possible solution to this problem by deforming the mass and coupling parameters

$$m_i \rightarrow m_i + \delta m_i \tag{323}$$

$$C_{ijk} \rightarrow C_{ijk} + \delta C_{ijk} \tag{324}$$

and then taking the sum of all the diagrams in the limit as $\delta m_i \rightarrow 0$, $\delta C_{ijk} \rightarrow 0$. This has not been tried yet.

A test run to find any fifth order poles in d_6 has been unsuccessful, although the search was not complete.

6 Comparisons

In this section we compare some results from the programs to those which have been calculated from hand in [9] and demonstrate complete agreement.

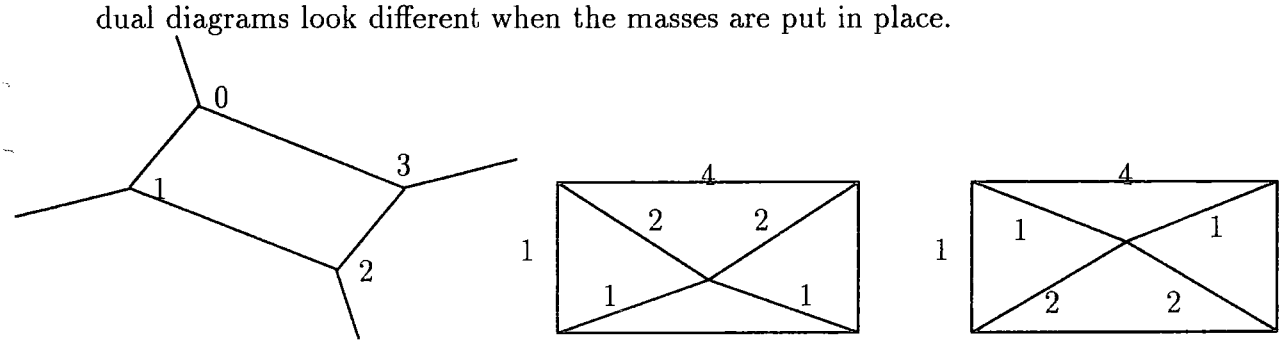
6.0.2 Second Order Diagrams

First of all let us take the second order pole of S_{14} at an angle of $\frac{\pi}{2}$ in e_7. Below is listed a printout of the diagrams together with the results:

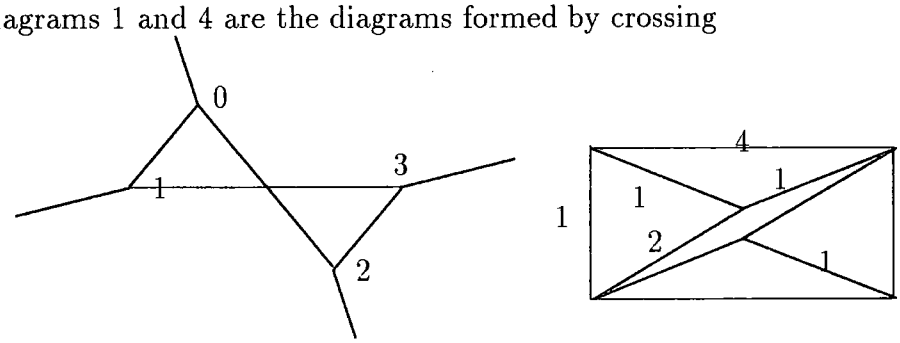
```
nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,2},{1,3},{2,3}}
masslst={ 1,4,1,4,1,2,1,1}
thetalst={ 9,0,27,18,17,4,1,35}
A.calculation is 1 B.integral no is 1 -> E.overall result is -0.326352 I -> F.result so far is -0.326351822333

nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,3},{1,2},{2,3}}
masslst={ 1,4,1,4,1,2,1,2}
thetalst={ 9,0,27,18,17,4,1,32}
A.calculation is 1 B.integral no is 2 -> E.overall result is 0.826352 I -> F.result so far is 0.50000000000000

nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,1},{0,3},{1,2},{2,3}}
masslst={ 1,4,1,4,2,1,2,1}
```

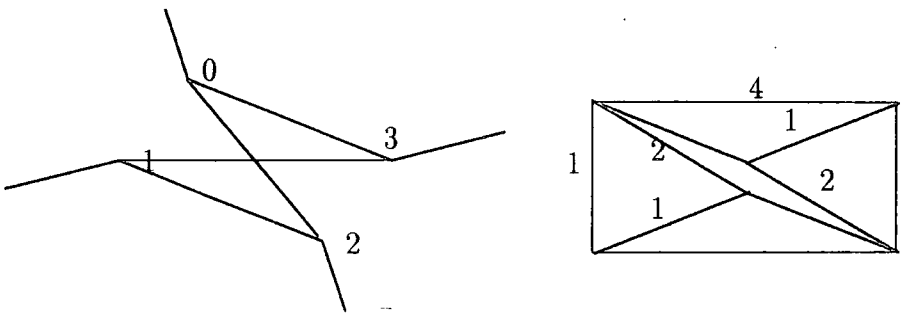
```
thetalst={ 9,0,27,18,14,1,4,35}
A.calculation is 1 B.integral no is 3 -> E.overall result is 0.826352 I -> F.result so far is 1.32635182233307

nodelst={0,1,2,3}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,3},{1,2},{1,3}}
masslst={ 1,4,1,4,2,1,1,1}
thetalst={ 9,0,27,18,14,1,1,35}
A.calculation is 1 B.integral no is 4 -> E.overall result is -0.326352 I -> F.result so far is 1.0000000000000
```

We take as a base diagram number 2 and number 3, which are simply mirror images of each other, ie they both have the same topology, but their dual diagrams look different when the masses are put in place.



Diagrams 1 and 4 are the diagrams formed by crossing





In terms of the notation used in [9], we define the triangle area's Δ as follows together with the calculated numerical factor:

$$\Delta = \frac{m_1 m_4}{2} = .921605 \tag{325}$$

$$\Delta_a = \Delta_{[114]} = 0.160035 \tag{326}$$

$$\Delta_b = \Delta_{[121]} = 0.460802 \tag{327}$$

$$\Delta'_a = \Delta_{[224]} = 0.76157 \tag{328}$$

$$\Delta'_b = \Delta_{[121]} = 0.460802 \tag{329}$$

The residues of the diagrams are denoted by r_i in the order listed in [9]. This is not the same order as was produced by the program. Their numerical factors are given below:

$$r_1 = \frac{\Delta'_a}{\Delta} = 0.826352 \tag{330}$$

$$r_2 = \frac{\Delta_a \Delta_b - \Delta'_a \Delta'_a}{\Delta^2} = -0.326352 \tag{331}$$

$$r_3 = \frac{\Delta_a \Delta'_b - \Delta'_a \Delta_b}{\Delta^2} = -0.326352 \tag{332}$$

r_1 corresponds to diagram 2 and 3, r_2 corresponds to diagram 1 and r_3 corresponds to diagram 4 in the printout above. The sum of these gives one both numerically in this instance, and algebraically in the formulas above.

$$2r_1 + r_2 + r_3 = 1 \tag{333}$$

6.0.3 Third Order Diagrams

We take as an example the 3 order pole in S_{27} at $\frac{8\pi}{18}$ in e_7. Below is listed a printout of the diagrams together with the results. They are split into two groups, the firt 9 for diagrams with one line which is the sum of external momenta, and which sum to -1 and the remaining 17 which sum to 2. This is again in complete agreement with [9]

```
nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,1,1,1,6,1,6,7}
thetalst={ 8,0,26,18,3,13,3,35,31,17,16}
A.calculation is 1 B.integral no is 1 -> E.overall result is -0.15597 -> F.result so far is -0.155970371254015

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,1,1,2,5,2,6,7}
thetalst={ 8,0,26,18,3,13,2,35,32,17,16}
A.calculation is 1 B.integral no is 2 -> E.overall result is -0.137158 -> F.result so far is -0.29312841385727

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,1,1,3,3,3,6,7}
thetalst={ 8,0,26,18,3,13,1,35,33,17,16}
A.calculation is 1 B.integral no is 3 -> E.overall result is -0.101802 -> F.result so far is -0.3949308436347(

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,2,2,1,6,1,5,7}
thetalst={ 8,0,26,18,2,14,3,35,31,17,16}
A.calculation is 1 B.integral no is 4 -> E.overall result is -0.137158 -> F.result so far is -0.5320888862379(

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,2,2,2,5,2,5,7}
thetalst={ 8,0,26,18,2,14,2,35,32,17,16}
A.calculation is 1 B.integral no is 5 -> E.overall result is -0.120615 -> F.result so far is -0.65270364466614

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,2,2,3,3,3,5,7}
thetalst={ 8,0,26,18,2,14,1,35,33,17,16}
A.calculation is 1 B.integral no is 6 -> E.overall result is -0.0895236 -> F.result so far is -0.7422271989685

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,3,3,1,6,1,3,7}
thetalst={ 8,0,26,18,1,15,3,35,31,17,16}
A.calculation is 1 B.integral no is 7 -> E.overall result is -0.101802 -> F.result so far is -0.84402962874599

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
```

```
masslst={ 2,7,2,7,3,3,2,5,2,3,7}
thetalst={ 8,0,26,18,1,15,2,35,32,17,16}
A.calculation is 1 B.integral no is 8 -> E.overall result is -0.0895236 -> F.result so far is -0.9335531830484

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,5},{2,5},{3,4},{4,5}}
masslst={ 2,7,2,7,3,3,3,3,3,3,7}
thetalst={ 8,0,26,18,1,15,1,35,33,17,16}A.calculation is 1 B.integral no is 9 -> E.overall result is -0.066446
```

For these, the residues have the same algebraic formula when expressed in terms of areas of triangles, however these areas vary between diagrams. In the notation of [9], the residue is

$$r = -\frac{\Delta_c \Delta_c'}{\Delta^2} \tag{334}$$

With

$$\Delta = \Delta_{[2,7,7]} = 2.19285 \tag{335}$$

For example for the first diagram listed

$$\Delta_c' = \Delta_c = \Delta_{[1,6,7]} \tag{336}$$

gives

$$r = -0.15597 \tag{337}$$

and for the second with

$$\Delta_c' = \Delta_{[2,5,7]} \tag{338}$$

and Δ_c the same gives

$$r = -0.137158 \tag{339}$$

A printout of the remaining 17 diagrams are listed below:

```
nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,3},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,1,1,3,3,1,3,2}
thetalst={ 8,0,26,18,3,13,1,35,31,17,2}
A.calculation is 2 B.integral no is 1 -> E.overall result is 0.0330866 -> F.result so far is 0.033086568350148

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,4},{1,5},{2,5},{3,4},{3,5}}
masslst={ 2,7,2,7,1,1,3,3,1,2,5}
thetalst={ 8,0,26,18,3,13,35,1,31,20,17}
A.calculation is 2 B.integral no is 2 -> E.overall result is 0.0145479 -> F.result so far is 0.047634488300837
```

```
nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,2},{0,4},{1,4},{1,5},{2,5},{3,4},{3,5}}
masslst={ 2,7,2,7,1,1,5,2,1,3,3}
thetalst={ 8,0,26,18,3,13,35,2,31,19,17}
A.calculation is 2 B.integral no is 3 -> E.overall result is 0.0145479 -> F.result so far is 0.062182408251527

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,1,1,2,5,2,6,3}
thetalst={ 8,0,26,18,3,13,2,35,32,17,1}
A.calculation is 2 B.integral no is 4 -> E.overall result is 0.532089 -> F.result so far is 0.5942712944894857

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,1,1,3,3,3,6,2}
thetalst={ 8,0,26,18,3,13,1,35,33,17,2}
A.calculation is 2 B.integral no is 5 -> E.overall result is 0.394931 -> F.result so far is 0.989202138124186

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,2,2,1,6,1,5,3}
thetalst={ 8,0,26,18,2,14,3,35,31,17,1}
A.calculation is 2 B.integral no is 6 -> E.overall result is 0.532089 -> F.result so far is 1.521291024362145

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,2,2,3,3,3,5,1}
thetalst={ 8,0,26,18,2,14,1,35,33,17,3}
A.calculation is 2 B.integral no is 7 -> E.overall result is 0.426022 -> F.result so far is 1.947313072122609

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,3,3,1,6,1,3,2}
thetalst={ 8,0,26,18,1,15,3,35,31,17,2}
A.calculation is 2 B.integral no is 8 -> E.overall result is 0.394931 -> F.result so far is 2.342243915757309

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,2},{1,4},{2,5},{3,5},{4,5}}
masslst={ 2,7,2,7,3,3,2,5,2,3,1}
thetalst={ 8,0,26,18,1,15,2,35,32,17,3}
A.calculation is 2 B.integral no is 9 -> E.overall result is 0.426022 -> F.result so far is 2.768265963517773

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,4},{1,5},{2,4},{2,5},{3,5}}
masslst={ 2,7,2,7,1,1,3,3,2,2,6}
thetalst={ 8,0,26,18,3,13,35,1,20,32,17}
A.calculation is 2 B.integral no is 10 -> E.overall result is -0.137158 -> F.result so far is 2.63110792091451

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,4},{1,5},{2,4},{2,5},{3,5}}
masslst={ 2,7,2,7,2,2,3,3,1,1,5}
thetalst={ 8,0,26,18,2,14,35,1,21,31,17}
A.calculation is 2 B.integral no is 11 -> E.overall result is -0.19934 -> F.result so far is 2.431767470059729

nodelst={0,1,2,3,4,5}
```

```
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,3},{0,4},{1,4},{1,5},{2,4},{2,5},{3,5}}
masslst={ 2,7,2,7,3,3,5,2,1,1,3}
thetalst={ 8,0,26,18,1,15,35,2,21,31,17}
A.calculation is 2 B.integral no is 12 -> E.overall result is -0.0787257 -> F.result so far is 2.3530417776331

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,4},{2,5},{3,4},{3,5}}
masslst={ 2,7,2,7,1,1,2,5,2,3,3}
thetalst={ 8,0,26,18,13,3,2,35,32,19,17}
A.calculation is 2 B.integral no is 13 -> E.overall result is -0.19934 -> F.result so far is 2.153701326778342

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,4},{2,5},{3,4},{3,5}}
masslst={ 2,7,2,7,1,1,3,3,3,2,5}
thetalst={ 8,0,26,18,13,3,1,35,33,20,17}
A.calculation is 2 B.integral no is 14 -> E.overall result is -0.0787257 -> F.result so far is 2.0749756343517

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,2},{1,4},{2,5},{3,4},{3,5}}
masslst={ 2,7,2,7,2,2,1,6,1,3,3}
thetalst={ 8,0,26,18,14,2,3,35,31,19,17}
A.calculation is 2 B.integral no is 15 -> E.overall result is -0.137158 -> F.result so far is 1.93781759174848

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,3},{1,4},{2,4},{2,5},{3,5}}
masslst={ 2,7,2,7,1,1,3,3,2,2,3}
thetalst={ 8,0,26,18,13,3,1,35,20,32,17}
A.calculation is 2 B.integral no is 16 -> E.overall result is 0.0310912 -> F.result so far is 1.96890879587424

nodelst={0,1,2,3,4,5}
linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},{0,4},{0,5},{1,3},{1,4},{2,4},{2,5},{3,5}}
masslst={ 2,7,2,7,2,2,3,3,1,1,3}
thetalst={ 8,0,26,18,14,2,1,35,21,31,17}
A.calculation is 2 B.integral no is 17 -> E.overall result is 0.0310912 -> F.result so far is 2.00000000000001
```

Diagram 8 has the correct lozenge shape to start with. Evaluating the Δ's for this diagram gives the following

$$\Delta'_a = \Delta_{[1,1,2]} = 0.460802 \tag{340}$$

$$\Delta'_b = \Delta_{[1,6,7]} = 0.866025 \tag{341}$$

$$\Delta_a = \Delta_{[2,2,2]} = 0.669713 \tag{342}$$

$$\Delta_b = \Delta_{[2,7,5]} = 0.76157 \tag{343}$$

$$\Delta''_a = \Delta_{[3,3,2]} = 1.06234 \tag{344}$$

$$\Delta''_b = \Delta_{[3,3,7]} = 0.565258 \tag{345}$$

We also need to evaluate the τ factors

$$\tau\Delta = \Delta''_a\Delta'_b - \Delta'_a\Delta''_b = 0.300767 \tag{346}$$

$$\tau'\Delta = \Delta''_a\Delta_b - \Delta_a\Delta''_b = 0.196312 \tag{347}$$

$$\tau''\Delta = \Delta_a\Delta'_b - \Delta'_a\Delta_b = 0.104455 \tag{348}$$

This gives the following values for the residues (again in the order given by [9]).

$$r_1 = \frac{\Delta_a + \tau''}{\Delta} = 0.394931 \tag{349}$$

$$r_2 = \frac{\Delta'_a + \tau)(\Delta'_a + \tau'')}{\Delta\Delta'_a} = 0.426022 \tag{350}$$

$$r_3 = \frac{\Delta''_a - \tau')(\Delta''_a - \tau)}{\Delta\Delta''_b} = 0.532089 \tag{351}$$

$$r_4 = -\tau''\frac{\Delta'_a + \tau}{\Delta\Delta'_a} = -0.0787257 \tag{352}$$

$$r_5 = -\tau'\frac{\Delta''_a - \tau'}{\Delta\Delta''_b} = -0.137158 \tag{353}$$

$$r_6 = -\frac{\Delta''_a - \tau}{\Delta}\left(\frac{(\tau'')^2}{\Delta'_a(\Delta_a - \tau'')} + \frac{\tau}{\Delta''_b}\right) = -0.19934 \tag{354}$$

$$r_7 = (\tau'')^2\frac{\Delta''_a - \tau}{\Delta\Delta'_a(\Delta_a - \tau'')} = 0.0145479 \tag{355}$$

$$r_8 = (\tau')^2\frac{\Delta'_a + \tau''}{\Delta\Delta''_b(\Delta_a - \tau'')} = 0.0310912 \tag{356}$$

$$r_9 = \frac{2\tau'\tau''}{\Delta(\Delta_a - \tau'')} = 0.0330866 \tag{357}$$

On account of the ordering built into the program, the printout lists the diagrams in a different order, eg r_1 corresponds to diagram 5 and 8, r_2 to diagram 7 and 9 etc. This time the sum of the diagrams adds up to 2:

$$2r_1 + 2r_2 + 2r_3 + 2r_4 + 2r_5 + 2r_6 + 2r_7 + 2r_8 + r_9 = 2 \tag{358}$$

6.1 Tables of Poles

We list only the table for d_6 poles, because since a mistake was discovered in the input data for e_6 e_7 and e_8 we have not listed them.

The notation that we will use is $\begin{array}{|c|c|}\hline \mathrm{O} & \theta \\ \hline \mathrm{R} & \mathrm{D} \\ \hline \end{array}$

Where

- O is the order of the pole
- θ is the angle at which the pole occurs
- R is the result of the calculation
- D is the number of diagrams calculated

Table 5: Poles in d_6

	1	2	3	4	s	s'
1				2 $\frac{5\pi}{10}$ 1 2		
2		2 $\frac{2\pi}{10}$ 1 3 2 $\frac{8\pi}{10}$ 1 3	2 $\frac{3\pi}{10}$ 1 4 2 $\frac{5\pi}{10}$ 1 2 2 $\frac{7\pi}{10}$ 1 4	3 $\frac{4\pi}{10}$ -i 21 3 $\frac{6\pi}{10}$ i 21	2 $\frac{5\pi}{10}$ 1 4	2 $\frac{5\pi}{10}$ 1 4
3		2 $\frac{3\pi}{10}$ 1 4 2 $\frac{5\pi}{10}$ 1 2 2 $\frac{7\pi}{10}$ 1 4	3 $\frac{4\pi}{10}$ -i 15 3 $\frac{6\pi}{10}$ i 15	4 $\frac{5\pi}{10}$? 214	2 $\frac{4\pi}{10}$ 1 3 2 $\frac{6\pi}{10}$ 1 3	2 $\frac{4\pi}{10}$ 1 3 2 $\frac{6\pi}{10}$ 1 3
4	2 $\frac{5\pi}{10}$ 1 2	3 $\frac{4\pi}{10}$ -i 21 3 $\frac{6\pi}{10}$ i 21	4 $\frac{5\pi}{10}$? 214	4 $\frac{4\pi}{10}$? 298 4 $\frac{6\pi}{10}$? 298	2 $\frac{3\pi}{10}$ 1 3 2 $\frac{5\pi}{10}$ 1 4 2 $\frac{7\pi}{10}$ 1 3	2 $\frac{3\pi}{10}$ 1 3 2 $\frac{5\pi}{10}$ 1 4 2 $\frac{7\pi}{10}$ 1 3
s		2 $\frac{5\pi}{10}$ 1 4	2 $\frac{4\pi}{10}$ 1 3 2 $\frac{6\pi}{10}$ 1 3	2 $\frac{3\pi}{10}$ 1 3 2 $\frac{5\pi}{10}$ 1 4 2 $\frac{7\pi}{10}$ 1 3		
s'		2 $\frac{5\pi}{10}$ 1 4	2 $\frac{4\pi}{10}$ 1 3 2 $\frac{6\pi}{10}$ 1 3	2 $\frac{3\pi}{10}$ 1 3 2 $\frac{5\pi}{10}$ 1 4 2 $\frac{7\pi}{10}$ 1 3		

For the fourth order poles the S_{46} matrix element at $\frac{5\pi}{12}$ has over 11000 diagrams

References

[1] D.Z. Albert Quantum Mechanics and Experience (See also Wheeler and Zurek, Chap 6).

[2] L.E. Ballentine *Quantum Mechanics* Prentice Hall 1990.

[3] G. Birkhoff and J. Von-Neuman *An Math 37 (1936) 823.*

[4] N. N. Bogolubov, A. A. Logunov, I. T. Todorov *Introduction to Axiomatic Quantum Field Theory* W. A. Benjamin, Advanced Book Program, 1975.

[5] H.W. Braden, E. Corrigan, P.E. Dorey, R. Sasaki: Affine Toda Field Theory: S-Matrix vs Perturbation: Second Workshop on Quantum Groups 1990.

[6] H.W. Braden, R. Sasaki *Affine Toda Purturbation Theory* YITP/U-01-40 (Oct 91).

[7] H.W. Braden, H.S. Cho, J.D. Kim, I.G, Koh, R. Sasaki *Singularity analysis in A_n Affine Toda Theories* Edinburgh-/91-92/01.

[8] H.W. Braden, E.Corrigan, P.E. Dorey, R.Sasaki: *Affine Toda Field Theory and Exact S-Matrices:* Nuclear Physics B338(1990) 689-746.

[9] H.W. Braden, E.Corrigan, P.E. Dorey, R.Sasaki: *Multiple Poles and other Features of Affine Toda Fiels Theory:* Nuclear Physics B356(1991) 469-498.

[10] M. Bruschi, D. Levi, M.A. Olshanetsky, A.M. Perelomov, R. Ragnisco *The Quantum Toda Lattice.*

[11] M. Brusch, O. Ragnisco *Lax Representation and Complete Integrability for the Periodic Toda Lattice* Phys Lett A 134 No 6 (1989) pp365-

[12] R.K. Bullough and P.J. Caudrey *Topics in Current Physics 17: Solitons* Berlin, New York, Springer-Verlag, 1976.

[13] B.G. Buchanan *Rule-based expert systems: the MYCIN experiments of the Stannford Heuristic programming Project* Imprint 1977.

[14] E. Cartan Collected Papers, Gauthier Villars 1955.

[15] A. Church *Introduction to Mathematical Logic* Princeton University Press, 1956.

[16] D. Deutch, R. Jozsa *Rapid solution of problems by quantum computation* Proc Royal Soc London, 439, 553-558 (1992).

[17] P. Dorey *Root systems and Purely Elastic S Matrices* SPhT/90-169 (1990), Nuclear Physics B 1991 Vol.358 No.3 pp.654-676
Root systems and Purely Elastic S Matrices II Saclay 91191.

[18] M.A. Ellis, B. Stroustrup *The annotated C++ reference manual* Addison-Wesley, 1991.

[19] H. Flaschka, Comm Math Phys vol.76 p.65 (1980).

[20] N. Ford *PROLOG Programming* Wiley, 1989.

[21] M.D. Freeman *On the Mass Spectrum of Affine Toda Field Theory* Phys Lett B261 (1991).

[22] N. Ganoulis *Quantum Toda Systems and Lax Pairs* Comm Math Phys 109, 23-32 (1987).

[23] R.W. Garden *Modern Logic and Quantum Mechanics* Adam Hilger 1984.

[24] G. Gentenzen *An Investigation Into Logical Deduction, in The Collected Papers of G.Gentenzen* North-Holland, 1969.

[25] R. Gilmore *Lie Groups, Lie Algebras and some of their applications* Wiley, New York, 1974.

[26] K. Gödel *The Consistancy of the Generalised Continuum Hypothosis* Princeton University Press, 1940.

[27] R.E. Grandy *Advanced Logic For Applications* D.Reidel 1977.

[28] R. Hall Thesis, Durham University 1994 (Unpublished).

[29] R. Hirota, K. Suzuki J. Phys Soc Japan, 28 1388- (1970).

[30] T. Hollowood *Quantum Solitons in Affine Toda Field Theories* PUPT-1286.

[31] T. Richards *Clausal Form Logic* Addison-Wesley 1989.

[32] G. Hughes, M. Cresswell *An Introduction to Modal Logic V1* London, Methuen, 1968.

[33] C.Itzykson and J.B.Zuber *Quantum Field Theory* McGraw-Hill New York 1980

[34] G.J. Klir, T.A. Folger *Fuzzy Sets, Uncertainty and Information* Prentice Hall, 1988.

[35] S.A. Kripke *Semantical analysis of intuistic logic.*

[36] G.L. Lamb and D.W. McLaughlin *Topics in Current Physics 17: Solitons, 2. Aspects of soliton physics* p65-106 (see 94). Berlin, New York, Springer-Verlag, 1976-

[37] L.D. Landau *On Analytic Properties of Vertex Parts in Quantum Field Theory* Nuclear Physics 13 p181-192 (1959).

[38] P.D. Lax Comm Pure Applied Math V21, p467 (1968).

[39] D.B. Lenat *The Role of Heuristics in Learning by Discovery* Machine Learning, an Artificial Inteligence approach.

[40] A. Matsuyama Periodic Toda Lattice In Quantum-Mechanics Ann Phys 220 300-334 (1992).

[41] A.V. Mikhailov, M.A. Olshanetsky, A.M. Perelomov *Two Dimensional Generalized Toda Lattice* Comm Math Phys 1981 Vol.79 No.4 pp.473-488.

[42] P. Nogueira *Automatic Feynman Graph Generation* IFM -7/91, J Comp Phys 1993 Vol.105 No.2 pp.279-289.

[43] A. Ocneanu Proc. XVIIIth Int Conference on Differential Geometrical Methods in Theoretical Physics (1989).

[44] J.E. Humphreys *Introduction to Lie Algebras and Representation Theory* Springer-Verlag 1972.

[45] R. Penrose *The Emperors New Mind* O.U.P. 1989.

[46] J.W. Robbin *Mathematical Logic* W.A.Benjamin, New York 1969.

[47] A.N. Whitehead, B. Russell *Principia Mathematica* Cambridge University Press.

[48] R. Rajaraman *Some non-purturbative methods in quantum field theory* Phys Rep 21C 227 (1975).

[49] R. Spencer-Smith *Model Logic* Artificial Intelligence Review 5(1991),pp5-34.

[50] J. M. Spivey *The Z notation : a reference manual* Prentice Hall, New York 1992.

[51] A. Tarski et al *Undecidable Theories* North Holand 1953.

[52] M. Toda *Theory of Nonlinear Lattices* Springer-Verlag, Berlin Heidelburg New York 1981.

[53] D.H.D. Warren *An Abstract Prolog Instruction Set* Tech. Note 309, SI International, 1983.

[54] G.M.T. Watts R.A Weston $G_2^{(1)}$ *Affine Toda Field Theory: A Numerical Test of Exact S-Matrix Results.*

[55] G.M.T. Watts R.A Weston *Quantum Mass Corrections for* $C_2^{(1)}$ *Affine Toda Theory Solitons* hep-th/9404065 (DAMTP-94-27)

[56] V.E. Zakharov *What Is Integrability* Springer-Verlag, Berlin 1991.

[57] A.B. Zamolodchikov, A.B. Zamolodchikov: *Factorised S-Matrices in Two Dimensions as the Exact Solutions of Certain Relativistic Quantum Field Theory Models* Annals of Physics 120(1979) 253-291.

V.A. Fateev, A.B. Zamolodchikov: *Conformal Field theory and Purely Elastic S-Matrices:* Int J. Modern Phys A5(1990) 1025-1048.

Appendix A: Program pattj3 (PROLOG)

```
/* pattj3 - PROLOG program to generate feynman graphs in Toda Field Theory
   Copyright (C) Paul Bayton 1994
       This program is free software; you can redistribute it and/or
     modify it under the terms of the GNU General Public License
     as published by the Free Software Foundation; either version 2
     of the License, or (at your option) any later version.

     This program is distributed in the hope that it will be useful,
     but WITHOUT ANY WARRANTY; without even the implied warranty of
     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
     GNU General Public License for more details.

     You should have received a copy of the GNU General Public License
     along with this program; if not, write to the Free Software
     Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA. */

not(X):-trig(Y),not2(Y,X),call(X),!,fail.
not(X):-not3(X).
trig(1).
trig(2).
not2(2,X):-retractall(not3(Y)),assert(not3(X)).

/* ?-op(600,xfx,\=). */
A \= B:-not(A=B).



type([]).
type([A|B]):-put(A),type(B).
comments:-repeat,comment(X),type(X),nl,X ="end".

comment(" @>,@< installed").
/* ?-op(500,xfy,@>).
?-op(510,xfy,@<). */
A @> B:-not(integer(A)),not(integer(B)),name(A,C),name(B,D),gt(C,D),!.
A @> B:-integer(A),integer(B),A > B,!.
A @> B:-integer(A),not(integer(B)),!.
gt([A|B],[C|D]):-A>C ; A=C,gt(B,D).
gt([A],[]).



comment("pnode(x) - possible node accessable").
pnode(1).
pnode(2).
pnode(3).
pnode(4).
pnode(5).
pnode(6).

comment("rhead(list,element,newlist) - removes all elements in list to elmnt").
rhead([A|B],A,B).
```

```
rhead([A|B],E,R):-rhead(B,E,R).

comment("rtail(list,element,new) - removes tail to element").
rtail([A|B],A,[]).
rtail([A|B],E,[A|C]):-rtail(B,E,C).

comment("reverse(list,rlist) reverses list").
:- op(500,xfx,reverse).
R1 reverse L:-reverse(L,[],R1).
reverse([],C,C).
reverse([A|B],C,D):-reverse(B,[A|C],D).

comment("esplit(A,b,c) - unknown (like rtail?)").
esplit([A|B],C,[A|E]):- esplit(B,C,E).
esplit([A],A,[]):-atom(A).

comment("member(elmnt,list) installed").
member(A,[A|C]).
member(A,[B|C]):-member(A,C).

comment("append(element,list,newlist) - installed").
append(E,L,[E|L]).

comment("collectFN - collects all vals of fn and asserts them in globallist").
max(0).
collectpnode:-pnode(A),max(N),M is N+1,retractall(max(X)),asserta(max(M)),
assertz(globallist(M,A)),fail.
 collectdios:-main(A),max(N),M is N+1,retractall(max(X)),asserta(max(M)),
assertz(globallist(M,A)),fail.

comment("baglist(1,list) - collects data in list fn and stores in [list]").
baglist(N,[A|B]):-globallist(N,A),M is N+1,baglist(M,B).
baglist(N,[]):-M is N-1,max(M).

comment("writeout(fn) - writes out fn").
write2(1,X):-write(X).
write2(2,X).
trig(1). trig(2). trig2(2).
writeout(F):-L1 =.. [F,A],
trig(Y),call(L1),write2(Y,A),nl,trig2(Y),!.

ttest(F):-L =.. [F,A],trig(Y),call(L),trig2(Y),!.

comment("part(unusednodelist,chain,new.u.l.l) - partitions list").
part([A|B],[A|C],D):-part(B,C,D).
part(A,[],A).

comment("removex(list,elmnt,newlist) - removes elmnt from list").
removex([E|B],E,B).
removex([A|B],E,[A|F]):-E \=A,removex(B,E,F).

comment("removex(list,elmnt,elmnt,newlist) - removes 2 elmnts from list").
removex([A|B],E,F,[A|Z]):-E \=A,F \=A,removex(B,E,F,Z).
removex([A|B],A,F,Z):-removex(B,F,Z).
removex([A|B],E,A,Z):-removex(B,E,Z).
```

```
comment("join(list,list,catlist) - joins two lists").
join([A|B],C,[A|D]):-join(B,C,D).
join([],C,C).


comment("nspare(poly,goodnodes,n) - spare points on poly").
nspare([A|B],G,N):-not(atom(A)),nspare2(A,G,M),nspare(B,G,P),N is M+P.
nspare2([A|B],G,N):-atom(A),noin(B,G,N).
noin([A|B],G,N):-member(A,G),noin(B,G,M),N is M+1.
noin([],G,0).

comment("getab(poly,a,b,goodnodes,list1,list2,list3) - gets end points and split list").
getab([[P|Q]|R],A,B,G,[[P|C1]],[C5|L6],L3):-geta(Q,A,G,C1,C5),
getbb(R,B,G,L6,L3).
getab([[P|Q]|R],A,B,G,[[P|Q]|L1],L2,L3):-getab(R,A,B,G,L1,L2,L3).

getbb([[P|Q]|R],B,G,[[P|C1]],[C2|R]):-geta(Q,B,G,C1,C2).
getbb([[P|Q]|R],B,G,[[P|Q]|L1],L2):-getbb(R,B,G,L1,L2).
geta([A|Q],A,G,[],[A|Q]):-member(A,G).
geta([P|Q],A,G,[P|X],Y):-geta(Q,A,G,X,Y).

comment("extend(poly,unusednodes,goodnodes,newpolys,newunused,newgood) - extend poly").
extend(P,U,G,[Pp1,Pp2],Nu,Ng):-getab(P,A,B,G,L1,L2,L3),
removex(G,A,G2),removex(G2,B,Ng),
part(U,Link1,Nu),append(A,Link1,Chain1),
Link2 reverse Link1,append(B,Link2,Chain2),
join(L1,[Chain1],L4),join(L4,L3,Pp1),
join(L2,[Chain2],Pp2).


comment("new(polylist,polylistdio,newnodes,goodnodes) - main dio gen").
new(P,[P],[],[]).
new(P,[P1|D],N,G):-extend(P,N,G,[P1,P2],Nn,Ng),new(P2,D,Nn,Ng).

comment("main(polylist) - main program").
main([[Ep1],[Ep2],[Ep3],[Ep4]|P]):-N1=[1,2,3,4,5,6,7,8],G1=N1,
part(N1,L1,N2),part(N2,L2,N3),part(N3,L3,N4),part(N4,L4,N5),
append(e1,L1,C1),append(e2,L2,C2),append(e3,L3,C3),append(e4,L4,C4),
new([C1,C2,C3,C4],P,N5,G1),
join(C1,[e2],Ep1),
join(C2,[e3],Ep2),
join(C3,[e4],Ep3),
join(C4,[e1],Ep4).

comment("genlist(list) - writes out all main(p) until last p = first p").
genlist(P):-repeat,main(Q),not(last(Q)),retractall(last(X)),asserta(last(Q)),
write(Q),nl,Q  = P.

comment("normpoly(chianlist,poly) - converts a chainlist to a poly:nodelist").
normpoly([A],A).
normpoly([A|B],J):-normpoly(B,C),join(A,C,J).

comment("args(list,n) - no of arguments in list").
args([],0).
```

```
args([A|B],N):-args(B,M),N is M+1.

comment("sort(unsorted,sorted) -  simple sort prog").
max([A],A,[]). max([A|B],M,[A|L]):-max(B,M,L),M>A. sort([],[]).
max([A|B],A,[M|L]):-max(B,M,L),A>=M. sort(U,[S|T]):-max(U,S,L),sort(L,T).

comment("theta(mb,mc,ma,theta:10/pi) reflexive over 1st two arguments").

ang(1,1,2,48).
ang(1,2,1,6).
ang(1,2,3,42).
ang(1,3,2,12).
ang(1,3,4,36).
ang(1,4,3,18).
ang(1,s1,s2,24).
ang(1,s2,s1,24).
ang(2,2,4,36).
ang(2,3,1,6).
ang(2,4,2,12).
ang(2,s1,s1,18).
ang(2,s2,s2,18).
ang(3,4,1,6).
ang(3,s1,s2,12).
ang(3,s2,s1,12).
ang(4,s1,s1,6).
ang(4,s2,s2,6).
ang(s1,s1,2,24).
ang(s1,s1,4,48).
ang(s1,s2,1,12).
ang(s1,s2,3,36).
ang(s2,s2,2,24).
ang(s2,s2,4,48).
tpi(120).
cox(10).
nm2(4).
s1(5).
s2(6).
pi(60).

c(1,2,1).
c(1,2,3).
c(1,3,2).
c(1,3,4).
c(1,4,3).
c(1,s1,s2).
c(1,s2,s1).
c(2,1,1).
c(2,1,3).
c(2,2,4).
c(2,3,1).
c(2,4,2).
c(2,s1,s1).
c(2,s2,s2).
c(3,1,2).
c(3,1,4).
```

```
c(3,2,1).
c(3,4,1).
c(3,s1,s2).
c(3,s2,s1).
c(4,1,3).
c(4,2,2).
c(4,3,1).
c(4,s1,s1).
c(4,s2,s2).
c(s1,1,s2).
c(s1,2,s1).
c(s1,3,s2).
c(s1,4,s1).
c(s1,s1,2).
c(s1,s1,4).
c(s1,s2,1).
c(s1,s2,3).
c(s2,1,s1).
c(s2,2,s2).
c(s2,3,s1).
c(s2,4,s2).
c(s2,s1,1).
c(s2,s1,3).
c(s2,s2,2).
c(s2,s2,4).


theta(A,B,C,T):-ang(A,B,C,T).
theta(A,B,C,T):-ang(B,A,C,T),B \= A.

comment("tri(a,b,c,d,e,f) std triangle fn").
tri(A,B,C,D,E,F):-theta(A,B,D,T1),theta(B,C,E,T2),theta(C,A,F,T3),20 is
T1+T2+T3.

comment("poly(n,list,list,sum=20) - checks if n-poly can be tiled").
poly(N,[H|L1],L2,S):-poly(N,[H|L1],L2,S,H).
poly(1,[A],[C],S,H):-theta(A,H,C,S).
poly(N,[A,B|L1],[C|L2],S,H):-N>1,theta(A,B,C,T),M is N-1,poly(M,[B|L1],L2,S2,H),
S is S2 + T.

comment("getlinks(poly,linklist,preset) - gets list of normal ordered links").
getlinks([H|T],L,S):-join([H|T],[H],H),getlinks2(H,L,S).
getlinks2([A,B|R],L,S):-getlinks2([B|R],L,S),norder(A,B,An,Bn),
member([An,Bn],L).
getlinks2([A,B|R],[[An,Bn]|L],S):-getlinks2([B|R],L,S),norder(A,B,An,Bn),
not(member([An,Bn],L)).
getlinks2([A],S,S).
norder(A,B,A,B):-A @> B. norder(A,B,B,A):-B @> A.

comment("awn(node,link,linklist) - associates links with node").
awn(N,[N,X],[[N,X]|B]).
awn(N,[X,N],[[X,N]|B]).
awn(N,L,[A|B]):-awn(N,L,B).

comment("lensort(unsorted,sorted) -  sort list by length").
```

```
min([[N,A]],[N,A],[]).
min([[N,A]|B],[M,Lm],[[N,A]|L]):-min(B,[M,Lm],L),M=<N.
min([[N,A]|B],[N,A],[[M,Lm]|L]):-min(B,[M,Lm],L),N<M.
lensort([],[]).
lensort(U,[S|T]):-min(U,S,L),lensort(L,T).

comment("lenlist(listoflists,[length,listoflists]) ").
lenlist([],[]).
lenlist([A|B],[[N,A]|C]):-length(A,N),lenlist(B,C).

comment("tilelists(dio,linklist,polylistlist) - producesall main lists").
tilelist(D,[[far1,e1],[far2,e2],[far3,e3],[far4,e4]|L],P):-
tilelist2(D,L,[],Q),lenlist(Q,R),lensort(R,P).
tilelist2([D1|Dr],L,S,[P1|Pr]):-normpoly(D1,P1),getlinks(P1,S2,S),
tilelist2(Dr,L,S2,Pr).
tilelist2([],L,L,[]).

thread([[A,B]|C],[[A,B,X]|D]):-thread(C,D).
thread([],[]).

comment("lenth(list,n) - no of elements in list").
lenth([A|B],N):-lenth(B,M),N is M+1.
lenth([],0).

comment("tileable(dio)").
tileable([X1,X2,X3,X4|D]):-tilelist(D,L,P),thread(L,T1),
tileable2(P,L,T1,Ap),tileable4(Ap),write(T1).

tileable2([P1|Pr],L,T1,[Ap1|Apr]):-
tileable3(P1,L,T1,Ap1),tileable2(Pr,L,T1,Apr).
tileable2([],L,T1,[]).

tileable3([N,[A,B,C|D]],L,T1,[N,Itl,Otl]):-
join([A,B,C|D],[A],I),join([A,B,C|D],[A,B],O),
inside(I,T1,Itl),outside(O,T1,Otl,L).

tileable4([[N,Itl,Otl]|Apr]):-
poly(N,Itl,Otl,20),
tileable4(Apr).
tileable4([]).

comment("inside(poly+head,thread,inside) - finds inside list").
inside([A,B|C],T1,[X|R]):-norder(A,B,An,Bn),member([An,Bn,X],T1),
inside([B|C],T1,R).
inside([A],T1,[]).

comment("outside(poly+1st2,thread,outside,links) ").
outside([A,B,C|D],T1,[X|R],L):-awn(B,[I,J],L),
norder(A,B,An,Bn),norder(B,C,Bm,Cm),
[I,J]\=[An,Bn],[I,J]\=[Bm,Cm],
member([I,J,X],T1),
outside([B,C|D],T1,R,L).
outside([A,B],T1,[],L).

no(A,B,C):-A =< B,B =< C.
```

```
even(0). even(X):-Y is X-2,Y >= 0,even(Y).
odd(X):-Y is X+1,even(Y).

/* c(A,B,C):-sort([A,B,C],[D,E,F]),cc(D,E,F).
cc(A,B,C):-nm2(M),A =< M,(0 is A+B-C;0 is A-B-C).
cc(A,B,C):-nm2(M),(even(M),c1(A,B,C);odd(M),c2(A,B,C)).
c1(A,B,C):-even(C),(s2(A),s2(B);s1(A),s1(B)).
c1(A,B,C):-s2(A),s1(B),odd(C).
c2(A,B,C):-s2(A),s1(B),even(C).
c2(A,B,C):-odd(C),(s2(A),s2(B);s1(A),s1(B)). */


tile([A,B,C,D,E,F]):-tpi(P),cox(H),
poly(3,[A,B,C],[D,E,F],P),H is D+E+F,
c(A,B,D),c(B,C,E),c(C,A,F),no(D,E,F).

/* new prog - all diograms */
ndlst([e1,e2,e3,e4,1,2,3,4]).
curentn([2,2,2,2,3,3,3,3]).



/* linkfrom(nodelist,currentlinks,return) */
linkfrom([A|B],[C|D],[G|H],Ext):-B \= [],C > 0,linkfrom2([A|B],[C|D],[G|H],0,Ext).

linkfrom([A|B],[0|D],H,Ext):-B \= [],linkfrom(B,D,H,Ext).

linkfrom([A],[0],[],_):-atomic(A).



linkfrom2([A|B],[C|D],[G|H],Os,Ext):-C > 1,linkto(B,D,N,Dd,Os,Os2,Ext,Ext2),
G = [A,N],Cc is C-1,linkfrom2([A|B],[Cc|Dd],H,Os2,Ext2).

linkfrom2([A|B],[1|D],[G|H],Os,Ext):-linkto(B,D,N,Dd,Os,Os2,Ext,Ext2),
G = [A,N],linkfrom(B,Dd,H,Ext2).



/* linkto(node-lint,currnt-links,node-selected,new-current-links,
offset,new-offset) */

linkto([A|B],[C|D],N,[C|Dd],Os,Osr,Ext,Extr):-B \= [],
Os2 is Os-1,Ext2 is Ext-1,
linkto(B,D,N,Dd,Os2,Osr2,Ext2,Extr2),
Osr is Osr2 + 1,Extr is Extr2+1.

linkto([A|B],[C|D],A,[Cc|D],Os,1,Ext,Ext):-C \= 0,Cc is C-1,Os =< 0,Ext > 0.

linkto([A|B],[C|D],A,[Cc|D],Os,1,0,1):-C \= 0,Cc is C-1,Os =< 0.



/* find3tuple(node,lnklist,mainlist,3tuple) */
```

```
find3tuple(N,[[N,M]|B],[C|D],[C|F]):-find3tuple(N,B,D,F).
find3tuple(N,[[M,N]|B],[C|D],[C|F]):-find3tuple(N,B,D,F).
find3tuple(N,[],[],[]).
find3tuple(N,[[P,Q]|B],[C|D],F):-N \= P,N \= Q,find3tuple(N,B,D,F).

find3tuple2(N,[[N,M]|B],[C|D],[C|F],[out|I]):-find3tuple2(N,B,D,F,I).
find3tuple2(N,[[M,N]|B],[C|D],[C|F],[in|I]):-find3tuple2(N,B,D,F,I).
find3tuple2(N,[],[],[],[]).
find3tuple2(N,[[P,Q]|B],[C|D],F,I):-N \= P,N \= Q,find3tuple2(N,B,D,F,I).

findtuple(N,[[N,M]|B],[C|D],R,[C|F]):-findtuple(N,B,D,R,F).
findtuple(N,[[M,N]|B],[C|D],R,[C|F]):-findtuple(N,B,D,R,F).
findtuple(N,[],[],[],[]).
findtuple(N,[[P,Q]|B],[C|D],[C|R],F):-N \= P,N \= Q,findtuple(N,B,D,R,F).


/* assocmass(linklist,nodelist,masslist) */
assocmass(L,N,M):-assocmass2(L,N,M).
assocmass2(L,[N1|Nr],M):-find3tuple(N1,L,M,[T1,T2,T3]),
c(T1,T2,T3),
assocmass2(L,Nr,M).



assocmass2(L,[],M).

/* assocang(linklist,nodelist,masslist,thetalist) */
assocang(L,[N1|Nr],M,T):-find3tuple2(N1,L,T,[T1,T2,T3],Io),
find3tuple(N1,L,M,[M1,M2,M3]),
(ang0ua([T1,T2,T3],[M1,M2,M3],Io);
ang1ua([T1,T2,T3],[M1,M2,M3],Io);
ang2ua([T1,T2,T3],[M1,M2,M3],Io);
ang3ua([T1,T2,T3],[M1,M2,M3],Io)),
assocang(L,Nr,M,T).
assocang(L,[],M,T).

/* ang1ua - ony one uninsantiated angle */
ang1ua([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]):-
ang1ua2([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]);
ang1ua2([T2,T3,T1],[M2,M3,M1],[I2,I3,I1]);
ang1ua2([T3,T1,T2],[M3,M1,M2],[I3,I1,I2]).

ang1ua2([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]):-
not(atomic(T1)),atomic(T2),atomic(T3),
correctang([T1,T2],[M1,M2,M3],[I1,I2,I3]),
correctang([T1,T3],[M1,M3,M2],[I1,I3,I2]).

/* ang2ua - two uninstantiated angles - instantiate both */
ang2ua([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]):-
ang2ua2([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]);
ang2ua2([T2,T3,T1],[M2,M3,M1],[I2,I3,I1]);
ang2ua2([T3,T1,T2],[M3,M1,M2],[I3,I1,I2]).

ang2ua2([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]):-
atomic(T1),not(atomic(T2)),not(atomic(T3)),
```

```
correctang([T2,T1],[M2,M1,M3],[I2,I1,I3]),
correctang([T3,T1],[M3,M1,M2],[I3,I1,I2]).

ang3ua([T1,T2,T3],[M1,M2,M3],Io):-
not(atomic(T1)),not(atomic(T2)),not(atomic(T3)).

angOua([T1,T2,T3],[M1,M2,M3],[I1,I2,I3]):-
atomic(T1),atomic(T2),atomic(T3),
correctang([T1,T2],[M1,M2,M3],[I1,I2,I3]),
correctang([T1,T3],[M1,M3,M2],[I1,I3,I2]),
correctang([T2,T3],[M2,M3,M1],[I2,I3,I1]).



:- op(10,xfx,eqrad).
A eqrad B:-tpi(Tpi),A is B,A >= 0,A < Tpi.
A eqrad B:-tpi(Tpi),A is B+Tpi,A < Tpi.
A eqrad B:-tpi(Tpi),A is B-Tpi,A >=0,A < Tpi.

correctang([T1,T2],[M1,M2,M3],Io):-
(Io=[in,out,_];Io=[out,in,_]),
theta(M1,M2,M3,Theta),
(T1 eqrad (T2 + Theta);T1 eqrad (T2 - Theta) ).

correctang([T1,T2],[M1,M2,M3],Io):-
(Io=[in,in,_];Io=[out,out,_]),
theta(M1,M2,M3,Theta),pi(Pi),
(T1 eqrad (T2 + Pi - Theta) ; T1 eqrad (T2 - Pi + Theta) ).


allinst([A|B]):-atomic(A),allinst(B).
allinst([]).

checkang(L,N,M,T):-checkang2(L,N,M,T),assocang(L,N,M,T).

checkang2(L,N,M,T):-assocang(L,N,M,T),
((not(allinst(T)),checkang2(L,N,M,T));(allinst(T))).

rewrite(L,N,M,T,L2,[Ng|N3],M2,T2):-getgoodnode(N,L,Ng,Nr),
findtuple(Ng,L,L,Lr,Lt),
findtuple(Ng,L,M,Mr,Mt),
findtuple(Ng,L,T,Tr,Tt),
rewrite(Lr,Nr,Mr,Tr,L3,N3,M3,T3),
join(Lt,L3,L2),join(Mt,M3,M2),join(Tt,T3,T2).

rewrite([],_,[],[],[],[],[],[]).


getgoodnode([N1|Nr],[[N1,L2]|Lr],N1,Nr).
getgoodnode([N1|Nr],[[L1,N2]|Lr],N1,Nr).
getgoodnode([N1|Nr],[[L1,L2]|Lr],Ng,[N1|N2]):-getgoodnode(Nr,Lr,Ng,N2).


/* couple(list1,list2,couplist) */
couple([A|B],[C|D],[[A,C],R]).
```

```
couple([],[],[]).

comment("main2(polylist) - main program").
main2(Nl,R2,M,Th):-
Nl=[e1,e2,e3,e4],Lp=[2,2,2,2],
linkfrom(Nl,Lp,R,4),
R2=[[ef1,e1],[ef2,e2],[e3,ef3],[e4,ef4]|R],
M=[2,2,2,2|Mr],
assocmass(R2,Nl,M),Th=[72,0,72,0|Thr],
checkang(R2,Nl,M,Th).

main3:-trig(Y),main2(N,L,M,T),
type("nodelst="),write(N),nl,
type("linklst="),write(L),nl,
type("masslst="),write(M),nl,
type("thetalst="),write(T),nl,
trig2(Y),!.

testlinks([[ef1, e1], [ef2, e2], [e3, ef3], [e4, ef4], [e1, e2], [e2, e3], [e3, e4], [e4,
e1]]).
testnodes([e1,e2,e3,e4]).
testmass([2, 2, 2, 2, 3, 1, 1, 1]).
testtheta([72,0,72,0,66,78,6,54]).
testvars(L,N,M,T):-testlinks(L),testmass(M),testtheta(T),testnodes(N).


/* ncount(0).  */
count:-retract(ncount(X)),Y is X+1,assert(ncount(Y)).


comment("end").
```

Appendix B: Program cth3.C (C++)

```
/* cth3.C - A program to generate Feynman graphs In Toda Field Theory
   Copyright (C) Paul Bayton 1994
      This program is free software; you can redistribute it and/or
     modify it under the terms of the GNU General Public License
     as published by the Free Software Foundation; either version 2
     of the License, or (at your option) any later version.

     This program is distributed in the hope that it will be useful,
     but WITHOUT ANY WARRANTY; without even the implied warranty of
     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
     GNU General Public License for more details.

     You should have received a copy of the GNU General Public License
     along with this program; if not, write to the Free Software
     Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA. */


// program ctg1.C
// object: calculation of feynman diograms in toda theory

#include <iostream.h>
#include <stdio.h>

#define GROUPE7
#define ORDER 2

// set external mass and angles
int em1=4;
int ea1=36;
int em2=4;
int ea2=0;


// ******************************************************
int dino2=1;

#ifdef GROUPD6
const cox=10; // coxeter number
const Mmasses=6;
const int PI=60;
#endif
#ifdef GROUPE6
const cox=12;
const Mmasses=6;
const PI=12;
#endif
#ifdef GROUPE7
const cox=18;
const Mmasses=7;
const PI=18;
#endif
#ifdef GROUPE8
```

```
const cox=30;
const Mmasses=8;
const PI=30;
#endif

#if ORDER==2
 const Mnodes=4, Mlinks=4; // second order
 char* nodestr="nodelst={0,1,2,3}";
#endif
#if ORDER==3
 const Mnodes=6, Mlinks=7;  // third order
 char* nodestr="nodelst={0,1,2,3,4,5}";
#endif
#if ORDER==4
 const Mnodes=8, Mlinks=10;  // fourth order
 char* nodestr="nodelst={0,1,2,3,4,5,6,7}";
#endif
#if ORDER==5
 const Mnodes=10, Mlinks=13; // fith order
 char* nodestr="nodelst={0,1,2,3,4,5,6,7,8,9}";
#endif

class Angle
{
public:
int t;
enum { TwoPi=2*PI, Pi=PI }; // defines radix
Angle(int i=0) { t=(i+TwoPi) % TwoPi; };
operator int() { return (TwoPi+this->t) % TwoPi; }; // convert angle to int
//Angle operator=(Angle a) { t=(a.t+TwoPi) % TwoPi; return *this; };
int operator==(int i) { return t==((i+TwoPi) % TwoPi); };
//Angle operator+(Angle a) { Angle b; b.t=(t+a.t) % TwoPi; return b; };
//Angle operator-(Angle a) { Angle b; b.t=(TwoPi+t-a.t) % TwoPi; return b; };
};

// change also Mass::prep external masses and Theta::prep external angles
// for second order, change opto=4 to upto=3 in Dio::make (186)
// change also p("nodelst=..... for particular order.

int pass;

class Dio {
public:
static int used[Mlinks];
static Dio *pd,*pdstart,*pdend;
int *start,*end,upto;
int search1(int i); // search used for non-zero starting at i
int update(),rest(); // make another to link
static int bd,before[5000][Mlinks+1][2];
static permlst[25][Mnodes+1];
static int init;
static int list[Mlinks+1][2]; // main linklist
int Dio::check();
void Dio::gen_permed_dios();
void gen_perms();
```

```
int make(); // main program-needs first link only
void prep(Dio* d);
};


int Dio::check() {
// check weather or not the current dio has been produced before with permutated nodes
// 1:true=>ok (not been before) 0=>fail (before)
int i,j,ok2;
for(i=0;i<bd;i++) {
ok2=1;
for(j=0;j<Mlinks && ok2;j++) {
 if(list[j][0]!=before[i][j][0]) ok2=0;
 if(list[j][1]!=before[i][j][1]) ok2=0;
 };
if(ok2) return 0;
};
return 1;
};



void Dio::gen_permed_dios() { // generate a list of dios which are permuted from current l
ist
int i,j,k,n,p,p3;
int list2[Mlinks+1][2],list3[Mlinks+1][2];
int used[Mlinks+1]; // once a link is used append to list3 and dont use again
int plist[Mlinks+1][2]; // partial list with plist[i][0]=node in common
int gooddio,t0,t1,u;
Dio tmp;
// go through node permutation list permuting list and storing in before
int mperm=24;
for(i=0;i<mperm;i++) {
gooddio=1;
p3=0;
 for(j=0;j<Mlinks;j++) {
  list2[j][0]=permlst[i][ list[j][0] ];
  list2[j][1]=permlst[i][ list[j][1] ];
  }
 for(j=0;j<Mlinks;j++) used[j]=1;
 for(n=0;n<Mnodes;n++) {

 // find set of links corresponding to node n
 p=0;
 for(j=0;j<Mlinks;j++) {
  if((n==list2[j][0] || n==list2[j][1]) && used[j]) {
    plist[p][0]=n;
    if(n==list2[j][0]) plist[p][1]=list2[j][1]; else plist[p][1]=list2[j][0];
    used[j]=0;
    p++; }
  }
 // now sort as (n,p)(n,q) with p<q
 for(j=0;j<p;j++) for(k=j;k<p;k++)
 if(plist[j][1] > plist[k][1]) {
  t0=plist[j][0]; t1=plist[j][1];
```

```
 plist[j][0]=plist[k][0]; plist[j][1]=plist[k][1];
 plist[k][0]=t0; plist[k][1]=t1;
 }

// append links to list3- a sorted version of list2 with node first.
for(j=0;j<p;j++) {
 list3[p3][0]=plist[j][0];
 list3[p3][1]=plist[j][1];
 p3++;
 };
                // cout<<"n is"<<n<<",p is"<<p<<",p3 is"<<p3<<"\n";
}; // end of for n

// check list3 for i>j in (i,j) and u order
u=4; gooddio=1;
for(j=0;j<Mlinks;j++) {
 if(list3[j][0]>list3[j][1]) gooddio=0;
 if(list3[j][1]==u+1) u++;
 if(list3[j][1]>u) gooddio=0;
 };

// store dio on before
if(gooddio) {
 for(j=0;j<Mlinks;j++) {
  before[bd][j][0]=list3[j][0];
  before[bd][j][1]=list3[j][1];
  };
 bd++;
 };
                    /* cout<<gooddio<<flush;

cout << "linklst={{ef0,0},{ef1,1},{e2,2},{e3,3},";
for(j=0;j<Mlinks;j++) {
cout << "{"<< list3[j][0] << "," << list3[j][1] << "}";
if(j<Mlinks-1) cout << ",";
};
cout << "}\n"<<flush;     */


 }; // end of for i (permutations)
};


void Dio::gen_perms() {
// generate perm list permute nodes 4,5,6..Mnodes-1 not 0,1,2,3
int i,j,pl,p,q,m;
pl=0;
for(i=0;i<Mnodes;i++) permlst[pl][i]=i;
for(;;) {
for(i=0;i<Mnodes;i++) permlst[pl+1][i]=permlst[pl][i];
pl++;
p=Mnodes-1;
do {
p--;
do {
```

```
permlst[pl][p]++;
m=0;
for(j=4;j<p;j++) if(permlst[pl][j]==permlst[pl][p]) m=1;
} while(m && permlst[pl][p]<Mnodes);
} while(permlst[pl][p]>=Mnodes && p>=4);
if(p<4) return;
p++;
for(q=p;q<Mnodes;q++) {
permlst[pl][q]=3;
do {
permlst[pl][q]++;
m=0;
for(j=4;j<q;j++) if(permlst[pl][j]==permlst[pl][q]) m=1;
} while(m);
};
};
};


struct Massdat {public: int l1,l2,l3,ninst,node,depth; };

class Mass:public Massdat
{
private:
enum { Mp1d=64, Mp2d=10 }; // *implimentation limit, ok for e8
static int c[Mmasses+1][Mmasses+1][Mmasses+1];
static struct S {int first,second; } p1[Mmasses+1][Mp1d];
static int p2[Mmasses+1][Mmasses+1][Mp2d];
static int p1max[Mmasses+1],p2max[Mmasses+1][Mmasses+1];
void gen();
void gen_p();
int update(),rest();
public:
enum { Mpd=50 }; // *impimentation limit max 2*propagators
static Mass *pdstart,*pdend,*pd;
int *m1,*m2,*m3;
static int init;
Mass() {gen(); gen_p(); };
void debug();
int make();
void prep(Dio& d,Mass*m);
static int list[Mlinks+4];
};


class Theta:public Massdat
{
public:
Angle t1,t2,t3;
int io1,io2,io3;
static Theta *pd,*pdstart,*pdend;
static int store[500][4],sd; // *implimentation limit, max number of couplings
static int th[Mmasses][Mmasses][Mmasses];
int update(),rest();
```

```
void set(int,int,int,int),gen();
public:
static int init;
int make();
static int list[Mlinks+4];
void prep(Dio& d,Mass*m,Theta*t);
Theta() { gen(); };
};

// ********************************************** end of headers

// overload p; // print functions
void p(int i) { cout<<i<<'\n'<<flush; };
void p(char* c) { cout<<c<<'\n'<<flush; };
void p(int*x,int n) {
cout<<"list :";
for(int i=0;i<n;i++) cout<<*(x+i)<<",";
cout<<".\n"<<flush;
};


ostream& operator<<(ostream& s,Dio&d) {
s << "linklst={{ef0,0},{ef1,1},{ef2,2},{ef3,3},";
for(int i=0;i<Mlinks;i++) {
s << "{"<< d.list[i][0] << "," << d.list[i][1] << "}";
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
};

#ifdef GROUPD6
ostream& operator<<(ostream& s,Mass&m) {
char*s1="s1",*s2="s2";
s << "masslst={ "<<m.list[Mlinks]<<","<<m.list[Mlinks+1]<<
","<<m.list[Mlinks+2]<<","<<m.list[Mlinks+3]<<",";
for(int i=0;i<Mlinks;i++) {
if(m.list[i]==Mmasses) s<<"s2"; else if(m.list[i]==Mmasses-1) s<<"s1"; else s<<m.list[i];
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
};
#endif

#ifdef GROUPE6
ostream& operator<<(ostream& s,Mass&m) {
s << "masslst={ "<<m.list[Mlinks]<<","<<m.list[Mlinks+1]<<
","<<m.list[Mlinks+2]<<","<<m.list[Mlinks+3]<<",";
for(int i=0;i<Mlinks;i++) {
s<<m.list[i];
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
```

```
};
@endif

#ifdef GROUPE7
ostream& operator<<(ostream& s,Mass&m) {
s << "masslst={ "<<m.list[Mlinks]<<","<<m.list[Mlinks+1]<<
","<<m.list[Mlinks+2]<<","<<m.list[Mlinks+3]<<",";
for(int i=0;i<Mlinks;i++) {
s<<m.list[i];
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
};
#endif

#ifdef GROUPE8
ostream& operator<<(ostream& s,Mass&m) {
s << "masslst={ "<<m.list[Mlinks]<<","<<m.list[Mlinks+1]<<
","<<m.list[Mlinks+2]<<","<<m.list[Mlinks+3]<<",";
for(int i=0;i<Mlinks;i++) {
s<<m.list[i];
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
};
#endif


ostream& operator<<(ostream& s,Theta&t) {
s << "thetalst={ "<<t.list[Mlinks]<<","<<t.list[Mlinks+1]<<
","<<t.list[Mlinks+2]<<","<<t.list[Mlinks+3]<<",";
for(int i=0;i<Mlinks;i++) {
s<<t.list[i];
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
};

/*
ostream& operator<<(ostream& s,Theta&t) {
s << "thetalst={ "<< ((2*PI)-t.list[Mlinks]) % (2*PI)<<","<< ((2*PI)-t.list[Mlinks+1]) % (
2*PI)<<
","<< ((2*PI)-t.list[Mlinks+2]) % (2*PI)<<","<<  ((2*PI)-t.list[Mlinks+3]) % (2*PI)<<",";
for(int i=0;i<Mlinks;i++) {
s<< ((2*PI)-t.list[i]) % (2*PI);
if(i<Mlinks-1) s << ",";
};
s << "}\n"<<flush;
return s;
};
*/
```

```
// ********************************************** end of print functions


inline int Dio::search1(int i)
 { while((!pd->used[i]) && (i < Mnodes)) i++; return i; }

void Dio::prep(Dio* d) {
for(int i=0;i<Mlinks+2;i++) {
pd=d+i;
pd->start=&list[i][0];
pd->end=&list[i][1];
*pd->start=*pd->end=0;
used[i]=3;
};
pdend=d+(Mlinks-1);
pdstart=d;
init=1;
};

int Dio::update()
{
// start is the same; end needs updateing
// p("debug:update"); cout<<int(pd-pdstart)<<*start<<*end<<upto<<":";
// cout<<used[0]<<used[1]<<used[2]<<used[3]<<used[4]<<used[5]; p("uused");
*end=search1(*end+1);
if(*end>upto) {
if(this==pdstart) { return 2; }
else { pd--; used[*pd->start]++; used[*pd->end]++; return 1; } };
if(pd>=pdend) { p("debug:1"); return 0; }
pd++; // setup used,upto,start for next call
used[*start]--; used[*end]--;
if(*end==upto && upto<(Mnodes-1)) pd->upto=upto+1; else pd->upto=upto;
if(used[*start]) { *pd->start=*start; *pd->end=search1(*end+1); }
else { *pd->start=search1(*start); *pd->end=search1(*pd->start+1); };
int i=pd->rest();
return i;
};

int Dio::rest()
{ // p("debug:rest"); cout<<int(pd-pdstart)<<*start<<*end<<upto<<":";
// cout<<used[0]<<used[1]<<used[2]<<used[3]<<used[4]<<used[5]; p("rused");
if(*end>upto) {
if(this==pdstart) { return 2; }
else { pd--; used[*pd->start]++; used[*pd->end]++; return 1; } };
if(pd>=pdend) { return 0; }
pd++; // setup used,upto,start for next call
used[*start]--; used[*end]--;
if(*end==upto && upto<(Mnodes-1)) pd->upto=upto+1; else pd->upto=upto;
if(used[*start]) { *pd->start=*start; *pd->end=search1(*end+1); }
else { *pd->start=search1(*start); *pd->end=search1(*pd->start+1); };
int i=pd->rest();
return i;
};

int Dio::make()
```

```
{
int l;
if(init) { init=0; pd=pdstart; *pd->start=0; *pd->end=1;
#if ORDER==2
pd->upto=3;  // >>>upto=3/4
#elif ORDER==3
pd->upto=4;
#elif ORDER==4
pd->upto=4;
#elif ORDER==5
pd->upto=4;
#endif

l=pd->rest();
if(l==0) return 1;
if(l==2) return 0;
};
for(;;) {
l=pd->update();
if(l==0) return 1;
if(l==2) return 0;
};
return 0;
};


// ************************************************ end of Dio functions

#ifdef GROUPD6
void Mass::gen()
{
int i,j,k;
int s1=Mmasses-1, s2=Mmasses;
for(i=1;i<=Mmasses;i++)
 for(j=1;j<=Mmasses;j++)
  for(k=1;k<=Mmasses;k++) c[i][j][k]=0;
// d_n n=even:
for(i=1;i<=Mmasses-2;i++)
 for(j=1;j<=Mmasses-2;j++)
  for(k=1;k<=Mmasses-2;k++) {
if(i+j-k==0) c[i][j][k]=1;
if(i-j+k==0) c[i][j][k]=1;
if(i-j-k==0) c[i][j][k]=1;
if(i+j+k==cox) c[i][j][k]=1;
   };
for(i=2;i<=Mmasses-2;i++,i++) {
c[i][s1][s1]=1; c[i][s2][s2]=1;
c[s1][i][s1]=1; c[s2][i][s2]=1;
c[s1][s1][i]=1; c[s2][s2][i]=1;
 };
for(i=1;i<=Mmasses-2;i++,i++) {
c[i][s1][s2]=1; c[i][s2][s1]=1;
c[s1][i][s2]=1; c[s2][i][s1]=1;
c[s1][s2][i]=1; c[s2][s1][i]=1;
};
```

```
// symerterise:
for(i=1;i<=Hmasses;i++)
 for(j=1;j<=Hmasses;j++)
  for(k=1;k<=Hmasses;k++)
   if(c[i][j][k]) c[i][j][k]=c[i][k][j]=c[j][i][k]=c[j][k][i]=c[k][i][j]=c[k][j][i]=1;
};  // of Hass::gen
#endif


#ifdef GROUPE6
void Hass::gen()
{
int i,j,k;
for(i=1;i<=Hmasses;i++)
 for(j=1;j<=Hmasses;j++)
  for(k=1;k<=Hmasses;k++) c[i][j][k]=0;

c[1][1][2]=1;
c[1][1][4]=1;
c[1][2][3]=1;
c[1][3][1]=1;
c[1][3][4]=1;
c[1][4][2]=1;
c[1][4][5]=1;
c[1][5][3]=1;
c[1][5][6]=1;
c[1][6][4]=1;
c[2][2][1]=1;
c[2][2][4]=1;
c[2][3][2]=1;
c[2][3][5]=1;
c[2][4][3]=1;
c[2][4][6]=1;
c[2][5][1]=1;
c[2][5][4]=1;
c[2][6][5]=1;
c[3][3][3]=1;
c[3][3][6]=1;
c[3][4][1]=1;
c[3][5][2]=1;
c[3][6][3]=1;
c[3][6][6]=1;
c[4][4][2]=1;
c[4][4][5]=1;
c[4][5][6]=1;
c[4][6][1]=1;
c[4][6][4]=1;
c[5][5][1]=1;
c[5][5][4]=1;
c[5][6][2]=1;
c[5][6][5]=1;
c[6][6][3]=1;
c[6][6][6]=1;

// symerterise:
```

```
for(i=1;i<=Hmasses;i++)
 for(j=1;j<=Hmasses;j++)
  for(k=1;k<=Hmasses;k++)
   if(c[i][j][k]) c[i][j][k]=c[i][k][j]=c[j][i][k]=c[j][k][i]=c[k][i][j]=c[k][j][i]=1;
};
#endif

#ifdef GROUPE7
void Hass::gen()
{
int i,j,k;
for(i=1;i<=Hmasses;i++)
 for(j=1;j<=Hmasses;j++)
  for(k=1;k<=Hmasses;k++) c[i][j][k]=0;

c[1][1][2]=1;
c[1][1][4]=1;
c[1][2][1]=1;
c[1][2][3]=1;
c[1][3][2]=1;
c[1][3][4]=1;
c[1][3][5]=1;
c[1][4][1]=1;
c[1][4][3]=1;
c[1][4][6]=1;
c[1][5][3]=1;
c[1][5][6]=1;
c[1][6][4]=1;
c[1][6][5]=1;
c[1][6][7]=1;
c[1][7][6]=1;
c[2][2][2]=1;
c[2][2][4]=1;
c[2][2][5]=1;
c[2][3][1]=1;
c[2][3][3]=1;
c[2][3][6]=1;
c[2][4][2]=1;
c[2][4][5]=1;
c[2][5][2]=1;
c[2][5][4]=1;
c[2][5][7]=1;
c[2][6][3]=1;
c[2][7][5]=1;
c[2][7][7]=1;
c[3][3][2]=1;
c[3][3][7]=1;
c[3][4][1]=1;
c[3][5][1]=1;
c[3][5][6]=1;
c[3][6][2]=1;
c[3][6][5]=1;
c[3][6][7]=1;
c[3][7][3]=1;
c[3][7][6]=1;
```

```
c[4][4][4]=1;
c[4][4][5]=1;
c[4][4][7]=1;
c[4][5][2]=1;
c[4][5][4]=1;
c[4][5][7]=1;
c[4][6][1]=1;
c[4][6][6]=1;
c[4][7][4]=1;
c[4][7][5]=1;
c[5][5][5]=1;
c[5][6][1]=1;
c[5][6][3]=1;
c[5][7][2]=1;
c[5][7][4]=1;
c[5][7][7]=1;
c[6][6][4]=1;
c[6][6][7]=1;
c[6][7][1]=1;
c[6][7][3]=1;
c[6][7][6]=1;
c[7][7][2]=1;
c[7][7][5]=1;
c[7][7][7]=1;



// symerterise:
for(i=1;i<=Mmasses;i++)
 for(j=1;j<=Mmasses;j++)
  for(k=1;k<=Mmasses;k++)
   if(c[i][j][k]) c[i][j][k]=c[i][k][j]=c[j][i][k]=c[j][k][i]=c[k][i][j]=c[k][j][i]=1;
};
#endif

#ifdef GROUPE8
void Mass::gen()
{
int i,j,k;
for(i=1;i<=Mmasses;i++)
 for(j=1;j<=Mmasses;j++)
  for(k=1;k<=Mmasses;k++) c[i][j][k]=0;



c[1][1][1]=1;
c[1][1][2]=1;
c[1][1][3]=1;
c[1][2][1]=1;
c[1][2][2]=1;
c[1][2][3]=1;
c[1][2][4]=1;
c[1][3][1]=1;
c[1][3][2]=1;
c[1][3][4]=1;
```

```
c[1][3][5]=1;
c[1][4][2]=1;
c[1][4][3]=1;
c[1][4][4]=1;
c[1][4][5]=1;
c[1][4][6]=1;
c[1][5][3]=1;
c[1][5][4]=1;
c[1][5][6]=1;
c[1][5][7]=1;
c[1][6][4]=1;
c[1][6][5]=1;
c[1][6][7]=1;
c[1][7][5]=1;
c[1][7][6]=1;
c[1][7][8]=1;
c[1][8][7]=1;
c[1][8][8]=1;
c[2][2][1]=1;
c[2][2][2]=1;
c[2][2][4]=1;
c[2][2][5]=1;
c[2][2][6]=1;
c[2][3][1]=1;
c[2][3][3]=1;
c[2][3][6]=1;
c[2][4][1]=1;
c[2][4][2]=1;
c[2][4][7]=1;
c[2][5][2]=1;
c[2][5][6]=1;
c[2][6][2]=1;
c[2][6][3]=1;
c[2][6][5]=1;
c[2][6][7]=1;
c[2][6][8]=1;
c[2][7][4]=1;
c[2][7][6]=1;
c[2][7][7]=1;
c[2][7][8]=1;
c[2][8][6]=1;
c[2][8][7]=1;
c[3][3][2]=1;
c[3][3][3]=1;
c[3][3][5]=1;
c[3][3][6]=1;
c[3][3][7]=1;
c[3][4][1]=1;
c[3][4][5]=1;
c[3][5][1]=1;
c[3][5][3]=1;
c[3][5][4]=1;
c[3][5][7]=1;
c[3][5][8]=1;
c[3][6][2]=1;
```

```
c[3][6][3]=1;
c[3][6][6]=1;
c[3][6][8]=1;
c[3][7][3]=1;
c[3][7][5]=1;
c[3][8][5]=1;
c[3][8][6]=1;
c[3][8][8]=1;
c[4][4][1]=1;
c[4][4][4]=1;
c[4][4][6]=1;
c[4][4][7]=1;
c[4][4][8]=1;
c[4][5][1]=1;
c[4][5][3]=1;
c[4][5][5]=1;
c[4][5][8]=1;
c[4][6][1]=1;
c[4][6][4]=1;
c[4][7][2]=1;
c[4][7][4]=1;
c[4][7][7]=1;
c[4][7][8]=1;
c[4][8][4]=1;
c[4][8][5]=1;
c[4][8][7]=1;
c[5][5][4]=1;
c[5][5][5]=1;
c[5][5][8]=1;
c[5][6][1]=1;
c[5][6][2]=1;
c[5][6][7]=1;
c[5][7][1]=1;
c[5][7][3]=1;
c[5][7][6]=1;
c[5][8][3]=1;
c[5][8][4]=1;
c[5][8][5]=1;
c[5][8][8]=1;
c[6][6][3]=1;
c[6][6][6]=1;
c[6][6][8]=1;
c[6][7][1]=1;
c[6][7][2]=1;
c[6][7][5]=1;
c[6][7][8]=1;
c[6][8][2]=1;
c[6][8][3]=1;
c[6][8][6]=1;
c[6][8][7]=1;
c[7][7][2]=1;
c[7][7][4]=1;
c[7][7][7]=1;
c[7][8][1]=1;
c[7][8][2]=1;
```

```
c[7][8][4]=1;
c[7][8][6]=1;
c[7][8][8]=1;
c[8][8][1]=1;
c[8][8][3]=1;
c[8][8][5]=1;
c[8][8][7]=1;
c[8][8][8]=1;


// symerterise:
for(i=1;i<=Mmasses;i++)
 for(j=1;j<=Mmasses;j++)
  for(k=1;k<=Mmasses;k++)
   if(c[i][j][k]) c[i][j][k]=c[i][k][j]=c[j][i][k]=c[j][k][i]=c[k][i][j]=c[k][j][i]=1;
};
#endif


void Mass::gen_p()
{
int i,j,k,n;
for(i=1;i<=Mmasses;i++) {
n=0;
 for(j=1;j<=Mmasses;j++)
  for(k=1;k<=Mmasses;k++)
   if(c[i][j][k]) { p1[i][n].first=j; p1[i][n++].second=k; };
p1max[i]=n;
};
for(i=1;i<=Mmasses;i++)
 for(j=1;j<=Mmasses;j++) {
  n=0;
  for(k=1;k<=Mmasses;k++) if(c[i][j][k]) p2[i][j][n++]=k;
p2max[i][j]=n;
};
};  // of Mass::gen_p

void Mass::debug()
{
for(int i=1;i<=Mmasses;i++)
 for(int j=1;j<=Mmasses;j++)
  for(int k=1;k<=Mmasses;k++)
if(c[i][j][k]) cout << i << j << k << "\n";
};

// ****************************************** end of mass setup functions

void Mass::prep(Dio& d,Mass* m) {
pd=pdstart=m;
int i,j,k,ni;
int link[Mnodes][3];
int inst[Mlinks+4]; // dummy links Mlinks..Mlinks+3
int nused[Mnodes]; // nodes used
// 1. set inst
for(i=0;i<Mlinks;i++) inst[i]=0;
```

```
inst[Mlinks]=inst[Mlinks+1]=inst[Mlinks+2]=inst[Mlinks+3]=1;
// 1b. set mass table
for(i=0;i<Mlinks+4;i++) list[i]=0;
list[Mlinks]=list[Mlinks+2]=em1;
list[Mlinks+1]=list[Mlinks+3]=em2;
// 2. generate links per node list
for(i=0;i<Mnodes;i++) {
nused[i]=0;
k=0;
if(pass==1) if(i<4) link[i][k++]=Mlinks+i;
if(pass>1) {
if(i==0) { link[i][k++]=Mlinks; link[i][k++]=Mlinks+1; }
if(i==1) link[i][k++]=Mlinks+2;
if(i==2) link[i][k++]=Mlinks+3;
}
for(j=0;j<Mlinks;j++) {
if(d.list[j][0]==i || d.list[j][1]==i) link[i][k++]=j; }
};
// 3. determin order of evaluation and set arguments
do {
// 3a. determin node to use next
ninst=0; node=-1;
for(i=0;i<Mnodes;i++) {
ni=0;
if(inst[link[i][0]]) ni++; if(inst[link[i][1]]) ni++; if(inst[link[i][2]]) ni++;
if(ni>ninst && ni<3) { node=i; ninst=ni; };
};
if(node==-1) { i=0; while(nused[i]) i++; node=i; ninst=3; } // check only now
// 3b. find out which links are instantated and which are not
k=0;  l1=link[node][0]; l2=link[node][1]; l3=link[node][2];
if(!inst[l1] && inst[l2]) { k=l1; l1=l2; l2=k; } // mini sort of l1..l3
if(!inst[l1] && inst[l3]) { k=l1; l1=l3; l3=k; }
if(!inst[l2] && inst[l3]) { k=l2; l2=l3; l3=k; }
// 3c. set l1,l2,l3, ninst,node in ma
pd->l1=l1;
pd->l2=l2;
pd->l3=l3;
pd->node=node;
pd->ninst=ninst;
pd->m1=&list[l1];
pd->m2=&list[l2];
pd->m3=&list[l3];
// 3d. update inst
inst[link[node][0]]++; inst[link[node][1]]++; inst[link[node][2]]++;
nused[node]=1;
pd++;
j=1; for(i=0;i<Mnodes;i++) { j=j && nused[i]; } // terminating condition: all nodes are us
ed up
} while(!j);
init=1;
pdend=pd-1;
};

int Mass::update()
{ // p("debug::update"); cout<<depth<<int(pd-pdstart)<<*m1<<*m2<<*m3; p("d,pd,m1,m2,m3");
```

```
depth++;
switch(ninst) {
case 1:
if(depth>=p1max[*m1]) { if(pd==pdstart) return 2; else { pd--; return 1; } };
*m2=p1[*m1][depth].first;
*m3=p1[*m1][depth].second;
break;
case 2:
if(depth>=p2max[*m1][*m2]) { if(pd==pdstart) return 2; else { pd--; return 1; } };
*m3=p2[*m1][*m2][depth];
break;
case 3:
if(pd==pdstart) return 2; else { pd--; return 1; }
};
if(pd>=pdend) return 0;
pd++;
return pd->rest();
};

int Mass::rest()
{ // p("debug::rest"); cout<<depth<<int(pd-pdstart)<<*m1<<*m2<<*m3; p("d,pd,m1,m2,m3");
depth=0;
switch(ninst) {
case 1:
if(depth>=p1max[*m1]) { if(pd==pdstart) return 2; else { pd--; return 1; } };
*m2=p1[*m1][depth].first;
*m3=p1[*m1][depth].second;
break;
case 2:
if(depth>=p2max[*m1][*m2]) { if(pd==pdstart) return 2; else { pd--; return 1; } };
*m3=p2[*m1][*m2][depth];
break;
case 3:
if(!c[*m1][*m2][*m3]) { if(pd==pdstart) return 2; else { pd--; return 1; } };
};
if(this>=pdend) return 0;
pd++;
return pd->rest();
};

int Mass::make() {
int l;
if(init) { init=0; pd=pdstart; l=pd->rest(); // p(l);
if(l==0) return 1;
if(l==2) return 0;
}
for(;;) {
l=pd->update();
if(l==0) return 1;
if(l==2) return 0;
};
return 0;
};
```

```
// ********************************************************** end of theta functions
Dio d; Mass m; Theta t;

void Theta::set(int i,int j,int k,int l) {
store[sd][0]=i; store[sd][1]=j; // store is temporary store for data
store[sd][2]=k; store[sd][3]=l;
sd++; };

#ifdef GROUPD6
void Theta::gen() {
// data taken from prolog
sd=0;
set(1,1,2,48);
set(1,2,1,6);
set(1,2,3,42);
set(1,3,2,12);
set(1,3,4,36);
set(1,4,3,18);
set(1,5,6,24);
set(1,6,5,24);
set(2,2,4,36);
set(2,3,1,6);
set(2,4,2,12);
set(2,4,4,24);
set(2,5,5,18);
set(2,6,6,18);
set(3,3,4,24);
set(3,4,1,6);
set(3,4,3,18);
set(3,5,6,12);
set(3,6,5,12);
set(4,4,2,12);
set(4,5,5,6);
set(4,6,6,6);
set(5,5,2,24);
set(5,5,4,48);
set(5,6,1,12);
set(5,6,3,36);
set(6,6,2,24);
set(6,6,4,48);
// make p1 list
for(int i=0;i<sd;i++) {
 th[store[i][0]][store[i][1]][store[i][2]]=store[i][3]; //p1[m1,m2,m3] is angle data
 th[store[i][1]][store[i][0]][store[i][2]]=store[i][3];
};
};
#endif


#ifdef GROUPE6
void Theta::gen() {
int i,j,k;
sd=0;
```

```
set(1,1,2,4);
set(1,1,4,10);
set(1,2,3,6);
set(1,3,1,3);
set(1,3,4,7);
set(1,4,2,1);
set(1,4,5,5);
set(1,5,3,3);
set(1,5,6,9);
set(1,6,4,2);
set(2,2,1,4);
set(2,2,4,10);
set(2,3,2,3);
set(2,3,5,7);
set(2,4,3,3);
set(2,4,6,9);
set(2,5,1,1);
set(2,5,4,5);
set(2,6,5,2);
set(3,3,3,4);
set(3,3,6,10);
set(3,4,1,2);
set(3,5,2,2);
set(3,6,3,1);
set(3,6,6,5);
set(4,4,2,2);
set(4,4,5,4);
set(4,5,6,6);
set(4,6,1,1);
set(4,6,4,3);
set(5,5,1,2);
set(5,5,4,4);
set(5,6,2,1);
set(5,6,5,3);
set(6,6,3,2);
set(6,6,6,4);


for(i=0;i<sd;i++) {
 th[store[i][0]][store[i][1]][store[i][2]]=store[i][3];
 th[store[i][1]][store[i][0]][store[i][2]]=store[i][3];
}
};
#endif


#ifdef GROUPE7
void Theta::gen() {
int i,j,k;
sd=0;

set(1,1,2,8);
set(1,1,4,16);
set(1,2,1,5);
set(1,2,3,11);
```

```
set(1,3,2,4);
set(1,3,4,8);
set(1,3,5,12);
set(1,4,1,1);
set(1,4,3,7);
set(1,4,6,15);
set(1,5,3,4);
set(1,5,6,10);
set(1,6,4,2);
set(1,6,5,6);
set(1,6,7,14);
set(1,7,6,3);
set(2,2,2,6);
set(2,2,4,10);
set(2,2,5,16);
set(2,3,1,3);
set(2,3,3,7);
set(2,3,6,13);
set(2,4,2,4);
set(2,4,5,10);
set(2,5,2,1);
set(2,5,4,5);
set(2,5,7,15);
set(2,6,3,3);
set(2,7,5,2);
set(2,7,7,8);
set(3,3,2,4);
set(3,3,7,16);
set(3,4,1,3);
set(3,5,1,2);
set(3,5,6,8);
set(3,6,2,2);
set(3,6,5,6);
set(3,6,7,10);
set(3,7,3,1);
set(3,7,6,5);
set(4,4,4,6);
set(4,4,5,8);
set(4,4,7,14);
set(4,5,2,3);
set(4,5,4,5);
set(4,5,7,11);
set(4,6,1,1);
set(4,6,6,7);
set(4,7,4,2);
set(4,7,5,4);
set(5,5,5,6);
set(5,6,1,2);
set(5,6,3,4);
set(5,7,2,1);
set(5,7,4,3);
set(5,7,7,7);
set(6,6,4,4);
set(6,6,7,8);
set(6,7,1,1);
```

```
set(6,7,3,3);
set(6,7,6,5);
set(7,7,2,2);
set(7,7,5,4);
set(7,7,7,6);


for(i=0;i<sd;i++) {
 th[store[i][0]][store[i][1]][store[i][2]]=store[i][3];
 th[store[i][1]][store[i][0]][store[i][2]]=store[i][3];
}
};
#endif

#ifdef GROUPE8
void Theta::gen() {
int i,j,k;
sd=0;

set(1,1,1,10);
set(1,1,2,18);
set(1,1,3,28);
set(1,2,1,6);
set(1,2,2,12);
set(1,2,3,16);
set(1,2,4,22);
set(1,3,1,1);
set(1,3,2,9);
set(1,3,4,17);
set(1,3,5,27);
set(1,4,2,5);
set(1,4,3,9);
set(1,4,4,13);
set(1,4,5,19);
set(1,4,6,23);
set(1,5,3,2);
set(1,5,4,8);
set(1,5,6,16);
set(1,5,7,26);
set(1,6,4,5);
set(1,6,5,11);
set(1,6,7,21);
set(1,7,5,3);
set(1,7,6,7);
set(1,7,8,25);
set(1,8,7,4);
set(1,8,8,14);
set(2,2,1,6);
set(2,2,2,10);
set(2,2,4,16);
set(2,2,5,22);
set(2,2,6,28);
set(2,3,1,5);
set(2,3,3,11);
```

```
set(2,3,6,21);
set(2,4,1,3);
set(2,4,2,7);
set(2,4,7,25);
set(2,5,2,4);
set(2,5,6,14);
set(2,6,2,1);
set(2,6,3,5);
set(2,6,5,11);
set(2,6,7,17);
set(2,6,8,27);
set(2,7,4,3);
set(2,7,6,9);
set(2,7,7,13);
set(2,7,8,19);
set(2,8,6,2);
set(2,8,7,8);
set(3,3,2,8);
set(3,3,3,10);
set(3,3,5,16);
set(3,3,6,18);
set(3,3,7,26);
set(3,4,1,4);
set(3,4,5,14);
set(3,5,1,1);
set(3,5,3,7);
set(3,5,4,9);
set(3,5,7,17);
set(3,5,8,25);
set(3,6,2,4);
set(3,6,3,6);
set(3,6,6,12);
set(3,6,8,22);
set(3,7,3,2);
set(3,7,5,8);
set(3,8,5,3);
set(3,8,6,5);
set(3,8,8,13);
set(4,4,1,4);
set(4,4,4,10);
set(4,4,6,14);
set(4,4,7,18);
set(4,4,8,28);
set(4,5,1,3);
set(4,5,3,7);
set(4,5,5,11);
set(4,5,8,21);
set(4,6,1,2);
set(4,6,4,8);
set(4,7,2,2);
set(4,7,4,6);
set(4,7,7,12);
set(4,7,8,16);
set(4,8,4,1);
set(4,8,5,5);
```

```
set(4,8,7,9);
set(5,5,4,8);
set(5,5,5,10);
set(5,5,8,18);
set(5,6,1,3);
set(5,6,2,5);
set(5,6,7,13);
set(5,7,1,1);
set(5,7,3,5);
set(5,7,6,9);
set(5,8,3,2);
set(5,8,4,4);
set(5,8,5,6);
set(5,8,8,12);
set(6,6,3,6);
set(6,6,6,10);
set(6,6,8,16);
set(6,7,1,2);
set(6,7,2,4);
set(6,7,5,8);
set(6,7,8,14);
set(6,8,2,1);
set(6,8,3,3);
set(6,8,6,7);
set(6,8,7,9);
set(7,7,2,4);
set(7,7,4,6);
set(7,7,7,10);
set(7,8,1,1);
set(7,8,2,3);
set(7,8,4,5);
set(7,8,6,7);
set(7,8,8,11);
set(8,8,1,2);
set(8,8,3,4);
set(8,8,5,6);
set(8,8,7,8);
set(8,8,8,10);


for(i=0;i<sd;i++) {
 th[store[i][0]][store[i][1]][store[i][2]]=store[i][3];
 th[store[i][1]][store[i][0]][store[i][2]]=store[i][3];
}
};
#endif

// ************************************************** end of theta setup
int DINO=-1;

void Theta::prep(Dio& d,Mass*m,Theta*t) {
Mass*mpd=m;
pd=pdstart=t;
// set links
list[Mlinks]=ea1; list[Mlinks+1]=ea2;
```

```
list[Hlinks+2]=(ea1+PI) % (2*PI); list[Hlinks+3]=(ea2+PI) % (2*PI);
for(; mpd<=m->pdend ; pd++, mpd++) {
pd->l1=mpd->l1; pd->l2=mpd->l2; pd->l3=mpd->l3;   // dup all exept depth=0
pd->ninst=mpd->ninst; pd->depth=0; pd->node=mpd->node;
pd->io1=(pd->node==d.list[pd->l1][0]) ? 0 : 1;
pd->io2=(pd->node==d.list[pd->l2][0]) ? 0 : 1;
pd->io3=(pd->node==d.list[pd->l3][0]) ? 0 : 1;
// set all external links inward
if(pd->l1==Hlinks+0) pd->io1=1; if(pd->l2==Hlinks+0) pd->io2=1; if(pd->l3==Hlinks+0) pd->i
o3=1;
if(pd->l1==Hlinks+1) pd->io1=1; if(pd->l2==Hlinks+1) pd->io2=1; if(pd->l3==Hlinks+1) pd->i
o3=1;
if(pd->l1==Hlinks+2) pd->io1=1; if(pd->l2==Hlinks+2) pd->io2=1; if(pd->l3==Hlinks+2) pd->i
o3=1;
if(pd->l1==Hlinks+3) pd->io1=1; if(pd->l2==Hlinks+3) pd->io2=1; if(pd->l3==Hlinks+3) pd->i
o3=1;
pd->t1=th[*mpd->m1][*mpd->m2][*mpd->m3];
pd->t2=th[*mpd->m2][*mpd->m3][*mpd->m1];
pd->t3=th[*mpd->m3][*mpd->m1][*mpd->m2];
}; // for
init=1;
pdend=pd-1;
};

int Theta::update() {
depth++;
Angle a1,a2,a3;
Angle Pi(Angle::Pi);
a1=list[l1]; a2=list[l2]; a3=list[l3];
if(dino2==DINO) {
p("update**************************************************update");
p(this-pdstart);
p(a1); p(a2); p(a3);
p("thetas"); p(t1); p(t2); p(t3);
p("links"); p(l1); p(l2); p(l3);
p("inst"); p(ninst); }
if(io1) a1=a1+Pi;
if(io2) a2=a2+Pi;
if(io3) a3=a3+Pi;
switch(ninst) {
case 1: // 2 posibilities: choose second else fail
if(dino2==DINO) p("case1");
switch(depth) {
case 1:
a2=a1-t1+Pi;
a3=a1+t3+Pi;
break;
default: if(pd==pdstart) return 2; else {pd--; return 1; };
};
if(io2) a2=a2+Pi; //revert to actual angle
if(io3) a3=a3+Pi;
if(dino2==DINO) { p("setting"); p(a2); p(a3); }
list[l2]=a2; list[l3]=a3;
break;
case 2: // fail for next
```

```
if(dino2==DINO) p("case2");
if(pd==pdstart) return 2; else {pd--; return 1; };
case 3:
if(dino2==DINO) p("case3");
if(pd==pdstart) return 2; else {pd--; return 1; };
}; // of switch ninst
if(pd>=pdend) return 0;
pd++;
return pd->rest();
};

int Theta::rest() {
depth=0;
Angle a1,a2,a3;
Angle Pi=Angle::Pi;
a1=list[l1]; a2=list[l2]; a3=list[l3];
if(dino2==DINO) {
p("rest************************************************");
p(this-pdstart); p(a1); p(a2); p(a3);
p("thetas"); p(t1); p(t2); p(t3);
p("links"); p(l1); p(l2); p(l3);
p("inst"); p(ninst); }
if(io1) a1=a1+Pi;
if(io2) a2=a2+Pi;
if(io3) a3=a3+Pi;
switch(ninst) {
case 1: // 2 posibilities rest: 1st only.
// switch(depth) { case 0:
if(dino2==DINO) p("case1");
a2=a1+t1+Pi;
a3=a1-t3+Pi;
if(io2) a2=a2+Pi; //revert to actual angle
if(io3) a3=a3+Pi;
if(dino2==DINO) { p("setting"); p(a2); p(a3); }
list[l2]=a2; list[l3]=a3;
break;
case 2: // 1 posibility, check and assoc
if(dino2==DINO)  p("case2");
if(a2==a1+t1+Pi) { a3=a1-t3+Pi; } else {
if(a2==a1-t1+Pi) { a3=a1+t3+Pi; } else {
if(pd==pdstart) return 2; else {pd--; return 1; };
};
};
if(io3) a3=a3+Pi;
if(dino2==DINO) { p("setting"); p(a3); };
list[l3]=a3;
break;
case 3: // check only
if(!((a2==a1+t1+Pi && a3==a1-t3+Pi) ||
(a2==a1-t1+Pi && a3==a1+t3+Pi))) {
if(pd==pdstart) return 2; else {pd--; return 1; }; };
}; // of switch ninst
if(dino2==DINO) p("endofswitch");
if(pd>=pdend) return 0;
pd++;
```

```
return pd->rest();
};

int Theta::make() {
int l;  Theta*qd; int j;
if(init) { init=0; pd=pdstart; l=pd->rest();
if(l==0) {
/*
cout<<"hit-first-time\n";
for(j=0;j<=(pdend-pdstart);j++) { qd=(pdstart+j);
cout<<"node: "<<qd->node<<" links: "<<qd->l1<<","<<qd->l2<<","<<qd->l3<<
" theta: "<<list[qd->l1]<<","<<list[qd->l2]<<","<<list[qd->l3]<<
" io: "<<qd->io1<<qd->io2<<qd->io3<<
" mass: "<<m.list[qd->l1]<<","<<m.list[qd->l2]<<","<<m.list[qd->l3]<<".\n"; };
*/
return 1; }
if(l==2) return 0;
}
for(;;) {
l=pd->update();
if(l==0) {
/*
cout<<"hit-secont-time\n";
for(j=0;j<=(pdend-pdstart);j++) { qd=(pdstart+j);
cout<<"node: "<<qd->node<<" links: "<<qd->l1<<","<<qd->l2<<","<<qd->l3<<
" theta: "<<list[qd->l1]<<","<<list[qd->l2]<<","<<list[qd->l3]<<
" io: "<<qd->io1<<qd->io2<<qd->io3<<
" mass: "<<m.list[qd->l1]<<","<<m.list[qd->l2]<<","<<m.list[qd->l3]<<".\n"; };
*/
return 1;
};
if(l==2) return 0;
};
return 0;
};

// ************************************************** end of Theta functions

class Intcheck {
public:
int node;
int lnk;
int depth;
int follow;
static int init;
static int bag[Mnodes];
static int startnode;
static Intcheck *pd,*pdstart,*pdend;
static int link[Mnodes][3];
static int nlinks[Mnodes];
static int linklst[Mlinks+1][2];
int update(),rest(),loop(),main(Theta&);
void prep(Dio& d,Mass*m,Intcheck*ic);
};
```

```
void Intcheck::prep(Dio& d,Hass*m,Intcheck*ic) {
int l;
Hass *mpdstart,*mpdend,*mpd;
mpd=mpdstart=m;
mpdend=m->pdend;
for(;mpd<=mpdend;mpd++) {
node=mpd->node;
link[node][0]=mpd->l3; link[node][1]=mpd->l2; link[node][2]=mpd->l1;
}
for(node=0;node<Hnodes;node++) nlinks[node]=3;
if(pass==1) nlinks[0]=nlinks[1]=nlinks[2]=nlinks[3]=2; // first Hass::l1 is to external no
des.
if(pass>1) { nlinks[0]=1; nlinks[1]=nlinks[2]=2; }
pd=pdstart=ic;
pdend=(pdstart+Hnodes-1);
for(l=0;l<Hlinks;l++) {
linklst[l][0]=d.list[l][0];
linklst[l][1]=d.list[l][1];
}
init=2;
};


// exit status: 0:suceed, 1:fail-recall posible, 2:fail-fail.
// bag[node] only cleared on update due to depth>maxdepth

int Intcheck::update() {
int nextnode,lastnode,maxdepth;
maxdepth=nlinks[node];
depth++;
if(depth>=maxdepth) {
if(pd!=pdstart) { pd--; bag[node]=0; return 1; } else return 2;
}

lnk=link[node][depth];
follow=(linklst[lnk][0]==node);
if(follow) nextnode=linklst[lnk][1]; else nextnode=linklst[lnk][0];

if(pd!=pdstart) {
lastnode=(pd-1)->node;
if(nextnode==lastnode) return 1;
if(nextnode==startnode) { pdend=pd; return 0; }
if(bag[nextnode]) return 1;
}

pd++;
pd->node=nextnode;
bag[node]=1;
return pd->rest();
};



int Intcheck::rest() {
int nextnode,lastnode;
depth=0;
```

```
lnk=link[node][depth];
follow=(linklst[lnk][0]==node);
if(follow) nextnode=linklst[lnk][1]; else nextnode=linklst[lnk][0];

if(pd!=pdstart) {
lastnode=(pd-1)->node;
if(nextnode==lastnode) return 1;
if(nextnode==startnode) { pdend=pd; return 0; }
if(bag[nextnode]) { return 1; }
}

pd++;
pd->node=nextnode;
bag[node]=1;
return pd->rest();
};

int Intcheck::loop() {
int l,i;
if(init==2) {
init=1;
startnode=3;
}
for(;;) {
if(init==1) {
init=0;
pd=pdstart;
pd->node=startnode;
for(i=0;i<Mnodes;i++) bag[i]=0;
l=pd->rest();
} else l=pd->update();
if(l==0) return 1;
if(l==2) { startnode++; init=1; if(startnode>=Mnodes) return 0; }
}
};


int Intcheck::main(Theta& t) {
int a1,a2,a3,amin,amax,bmin,bmax; int lp;
Intcheck *lpd;
int tok=1,ok=1;

while(lp=loop() && ok) {
amin=Angle::TwoPi; amax=-Angle::TwoPi;
ok=0;

for(lpd=pdstart;lpd<=pdend && !ok;lpd++) {
a1=t.list[lpd->lnk];
if(!lpd->follow) a1=(a1+Angle::Pi) % Angle::TwoPi;
a2=(a1+(3*Angle::Pi/2)) % Angle::TwoPi;
a3=(a1+(Angle::Pi/2)) % Angle::TwoPi;
if(lpd==pdstart) {amin=a2; amax=a3; }
bmin=amin; bmax=amax;
if(a3>a2) {  // included range a2....a3 (normal)
```

```
if(amin>a2 && amin<a3 && amax>a2 && amax<a3) { ok=1; break; }
if(amin>a2 && amin<a3) bmin=a2;
if(amax>a2 && amax<a3) bmax=a3;
} else { // included range a2..0,0..a3 (goes through zero)
if((amin>a2 || amin<a3) && (amax>a2 || amax<a3)) { ok=1; break; }
if(amin>a2 || amin<a3) bmin=a2;
if(amax>a2 || amax<a3) bmax=a3;
}
amax=bmax; amin=bmin;
};

tok=tok && ok;
};
return ok;
};


// ************************************************** end of intcheck functions
// ****************************************Global definitions for main and listout

Dio dio[Mlinks+4];
Mass mass[Mass::Mpd];
Theta theta[Mass::Mpd];
Intcheck ic[Mass::Mpd];
 Intcheck i;

int dino0=1;
int dino1=1;
//int dino2=1;
int dino3=1;
int dino11=1;

void listout() {
int c1,c2,c3;
pass=1;
d.prep(dio);
d.used[0]=d.used[1]=d.used[2]=d.used[3]=2; // first pass
d.bd=0; // before dio pointer to.

cout<<"%ma="<<em1<<"\n"<<flush;
cout<<"%ta="<<ea1<<"\n"<<flush;
cout<<"%mb="<<em2<<"\n"<<flush;
cout<<"%tb="<<ea2<<"\n"<<flush;
p("%start");

while(d.make()) {
 m.prep(dio[0],mass);
 c3=0;
 while(m.make()) {
  t.prep(dio[0],mass,theta);
   while(t.make()) {
//p(dino1); p(dino2);  cout<<d; cout<<m; cout<<t;
//p(t.make());
// cout<<d; cout<<m; cout<<t;
   i.prep(dio[0],mass,ic);
```

```
   c1=i.main(t);
   if(c1) {
   if(d.bd==0) c2=1; else c2=d.check();
    if(c2) {
     c3=1;
 //    //p(dino);
   p(nodestr);
  cout<<d;
  cout<<m;
  cout<<t;
dino0++;
     }
dino11++;
    }
   dino1++;
   } // end of while(t.make)
dino2++;
  } // end of while(m.make)
if(c3) d.gen_permed_dios();
dino3++;
 } // end if while(d.make)
};

main()
{

d.gen_perms();
// 1. ordinary diograms:
pass=1;
d.prep(dio);
d.used[0]=d.used[1]=d.used[2]=d.used[3]=2; // first pass
d.bd=0; // before dio pointer to.

#ifdef GROUPD6
p("%group=d6");
#endif
#ifdef GROUPE6
p("%group=e6");
#endif
#ifdef GROUPE7
p("%group=e7");
#endif
#ifdef GROUPE8
p("%group=e8");
#endif

#if ORDER==2
p("%order=2");
#include "02POLES"
#elif ORDER==3
p("%order=3");
#include "03POLES"
#elif ORDER==4
p("%order=4");
#include "04POLES"
```

```
#elif ORDER==5
p("%order=5");
#include "05POLES"
#endif

p("%finished");

/* debug information: printout number of different types of diagrams

cout<<"%ma="<<em1<<"\n"<<flush;
cout<<"%ta="<<ea1<<"\n"<<flush;
cout<<"%mb="<<em2<<"\n"<<flush;
cout<<"%tb="<<ea2<<"\n"<<flush;
listout();

p(dino0);
p(dino1);
p(dino2);
p(dino3);
p(dino11);
p(d.bd);
*/

// test ************************************************************
// ************************************************end of program
/* Angle a1,a2,a3,t1,t2,t3,Pi;
Pi=Angle::Pi;
t1=42;
t2=12;
t3=6;
a1=72;
a2=90;
a3=18;
p(a2==a1+t1+Pi);
p(a3==a1-t3+Pi);
p(a2==a1-t1+Pi);
p(a3==a1+t3+Pi);
p(!((a2==a1+t1+Pi && a3==a1-t3+Pi) ||
(a2==a1-t1+Pi && a3==a1+t3+Pi) ) );

for(;;) { }
*/
/*
// 2a. second pass:
pass=2;
d.used[0]=1; d.used[1]=d.used[2]=2;
d.bd=0;
listout();

cout<<d.bd<<"\n"<<flush;
*/


};
```

```
// ********** DEFINE OBJECTS THAT ARE ONLY DECLARED ***************

int Dio::used[Mlinks];
Dio* Dio::pd;
Dio* Dio::pdstart;
Dio* Dio::pdend;
int Dio::bd;
int Dio::before[5000][Mlinks+1][2];
int Dio::permlst[25][Mnodes+1];
int Dio::init;
int Dio::list[Mlinks+1][2]; // main linklist

int Mass::c[Mmasses+1][Mmasses+1][Mmasses+1];
struct Mass::S Mass::p1[Mmasses+1][64];
int Mass::p2[Mmasses+1][Mmasses+1][Mp2d];
int Mass::p1max[Mmasses+1],Mass::p2max[Mmasses+1][Mmasses+1];
Mass* Mass::pdstart,*Mass::pdend,*Mass::pd;
int Mass::init;
int Mass::list[Mlinks+4];

//class Theta:public Massdat
Theta* Theta::pd,*Theta::pdstart,*Theta::pdend;
int Theta::store[500][4],Theta::sd; //
int Theta::th[Mmasses][Mmasses][Mmasses];
int Theta::init;
int Theta::list[Mlinks+4];

//class Intcheck {
int Intcheck::init;
int Intcheck::bag[Mnodes];
int Intcheck::startnode;
Intcheck *Intcheck::pd,*Intcheck::pdstart,*Intcheck::pdend;
int Intcheck::link[Mnodes][3];
int Intcheck::nlinks[Mnodes];
int Intcheck::linklst[Mlinks+1][2];
```

Appendix C: Program mj1 (Mathematica)

```
(*  Copyright (C) Paul Bayton 1994
    mj1 - A program to calculate Feynman diagrams
     This program is free software; you can redistribute it and/or
     modify it under the terms of the GNU General Public License
     as published by the Free Software Foundation; either version 2
     of the License, or (at your option) any later version.

     This program is distributed in the hope that it will be useful,
     but WITHOUT ANY WARRANTY; without even the implied warranty of
     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
     GNU General Public License for more details.

     You should have received a copy of the GNU General Public License
     along with this program; if not, write to the Free Software
     Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA. *)
setprop:=Block[{},
(* set defvar,defval: fixed part for int *)
(* set intvar: vars to integrate over *)
ivlst={}; ivtf={{False,False},{False,False},{False,False},{False,False}};
den={};
ma=m[masslst[[1]]]; mb=m[masslst[[2]]];
thetaa=2 Pi thetalst[[1]]/tpi; thetab=2 Pi thetalst[[2]]/tpi;
 (* stdsub={l[1,1]->ma Cos[thetaa], l[1,2]->ma Sin[thetaa],
 l[2,1]->mb Cos[thetab], l[2,2]->mb Sin[thetab],
 l[3,1]->ma Cos[thetaa], l[3,2]->ma Sin[thetaa],
 l[4,1]->mb Cos[thetab], l[4,2]->mb Sin[thetab]}; *)
stdsub={l[1,1]->0, l[1,2]->0,
l[2,1]->0, l[2,2]->0,
l[3,1]->0, l[3,2]->0,
l[4,1]->0, l[4,2]->0};
For[i=5,i<=Length[linklst],i++,
theta=2 Pi thetalst[[i]]/tpi;
mass=m[masslst[[i]]]; angle=2 Pi thetalst[[i]]/tpi;
tmat={{ma Cos[thetaa],mb Cos[thetab]},{ma Sin[thetaa],mb Sin[thetab]}};
abvec=Inverse[tmat] . {mass Cos[theta],mass Sin[theta]};
pden=2 (l[i,1] mass Cos[angle] + l[i,2] mass Sin[angle]) +
abvec[[1]] abvec[[2]] + (ie (1)) ;
AppendTo[ivtf,{True,True}];
(*  mass^2 ma mb (Cos[thetaa + thetab] - Cos[thetaa+thetab+theta])/2  *)
AppendTo[den,pden];
]; (* den is proplist uncompressed *)
den=den /. stdsub;
(* Print["fin part 1"]; *)
eqlst={};
For[i=1,i<=Length[nodelst],i++,
tup={}; nd=nodelst[[i]];
For[j=1,j<=Length[linklst],j++,
If[TrueQ[linklst[[j,1]]==nd],AppendTo[tup,{j,1}],
If[TrueQ[linklst[[j,2]]==nd],AppendTo[tup,{j,2}]]];
];

Print["links for node ",i," found"];
```

```
Print[tup];
mm1=masslst[[tup[[1,1]]]]; mm2=masslst[[tup[[2,1]]]]; mm3=masslst[[tup[[3,1]]]];
tt1=thetalst[[tup[[1,1]]]]; tt2=thetalst[[tup[[2,1]]]]; tt3=thetalst[[tup[[3,1]]]];
tt1=tt1*Pi/12; tt2=tt2*Pi/12; tt3=tt3*Pi/12;
Print["mass:",masslst[[tup[[1,1]]]],masslst[[tup[[2,1]]]],masslst[[tup[[3,1]]]]];
mm1=m[mm1]; mm2=m[mm2]; mm3=m[mm3];
Print["angles:",thetalst[[tup[[1,1]]]],":",thetalst[[tup[[2,1]]]],":",thetalst[[tup[[3,1]]
]]];
Print["coses:",N[mm1*Cos[tt1]],":",N[mm2*Cos[tt2]],":",N[mm3*Cos[tt3]]];
Print["sins :",N[mm1*Sin[tt1]],":",N[mm2*Sin[tt2]],":",N[mm3*Sin[tt3]]];

eq1=eq2=0;
For[j=1,j<=3,j++,
pad1=l[tup[[j,1]],1];
If[tup[[j,2]]==1,eq1=eq1-pad1,eq1=eq1+pad1];
pad2=l[tup[[j,1]],2];
If[tup[[j,2]]==1,eq2=eq2-pad2,eq2=eq2+pad2];
];
AppendTo[eqlst,eq1]; AppendTo[eqlst,eq2];
]; (* for i<=nodelst... eqlst is now set *)
oldeqlst=eqlst;
eqlst=eqlst /. stdsub;
(* Print["end of eqlist"]; *) (* Abort[]; *)
done=True;
While[done,
subn=0; subm=Infinity;
For[i=1,i<=Length[eqlst],i++,
lce=LeafCount[eqlst[[i]]];
If[lce<subm && !TrueQ[eqlst[[i]]==0],subm=lce; subn=i]];
(* Print["eqlst is now ",eqlst]; Print["smallest eqe is ",subn]; *)
nm=True;
For[q=1,q<=2 && nm,q++,
For[p=1,p<=Length[eqlst] && nm,p++,
If[MemberQ[{eqlst[[subn]]},l[p,q],Infinity],nm=False; p0=p; q0=q] ];
]; (* Print["attack var",p0,q0,nm]; *)
ivtf[[p0,q0]]=False;
(* substitution of l[i,j] *)
subs=Solve[eqlst[[subn]]==0,l[p0,q0]][[1,1]];
   (* If[subs=={{}}[[1,1]],Abort[]]; *)
(* Print["solution to ",eqlst[[subn]]," is"]; Print[subs]; Print[" "]; *)
eqlst=eqlst /. subs;
den=den /. subs;
done=False;
For[k=1,k<=Length[eqlst],k++,done=done || !TrueQ[eqlst[[k]]==0]];
];
(* produce intvarlist *)
For[k=1,k<=Length[ivtf],k++,
 For[kk=1,kk<=2,kk++,
 If[ivtf[[k,kk]],AppendTo[ivlst,l[k,kk]]] ]];
defvar=Append[ivlst,e];
defval=Append[Table[i,{i,1,Length[ivlst]}],0.1];
den2=Table[{den[[k]],1},{k,1,Length[den]}];
den={Join[{{1,1}},den2]};
(* den=necsimp2[den,ivlst[[1]]]; *)
(* Print[den]; *)
```

```
];


l[i_,j_]:=ToExpression[StringJoin["l",ToString[i],"x",ToString[j]]];

find3tuple[nd_]:=Block[{i},
res={};
For[i=1,i<=Length[linklst],i++,
If[TrueQ[linklst[[i,1]]==nd || linklst[[i,2]]==nd],AppendTo[res,i]];
];
Return[res];
];

(*

ivlst={};
(* compress into normal form *)
den2=Table[{den[[k]],1},{k,1,Length[den]}];
den3=N[den2];
den4={};
For[k=1,k<=Length[den3],k++,
If[Count[den

oldden=den; den={{1,1}};
 For[k=1,k<=Length[oldden],k++,
 nin=True;
 For[kk=1,kk<=Length[den],kk++,
 Print[kk,TrueQ[N[den[[kk,1]]]==N[oldden[[k]]]]];
 If[N[den[[kk,1]]]==N[oldden[[k]]],nin=False] ];
Print[k,nin];
If[nin,AppendTo[den,{oldden[[k]],noinlist[oldden,oldden[[k]]]}] ];
 ];  *)

noinlist[list_,e_]:=Block[{i,ct},
(* number of elements in list (list,element):number *)
ct=0;
i=1;
While[i<=Length[l],
If[N[list[[i]]]==N[e],ct++];
i++;
];
Return[ct];
];


trm[list_]:=Block[{k,p},
p=1/(Product[list[[k,1]]^list[[k,2]],{k,2,Length[list]}]);
Return[p];
];


cvar[l_]:=Block[{},
lr=l; ltlst[0]={};
oldivlst=ivlst; ivlst={};
For[k=1,k<=Length[oldivlst],k=k+2,
```

```
 v1=oldivlst[[k]]; v2=oldivlst[[k+1]];
 bv1=True; i=0; j=1; ml=Length[l[[1]]];
 While[bv1,
  If[i<ml,i++,i=1; j++; ];
  If[i==ml && j==ml,Print["error"]; Abort[]; ];
  expra=l[[1,i,1]];
  exprb=l[[1,j,1]];
  mat={{Coefficient[expra,v1,1],Coefficient[expra,v2,1]},{Coefficient[exprb,v1,1],Coeffici
ent[exprb,v2,1]}};
  bv1=SameQ[Det[mat],0];
  ];
 va=(mat[[1,1]] v1 + mat[[1,2]] v2);
 vb=(mat[[2,1]] v1 + mat[[2,2]] v2);
 {{v1e},{v2e}}=Simplify[Inverse[mat] . {{cf[0,k]},{cf[0,k+1]}}];
 AppendTo[ltlst[0],cf[0,k]]; AppendTo[ltlst[0],cf[0,k+1]];
 ivlst=Join[ivlst,{va,vb}];
 For[p=1,p<=Length[l[[1]]],p++,
  expr=l[[1,p,1]]; expr0=Coefficient[Coefficient[expr,v1,0],v2,0];
  expr1=Coefficient[expr,v1,1]; expr2=Coefficient[expr,v2,1];
  If[Not[expr1 === 0 && expr2 === 0],
   ksub=Solve[{ka mat[[1,1]] + kb mat[[2,1]],ka mat[[1,2]] + kb mat[[2,2]]} == {expr1,expr
2},{ka,kb} ][[1]];
Print[ksub]; Pause[1];
   lr[[1,p,1]]=expr0+ka cf[0,k] + kb cf[0,k+1] /. ksub;
   ];
  ];
 ];
Return[lr];
];



diff2[t_,v_,n_]:=Block[{sret,i,j,expr,power,pret},
sret={};
For[i=1,i<=Length[t],i++,
 For[j=2,j<=Length[t[[i]]],j++,
  power=t[[i,j,2]];
  expr=t[[i,j,1]];
  If[!TrueQ[Coefficient[expr,v,1] == 0],
   pret=t[[i]];
   pret[[1,1]]*=-power * Coefficient[expr,v,1];
   pret[[j,2]]++;
   AppendTo[sret,pret];
  ]; (*if*)
]; ];
If[n==1,Return[sret],Return[diff2[sret,v,n-1]]];
]

(* third version - simple replacement *)
substit[p_]:=Block[{i,q},
q=p;
For[i=1,i<=Length[defvar],i++,q=q /. defvar[[i]] -> defval[[i]] ];
Return[q];
]
```

```
encode[list_]:=Block[{i,j,k},   (* set coefficients of list *)
m=1; stage++; lr={}; ltlst[stage]={};
For[i=1,i<=Length[list],i++,
 pl=list[[i]]; plr={};
  For[j=1,j<=Length[pl],j++,
   expr=pl[[j,1]]; power=pl[[j,2]];
   exprr=0;
   For[k=1,k<=Length[ivlst],k++,
   iv=ivlst[[k]];
   c1=Coefficient[expr,iv,1];
   lts=ltlst[stage]; bl1=(lts != {}); lls=Length[lts]; kk=0; bl3=bl4=False; bl1=False;
   While[bl1,
    kk++;
    bl2=kk<lls;
    lsk=ltlst[stage][[kk]];
    bl3=TrueQ[c1==lsk];
    bl4=TrueQ[c1==-lsk];
    bl1=bl2 && Not[bl3] && Not[bl4];
    ];
   If[bl3,exprr=exprr + iv cf[stage,kk]; ];
   If[bl4,exprr=exprr - iv cf[stage,kk]; ];
   If[Not[bl3 || bl4],
    exprr=exprr + iv cf[stage,m]; m++;
    AppendTo[ltlst[stage],c1];
    expr=expr - c1 iv;
    ];
   ];
  If[Not[TrueQ[expr == 0]],
   exprr=exprr + cf[stage,m]; m++;
   AppendTo[ltlst[stage],expr];
   ];
  AppendTo[plr,{exprr,power}];
  ];
 AppendTo[lr,plr];
 ];
Return[lr];
];


decode[list_]:=Block[{i,j},
lr=list;
For[i=stage,i>=1,i--,
 For[j=1,j<=Length[ltlst[i]],j++,
  lr=lr /. cf[i,j] -> ltlst[i][[j]]
  ];
 ];
Return[lr];
];

decoden[list_,i_]:=Block[{j},
lr=list;
 For[j=1,j<=Length[ltlst[i]],j++,
  lr=lr /. cf[i,j] -> ltlst[i][[j]]
  ];
Return[lr];
```

```
];


(* form of term for integration is
{ {{i1,1} ,{A,n}, {B,n}},{{i2,1},{C,n},{D,n}.... } representing
1/(i1 A^n B^n) + 1/(i2 C^n D^n ... ------------------------------- *)
int3[tt_,v_]:=Block[{}, (* Print["next"]; *)
t=tt; (* t=encode[tt]; *) Print["nnext"];
ipm=pm1[Coefficient[t,v,1],Coefficient[t,ie,1]];
(* Print[ipm]; *)
sr={}; (* sum-term return *)
lt=Length[t];
For[ii=1,ii<=lt,ii++,
 pt=t[[ii]]; (* prod term  Print["4444"]; *)
 For[ji=2,ji<=Length[pt],ji++,
(*  Print["ji is -----=====-=-=-=-== ",ji];  *)
  fexpr=pt[[ji]]; expr=fexpr[[1]];  power=fexpr[[2]];
  cr0=Coefficient[expr,v,0];  cr1=Coefficient[expr,v,1];
  c1=Not[TrueQ[cr1 == 0]];
  cre=Coefficient[expr,ie,1];
  If[Not[PloynomialQ[expr,v]],Print["error- not poly"]; Abort[] ];
(*  Print["c1 is -------------- ",c1]; *)
  If[c1,
   polesub=v -> -cr0/cr1;
   intq=N[substit[cr1/cre]]; (* Print[intq]; *)
(* if intq>0, x cr1 + I e=0, x = -Ie/cr1 <0 below line *)
   If[intq>0,
    ptr=ucomp[pt,ji]; (* list excluding working term *)
    ptr[[1,1]]=ptr[[1,1]]/(cr1^power);
    If[power==1,gz={ptr},gz=diff2[ptr,v,power-1];
                      For[k=1,k<=length[gz],k++,
                      gz[[k,1,1]]=gz[[k,1,1]]/((power-1)!)] ];
    gz=gz /. polesub;
    sr=Join[sr,gz]; abort[];
    ]
   ] (* if c1 *)
  ]
 ]; (* for i *)
Return[sr];
]; (* block *)


int4[tt_,v_]:=Block[{}, t=tt; sr={};
criel=Coefficient[t,ie,1];
cr1l=Coefficient[t,v,1];
cr0l=Coefficient[t,v,0];
pm=pmf[substit[cr1l],substit[criel]];
(* Print[pm];
Print["criel is ",criel]; *)
For[ii=1,ii<=Length[criel],ii++,For[ji=1,ji<=Length[criel[[ii]]],ji++,
If[(criel[[ii,ji,1]]^2 <0.0000001) && (criel[[ii,ji,1]] !=0),Print["bomb999"]]]];
lt=Length[t];
For[ii=1,ii<=lt,ii++,
(* Print["**********************************doing sum list: ",ii]; *)
 t1=t[[ii]]; pm1=pm[[ii]]; lpt=Length[t1];
```

```
pmpc=0; pmmc=0; pmec=0;
For[ji=2,ji<=lpt,ji++,
pm2=pm1[[ji,1]];
Switch[pm2,
1,pmpc++,
-1,pmmc++,
9,pmec++]; ];
(* Print[" hisogram(-1,1,err): ",pmmc," ",pmpc," ",pmec]; *)
If[pmec>0,Print["no ie"]];
If[pmpc pmmc>0 || pmec>0,
(* Print["plus,minus,error"]; Print[pmpc]; Print[pmmc]; Print[pmec]; *)
If[pmpc<pmmc,
(* ======================================== integrate top contour *)
(*  Print["top"]; *)
 For[ji=2,ji<=Length[t1],ji++,
 t2=t1[[ji]]; expr=t2[[1]];  power=t2[[2]];
  If[Not[PloynomialQ[expr,v]],Print["error- not poly"]; Abort[] ];
  pm2=pm1[[ji,1]]; cr0=cr01[[ii,ji,1]]; cr1=cr11[[ii,ji,1]];
  If[pm2==1 || pm2==9, (* Print["ji=",ji]; *)
   polesub=v -> -cr0/cr1;
   ptr=ucomp[t1,ji]; (* list excluding working term *)
   ptr[[1,1]]=ptr[[1,1]]/(cr1^power);
If[pm2==9,ptr[[1,1]]=ptr[[1,1]]/2];
   If[power==1,gz={ptr},gz=diff2[{ptr},v,power-1];
                    For[k=1,k<=Length[gz],k++,
                    gz[[k,1,1]]=gz[[k,1,1]]/((power-1)!)] ];
   gz=gz /. polesub;
   sr=Join[sr,gz];
  ];
 ],
(* ======================================== integrate bottom contour *)
(*  Print["bottom"]; *)
 For[ji=2,ji<=Length[t1],ji++,
 t2=t1[[ji]]; expr=t2[[1]];  power=t2[[2]];
  If[Not[PloynomialQ[expr,v]],Print["error- not poly"]; Abort[] ];
  pm2=pm1[[ji,1]]; cr0=cr01[[ii,ji,1]]; cr1=cr11[[ii,ji,1]];
  If[pm2==-1 || pm2==9, (* Print["ji==",ji]; *)
   polesub=v -> -cr0/cr1;
(* Print["cr0,cr1 is ",cr0,"   ",cr1]; *)
   ptr=ucomp[t1,ji]; (* list excluding working term *)
   ptr[[1,1]]=-ptr[[1,1]]/(cr1^power);
If[pm2==9,ptr[[1,1]]=ptr[[1,1]]/2];
   If[power==1,gz={ptr},gz=diff2[{ptr},v,power-1];
                    For[k=1,k<=Length[gz],k++,
                    gz[[k,1,1]]=gz[[k,1,1]]/((power-1)!)] ];
   gz=gz /. polesub;
   sr=Join[sr,gz];
   ];
  ];
 ];
(* ======================================== all zero- ignoor *)
];
]; (* for i *)
Return[sr];
]; (* block *)
```

```
pmf[x_,y_]:=If[x==0,Return[0],If[y==0,Return[9],If[x/y>0,Return[1],If[x/y<0,Return[-1],Ret
urn[0]]]]];
SetAttributes[pmf,Listable];

ucomp[lst_,pos_]:=Join[Table[lst[[uci]],{uci,1,pos-1}],Table[lst[[uci]],{uci,pos+1,Length[
lst]}]];
(* user compliment *)

(*
trm[list_]:=Block[{k,p},
p=1/(Product[list[[k]],{k,1,Length[list]}]);
Return[p];
] *)

(* ldec: list-decode. Decode from internal format to mathematica expression *)
ldec[list_]:=Sum[list[[ldi,1,1]]/Product[list[[ldi,ldj,1]]^list[[ldi,ldj,2]],{ldj,2,Length
[list[[ldi]]]}],{ldi,1,Length[list]}]

ldec2[list_]:=Sum[
ldb=True;
ldt1=Product[
lsl=list[[ldi,ldj,1]]^Round[list[[ldi,ldj,2]]];
ldc=Coefficient[lsl,ie,0];
If[(Abs[ldc]<10^-7),ldb=False];
lsl,
{ldj,2,Length[list[[ldi]]]}];
If[ldb,list[[ldi,1,1]]/ldt1,0],
{ldi,1,Length[list]}]

ldec3[list_]:=Expand[(Sum[
(list[[ldi,1,1]]*
Product[
Product[list[[ldk,ldj,1]]^Round[list[[ldk,ldj,2]]],{ldj,2,Length[list[[ldk]]]}],
{ldk,1,ldi-1}]*
Product[
Product[list[[ldk,ldj,1]]^Round[list[[ldk,ldj,2]]],{ldj,2,Length[list[[ldk]]]}],
{ldk,ldi+1,Length[list]}]),
{ldi,1,Length[list]}])]/Expand[
(Product[Product[list[[ldi,ldj,1]]^Round[list[[ldi,ldj,2]]],{ldj,2,Length[list[[ldi]]]}],{
ldi,1,Length[list]}])];

(* get rid of small numbers *)
(* grsn[lst_,lvl]:=If[TrueQ[lst<=10^-15],Return[0],Return[lst]]
SetAttributes[grsn,Listable] *)
grsn[lst_,lvl_]:=Block[{i,l,sub,li},l=lst;
If[AtomQ[l],
 (*  Print[l]; *)
   If[TrueQ[Abs[l]<10^-15],Print["zero"]; Return[0],Return[l]],
 Print["length lst is ",Length[l]];
 For[i=1,i<=Length[l],i++,
  li=l[[i]];
  sub=grsn[li,lvl+1];
```

```
If[lvl==1,Print["i is ",i," li is ",FullForm[l[[i]]]," sub is ",FullForm[sub]]];
 l=ReplacePart[l,sub,i];
  ];
Return[l]]];

grsn2[lst_]:=Block[{i,l,sub,li},l=lst;
If[AtomQ[l],
 (*  Print[l]; *)
   If[TrueQ[Abs[l]<10^-15],Print["**********small number (zero?) eliminated"]; Return[0],Re
turn[l]],
 For[i=1,i<=Length[l],i++,
  li=l[[i]];
  sub=grsn2[li];
 l=ReplacePart[l,sub,i];
  ];
Return[l]]];

grsn3[lst_]:=FixedPoint[grsn2,lst];


necsimp[list_]:=Block[{i,i0,j},
outlst={};
For[i=1,i<=Length[list],i++,
 nm=True;
 For[j=1,j<=Length[outlst] && nm,j++,
  If[N[list[[i,1]],10]==N[outlst[[j,1]],10],nm=False; j0=j ];
  ];
Print[i,nm];
 If[nm,AppendTo[outlst,list[[i]]],outlst[[j0,2]]++];
 ];
Return[outlst];
];




necsimp2[lst_,v_]:=Block[{(* i,j,s,p,tmp1,tmp2,tmp3,lst5*)},
(*  lst5-prod-result lst9-sum-result *)
lst9={};
For[i=1,i<=Length[lst],i++,
 plst=lst[[i]]; tmp1={}; tmp2={}; tmp3={};
 For[j=2,j<=Length[plst],j++,
  trm=plst[[j,1]];
  cr0=Coefficient[trm,v,0];
  cr1=Coefficient[trm,v,1];
  scr0=N[substit[cr0],20];
  scr1=N[substit[cr1],20];
If[scr1 != 0,ppole=N[scr0/scr1,10],ppole=N[scr0,10]];
Print[scr1,"=====",scr0];
  AppendTo[tmp1,ppole];
  AppendTo[tmp2,cr1];
  ];
 lst5={First[plst]};
 plst=Rest[plst];
```

```
 For[j=1,j<=Length[tmp1],j++,
  elm=tmp1[[j]];
  If[Count[tmp3,elm]==0,
   p=Position[tmp1,elm]; Print[p];
   power=Sum[plst[[p[[k,1]],2]],{k,1,Length[p]}]; Print["power is ",power];
   cr0=tmp2[[j]];
   scr0=N[substit[cr0],20];
Print[scr0];
  If[scr0 !=0, pnumerator=Product[(tmp2[[p[[k,1]]]]/cr0)^plst[[p[[k,1]],2]],
{k,1,Length[p]}],pnumerator=1];
   AppendTo[tmp3,elm];
   AppendTo[lst5,{plst[[j,1]],power}];
   lst5[[1,1]]=lst5[[1,1]]*pnumerator;
   ];
  ];
 AppendTo[lst9,lst5];
 ];
Return[lst9];
];


necsimp3[lst_,v_]:=Block[{(* i,j,s,p,tmp1,tmp2,tmp3,lst5 *)},
(*  lst5-prod-result lst9-sum-result *)
lst9={};
For[i=1,i<=Length[lst],i++,
 plst=lst[[i]]; tmp1={}; tmp2={}; tmp3={};
 rlst=Rest[plst];  lst5={First[plst]};
 For[j=1,j<=Length[rlst],j++,
  trm=rlst[[j,1]]; lpow=rlst[[j,2]];
  cr0=Coefficient[trm,v,0];
  cr1=Coefficient[trm,v,1];
  scr0=N[substit[cr0],20];
  scr1=N[substit[cr1],20];
  If[scr1 != 0,
   ppole=N[scr0/scr1,10];
   If[Count[tmp1,ppole]==0,
    AppendTo[lst5,rlst[[j]]];
    AppendTo[tmp1,ppole]; AppendTo[tmp2,cr1];
     AppendTo[tmp3,Length[lst5]];,
    p=Position[tmp1,ppole][[1,1]];
    cr1r=tmp2[[p]];
    q=tmp3[[p]];
    pnumerator=(cr1/cr1r)^lpow;
    lst5[[1,1]] *= pnumerator;
    lst5[[q,2]] += lpow;
    ];,
   AppendTo[lst5,rlst[[j]]];
   ];
  ];
 AppendTo[lst9,lst5];
 ];
Return[lst9];
];
```

```
necsimp4[lst_,v_]:=Block[{},(* do necessary simplification *)
newlst={};
For[i=1,i<=Length[lst],i++,
newprodlst={lst[[i,1]]}; cmplst={};
For[j=2,j<=Length[lst[[i]]],j++,
cr0=Simplify[Coefficient[lst[[i,j,1]],v,0]];
cr1=Simplify[Coefficient[lst[[i,j,1]],v,1]];
power=Round[lst[[i,j,2]]];
If[cr1!=0,
newprodlst[[1,1]]*=((1/cr1)^power);
normterm=v+Simplify[cr0/cr1],
normterm=cr0];
cmp=N[normterm,30];
(* Print[Count[cmplst,cmp]]; *)
If[Count[cmplst,cmp]!=0,
pos=Position[cmplst,cmp][[1,1]]+1;
newprodlst[[pos,2]]+=power,
AppendTo[newprodlst,{normterm,power}];
AppendTo[cmplst,cmp];
];
];
AppendTo[newlst,newprodlst];
];
Return[newlst];
];


simplst[lst_]:=Table[necsimp2[lst[[sli]]],{sli,1,Length[lst]}];


(* ========================== postcript in mathematica ==================== *)
(* global data:
   1. linklst,masslst,thetalst : basic lists from prolog
   2. tpi: two Pi
   3. pointlst: generated point list consisting of
     { ... { {x,y},{link},{link}.. } .... }
   4. nodelst: list of nodes, first four are external
*)
(* wanps write annother diogram to postscript *)
wanps:=Block[{},
wanind1++;
wanind2=Mod[wanind1,3];
wanind3=Mod[Quotient[wanind1,3],5];
wanind4=Mod[wanind1,15];
Write[file,OutputForm["matrix defaultmatrix setmatrix "],wanind2*180+90,OutputForm[" "],wa
nind3*160+80,OutputForm[" translate 113 113 scale"]];
drawdual;
Write[file,OutputForm["0 4.5 translate"]];
drawnorm;
Write[file,OutputForm["0 0 moveto (pic:"],wanind1,OutputForm[") show"]];
If[wanind4==15,Write[file,OutputForm["showpage"]]];
]

(* crecpoints - create points for diagram *)
crecpoints:=Block[{},
```

```
externbox;
mar={};
For[cpi=1,cpi<=Length[nodelst],cpi++,
AppendTo[mar,internbox[cpi]]];
While[mab=True;
For[mai=1,mai<=Length[mar],mai++,mab=mab && mar[[mai]]];
Not[mab],
Print["failed ********************"];
mar={};
For[cpi=1,cpi<=Length[nodelst],cpi++,
AppendTo[mar,internbox[cpi]] ];
]

sv2ep[sp]=ep
sv2ep[ep]=sp

(* drawnorm: draw a normal (non dual) diagram in postscript *)
olddrawnorm:=Block[{n},
n=4;
intern={};
extern={1,2,3,4};
ne=Length[extern];
tlist={{1,2},{2,3},{3,4},{4,1}};
rad=1;
dtheta=2 Pi/n;
stheta=Pi - dtheta/2;
For[i=0,i<n,i++,atheta[i]=stheta+ i dtheta;
x1=N[rad Cos[atheta[i]]];
y1=N[rad Sin[atheta[i]]];
spc=OutputForm[" "];
Write["test.ps",x1,spc,y1,spc,OutputForm["/v"],i,OutputForm[" sto2 blob"]];
markused[i]=False; ]; (* for *)
For[i=0,i<ne,i++,lab[extern[[i+1]]]=Round[i ne/n]; markused[Round[i ne/n]]=True];
For[i=0,i<n,i++,If[!markused[i],lab[intern[[i+1]]]=i]];
For[i=1,i<=Length[tlist],i++,
pt1=lab[tlist[[i,1]]]; pt2=lab[tlist[[i,2]]];
Write["test.ps",OutputForm[" /v"],pt1,
OutputForm[" rcl /v"],pt2,OutputForm[" rcl arrowft"]];
]; (* for i *)
Write["test.ps",OutputForm["0.02 setlinewidth \n"]];
pt1=lab[extern[[1]]];
Write["test.ps",OutputForm[" /v"],pt1,OutputForm[" rcl \n
/tmp sto2 0.3 add exch -1 add exch /tmp rcl arrowft \n"]];
pt1=lab[extern[[2]]];
Write["test.ps",OutputForm[" /v"],pt1,OutputForm[" rcl \n
/tmp sto2 -0.3 add exch -1 add exch /tmp rcl arrowft \n"]];
pt1=lab[extern[[3]]];
Write["test.ps",OutputForm[" /v"],pt1,OutputForm[" rcl \n
2 copy -0.3 add exch 1 add exch arrowft \n"]];
pt1=lab[extern[[4]]];
Write["test.ps",OutputForm[" /v"],pt1,OutputForm[" rcl \n
2 copy 0.3 add exch 1 add exch arrowft \n"]];
Write[OutputForm["showpage"]];
Close["test.ps"];
];
```

```
(* opens file "file" and writes postscript utils to it *)
drawdefaults:=Block[{},
OpenWrite[file];
(* x y blob - draw blob.
x1 y1 x2 y2 arrowft - arrow from too.
x1 y1 r theta arrowsa - arrow start angle.
*)
Write[file,OutputForm["%!PostScript \n
matrix defaultmatrix setmatrix \n
/Times_Roman findfont 12 113 div scalefont setfont \n
1 setlinecap \n
1 setlinejoin \n
3 setmiterlimit \n
1 113 div setlinewidth \n
300 500 translate 113 113 scale \n
/blob { newpath 0.02 0 360 arc fill } def \n
/arrowft { newpath 4 copy moveto lineto stroke newpath \n
2 1 roll 3 1 roll 4 copy \n
exch sub /dy exch def exch sub /dx exch def \n
add 2 div /cntry exch def \n
add 2 div /cntrx exch def \n
dx 0.2 mul cntrx add dy 0.2 mul cntry add moveto \n
dy -0.1 mul cntrx add dx 0.1 mul cntry add lineto \n
dy 0.1 mul cntrx add dx -0.1 mul cntry add lineto \n
closepath fill } def \n
/sto2 { [ 4 copy pop /tmps exch def ] tmps exch def pop } def \n
/sto4 { [ 6 copy pop /tmps exch def ] tmps exch def pop } def \n
/pop4 { pop pop pop pop } def \n
/rcl { load aload pop} def \n
/arrowsa { 2 copy cos mul /tmp exch def \n
sin mul tmp arrowft } def \n
"]];
]



(* externbox:produces initial pointlst due to external mopmenta
assumes first four links are external *)
externbox:=Block[{},
t1={0,0};
t2=t1+finddxdy[1,sp];
t3=t2+finddxdy[2,sp];
t4=t3+finddxdy[3,ep];
pointlst={{t1,{1,sp},{4,sp}},{t2,{2,sp},{1,ep}},{t3,{3,ep},{2,ep}},{t4,{4,ep},{3,sp}}};
searchpath={{1,2},{1,3},{2,2},{2,3},{3,2},{3,3},{4,2},{4,3}};
ndpointlst={};
]

(* internbox: produces rest of pointlst due to internal momenta *)
internbox[n_]:=Block[{},
(* match nodes - order weighted *)
nd=nodelst[[n]];
p0={pa,pb,pc}=form3pts[nd];
```

```
{match,i,j,k,l}=locate[p0];
If[match,
p1=p0[[k,l]];
If[l==1,p2=p0[[k,2]],p2=p0[[k,1]]];
If[TrueQ[p1[[2]]==ep],pa1={p1[[1]],sp},pa1={p1[[1]],ep}];
If[TrueQ[p2[[2]]==ep],pa2={p2[[1]],sp},pa2={p2[[1]],ep}];
{matcha,ia,ja,ka,la}=locate[{{pa1}}];
{match2,i2,j2,k2,l2}=locate[{{p2}}];
{matchb,ib,jb,kb,lb}=locate[{{pa2}}];
{k1,l1}=locate2[p1];
{k2,l2}=locate2[p2];
{ka,la}=locate2[pa1];
{kb,lb}=locate2[pa2];
(* only one point in pointlist, kb is new node*)
newpoint={pointlst[[i,1]] + finddxdy[p2[[1]],p2[[2]]] };
Print["xxxx1"];
AppendTo[pointlst,newpoint]; Print["xxxxa"];
addto[Length[pointlst],{p0[[kb,1]],p0[[kb,2]]}]; Print["xxxx2"];
addto[i,p0[[k]]]; Print["xxxx3"];
addto[ia,p0[[ka]]]; Print["xxxx1"];
pav=(pointlst[[i,1]] + newpoint[[1]] + pointlst[[ia,1]])/3; Print["xxxx4"];
AppendTo[ndpointlst,{nd,pav}]; Print["xxxx5"];
];
Return[match];
]


allsimp[lst_]:=Block[{},
olst=lst;
For[asi=1,asi<=Length[ivlst],asi++,
nlst=necsimp3[olst,ivlst[[asi]]]; olst=nlst; ];
Return[nlst];
];

checksetprop:=Block[{},
OpenRead[filein];
For[mi=1,mi<=7,mi++,
Read[filein]; Read[filein]; Read[filein]; Read[filein];
setprop;
Print[""]; Print[""];
Print["------------------------------------------------------------",mi];
Print[""]; Print[""];
(* Print[Simplify[N[allsimp[den]]]];  *)
];
];


addto[i_,pti_]:=Block[{k,l,nm},
(*  i- index for pointlst  pti- insert *)
For[k=1,k<=Length[pti],k++,
nm=True;
For[l=2,l<=Length[pointlst[[i]]],l++,
If[TrueQ[pti[[k]]==pointlst[[i,l]] ],nm=False];
];
If[nm, AppendTo[pointlst[[i]], pti[[k]]];
```

```
AppendTo[searchpath,{i,Length[pointlst[[i]]]}]; ];
];
]


locate[pl_]:=Block[{n,nd,i,j,k,l,m,nm,i0,j0,k0,l0},
(* match nodes - order weighted -
returns nodefound: pointlst[[i,j]] and pl[[k,l]] *)
nm=True; ml=0;
For[m=1,m<=Length[searchpath] && nm,m++,
i=searchpath[[m,1]]; j=searchpath[[m,2]];
For[k=1,k<=Length[pl] && nm,k++,
For[l=1,l<=Length[pl[[k]]] && nm,l++,
Print[i,j,k,l];
 If[nm && TrueQ[pointlst[[i,j]]==pl[[k,l]] ],nm=False; i0=i; j0=j; k0=k; l0=l;] (* must us
e lazy evaluation *)
]]];
Return[{!nm,i0,j0,k0,l0}];
]

yesprint[x_]:=Block[{},Print[x]; Return[True]]

locate2[p_]:=Block[{i,j,i0,j0},
(* match nodes - order weighted - as locate
but returns p[[i,j]] *)
nm=True;
For[i=1,i<=Length[p0] && nm,i++,
For[j=1,j<=Length[p0[[i]]] && nm,j++,
 If[nm && TrueQ[p0[[i,j]]==p], nm=False; i0=i; j0=j; ];
]];
Return[{i0,j0}];
]


(* gettri: produces list {l1,l2,l3} ln:index of links
with node in common from "linklst" *)
gettri[n_]:=Block[{ll,i},
ll={};
For[i=1,i<=Length[linklst],i++,
If[TrueQ[linklst[[i,1]]==n || linklst[[i,2]]==n],AppendTo[ll,i]; ];
];
Return[ll];
]

(* form3pts: locate 3 arcs with node n'literal in common
out of "linklst" *)
form3pts[n_]:=Block[{},
ll=gettri[n];
p1={}; p2={}; p3={};
l1=ll[[1]]; l2=ll[[2]]; l3=ll[[3]];
theta1=thetalst[[l1]]; theta2=thetalst[[l2]];
p1={{l1,sp}}; p2={{l1,ep}};
pi=tpi/2;
If[TrueQ[linklst[[l1,1]]==n],ac1=1,ac1=-1]; (* out->anticlockwise ac=1 *)
If[TrueQ[linklst[[l2,1]]==n],ac2=1,ac2=-1];
```

```
If[TrueQ[linklst[[l3,1]]==n],ac3=1,ac3=-1];
If[ac1 ac2 == -1,theta2=theta2+pi];
dtheta=Mod[theta2-theta1,tpi];
If[dtheta>pi,acc1=1,acc1=-1];
 If[acc1==ac1,(* ------------------------------ trangle is right way round *)
 If[ac1 ac2 == 1,AppendTo[p2,{l2,sp}]; p3={{l2,ep}},
 AppendTo[p2,{l2,ep}]; p3={{l2,sp}}];
 If[ac1 ac3 == 1,AppendTo[p1,{l3,ep}]; AppendTo[p3,{l3,sp}],
 AppendTo[p1,{l3,sp}]; AppendTo[p3,{l3,ep}] ];
 ,(* ---------------------------------------- triangle is wrong way round *)
 If[ac1 ac2 == 1,AppendTo[p1,{l2,ep}]; p3={{l2,sp}},
 AppendTo[p1,{l2,sp}]; p3={{l2,ep}}];
 If[ac1 ac3 == 1,AppendTo[p2,{l3,sp}]; AppendTo[p3,{l3,ep}],
 AppendTo[p2,{l3,ep}]; AppendTo[p3,{l3,sp}] ];
 ];
Return[{p1,p2,p3}];
]


(* calculate dx,dy of a link from startingpoint or endingpoint *)
finddxdy[lnk_,spep_]:=Block[{mass,angle,r,theta,dx,dy},
  mass=m[masslst[[lnk]]];
  angle=thetalst[[lnk]];
  r=mass;
  theta=angle 2 Pi/tpi;
  dx=N[r Cos[theta]]; dy=N[r Sin[theta]];
If[TrueQ[spep==ep],dx=-dx; dy=-dy];
Return[{dx,dy}];
]


(* final ps output routine to "file" *)
drawdual:=Block[{},
pl=pointlst;
For[i=1,i<=Length[pl],i++,
 pp=pl[[i]];
 px=pp[[1,1]]; py=pp[[1,2]];
 For[j=2,j<=Length[pp],j++,
 {dx,dy}=finddxdy[pp[[j,1]],pp[[j,2]]];
  If[TrueQ[pp[[j,2]] == sp],x1=px; y1=py; x2=px+dx; y2=py+dy,
       x2=px; y2=py; x1=px+dx; y1=py+dy; ];
  Write[file,CForm[x1],OutputForm[" "],CForm[y1],
    OutputForm[" "],CForm[x2],OutputForm[" "],CForm[y2],
    OutputForm[" arrowft \n"]];
  ]; (* for j *)
 ]; (* for i *)
]; (* block *)

drawnorm:=Block[{},
totpav={0,0};
For[i=1,i<=Length[ndpointlst],i++,
 {x1,y1}=ndpointlst[[i,2]];
 totpav=totpav+{x1,y1};
 Write[file,CForm[x1],OutputForm[" "],CForm[y1],OutputForm[" "],OutputForm["/k"],i,OutputF
orm[" sto2 blob \n"]];
```

```
 ];
totpav=totpav/Length[ndpointlst];
For[i=5,i<=Length[linklst],i++,
 nd1=linklst[[i,1]]; nd2=linklst[[i,2]];
{mat1,j1}=locatenpt[nd1]; {mat2,j2}=locatenpt[nd2];
Print[nd1]; Print[nd2]; Print[j1]; Print[j2];
 Write[file,OutputForm["/k"],j1,OutputForm[" rcl /k"],j2,OutputForm[" rcl arrowft\n"]]
 ];
For[i=1,i<=4,i++, (* do first four external lines *)
 lk=linklst[[i]];
 {mat1,j1}=locatenpt[lk[[1]]];
 {mat2,j2}=locatenpt[lk[[2]]];
 If[mat2,
 k2=j2;
 xy2=ndpointlst[[j2,2]];
 dxy=(xy2-totpav);
 {x1,y1}=xy2+dxy;
 Write[file,CForm[x1],OutputForm[" "],CForm[y1],OutputForm[" /k"],k2,OutputForm[" rcl arro
wft % xxx \n"] ];
  ];
 If[mat1,
  k1=j1;
  xy1=ndpointlst[[j1,2]];
  dxy=(xy1-totpav);
  {x2,y2}=xy1+dxy;
  Write[file,OutputForm[" /k"],k1,OutputForm[" rcl "],CForm[x2],OutputForm[" "],CForm[y2],
OutputForm[" arrowft % yyy \n"]];
  ]; (* if *)
 ]; (* for *)
]; (* block *)

locatenpt[pt_]:=Block[{j,j0,nm},
nm=True;
For[j=1,j<=Length[ndpointlst],j++,
If[TrueQ[ndpointlst[[j,1]]==pt],nm=False; j0=j];
 ];
Return[{!nm,j0}];
];

die[n_]:=Simplify[D[ie*rc[[n]],ie]];
crb:=Block[{},
nif=0;
comin={};
rc=Table[rb[[i]],{i,1,Length[rb]}];
rd=rc /. ie->0;
For[i=1,i<=Length[rb],i++,
If[rd[[i]]!=ComplexInfinity,nif+=rd[[i]],AppendTo[comin,i] ];
];
cid=Table[die[comin[[i]]],{i,1,Length[comin]}];
cie=cid /. ie->0;
cif=Sum[cie[[i]],{i,1,Length[comin]}];
Return[cif+nif];
];

finite[t_]:=If[TrueQ[t==Indeterminate || t==ComplexInfinity],
```

```
False,True];

die2:=Block[{ i,ra,term,aterm,endres0,endres1,endres0i},
(* takes global:"result" in normal form and calculates the limit ie->0 *)
endres0=0; endres1=0; endres0i=0;
ra=Table[ldec[{result[[i]]}],{i,1,Length[result]}];
For[i=1,i<=Length[ra],i++,
term=ra[[i]];
If[finite[term /. ie->0],
endres0 += (term /. ie->0),
aterm=Simplify[ie*term];
endres1 += D[aterm,ie] /.ie->0;
endres0i += aterm /. ie->0;
];
];

 Print["ordinary term: ",endres0];
Print["prepole A1: ",endres1];
Print["prepole A0: ",endres0i];
If[!TrueQ[endres0i == 0],Print["*************possible infinite expression in die2"]];
Return[endres0+endres1];
];

die3:=Block[{},
(* old version *)
resultb=ldec[result];
resultc2=resultb /. ie->0;
rh=resultb;
rb=Table[ldec[{result[[i]]}],{i,1,Length[result]}];
rcrb=crb;
resultc3={rh /. ie->-0.0001,rh /. ie->-0.00005, rh /. ie->-0.00001, rh /. ie->0,rh /. ie->
0.00001,rh /. ie->0.00005, rh /. ie->0.0001,rcrb};
If[resultc3[[4]]==Indeterminate,
resultc3[[4]]=(resultc3[[3]]+resultc3[[5]])/2];
Return[resultc3];
];

readblock:=Block[{},
(* reads in next block from file "filein" ignooring comments *)

rba=True;
While[rba,
(* Print["entering loop"]; *)
rbstr=Read[filein,String];
If[rbstr==EndOfFile,finish=True; rba=False ];
If[StringTake[rbstr,1]=="%", (* contains comment/start *)
 Print["Comment:",rbstr];
  If[rbstr=="%start",
  Print["****************start of S matrix calculation*****************"];
  finres=0; ressof=0;
  calcnumber=calcnumber+1;
  intnumber=0; ];
 rba=True;
 ,
 ToExpression[rbstr];
```

```
 ToExpression[Read[filein,String]];
 ToExpression[Read[filein,String]];
 ToExpression[Read[filein,String]];
 intnumber=intnumber+1;
 rba=False;
 ]];
];


main1:=Block[{},
mi=0; finish=False;
ressof=0;
reschain={};
setgroupe6;
OpenRead[filein];
(* OpenWrite["intlst4"]; *)
(* wanind1=-1; drawdefaults; *)
(* finres={0,0,0,0,0,0,0,0}; *)
(********************************** determin offset *)
(* ioffset=0; coffset=0; *)
calcnumber=0; intnumber=0;
readblock;
While[(ioffset>intnumber && coffset == calcnumber) ||
coffset>calcnumber,readblock;];

(* ******************************** begin main program *)
While[! finish,
(* postscript:: externbox; crecpoints; wanps; *)
l=.; setprop;
result=Simplify[N[den,40]];

(* do sucessive integrals *)
For[km=1,km<=Length[ivlst],km++,tres=result; (* Print["next..."]; *)
result=necsimp4[tres,ivlst[[km]]];
result=grsn3[result];
result=int4[result,ivlst[[km]]];
];

(* integral done, now calculate I as ie->0 *)
uint=die2;
of=N[overallfactor];
ovres=of*uint;
ressof += ovres;

(* print out answer *)
Print["A.calculation is ",calcnumber," B.integral no is ",intnumber];
Print["C.integral is ",N[uint]," D.overall factor is ",of];
Print["E.overall result is ",ovres];
Print["F.result so far is ",N[ressof,20]];
Print["------------------------------------------------------------"];
readblock;
];

];
```

```
overallfactor:=Block[{l,p},
delta:=m[masslst[[i]]] m[masslst[[2]]] Sin[2 Pi (thetalst[[2]]-thetalst[[1]])/tpi]/2;
If[N[delta]<0,delta=-delta];
p=Length[linklst]-4;
l=Length[ivlst]/2;
o=p- 2 l;
(* note 1/(s-iso)^order beta/sqrt(2 h) not included *)
of= ((2 Pi)^(4 o - 2 p - 2 + 2 l)) (- 1/(8 delta)) ((-1)^l) ((1/(4 I delta))^o);
of=of*Product[mt=masslst[[find3tuple[nodelst[[i]]]]];
c[mt],{i,1,Length[nodelst]}];
Return[of];
];


setgroupd6:=Block[{},
(* data for group=d_n masses and coupling constants *)
SetAttributes[s1,Protected];
SetAttributes[s2,Protected];
n=6;
h=2*(n-1);
tpi=12 h;

For[i=1,i<=4,i++,
m[i]=Sin[i Pi/h]; ];
m[5]=m[6]=m[s1]=m[s2]=1/2;

c[mt_]:=c[mt]=c2[Sort[mt]];

c2[{i_,j_,k_}]:=If[IntegerQ[i] && IntegerQ[j] && IntegerQ[k],
 If[i+j+k==h,
c2[{i,j,k}]= - Sqrt[8] m[i] m[j] Sin[Pi (i+j)/h],
 If[i+j-k==0,
c2[{i,j,k}]= Sqrt[8] m[i] m[j] Sin[Pi k/h]
]]];
c2[{a_,s1,s1}]:=If[EvenQ[a],Sqrt[8] m[s1] m[s1] Sin[(1-2a/h) Pi]];
c2[{a_,s2,s2}]:=c2[{a,s1,s1}];
c2[{a_,s1,s2}]:=If[OddQ[a],Sqrt[8] m[s1] m[s1] Sin[(1-2a/h) Pi]];
];

zcp[i_,j_,k_,a_,t_]:=Block[{},
c[{i,j,k}]=Sqrt[8]*a*m[i]*m[j]*Sin[t*Pi/h];
c[{j,i,k}]=c[{j,k,i}]=c[{k,i,j}]=c[{k,j,i}]=c[{i,k,j}]=c[{i,j,k}];
];

setgroupe6:=Block[{},
(* data for group=e6 *)
n=6;
h=12;
tpi=24;

m[1] = Sqrt[3 - Sqrt[3]];
m[2] = m[1];
m[3] = Sqrt[2] * m[1];
m[4] = Sqrt[3 + Sqrt[3]];
m[5] = m[4];
```

```
m[6] = Sqrt[2] * m[4];

zcp[1,1,2,-I,8];
zcp[1,1,4,-I,2];
zcp[1,2,3,1,6];
zcp[1,3,1,1,9];
zcp[1,3,4,1,5];
zcp[1,4,2,-I,11];
zcp[1,4,5,-I,7];
zcp[1,5,3,1,9];
zcp[1,5,6,1,3];
zcp[1,6,4,1,10];
zcp[2,2,1,I,8];
zcp[2,2,4,I,2];
zcp[2,3,2,1,9];
zcp[2,3,5,1,5];
zcp[2,4,3,1,9];
zcp[2,4,6,1,3];
zcp[2,5,1,I,11];
zcp[2,5,4,I,7];
zcp[2,6,5,1,10];
zcp[3,3,3,1,8];
zcp[3,3,6,1,2];
zcp[3,4,1,1,10];
zcp[3,5,2,1,10];
zcp[3,6,3,1,11];
zcp[3,6,6,-1,7];
zcp[4,4,2,-I,10];
zcp[4,4,5,I,8];
zcp[4,5,6,-1,6];
zcp[4,6,1,1,11];
zcp[4,6,4,-1,9];
zcp[5,5,1,-I,10];
zcp[5,5,4,I,8];
zcp[5,6,2,1,11];
zcp[5,6,5,-1,9];
zcp[6,6,3,-1,10];
zcp[6,6,6,1,8];
];

setgroupe7:=Block[{},
(* data for group=e7 *)
n=7;
h = 18;
tpi=18;

m[1] = Sqrt[8] * Sin[Pi / 9];
m[2] = Sqrt[8 * Sqrt[3] * Sin[Pi / 18] * Sin[2 * Pi / 9]];
m[3] = Sqrt[8] * Sin[2 * Pi / 9];
m[4] = Sqrt[8 * Sqrt[3] * Sin[5 * Pi / 18] * Sin[Pi / 9]];
m[5] = Sqrt[8] * Sin[Pi / 3];
m[6] = Sqrt[8] * Sin[4 * Pi / 9];
m[7] = Sqrt[8 * Sqrt[3] * Sin[7 * Pi / 18] * Sin[4 * Pi / 9]];

zcp[1,1,2,-1,10];
```

```
zcp[1,1,4,-1,2];
zcp[1,2,1,-1,13];
zcp[1,2,3,1,7];
zcp[1,3,2,1,14];
zcp[1,3,4,1,10];
zcp[1,3,5,1,6];
zcp[1,4,1,-1,17];
zcp[1,4,3,1,11];
zcp[1,4,6,1,3];
zcp[1,5,3,1,14];
zcp[1,5,6,1,8];
zcp[1,6,4,1,16];
zcp[1,6,5,1,12];
zcp[1,6,7,1,4];
zcp[1,7,6,1,15];
zcp[2,2,2,-1,12];
zcp[2,2,4,-1,8];
zcp[2,2,5,-1,2];
zcp[2,3,1,1,15];
zcp[2,3,3,-1,11];
zcp[2,3,6,-1,5];
zcp[2,4,2,-1,14];
zcp[2,4,5,-1,8];
zcp[2,5,2,-1,17];
zcp[2,5,4,-1,13];
zcp[2,5,7,-1,3];
zcp[2,6,3,-1,15];
zcp[2,7,5,-1,16];
zcp[2,7,7,1,10];
zcp[3,3,2,-1,14];
zcp[3,3,7,-1,2];
zcp[3,4,1,1,15];
zcp[3,5,1,1,16];
zcp[3,5,6,1,10];
zcp[3,6,2,-1,16];
zcp[3,6,5,1,12];
zcp[3,6,7,1,8];
zcp[3,7,3,-1,17];
zcp[3,7,6,1,13];
zcp[4,4,4,-1,12];
zcp[4,4,5,1,10];
zcp[4,4,7,-1,4];
zcp[4,5,2,-1,15];
zcp[4,5,4,1,13];
zcp[4,5,7,1,7];
zcp[4,6,1,1,17];
zcp[4,6,6,1,11];
zcp[4,7,4,-1,16];
zcp[4,7,5,1,14];
zcp[5,5,5,1,12];
zcp[5,6,1,1,16];
zcp[5,6,3,1,14];
zcp[5,7,2,-1,17];
zcp[5,7,4,1,15];
zcp[5,7,7,-1,11];
```

```
zcp[6,6,4,1,14];
zcp[6,6,7,-1,10];
zcp[6,7,1,1,17];
zcp[6,7,3,1,15];
zcp[6,7,6,-1,13];
zcp[7,7,2,1,16];
zcp[7,7,5,-1,14];
zcp[7,7,7,1,12];
];

setgroupe8:=Block[{},
(* data for group=e8 *)
n=8;
h=30;
tpi=30;

m[1] = Sqrt[4 * Sqrt[3] * Sin[Pi / 30] * Sin[Pi / 5]];
m[2] = Sqrt[16 * Sqrt[3] * Sin[Pi / 30] * Sin[Pi / 5]] * Cos[Pi / 5];
m[3] = Sqrt[16 * Sqrt[3] * Sin[Pi / 30] * Sin[Pi / 5]] * Cos[Pi / 30];
m[4] = Sqrt[64 * Sqrt[3] * Sin[Pi / 30] * Sin[Pi / 5]] * Cos[Pi / 5] * Cos[7 * Pi / 30];
m[5] = Sqrt[4 * Sqrt[3] * Sin[11 * Pi / 30] * Sin[Pi / 5]];
m[6] = Sqrt[4 * Sqrt[3] * Sin[7 * Pi / 30] * Sin[2 * Pi / 5]];
m[7] = Sqrt[4 * Sqrt[3] * Sin[13 * Pi / 30] * Sin[2 * Pi / 5]];
m[8] = Sqrt[256 * Sqrt[3] * Sin[Pi / 30] * Sin[Pi / 5]] * Cos[2 * Pi / 15] * Cos[Pi / 5] *
 Cos[Pi / 5];

zcp[1,1,1,-1,20];
zcp[1,1,2,1,12];
zcp[1,1,3,-1,2];
zcp[1,2,1,1,24];
zcp[1,2,2,-1,18];
zcp[1,2,3,1,14];
zcp[1,2,4,-1,8];
zcp[1,3,1,-1,29];
zcp[1,3,2,1,21];
zcp[1,3,4,1,13];
zcp[1,3,5,1,3];
zcp[1,4,2,-1,25];
zcp[1,4,3,1,21];
zcp[1,4,4,-1,17];
zcp[1,4,5,-1,11];
zcp[1,4,6,1,7];
zcp[1,5,3,1,28];
zcp[1,5,4,-1,22];
zcp[1,5,6,1,14];
zcp[1,5,7,-1,4];
zcp[1,6,4,1,25];
zcp[1,6,5,1,19];
zcp[1,6,7,1,9];
zcp[1,7,5,-1,27];
zcp[1,7,6,1,23];
zcp[1,7,8,1,5];
zcp[1,8,7,1,26];
zcp[1,8,8,1,16];
zcp[2,2,1,-1,24];
```

```
zcp[2,2,2,1,20];
zcp[2,2,4,1,14];
zcp[2,2,5,1,8];
zcp[2,2,6,-1,2];
zcp[2,3,1,1,25];
zcp[2,3,3,1,19];
zcp[2,3,6,1,9];
zcp[2,4,1,-1,27];
zcp[2,4,2,1,23];
zcp[2,4,7,1,5];
zcp[2,5,2,1,26];
zcp[2,5,6,1,16];
zcp[2,6,2,-1,29];
zcp[2,6,3,1,25];
zcp[2,6,5,1,29];
zcp[2,6,7,1,13];
zcp[2,6,8,1,3];
zcp[2,7,4,1,27];
zcp[2,7,6,1,21];
zcp[2,7,7,-1,17];
zcp[2,7,8,1,11];
zcp[2,8,6,1,28];
zcp[2,8,7,1,22];
zcp[3,3,2,1,22];
zcp[3,3,3,1,20];
zcp[3,3,5,1,14];
zcp[3,3,6,1,12];
zcp[3,3,7,1,4];
zcp[3,4,1,1,26];
zcp[3,4,5,1,16];
zcp[3,5,1,1,29];
zcp[3,5,3,1,23];
zcp[3,5,4,1,21];
zcp[3,5,7,1,13];
zcp[3,5,8,1,5];
zcp[3,6,2,1,26];
zcp[3,6,3,1,24];
zcp[3,6,6,1,18];
zcp[3,6,8,1,8];
zcp[3,7,3,1,28];
zcp[3,7,5,1,22];
zcp[3,8,5,1,27];
zcp[3,8,6,1,25];
zcp[3,8,8,-1,17];
zcp[4,4,1,-1,26];
zcp[4,4,4,-1,20];
zcp[4,4,6,1,16];
zcp[4,4,7,-1,12];
zcp[4,4,8,-1,2];
zcp[4,5,1,-1,27];
zcp[4,5,3,1,23];
zcp[4,5,5,-1,19];
zcp[4,5,8,1,9];
zcp[4,6,1,1,28];
zcp[4,6,4,1,22];
```

```
zcp[4,7,2,1,28];
zcp[4,7,4,-1,24];
zcp[4,7,7,1,18];
zcp[4,7,8,-1,14];
zcp[4,8,4,-1,29];
zcp[4,8,5,1,25];
zcp[4,8,7,-1,21];
zcp[5,5,4,-1,22];
zcp[5,5,5,1,20];
zcp[5,5,8,-1,12];
zcp[5,6,1,1,27];
zcp[5,6,2,1,25];
zcp[5,6,7,-1,17];
zcp[5,7,1,-1,29];
zcp[5,7,3,1,25];
zcp[5,7,6,-1,21];
zcp[5,8,3,1,28];
zcp[5,8,4,1,26];
zcp[5,8,5,-1,24];
zcp[5,8,8,-1,18];
zcp[6,6,3,1,24];
zcp[6,6,6,-1,20];
zcp[6,6,8,-1,14];
zcp[6,7,1,1,28];
zcp[6,7,2,1,26];
zcp[6,7,5,-1,22];
zcp[6,7,8,-1,16];
zcp[6,8,2,1,29];
zcp[6,8,3,1,27];
zcp[6,8,6,-1,23];
zcp[6,8,7,-1,21];
zcp[7,7,2,-1,26];
zcp[7,7,4,1,24];
zcp[7,7,7,-1,20];
zcp[7,8,1,1,29];
zcp[7,8,2,1,27];
zcp[7,8,4,-1,25];
zcp[7,8,6,-1,23];
zcp[7,8,8,1,19];
zcp[8,8,1,1,28];
zcp[8,8,3,-1,26];
zcp[8,8,5,-1,24];
zcp[8,8,7,1,22];
zcp[8,8,8,1,20];
];


(*********************************************************initialise *)
intern={};
extern={1,2,3,4};
nodelst={e1,e2,e3,e4};
linklst={{ef1,e1},{ef2,e2},{e3,ef3},{e4,ef4},{e1,e2},{e2,e3},{e3,e4},{e4,e1}};
masslst={2,2,2,2,3,1,1,1};
thetalst={72,0,72,0,66,78,6,54};
```

```
ndpointlst={};
file="test.ps"
filein="o2e6"
reschain={};
fileinl=214;
fileins=1;
ioffset=0; coffset=0;


(* temp ------------------------
zz= (-41356.80831212664*(2.646977960169688*10^-22*ie^5 -
       7.895778862130056*10^-8*ie^6 + 4.750150216720283*10^-6*ie^7 -
       0.0001071566843994093*ie^8 + 0.001074278674101529*ie^9 -
       0.0040384445235575644*ie^10))/
   (0.0001730702717123939348*ie^6 - 0.015618024574336l1*ie^7 +
     0.5872181356762538*ie^8 - 11.77477228058547*ie^9 +
     132.8033622983493*ie^10 - 798.8133663906054*ie^11 +
     2001.941154491274*ie^12)

zz= 2.646977960169688*10^-22*ie^5 - 7.895778862130056*10^-8*ie^6 +
   4.750150216720284*10^-6*ie^7 - 0.0001071566843994093*ie^8 +
   0.001074278674101529*ie^9 - 0.0040384445235575644*ie^10

nodelst={e1,e2,e3,e4,1,2}
linklst={{ef1,e1},{ef2,e2},{e3,ef3},{e4,ef4},{e1,1},{e1,2},{e2,1},{e2,2},{e3,e4},{e3,1},{e
4,2}}
masslst={3,3,3,3,1,2,s1,s2,s1,s2,s2}
thetalst={84,0,84,0,72,90,108,12,12,36,48}
res=int3[int3[int3[den,18x1],18x2],111x1];
Simplify[N[res]]


*)
```