Maxima by Example:
Ch. 12, Dirac Algebra and Quantum Electrodynamics [*]

Edwin L. Woollett

June 8, 2019

Contents

[*]We use **Maxima 5.36.1** in these notes and **wxMaxima 17.10.0** in the worksheets. Check `http://www.csulb.edu/~woollett/` for the latest version of these notes. Send comments and suggestions to `woollett@charter.net`

Preface

Feedback from readers is the best way for this series of notes to become more helpful to new users of Maxima. *All* comments and suggestions for improvements will be appreciated and carefully considered.

```
LOADING FILES
The defaults allow you to use the brief version load (dirac3) to load in the
Maxima file dirac3.mac, provided you have the file dirac3.mac in your work
folder, say c:\work5, and you have a line like

file_search_maxima : append(["c:/work5/###.{mac,mc}"],file_search_maxima )$

in your startup file maxima-init.mac. (See Maxima by Example, Ch. 1, for more
details about this startup file.)

Otherwise you need to provide a complete path in double quotes,
as in load("c:/work5/dirac3.mac"),

We always use the brief load version in our examples, which are generated
using the XMaxima graphics interface on a Windows XP computer, and copied
into a fancy verbatim environment in a latex file which uses the fancyvrb
and color packages.
```

```
Maxima.sourceforge.net. Maxima, a Computer Algebra System. Version 5.36.1
     http://maxima.sourceforge.net/
```

The author would like to thank Maxima developer Barton Willis for his initial suggestion to use the Maxima code file **`simplifying.lisp`** and his patient help in understanding how to make use of that code. The new code has also been simplified (as compared with the rather long-winded code in version 1) by using a set of functions shared by Barton Willis which are used in making decision branches. These functions are

```
  mplusp, mtimesp, mnctimesp, mexptp, mncexptp.
```

(The "nc" additions refer to "non-commutative"). General expansions of multiple term arguments have been simplified by using Barton Willis' methods which combine **`lambda`** and **`xreduce`**.

1 Introduction

Both symbolic and explicit matrix Dirac algebra methods are available in this Dirac package. One can gain confidence in an unexpected result by doing the same calculation in multiple ways.

The **explicit** (matrix) Dirac spinor methods, which use an explicit representation of the gamma matrices, are bug free, fast, and the route to polarized amplitudes (rather that the square of polarized amplitudes). Moreover, summing the absolute square of the polarized amplitudes over all helicity values leads quickly and reliably to the unpolarized cross section in terms of the frame dependent kinematic variables such as scattering angle.

However the version available with this package is not a covariant method and is always frame dependent. The **explicit matrix** methods using the Maxima function **`mat_trace`** (or the equivalent **`m_tr`** function) are inherently less bug prone and also are faster in execution than the purely symbolic methods.

However, the **purely symbolic** contraction and trace methods available in this package are capable of introducing the kinematic invariants s, t, and u into the calculation in a simple way, or to reducing the types of four vector dot products to two, leading to useful covariant results.

In the following examples, we use a variety of methods to arrive at the same answer. We generally start with a purely symbolic method which allows the production of a frame independent result in terms of Mandelstam variables. This is usually the most complicated method with more steps to the final result.

Faster methods to the final answer appear later, and the fastest methods to an unpolarized differential scattering cross section are either the **`nc_tr`** plus **`mcon(expr, indices)`** method or else the explicit matrix method **`m_tr`** plus **`mcon`** method. So if you are looking for the fastest method to a frame dependent solution, you should bear this in mind.

2 Kinematics and Mandelstam Variables

We use the same conventions as Peskin & Schroeder's **An Introduction to Quantum Field Theory** (see the References section at the end). The diagonal elements of the metric tensor are $(1, -1, -1, -1)$,

$$g_{\mu\nu} = g^{\mu\nu} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.1}$$

with Greek indices running over $0, 1, 2, 3$.

We are using natural units in which the momentum vector $\vec{p}$, energy E and mass M all have the same units, and the velocity vector $\vec{v}$ is dimensionless. Thus for a given particle with energy E, momentum vector $\vec{p}$, mass m and velocity vector $\vec{v}$

$$E = m\,\gamma, \quad \vec{p} = m\,\gamma\,\vec{v}, \quad E^2 = m^2 + |\vec{p}|^2, \quad \gamma(v) = \frac{1}{\sqrt{1-v^2}} \tag{2.2}$$

If p_a and p_b are the four-momenta of two particles a and b, then the quantity

$$M_{ab}^2 = (p_a + p_b)^2 = (E_a + E_b)^2 - (\vec{p}_a + \vec{p}_b)^2\,, \tag{2.3}$$

is called the effective mass squared of a and b. If a particle **R** decays to a and b, then M_{ab} is the mass of the parent particle **R**.

2.1 $1 + 2 \to 3 + 4$ Scattering for Non-Equal Masses

We can use Maxima and the author's **qdraw** program (see Ch. 13) to make a diagram of $2 \to 2$ scattering with our notation conventions.

```
(%i19) qdraw(xr(-5,7),yr(-5,5), line(-4,-4,-2,0,lw(4)),line(-2,0,-4,4,lw(4)),
              line(4,-4,2,0,lw(4)),line(2,0,4,4,lw(4)),
               arrowhead(-3,-2,63.43,0.4),arrowhead(-3,2,116.57,0.4),
               arrowhead(3,-2,116.57,0.4),arrowhead(3,2,63.43,0.4),
               line(5.4,-2,5.4,2,lw(2),lc(blue)),arrowhead(5.4,2,90,0.4,lc(blue)),
                label(["{/=18 P_{1}}",-4,-2],["{/=18 P_{3}}",-3.8,1.65],
                       ["{/=18 P_{4}}",3.3,1.65],["{/=18 P_{2}}",3.5,-2],
                        ["{/=12 t}",5.3,2.3]),circle(0,0,2.8,lc(black),fill(gray)), cut(all) )$
```

which produces the plot

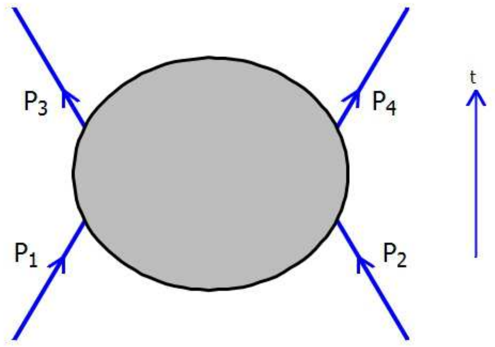

Figure 1: $2 \to 2$ Scattering

For the scattering event $1 + 2 \to 3 + 4$, let the corresponding particles have masses M_1, M_2, M_3, and M_4 and four-momenta $p_1 = (E_1; \vec{p}_1)$, $p_2 = (E_2; \vec{p}_2)$, etc. Each four-momentum satisfies $p_a^2 = p_a \cdot p_a = M_a^2$. Conservation of total four-momentum $p_1 + p_2 = p_3 + p_4$ gives four component equations:

$$p_1^\mu + p_2^\mu = p_3^\mu + p_4^\mu, \quad \text{for} \quad \mu = 0, 1, 2, 3. \tag{2.4}$$

Recall that the four-vector scalar product is defined as (using the summation convention for repeated Lorentz indices)

$$a \cdot b = a^\mu \, b_\mu = a^0 \, b^0 - \vec{a} \cdot \vec{b}. \tag{2.5}$$

We define the three Mandelstam variables s, t, and u (also making use of 4-momentum conservation)

$$s = (p_1 + p_2)^2 = (p_3 + p_4)^2, \tag{2.6}$$

$$t = (p_1 - p_3)^2 = (p_2 - p_4)^2, \tag{2.7}$$

$$u = (p_1 - p_4)^2 = (p_2 - p_3)^2. \tag{2.8}$$

Only two of these three Mandelstam variables can be independent, but it is conventional to define the three variables for the sake of symmetry. We have the identity.

$$s + t + u = M_1^2 + M_2^2 + M_3^2 + M_4^2, \tag{2.9}$$

since

$$\begin{aligned} s + t + u &= (p_1 + p_2)^2 + (p_1 - p_3)^2 + (p_1 - p_4)^2 \\ &= 3p_1^2 + p_2^2 + p_3^2 + p_4^2 + 2p_1 \cdot p_2 - 2p_1 \cdot p_3 - 2p_1 \cdot p_4 \\ &= p_1^2 + p_2^2 + p_3^2 + p_4^2 + 2p_1 \cdot (p_1 + p_2 - p_3 - p_4 = 0) \\ &= p_1^2 + p_2^2 + p_3^2 + p_4^2 \\ &= M_1^2 + M_2^2 + M_3^2 + M_4^2. \end{aligned}$$

Since s, t, and u are Lorentz scalars, they do not change from frame to frame. They can be evaluated in any frame that is convenient. The quantities s, t, and u are called Mandelstam variables after Stanley Mandelstam who introduced them in 1958.

In the high energy limit, in which we can neglect all particle masses compared to the particle energies,

$$\begin{aligned}
&s \approx 2\,p_1 \cdot p_2 \approx 2\,p_3 \cdot p_4 \\
&t \approx -2\,p_1 \cdot p_3 \approx -2\,p_2 \cdot p_4 \\
&u \approx -2\,p_1 \cdot p_4 \approx -2\,p_2 \cdot p_3 \\
&s + t + u \approx 0.
\end{aligned}$$

All Lorentz-invariant combinations of the four momenta can be expressed in terms of the Mandelstam variables. For example, the Lorentz invariant 4-vector products of any two momenta are

$$\begin{aligned}
2\,p_1 \cdot p_2 &= (p_1 + p_2)^2 - p_1^2 - p_2^2 = s - M_1^2 - M_2^2 \\
2\,p_3 \cdot p_4 &= (p_3 + p_4)^2 - p_3^2 - p_4^2 = s - M_3^2 - M_4^2 \\
2\,p_1 \cdot p_3 &= p_1^2 + p_3^2 - (p_1 - p_3)^2 = M_1^2 + M_3^2 - t \\
2\,p_2 \cdot p_4 &= p_2^2 + p_4^2 - (p_2 - p_4)^2 = M_2^2 + M_4^2 - t \\
2\,p_1 \cdot p_4 &= p_1^2 + p_4^2 - (p_1 - p_4)^2 = M_1^2 + M_4^2 - u \\
2\,p_2 \cdot p_3 &= p_2^2 + p_3^2 - (p_2 - p_3)^2 = M_2^2 + M_3^2 - u.
\end{aligned}$$

Using these invariant products, you can show, for example, that for arbitrary masses

$$(p_1 + p_3) \cdot (p_2 + p_4) = s - u. \tag{2.10}$$

If we consider "elastic scattering" in the technical sense $M_1 = M_3 = m$ and $M_2 = M_4 = M$, then the above invariant products show that

$$\begin{aligned}
&p_1 \cdot p_2 = p_3 \cdot p_4, \\
&p_1 \cdot p_4 = p_2 \cdot p_3 \\
&p_2 \cdot p_4 = M^2 - m^2 + p_1 \cdot p_3.
\end{aligned}$$

2.2 Mandelstam Variable Tools at Work

We show an example of the use of one of our Mandelstam variable tools here. We are using the xMaxima interface here with **display2d** set equal to **false** at the end of the file **dirac3.mac**. Loading **dirac3.mac** causes five other files to also load.

```
(%i1) load(dirac3);
dirac3.mac
simplifying-new.lisp
dgtrace3.mac
dgcon3.mac
dgeval3.mac
dgmatrix3.mac
current symbol assignments

  scalarL =    [c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]
  indexL =    [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,la,mu,nu,rh,si,ta,al,be,ga,de,ep]
 massL =    [m,M]
  Nlast =    0
 reserved program capital letter name use:
 Chi, Con, D, Eps, EpsL,  G, G(1), G5, G5p
 Gam, Gm, Gtr, KD,  LI, Nlast, UI,  P,  S
 Sig, Sigb, SIG, UU, VP, VV
 I2, Z2, CZ2, I4, Z4, CZ4, RZ4, N1,N2,...
 reserved array names: gmet, eps4, eps4L
```

```
  invar_flag =    true
  stu_flag =    false
(%o1) "c:/work5/dirac3.mac"
```

The list **invarR** (R for “rules”) is initially empty.

```
(%i2) invarR;
(%o2) []
```

We will illustrate a case in which all four masses are distinct. The list **massL** contains symbols which are to be recognised as symbols for particle masses. The default are the symbols **m** and **M**. We can add additional symbols to this list using the function **Mmass**. We then interrogate the new contents of the list **massL**.

```
(%i3) Mmass (m1,m2,m3,m4)$
(%i4) massL;
(%o4) [m,M,m1,m2,m3,m4]
```

We now populate the list **invarR** with a set of identities which express 4-vector dot products of 4-momentum vectors in terms of the Mandelstam variables, using the package function **set_invarR**. This particular case can be simply pasted in from code in the text file **mandelstam.txt**. We then interrogate the contents of the list **invarR**.

```
(%i5) set_invarR (D(p1,p1) = m1^2,
         D(p1,p2) =  (s - m1^2 - m2^2)/2,
         D(p1,p3) = (m1^2 + m3^2 - t)/2,
         D(p1,p4) = (m1^2 + m4^2 - u)/2,
         D(p2,p2) = m2^2,
         D(p2,p3) = (m2^2 + m3^2 - u)/2,
         D(p2,p4) = (m2^2 + m4^2 - t)/2,
         D(p3,p3) = m3^2,
         D(p3,p4) = (s - m3^2 - m4^2)/2,
         D(p4,p4) = m4^2)$
(%i6) invarR;
(%o6) [D(p4,p4) = m4^2,D(p3,p4) = (s-m4^2-m3^2)/2,D(p3,p3) = m3^2,
       D(p2,p4) = ((-t)+m4^2+m2^2)/2,D(p2,p3) = ((-u)+m3^2+m2^2)/2,
       D(p2,p2) = m2^2,D(p1,p4) = ((-u)+m4^2+m1^2)/2,
       D(p1,p3) = ((-t)+m3^2+m1^2)/2,D(p1,p2) = (s-m2^2-m1^2)/2,
       D(p1,p1) = m1^2]
```

The list **invarR** now holds a list of “replacement rules” which allow some of our package functions to replace symbolic 4-vector dot products **D(pa,pb)** with an expression depending on the masses and Mandelstam variables. For example, our symbolic trace function **tr(expr)** makes use of the list **invarR**. Also the combined symbolic and non-covariant trace function **nc_tr(expr)** makes use of the list **invarR**.

A third function which uses this list is **ev_Ds(expr)**, which we will demonstrate here. (The **expr** can be a product and or sum of sub-expressions containing **D’s**.)

```
(%i7) ev_Ds ( D(p1+p3, p2 + p4));
(%o7) s-u
```

This has reproduced the formula we found above:

$$(p_1 + p_3) \cdot (p_2 + p_4) = s - u. \tag{2.11}$$

The package function **ev_Ds(expr)** first calls **Dexpand(expr)**, and then calls **ev(result,invarR)**, and finally calls **expand** on the last result.

```
(%i8) fundef (ev_Ds);
(%o8) ev_Ds(eeexpr):=(Dexpand(eeexpr),ev(%%,invarR),expand(%%))
```

If we instead worked this out step by step,

```
(%i9) Dexpand ( D(p1+p3, p2 + p4) );
(%o9) D(p3,p4)+D(p2,p3)+D(p1,p4)+D(p1,p2)
(%i10) ev(%,invarR);
(%o10) ((-u)+m4^2+m1^2)/2+((-u)+m3^2+m2^2)/2+(s-m4^2-m3^2)/2+(s-m2^2-m1^2)/2
(%i11) expand(%);
(%o11) s-u
```

we get the same result as the use of **ev_Ds(expr)**.

Our function **ev_invar(expr)** is equivalent to **ev(expr,invarR)**.

```
(%i12) fundef(ev_invar);
(%o12) ev_invar(_expr%):=ev(_expr%,invarR)
```

The 4-vector product of a pair of 4-vectors is equivalent to a contraction process. Once the list **invarR** has been created (by using **set_invarR**, for example), the symbolic contraction functions **Con** and **scon** make use of the list **invarR** to simplify contractions of 4-momentum vectors. As shown here, it is easier to use **ev_Ds(expr)** to get these simplified results.

```
(%i13) Con(UI(p1,mu)*UI(p1,mu));
(%o13) m1^2
(%i14) ev_Ds(D(p1,p1));
(%o14) m1^2
(%i15) Con(UI(p1,mu)*UI(p2,mu));
(%o15) s/2-m2^2/2-m1^2/2
(%i16) ev_Ds(D(p1,p2));
(%o16) s/2-m2^2/2-m1^2/2
(%i17) Con(UI(p1,mu)*UI(p3,mu));
(%o17) (-t/2)+m3^2/2+m1^2/2
(%i18) Con(UI(p1,mu)*UI(p4,mu));
(%o18) (-u/2)+m4^2/2+m1^2/2
```

The symbolic contraction functions **Con** and **scon** automatically contract on repeated Lorentz indices (depending on the contents of the list indexL to recognise such indices), but you have the option of including explicit summation indices to be summed over.

The process $1+2 \to 3+4$ is known as the *s-channel* process because *s* is the energy variable (equal to W^2, the square of the center of momentum frame total energy). In the CM frame, $\vec{p}_2 = -\vec{p}_1$, so

$$s = (p_1+p_2)^2 = (E_1+E_2)^2 - (\vec{p}_1+\vec{p}_2)^2 = (E_1+E_2)^2 = W^2. \tag{2.12}$$

We can also consider the corresponding *t-channel* process $1+\bar{3} \to \bar{2}+4$, where *t* is the energy variable, or the corresponding *u-channel* process $1+\bar{4} \to \bar{2}+3$, where *u* is the energy variable. The antiparticle of 3 is $\bar{3}$, etc. It turns out that these three different processes are intimately related.

2.3 CM Frame Scattering Description

With the convention

$$\vec{p}_i = \vec{p}_1, \quad \vec{p}_f = \vec{p}_3, \tag{2.13}$$

we set

$$p_1 = (E_1; 0, 0, p_i)\,,$$
$$p_2 = (E_2; 0, 0, -p_i)\,,$$

so the initial particles move in the z-direction. The final particles can scatter at some angle, so

$$p_3 = (E_3; \vec{p}_f)\,,$$
$$p_4 = (E_4; -\vec{p}_f)\,,$$

where $\vec{p}_f$ can be taken to be in the $x - z$ plane,

$$\vec{p}_f = (p_f \sin\theta, 0, p_f \cos\theta)\,. \tag{2.14}$$

The angle θ is the CM scattering angle, and $\cos\theta = \hat{p}_i \cdot \hat{p}_f = \hat{p}_1 \cdot \hat{p}_3$. Thus, in the CM frame

$$s = (E_1 + E_2)^2 = \left[\sqrt{M_1^2 + p_i^2} + \sqrt{M_2^2 + p_i^2}\right]^2 \tag{2.15}$$

which can be solved (see below) for p_i^2,

$$p_i^2 = \frac{\left[s - (M_1 + M_2)^2\right]\,\left[s - (M_1 - M_2)^2\right]}{4\,s}. \tag{2.16}$$

In order to use Maxima's **`solve(eqn,var)`** function, we need to square the original equation enough times to remove the square root functions, and finally arrive at a polynomial in the unknown. Thus we can, for example, proceed in steps, letting x represent p_i^2, first expanding the squared term,

$$s = M_1^2 + x + M_2^2 + x + 2\sqrt{\left(M_1^2 + x\right)\,\left(M_2^2 + x\right)} \tag{2.17}$$

then isolating the square root on one side

$$2\sqrt{\left(M_1^2 + x\right)\,\left(M_2^2 + x\right)} = s - M_1^2 - M_2^2 - 2\,x \tag{2.18}$$

then squaring both sides

$$4\,\left(M_1^2 + x\right)\,\left(M_2^2 + x\right) = \left(s - M_1^2 - M_2^2 - 2\,x\right)^2 \tag{2.19}$$

and finally bringing all terms to the left hand side to allow for some cancellations

$$4\,\left(M_1^2 + x\right)\,\left(M_2^2 + x\right) - \left(s - M_1^2 - M_2^2 - 2\,x\right)^2 = 0. \tag{2.20}$$

We let **`x`** represent p_i^2 in the following Maxima session.

```
(%i1) assume(s>0,m1>0,m2>0);
(%o1) [s > 0,m1 > 0,m2 > 0]
(%i2) eqn : expand( 4*(m1^2+x)*(m2^2+x) - (s-m1^2-m2^2-2*x)^2 );
(%o2) 4*s*x-s^2+2*m2^2*s+2*m1^2*s-m2^4+2*m1^2*m2^2-m1^4
(%i3) solve(eqn,x);
(%o3) [x = (s^2+((-2*m2^2)-2*m1^2)*s+m2^4-2*m1^2*m2^2+m1^4)/(4*s)]
(%i4) soln : rhs(%[1]);
(%o4) (s^2+((-2*m2^2)-2*m1^2)*s+m2^4-2*m1^2*m2^2+m1^4)/(4*s)
(%i5) solnf : factor(soln);
(%o5) ((s-m2^2-2*m1*m2-m1^2)*(s-m2^2+2*m1*m2-m1^2))/(4*s)
(%i6) display2d:true$
(%i7) solnf;
                     2                2         2                2
              (s - m2  - 2 m1 m2 - m1 ) (s - m2  + 2 m1 m2 - m1 )
(%o7)         ---------------------------------------------------
                                        4 s
```

which can be further simplified by hand to get agreement, or other brute force Maxima methods can be used, as shown here:

```
(%i8) display2d:false$
(%i9) Nf : num(solnf);
(%o9) (s-m2^2-2*m1*m2-m1^2)*(s-m2^2+2*m1*m2-m1^2)
(%i10) Nf1 : part(Nf,1);
(%o10) s-m2^2-2*m1*m2-m1^2
(%i11) Nf1ms : Nf1-s;
(%o11) (-m2^2)-2*m1*m2-m1^2
(%i12) Nf1msf : factor(Nf1ms);
(%o12) -(m2+m1)^2
```

```
(%i13) Nf1 : Nf1msf + s;
(%o13) s-(m2+m1)^2
(%i14) Nf2 : part(Nf,2);
(%o14) s-m2^2+2*m1*m2-m1^2
(%i15) Nf2ms : Nf2 -s;
(%o15) (-m2^2)+2*m1*m2-m1^2
(%i16) Nf2msf : factor(Nf2ms);
(%o16) -(m2-m1)^2
(%i17) Nf2 : Nf2msf + s;
(%o17) s-(m2-m1)^2
(%i18) solnf_new : Nf1*Nf2/(4*s);
(%o18) ((s-(m2-m1)^2)*(s-(m2+m1)^2))/(4*s)
(%i19) display2d:true$
(%i20) solnf_new;
                                      2                2
                       (s - (m2 - m1) ) (s - (m2 + m1) )
(%o20)                 ---------------------------------
                                      4 s
```

We can then find E_1^2, and then E_1

```
(%i21) factor(m1^2 + solnf_new);
                                       2     2 2
                               (s - m2  + m1 )
(%o21)                         ----------------
                                     4 s
(%i22) sqrt(%);
                                !      2     2!
                                !s - m2  + m1 !
(%o22)                          ---------------
                                   2 sqrt(s)
```

Thus

$$E_1 = \frac{\left(s + M_1^2 - M_2^2\right)}{2\sqrt{s}}. \tag{2.21}$$

Likewise for E_2,

```
(%i23) factor(m2^2 + solnf_new);
                                       2     2 2
                               (s + m2  - m1 )
(%o23)                         ----------------
                                     4 s
```

Thus

$$E_2 = \frac{\left(s + M_2^2 - M_1^2\right)}{2\sqrt{s}}. \tag{2.22}$$

Similarly, from the equation

$$s = (E_3 + E_4)^2 = \left[\sqrt{M_3^2 + p_f^2} + \sqrt{M_4^2 + p_f^2}\right]^2 \tag{2.23}$$

we can take over the previous solutions by the replacements

$$E_1 \to E_3,\ E_2 \to E_4,\ M_1 \to M_3,\ M_2 \to M_4,\ p_i^2 \to p_f^2 \tag{2.24}$$

to get

$$p_f^2 = \frac{\left[s - (M_3 + M_4)^2\right]\ \left[s - (M_3 - M_4)^2\right]}{4\,s},$$

$$E_3 = \frac{\left(s + M_3^2 - M_4^2\right)}{2\sqrt{s}},$$

$$E_4 = \frac{\left(s + M_4^2 - M_3^2\right)}{2\sqrt{s}}.$$

The Mandelstam variables t and u involve the CM frame scattering angle θ.

$$\begin{aligned} t &= (p_1 - p_3)^2 \\ &= M_1^2 + M_3^2 - 2\,p_1 \cdot p_3 \\ &= M_1^2 + M_3^2 - 2\,E_1\,E_3 + 2\,p_i\,p_f\,\cos\theta \end{aligned}$$

and likewise

$$\begin{aligned} u &= (p_1 - p_4)^2 \\ &= M_1^2 + M_4^2 - 2\,p_1 \cdot p_4 \\ &= M_1^2 + M_4^2 - 2\,E_1\,E_4 - 2\,p_i\,p_f\,\cos\theta. \end{aligned}$$

An alternative way to write down the values of p_i^2 and p_f^2 is based on the definition of a function

$$\lambda(a,b,c) = a^2 + b^2 + c^2 - 2\,(ab + ac + bc), \tag{2.25}$$

which is a symmetric function of its arguments $\lambda(b,a,c) = \lambda(a,b,c)$ etc. Using this function, we can write

$$p_i^2 = \frac{\lambda\left(s, M_1^2, M_2^2\right)}{4s} \tag{2.26}$$

$$p_f^2 = \frac{\lambda\left(s, M_3^2, M_4^2\right)}{4s} \tag{2.27}$$

We can check the equivalence, using Maxima, for p_i^2:

```
(%i1) lam(a,b,c) := a^2 + b^2 + c^2 -2*(a*b + a*c + b*c)$
(%i2) val1 : expand (lam(s,m1^2,m2^2));
(%o2) s^2-2*m2^2*s-2*m1^2*s+m2^4-2*m1^2*m2^2+m1^4
(%i3) val2 : expand( (s - (m1+m2)^2)*(s - (m1-m2)^2) );
(%o3) s^2-2*m2^2*s-2*m1^2*s+m2^4-2*m1^2*m2^2+m1^4
(%i4) val1 - val2;
(%o4) 0
(%i5) is(equal(lam(b,a,c),lam(a,b,c)));
(%o5) true
```

From the above expressions for p_i^2 and p_f^2, it is obvious that if we consider *elastic* scattering, in the sense that $m_1 = m$, $m_3 = m$, $m_2 = M$, and $m_4 = M$, then $p_i = p_f$. Which implies that $|\vec{p}_1| = |\vec{p}_3|$ and $|\vec{p}_2| = |\vec{p}_4|$.

If we express $\cos\theta$ in terms of t, we have

$$\begin{aligned} 2p_ip_f\cos\theta &= t - M_1^2 - M_3^2 + 2E_1E_3 \\ &= t - M_1^2 - M_3^2 + 2\frac{\left(s + M_1^2 - M_2^2\right)}{2\sqrt{s}}\frac{\left(s + M_3^2 - M_4^2\right)}{2\sqrt{s}} \\ &= \frac{2s\left(t - M_1^2 - M_3^2\right) + \left(s + M_1^2 - M_2^2\right)\left(s + M_3^2 - M_4^2\right)}{2s} \end{aligned}$$

Using the abbreviations

$$\begin{aligned} \lambda_{s12} &\equiv \lambda\left(s, M_1^2, M_2^2\right), \\ \lambda_{s34} &\equiv \lambda\left(s, M_3^2, M_4^2\right) \end{aligned}$$

we get for the cosine of the scattering angle θ in the CM frame:

$$\begin{aligned} \cos\theta &= \frac{2s\left(t - M_1^2 - M_3^2\right) + \left(s + M_1^2 - M_2^2\right)\left(s + M_3^2 - M_4^2\right)}{4sp_ip_f} \\ &= \frac{2s\left(t - M_1^2 - M_3^2\right) + \left(s + M_1^2 - M_2^2\right)\left(s + M_3^2 - M_4^2\right)}{\left[\lambda_{s12}\lambda_{s34}\right]^{1/2}} \end{aligned}$$

2.4 CM Frame Differential Scattering Cross Section for $(AB) \to (CD)$

For the $2 \to 2$ scattering process $(AB) \to (CD)$,

$$(M1, p_1) + (M2, p_2) \to (M3, p_3) + (M4, p_4) \tag{2.28}$$

the differential scattering cross section in the CM frame, for specified helicities of the non-scalar particles, is (see, for example, Francis Halzen and Alan D. Martin, "Quarks and Leptons: An Introductory course ...," Sec. 4.3, or Mark Thomson, Modern Particle Physics, Sec. 3.5.1)

$$\frac{d\sigma}{d\Omega}(\text{cms}) = \frac{1}{64\pi^2 s}\frac{p_f}{p_i}|\mathcal{M}_{fi}|^2. \tag{2.29}$$

In the CM frame $p_i \equiv |\vec{p}_1| = |\vec{p}_2|$ and $p_f \equiv |\vec{p}_3| = |\vec{p}_4|$.

If we define a "reduced" amplitude $\mathcal{M}_r$ by

$$\mathcal{M}_{fi} = -e^2\,\mathcal{M}_r, \tag{2.30}$$

and use

$$e^2 = 4\pi\alpha \tag{2.31}$$

we can instead use the simpler formula :

$$\frac{d\sigma}{d\Omega}(\text{cms}) = \frac{\alpha^2}{4s}\frac{p_f}{p_i}|\mathcal{M}_r|^2 = A\,|\mathcal{M}_r|^2, \tag{2.32}$$

where

$$A = \frac{\alpha^2}{4s}\frac{p_f}{p_i}. \tag{2.33}$$

These formulas (2.29) and (2.32) assume the non-scalar particles have specified helicities.

An "unpolarized" differential cross section can be then found by averaging over the helicities of the incident particles, and summing over the helicities of the final particles.

For *elastic* scattering, in the sense that $m_1 = m$, $m_3 = m$, $m_2 = M$, and $m_4 = M$, we have the simplification $p_f = p_i$.

2.5 Lab Frame Differential Scattering Cross Section for Elastic Scattering $(A\,B) \to (A\,B)$

See Peter Renton, Electroweak Interactions, Eq. (4.34), and David Griffiths, Introduction to Elementary Particles, Prob. 6.10.

For the elastic scattering process in which particle B with mass M is initially at rest, and particle A with mass m, initial energy E_1, and initial 3-momentum magnitude p_1 is scattered into solid angle element $d\Omega$ with final energy E_3, and final 3-momentum magnitude p_3, the differential scattering cross section for specified helicities is

$$\frac{d\sigma}{d\Omega} = \frac{1}{64\,\pi^2\,M}\left(\frac{p_3}{p_1}\right)\frac{|\mathcal{M}_{fi}|^2}{\left|E_1 + M - \left(\frac{p_1\,E_3}{p_3}\right)\,\cos\theta\right|}, \tag{2.34}$$

in which we have

$$\begin{aligned}
\mathbf{p_1}\cdot\mathbf{p_3} &= p_1\,p_3\,\cos\theta \\
E_1^2 &= m^2 + p_1^2 \\
E_3^2 &= m^2 + p_3^2 \\
E_3 &< E_1 + M \\
E_3\,(M + E_1) - p_1\,p_3\,\cos\theta &= E_1\,M + m^2
\end{aligned}$$

This last relation follows from conservation of 4-momentum and can be used in principle to solve for p_3 in terms of p_1. Since $(p_1+p_2)^\mu = (p_3+p_4)^\mu$, the 4-vector dot products $(p_1+p_2)^2$ and $(p_3+p_4)^2$ are equal, which implies $p_1\cdot p_2 = p_3\cdot p_4$. Then, with $\mathbf{p_2} = \mathbf{0}$, and $\mathbf{p_1}\cdot\mathbf{p_3} = p_1\, p_3\, \cos\theta$, one obtains the relation desired.

If we define a "reduced" amplitude $\mathcal{M}_r$ by

$$\mathcal{M}_{fi} = -e^2\,\mathcal{M}_r, \tag{2.35}$$

and use

$$e^2 = 4\pi\alpha \tag{2.36}$$

we can instead use the formula :

$$\frac{d\sigma}{d\Omega}(\text{cms}) = A_{\text{lab}}\,|\mathcal{M}_r|^2, \tag{2.37}$$

where

$$A_{\text{lab}} = \frac{\alpha^2}{4M}\frac{p_3}{p_1}\,\frac{1}{\left|E_1 + M - \left(\frac{p_1\,E_3}{p_3}\right)\,\cos\theta\right|} \tag{2.38}$$

In the limit $m\to 0$, let $p_1 \to k$ and $p_3 \to k'$. We then get

$$k' = \frac{k}{\left[1 + \frac{k}{M}\,(1-\cos\theta)\right]}. \tag{2.39}$$

and

$$A_{\text{lab}} = \frac{\alpha^2}{4M^2}\left(\frac{k'}{k}\right)^2. \tag{2.40}$$

2.6 Decay Rate for Two Body Decay

The total decay rate of an unstable particle is the sum of the decay rates for individual decay modes. Here we consider one decay mode of particle a to two particles

$$a \to 1\,+\,2. \tag{2.41}$$

Sec. 3.3 of Thomson (Modern Particle Physics) derives the decay rate for any such two-body decay [see also Griffith (Introduction to Elementary Paticles), pp. 194 - 198],

$$\Gamma_{f\,i} = \frac{p^*}{32\,\pi^2\,m_a^2}\int|\mathcal{M}_{f\,i}|^2\,d\,\Omega, \tag{2.42}$$

in which p^* is the center of momentum frame 3-momentum magnitude of each of the final particles, (see our derivation above in Sec. 2.3) given by [note Thomson's 2015 3rd printing has an extraneous, unmatched, parenthesis]

$$p^* = \frac{1}{2\,m_a}\sqrt{[m_a^2-(m_1+m_2)^2]\;[m_a^2-(m_1-m_2)^2]}. \tag{2.43}$$

As Griffith notes on p. 198, " ...in the case of three or more particles in the final state, the integrals cannot be done until we know the specific functional form of $\mathcal{M}_{f\,i}$."

3 Spin Sums

Let $M(\sigma_1,\sigma_2,\sigma_3,\sigma_4)$ be the amplitude for each set of initial and final helicity states for the process (as a concrete example) $e^+e^-\to\mu^+\mu^-$. Quoting Mark Thomson, Modern Particle Physics, Section 6.2.1, "Spin Sums,"

> …the process $e^+e^-\to\mu^+\mu^-$ consists of sixteen possible *orthogonal* helicity combinations, each of which constitutes a separate physical process, for example $e^+_Re^-_R\to\mu^+_R\mu^-_R$ and $e^+_Re^-_R\to\mu^+_R\mu^-_L$, where R stands for $\sigma = 1$ and L stands for $\sigma = -1$. Because the helicity states involved are orthogonal, the processes for the different helicity configurations do not interfere, and the matrix element squared for each of the sixteen

possible helicity configurations can be considered independently.

For a particular initial-state spin configuration, the total $e^+e^- \to \mu^+\mu^-$ annihilation rate is given by the sum of the *rates* for the four possible $\mu^+\mu^-$ helicity states (each of which is a separate process). Therefore, for a given *initial-state* helicity configuration, the cross section is obtained by taking the sum of the four corresponding $|M|^2$ terms. For example, for the case where the colliding electron and positron are both in right-handed helicity states,

$$\sum |M_{RR}|^2 = |M_{RR\to RR}|^2 + |M_{RR\to RL}|^2 + |M_{RR\to LR}|^2 + |M_{RR\to LL}|^2. \tag{3.1}$$

In most e^+e^- colliders, the colliding electron and positron beams are unpolarized, which means that there are equal numbers of positive and negative helicity electrons/positrons present in the initial state. In this case, the helicity configuration for a particular collision is equally likely to occur in any one of the four possible helicity states of the e^+e^- initial state. This is accounted for by defining the spin-averaged summed matrix element squared,

$$\begin{aligned}\langle |M_{fi}|^2\rangle &= \frac{1}{4}\left(|M_{RR}|^2 + |M_{RL}|^2 + |M_{LR}|^2 + |M_{LL}|^2\right)\\ &= \frac{1}{4}\left(|M_{RR\to RR}|^2 + |M_{RR\to RL}|^2 + \ldots + |M_{RL\to RR}|^2 + \ldots\right)\\ &= \frac{1}{4}\sum_{\sigma_1}\sum_{\sigma_2}\sum_{\sigma_3}\sum_{\sigma_4} |M(\sigma_1,\sigma_2,\sigma_3,\sigma_4)|^2\end{aligned}$$

or more compactly as

$$\langle |M_{fi}|^2\rangle = \frac{1}{4}\sum_{\text{spins}} |M|^2, \tag{3.2}$$

where the sum corresponds to all possible helicity configurations. . . . This sum can be performed in two ways. One possibility is to use trace techniques . . . where the sum is calculated directly using the properties of the Dirac spinors. The second possibility is to calculate each of the sixteen individual helicity amplitudes. This direct calculation of the helicity amplitudes involves more steps, but has the advantages of being conceptually simpler and of leading to a deeper physical understanding of the helicity structure of the QED interaction.

4 Spin $\frac{1}{2}$ Examples

Reading and working through the worksheets in the first few examples will give the reader a quick overview of the use of the Dirac3 package tools.

4.1 $e^-\, e^+ \to \mu^-\, \mu^+$, ee-mumu-AE.wxm

We discuss this simple example in detail in two wxMaxima worksheets.

The high energy limiting case in which we can negect the masses of the electron and muon is discussed in **`ee-mumu-HE.wxm`**. The arbitrary energy case, in which we retain the mass of the electron **`m`** and the mass of the muon **`M`** is discussed in **`ee-mumu-AE.wxm`**.

In all cases, we introduce what we call the "reduced amplitude" $\mathcal{M}_r$ by pulling out a factor $-e^2$ from the amplitude $\mathcal{M}$:

$$\mathcal{M} = -e^2\,\mathcal{M}_r. \tag{4.1}$$

We always interpret the Feynman rules as defining $-i\,\mathcal{M}$, following the practice of David Griffiths (Introduction to Elementary Particles, 1987) and Francis Halzen & Alan D. Martin (Quarks and Leptons, 1984).

The reduced amplitude for the process

$$e^-(p_1,\sigma_1)\,e^+(p_2,\sigma_2) \to \mu^-(p_3,\sigma_3)\,\mu^+(p_4,\sigma_4) \tag{4.2}$$

is

$$\mathcal{M}_r(\sigma_1,\sigma_2,\sigma_3,\sigma_4) = \frac{(\overline{v}_2\,\gamma^\mu\,u_1)\;(\overline{u}_3\,\gamma_\mu\,v_4)}{s} \tag{4.3}$$

and in which, for example, $u_1 = u(p_1,\sigma_1)$, etc. and a sum over the Lorentz index μ is understood (contraction of a product). The denominator s is the Mandelstam variable

$$s = (p_1+p_2)\cdot(p_1+p_2)\,. \tag{4.4}$$

4.2 $e^-\,\mu^- \to e^-\,\mu^-$, e-mu-scatt.wxm

We discuss this simple scattering example in detail in one wxMaxima worksheet, **`e-mu-scatt.wxm`**.

We neglect the mass of the electron `m` and retain the mass of the muon `M`. The "reduced amplitude" $\mathcal{M}_r$ is defined by pulling out a factor $-e^2$ from the amplitude $\mathcal{M}$:

$$\mathcal{M} = -e^2\,\mathcal{M}_r. \tag{4.5}$$

We always interpret the Feynman rules as defining $-i\,\mathcal{M}$, following the practice of David Griffiths (Introduction to Elementary Particles, 1987) and Francis Halzen & Alan D. Martin (Quarks and Leptons, 1984).

The reduced amplitude for the process

$$e^-(p_1,\sigma_1)\,\mu^-(p_2,\sigma_2) \to e^-(p_3,\sigma_3)\,\mu^-(p_4,\sigma_4) \tag{4.6}$$

is

$$\mathcal{M}_r(\sigma_1,\sigma_2,\sigma_3,\sigma_4) = \frac{(\overline{u}_3\,\gamma^\mu\,u_1)\;(\overline{u}_4\,\gamma_\mu\,u_2)}{t} \tag{4.7}$$

and in which, for example, $u_1 = u(p_1,\sigma_1)$, etc. and a sum over the Lorentz index μ is understood (contraction of a product). The denominator t is the Mandelstam variable

$$t = (p_1-p_3)\cdot(p_1-p_3)\,. \tag{4.8}$$

4.3 $e^-\,e^- \to e^-\,e^-$ (Moller Scattering), em-em-scatt-AE.wxm

We discuss this Moller scattering example in detail in two wxMaxima worksheets, **`em-em-scatt-HE.wxm`** for the high energy limiting case, and **`em-em-scatt-AE.wxm`** for the arbitrary energy case.

The "reduced amplitude" $\mathcal{M}_r$ is defined by pulling out a factor $-e^2$ from the amplitude $\mathcal{M}$:

$$\mathcal{M} = -e^2\,\mathcal{M}_r. \tag{4.9}$$

We always interpret the Feynman rules as defining $-i\,\mathcal{M}$, following the practice of David Griffiths (Introduction to Elementary Particles, 1987) and Francis Halzen & Alan D. Martin (Quarks and Leptons, 1984).

The reduced amplitude for the process

$$e^-(p_1,\sigma_1)\,e^-(p_2,\sigma_2) \to e^-(p_3,\sigma_3)\,e^-(p_4,\sigma_4) \tag{4.10}$$

is

$$\mathcal{M}_r(\sigma_1,\sigma_2,\sigma_3,\sigma_4) = \frac{(\overline{u}_3\,\gamma^\mu\,u_1)\;(\overline{u}_4\,\gamma_\mu\,u_2)}{t} - \frac{(\overline{u}_4\,\gamma^\mu\,u_1)\;(\overline{u}_3\,\gamma_\mu\,u_2)}{u} \tag{4.11}$$

and in which, for example, $u_1 = u(p_1,\sigma_1)$, etc. and a sum over the Lorentz index μ is understood (contraction of a product). The denominator t is the Mandelstam variable

$$t = (p_1-p_3)\cdot(p_1-p_3)\,. \tag{4.12}$$

and the denominator u is the Mandelstam variable

$$u = (p_1-p_4)\cdot(p_1-p_4)\,. \tag{4.13}$$

4.4 $e^- \, e^+ \to e^- \, e^+$ (Bhabha Scattering), em-ep-scatt.AE.wxm

We discuss this Bhabha scattering example in detail in two wxMaxima worksheets, **`em-ep-scatt-HE.wxm`** for the high energy limiting case, and **`em-ep-scatt-AE.wxm`** for the arbitrary energy case.

The "reduced amplitude" $\mathcal{M}_r$ is defined by pulling out a factor $-e^2$ from the amplitude $\mathcal{M}$:

$$\mathcal{M} = -e^2 \, \mathcal{M}_r. \tag{4.14}$$

We always interpret the Feynman rules as defining $-i \, \mathcal{M}$, following the practice of David Griffiths (Introduction to Elementary Particles, 1987) and Francis Halzen & Alan D. Martin (Quarks and Leptons, 1984).

The reduced amplitude for the process

$$e^-(p_1, \sigma_1) \, e^+(p_2, \sigma_2) \to e^-(p_3, \sigma_3) \, e^+(p_4, \sigma_4) \tag{4.15}$$

is

$$\mathcal{M}_r(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \frac{(\overline{u}_3 \, \gamma^\mu \, u_1) \; (\overline{v}_2 \, \gamma_\mu \, v_4)}{t} - \frac{(\overline{v}_2 \, \gamma^\mu \, u_1) \; (\overline{u}_3 \, \gamma_\mu \, v_4)}{s} \tag{4.16}$$

and in which, for example, $u_1 = u(p_1, \sigma_1)$, etc. and a sum over the Lorentz index μ is understood (contraction of a product). The denominator t is the Mandelstam variable

$$t = (p_1 - p_3) \cdot (p_1 - p_3) \, . \tag{4.17}$$

and the denominator s is the Mandelstam variable

$$s = (p_1 + p_2) \cdot (p_1 + p_2) \, . \tag{4.18}$$

5 Theorems for Physical External Photons

5.1 Photon Real Linear Polarization 3-Vector Theorems: photon1.wxm

In the wxMaxima worksheet **`photon1.wxm`** we prove two theorems that hold for sets of real linear polarization vectors for physical photons.

Let $\mathbf{e}_{\mathbf{k}\,\mathbf{s}}$ be a real photon polarization 3-vector which is transverse to the initial photon 3-momentum $\mathbf{k}$, with $s = 1$ being parallel to the scattering plane, and $s = 2$ being perpendicular to the scattering plane.

Let $\mathbf{e}_{\mathbf{k}'\,\mathbf{r}}$ be a real photon polarization 3-vector which is transverse to the final photon 3-momentum $\mathbf{k}'$ with $r = 1$ being parallel to the scattering plane, and $r = 2$ being perpendicular to the scattering plane. Then we have the relations

$$\sum_{\mathbf{s}=\mathbf{1}}^{\mathbf{2}} \left(\mathbf{e}_{\mathbf{k}\,\mathbf{s}} \cdot \mathbf{e}_{\mathbf{k}'\,\mathbf{r}}\right)^{\mathbf{2}} = \mathbf{1} - \left(\hat{\mathbf{k}} \cdot \mathbf{e}_{\mathbf{k}'\,\mathbf{r}}\right)^{\mathbf{2}} \tag{5.1}$$

$$\sum_{\mathbf{s},\mathbf{r}=\mathbf{1}}^{\mathbf{2}} \left(\mathbf{e}_{\mathbf{k}\,\mathbf{s}} \cdot \mathbf{e}_{\mathbf{k}'\,\mathbf{r}}\right)^{\mathbf{2}} = \mathbf{1} + \mathbf{cos}^{\mathbf{2}} \boldsymbol{\theta} \tag{5.2}$$

5.2 Photon Circular Polarization 4-Vectors, photon2.wxm

In the wxMaxima worksheets **`photon2.wxm`** and **`photon2.wxmx`** we construct explicit photon polarization 4-vectors for several common cases. The derivations involve starting with normalized and mutually perpendicular real linear polarization 3-vectors $\mathbf{e_1}$ and $\mathbf{e_2}$ such that $\mathbf{e_1} \times \mathbf{e_2} = \hat{\mathbf{k}}$, where $\hat{\mathbf{k}}$ is a unit 3-vector in the direction of propagation of the photon.

We then construct a complex positive helicity (right-handed) polarization 3-vector $\mathbf{e_R}$ according to (See, for example, Mark Thompson, Modern Particle Physics, pp. 527 - 528, Peskin and Schroeder, Quantum Field Theory, p. 804, and Halzen and Martin, Quarks and Leptons, p. 135, Eq (6.69)):

$$\mathbf{e_R} = -\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\,(\mathbf{e_1} + \mathbf{i}\,\mathbf{e_2}) \tag{5.3}$$

and a complex negative helicity (left-handed) polarization 3-vector $\mathbf{e_L}$ according to:

$$\mathbf{e_L} = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\,(\mathbf{e_1} - \mathbf{i}\,\mathbf{e_2}). \tag{5.4}$$

Both $\mathbf{e_R}$ and $\mathbf{e_L}$ are orthogonal to $\hat{\mathbf{k}}$ and have the properties

$$\begin{aligned} \mathbf{e_R} \cdot \mathbf{e_R}^* &= 1 \\ \mathbf{e_L} \cdot \mathbf{e_L}^* &= 1 \\ \mathbf{e_R} \cdot \mathbf{e_L}^* &= 0. \end{aligned}$$

A "transverse gauge" complex positive helicity (right-handed) polarization 4-vector e_R has components given by

$$e_R = (0, \mathbf{e_R}) \tag{5.5}$$

and the negative helicity (left-handed) polarization 4-vector e_L has the components

$$e_L = (0, \mathbf{e_L})\,. \tag{5.6}$$

Define the (real) four vector k with components

$$k = \left(0, \hat{\mathbf{k}}\right). \tag{5.7}$$

Then, we have the properties (in our choice of metric), in terms of the 4-vector (scalar) dot product, (the second through fifth lines reflect the Lorentz gauge condition)

$$\begin{aligned} k \cdot k &= -1 \\ k \cdot e_R &= 0 \\ k \cdot e_R^* &= 0 \\ k \cdot e_L &= 0 \\ k \cdot e_L^* &= 0 \\ e_R \cdot e_R^* &= -1 \\ e_L \cdot e_L^* &= -1 \\ e_R \cdot e_L^* &= 0. \end{aligned}$$

6 Examples for Spin $1/2$ and External Photons

6.1 $e^- e^+ \to \gamma\,\gamma$, ee-gaga-AE.wxm

We discuss this pair annihilation example in detail in two wxMaxima worksheets, **`ee-gaga-HE.wxm`** for the high energy limiting case, and **`ee-gaga-AE.wxm`** for the arbitrary energy case.

The “reduced amplitude” $\mathcal{M}_r$ is defined by pulling out a factor $-e^2$ from the amplitude $\mathcal{M}$:

$$\mathcal{M} = -e^2\,\mathcal{M}_r. \tag{6.1}$$

We always interpret the Feynman rules as defining $-i\,\mathcal{M}$, following the practice of David Griffiths (Introduction to Elementary Particles, 1987) and Francis Halzen & Alan D. Martin (Quarks and Leptons, 1984).

The reduced amplitude for the process

$$e^-(p_1,\sigma_1)\,e^+(p_2,\sigma_2) \to \gamma(k_1,\lambda_1)\,\gamma(k_2,\lambda_2) \tag{6.2}$$

is

$$\mathcal{M}_r(\sigma_1,\sigma_2,\lambda_1,\lambda_2) = M_1 + M_2, \tag{6.3}$$

where

$$M_1 = -\frac{(\overline{v}_2\,\not\epsilon_2^*\;(\not p_1 - \not k_1 + m)\;\not\epsilon_1^*\,u_1)}{t - m^2}$$
$$M_2 = -\frac{(\overline{v}_2\,\not\epsilon_1^*\;(\not p_1 - \not k_2 + m)\;\not\epsilon_2^*\,u_1)}{u - m^2}$$

and in which $u_1 = u(p_1,\sigma_1)$ and $v_2 = v(p_2,\sigma_2)$.

t and u are Mandelstam variables

$$t = (p_1 - k_1)\cdot(p_1 - k_1)$$
$$u = (p_1 - k_2)\cdot(p_1 - k_2)\,.$$

6.2 $\gamma\, e^- \to \gamma\, e^-$, compton1.wxm, compton-CMS-HE.wxm

We discuss Compton scattering in detail in two wxMaxima worksheets: **`compton-CMS-HE.wxm`** for the high energy limiting case in the center of momentum frame, and **`compton1.wxm`** for the arbitrary energy case in the frame in which the initial electron is at rest.

In the high energy center of momentum frame worksheet, we include the calculation of helicity amplitudes.

The “reduced amplitude” $\mathcal{M}_r$ is defined by pulling out a factor $-e^2$ from the amplitude $\mathcal{M}$:

$$\mathcal{M} = -e^2\,\mathcal{M}_r. \tag{6.4}$$

We always interpret the Feynman rules as defining $-i\,\mathcal{M}$, following the practice of David Griffiths (Introduction to Elementary Particles, 1987) and Francis Halzen & Alan D. Martin (Quarks and Leptons, 1984).

The reduced amplitude for the process

$$e^-(p_1,\sigma_1)\,\gamma(k_1,\lambda_1) \to \gamma(k_2,\lambda_2)\,e^+(p_2,\sigma_2) \tag{6.5}$$

is

$$\mathcal{M}_r(\sigma_1,\lambda_1,\sigma_2,\lambda_2) = M_1 + M_2, \tag{6.6}$$

where

$$M_1 = -\frac{(\overline{u}_2\,\not\epsilon_1\;(\not p_1 - \not k_2 + m)\;\not\epsilon_2^*\,u_1)}{u - m^2}$$
$$M_2 = -\frac{(\overline{u}_2\,\not\epsilon_2^*\;(\not p_1 + \not k_1 + m)\;\not\epsilon_1\,u_1)}{s - m^2}$$

and in which $u_1 = u(p_1,\sigma_1)$ and $u_2 = u(p_2,\sigma_2)$.

s and u are Mandelstam variables

$$s = (p_1 + k_1)\cdot(p_1 + k_1)$$
$$u = (p_1 - k_2)\cdot(p_1 - k_2)\,.$$

6.3 Covariant Physical External Photon Polarization 4-Vectors - A Review

Instead of using explicit circular polarization 4-vectors (as we have in the above examples), one can instead use the following formalism for covariant physical photon polarization 4-vectors which are 4-orthogonal to the photon 4-momentum vector. This formalism (Sterman, p. 220, De Wit/Smith, p.141, Schwinger, p. 73 and pp. 291 - 294, Jauch/Rohrlich, p. 440) uses gauge freedom within the Lorentz gauge.

The two physical polarization states of real external photons corresond to $\lambda = 1, 2$, and for $\mu,\ \nu = 0, 1, 2, 3$,

$$P^{\mu\nu}(k) = \sum_{\lambda=1,2} e^{\mu}_{k\,\lambda}\, e^{\nu *}_{k\,\lambda} = -g^{\mu\nu} + \frac{(k^{\mu}\,\bar{k}^{\nu} + k^{\nu}\,\bar{k}^{\mu})}{k\cdot\bar{k}}, \tag{6.7}$$

where there exists a reference frame in which if $k^{\mu} = (k^0, \mathbf{k})$ then $\bar{k}^{\mu} = (k^0, -\mathbf{k})$.

Let n^{μ} be a unit timelike 4-vector which is 4-orthogonal to $e^{\mu}_{k\,\lambda}$. Then define

$$\bar{k}^{\mu} = -k^{\mu} + 2\,n^{\mu}\,(k\cdot n). \tag{6.8}$$

In an arbitrary frame, $k\cdot k = 0$, $\bar{k}\cdot\bar{k} = 0$, $k\cdot e_{k\,\lambda} = 0$, $\bar{k}\cdot e_{k\,\lambda} = 0$, $n\cdot e_{k\,\lambda} = 0$, $n\cdot n = +1$, $k\cdot\bar{k} = 2\,(k\cdot n)^2$ and

$$e^{*}_{k\,\lambda}\cdot e_{k\,\lambda'} = -\delta_{\lambda\,\lambda'}, \quad \lambda, \lambda' = 1, 2 \tag{6.9}$$

An alternative but less compact form of the physical photon polarization tensor $P^{\mu\nu}(k)$ is

$$P^{\mu\nu}(k) = \sum_{\lambda=1,2} e^{\mu}_{k\,\lambda}\, e^{\nu *}_{k\,\lambda} = -g^{\mu\nu} - \frac{k^{\mu}\,k^{\nu}}{(k\cdot n)^2} + \frac{(k^{\mu}\,n^{\nu} + k^{\nu}\,n^{\mu})}{k\cdot n} \tag{6.10}$$

We can always identify the unit timelike 4-vector n^{μ} with either the total 4-momentum vector (thus working in the CM frame) or the 4-momentum of a particular particle (thus working in the rest frame of that particle).

Thus if in a chosen frame, $\{p^{\mu}\} = \{m, 0, 0, 0\}$, we identify $n^{\mu} = p^{\mu}/m$, so that $\{n^{\mu}\} = \{1, 0, 0, 0\}$, then in that frame $\bar{k}^0 = k^0$ and $\bar{k}^j = -k^j$.

Then $n\cdot e_{k\,\lambda} = 0$ implies that $e^0_{k\,\lambda} = 0$, and then $k\cdot e_{k\,\lambda} = 0$ reduces to $\mathbf{k}\cdot\mathbf{e}_{k\,\lambda} = 0$ (for $\lambda = 1, 2$), and the two polarization 3-vectors $\mathbf{e}_{k\,\lambda}$ must be chosen perpendicular to $\mathbf{k}$ and to each other: $\mathbf{e}^{*}_{k\,1}\cdot\mathbf{e}_{k\,2} = 0$.

7 Examples Which Involve Scalar Particles

These examples will familiarize the reader with the use of the functions **`set_invarR`**, **`ev_Ds`**, and the list **`invarR`**. We write the cross section in a frame independent way in terms of the Mandelstam variables s, t, and u, and then specialize to the center of momentum frame.

Recall the definitions of the Mandelstam variables appropriate to the process $p_1 + p_2 \to p_3 + p_4$.

$$s = (p_1 + p_2)^2 = (p_3 + p_4)^2, \tag{7.1}$$

$$t = (p_1 - p_3)^2 = (p_2 - p_4)^2, \tag{7.2}$$

$$u = (p_1 - p_4)^2 = (p_2 - p_3)^2. \tag{7.3}$$

The Feynman rules for scalar electrodynamics are discussed in:
(See the reference list at end of chapter 12)
G/R, QED, p. 434 ff, Renton, p.180 ff, I/Z, p. 282 ff,
B/D, RQM, p. 195, Aitchison, p. 51 ff, A/H, p. 158 ff,
Kaku, p. 158 ff, Quigg, p. 49, Schwinger, p. 284 ff,
H/M, Ch. 4.

7.1 $\pi^+ K^+ \to \pi^+ K^+$

We calculate the differential scattering cross section in the center of momentum frame for the elastic scattering of a pion by a kaon in the wxMaxima worksheet **`piKaon.wxm`**.

We ignore the finite size and quark content (and potential strong interaction effects in small impact parameter collisions) in this example and only take into account the electromagnetic interactions of these spinless bosons (thus treated as point particles).

We interpret the Feynman diagram rules to produce an expression for $-i\,\mathcal{M}_{fi}$, following the convention of Griffiths and Halzen/Martin.

The Lorentz invariant amplitude for the process

$$\pi^+(m,p_1) + K^+(M,p_2) \to \pi^+(m,p_3) + K^+(M,p_4) \tag{7.4}$$

is, using the Feynman diagram rules, (external spinless boson legs each produce a factor of 1)

$$\mathcal{M}_{fi} = -e^2\,\frac{(p_1+p_3)\cdot(p_2+p_4)}{(p_1-p_3)^2}, \tag{7.5}$$

and we let

$$\mathcal{M}_{fi} = -e^2\,\mathcal{M}_r, \tag{7.6}$$

thus defining a “reduced” amplitude $\mathcal{M}_r$.

We use the symbolic notation **`D(pa,pb)`** in our package to represent the 4-momentum dot product, and **`D`** is declared to be a symmetric function of its arguments. We can then write $\mathcal{M}_r$ in terms of **`D`**’s, and later use the Dirac3 package function **`ev_Ds(expr)`**, which makes use of the list **`invarR`**.

7.2 $\pi^+\pi^+ \to \pi^+\pi^+$

We calculate the differential scattering cross section in the center of momentum frame for the elastic scattering of a positive pion by a positive pion in the wxMaxima worksheet **`pi-pi-scatt.wxm`**.

We ignore the finite size and quark content (and potential strong interaction effects in small impact parameter collisions) in this example and only take into account the electromagnetic interactions of these spinless bosons (thus treated as point particles).

We interpret the Feynman diagram rules to produce an expression for $-i\,\mathcal{M}_{fi}$, following the convention of Griffiths and Halzen/Martin.

The Lorentz invariant amplitude for the process,

$$\pi^+(p_1) + \pi^+(p_2) \to \pi^+(p_3) + \pi^+(p_4) \tag{7.7}$$

is, using our conventions, $\mathcal{M}_{fi} = -e^2\,\mathcal{M}_r$, where the “reduced” amplitude $\mathcal{M}_r = M_1 + M_2$, in which the “t-channel photon” diagram corresponds to the (reduced) amplitude

$$M_1 = \frac{(p_1+p_3)\cdot(p_2+p_4)}{t}, \tag{7.8}$$

and the “u-channel photon” diagram corresponds to the (reduced) amplitude

$$M_2 = \frac{(p_1+p_4)\cdot(p_2+p_3)}{u}. \tag{7.9}$$

The Mandelstam variables used here are $t = (p_1 - p_3)^2$ and $u = (p_1 - p_4)^2$.

We first express the reduced amplitude in terms of the symbolic 4-vector dot product **`D(pa,pb)`** symbols.
The fastest path to the cross section is to then use **`comp_def`** to define the frame dependent 4-momenta and then **`noncov_ratio(expr)`** to make use of the definitions provided by first using **`comp_def`**.

We then follow the longer and more involved path in which we write the amplitude in terms of the Mandelstam variables. This path requires the use of **`set_invarR`**, **`ev_Ds`**, **`sub_stu`**, and **`VP`**. (Of course also needed is **`comp_def`**, but this was already used in the first method.)

7.3 $\mathbf{e^- \pi^+ \to e^- \pi^+}$

We calculate the differential scattering cross section in the center of momentum frame for the elastic scattering of an electron (mass **`m`**) by a positive pion (mass **`M`**), the process

$$e^-(p_1) + \pi^+(p_2) \to e^-(p_3) + \pi^+(p_4), \tag{7.10}$$

in the wxMaxima worksheets **`ePi-HE.wxm`** and **`ePi-AE.wxm`**.

We ignore the finite size and quark content (and potential strong interaction effects in small impact parameter collisions) in this example and only take into account the electromagnetic interactions of the pion with the electron.

We interpret the Feynman diagram rules to produce an expression for $-i\,\mathcal{M}_{fi}$, following the convention of Griffiths and Halzen/Martin.

For *elastic* scattering, in the sense that $M_1 = m$, $M_3 = m$, $M_2 = M$, and $M_4 = M$, the magnitude of the initial 3-momentum is equal to the magnitude of the final 3-momentum.

The differential scattering cross section in CM frame for our case, when summing over the spin of the final electron (4-momentum p_3) and averaging over the spin of the initial electron (4-momentum p_1) is

$$\frac{d\sigma}{d\Omega}(\text{cms}) = \frac{1}{64\pi^2 s}\left\langle |\mathcal{M}_{fi}|^2 \right\rangle, \tag{7.11}$$

in which

$$\left\langle |\mathcal{M}_{fi}|^2 \right\rangle = \frac{1}{2}\sum_{\sigma_1}\sum_{\sigma_3} |\mathcal{M}_{fi}(\sigma_1, \sigma_3)|^2 \tag{7.12}$$

which becomes (using $e^2 = 4\pi\alpha$ in natural units)

$$\left\langle |\mathcal{M}_{fi}|^2 \right\rangle = \left(\frac{4\pi\alpha}{t}\right)^2 L_{\mu\nu} T^{\mu\nu}. \tag{7.13}$$

The "lepton tensor" $L^{\mu\nu}$ (see A/H, p. 182)

$$L^{\mu\nu} = \frac{1}{2} Tr\left\{(\not{p}_3 + m)\,\gamma^\mu\,(\not{p}_1 + m)\,\gamma^\nu\right\} \tag{7.14}$$

is a matrix product, summed over diagonal elements (i.e., a *trace*), and the "pion tensor" is

$$T^{\mu\nu} = (p_2 + p_4)^\mu\,(p_2 + p_4)^\nu\,. \tag{7.15}$$

We let **`L`** represent the lepton tensor and **`T`** represent the pion tensor in our wxMaxima worksheet.

The helicity specific amplitude referred to above is, using our conventions,

$$\mathcal{M}_{fi}(\sigma_1, \sigma_3) = \frac{e^2}{t}\;\left(\overline{u}\,(p_3, \sigma_3)\;\gamma^\mu\,u\,(p_1, \sigma_1)\right)\;(p_2 + p_4)_\mu\,, \tag{7.16}$$

in which $t = (p_1 - p_3)^2$ and $e^2 = 4\,\pi\,\alpha$.

7.4 $\gamma\,\pi^+ \to \gamma\,\pi^+$

We calculate the differential scattering cross section for the scattering of a photon by a positive pion in the rest frame of the initial pion

$$\gamma(k_1, \lambda_1) + \pi^+(p_1) \to \gamma(k_2, \lambda_2) + \pi^+(p_2), \tag{7.17}$$

in the wxMaxima worksheet **`compton0.wxm`**.

The differential cross sections are first calculated for specific photon helicities λ_1 and λ_2, and these results are then turned into the unpolarized differential cross section.

See Peter Renton, Electroweak Interactions, Fig. 4.10 (p. 181) and the bottom of p. 184 and following for details of the lowest order amplitude for this process.

There are three diagrams, and the amplitude is written in terms of the photon polarization 4-vectors ε_1 and ε_2 and the mass m of the pion:

$$M = M_1 + M_2 + M_3, \tag{7.18}$$

where

$$M_1 = \frac{e^2}{s - m^2}\,\varepsilon_1 \cdot (p_1 + q)\ \varepsilon_2^* \cdot (p_2 + q)\,, \tag{7.19}$$

in which $s = (p_1 + k_1)^2$, $q = p_1 + k_1$, $\varepsilon_1 = \varepsilon\,(k_1, \lambda_1)$, and $\varepsilon_2 = \varepsilon\,(k_2, \lambda_2)$.

We also then have

$$M_2 = \frac{e^2}{u - m^2}\,\varepsilon_2^* \cdot \left(p_1 + q'\right)\ \varepsilon_1 \cdot \left(p_2 + q'\right)\,, \tag{7.20}$$

in which $u = (p_1 - k_2)^2$, $q' = p_1 - k_2$. The third term is

$$M_3 = -2\,e^2\,\varepsilon_2^* \cdot \varepsilon_1. \tag{7.21}$$

The photon polarization 4-vectors are constructed (in **`photon2.wxm`**) to satisfy the Lorentz gauge conditions $k_1 \cdot \varepsilon_1 = 0$ and $k_2 \cdot \varepsilon_2 = 0$ so M_1 and M_2 can be simplified to (after using $p_1 + k_1 = p_2 + k_2$)

$$M_1 = \frac{4\,e^2}{s - m^2}\,\varepsilon_1 \cdot p_1\,\varepsilon_2^* \cdot p_2, \tag{7.22}$$

$$M_2 = \frac{4\,e^2}{u - m^2}\,\varepsilon_2^* \cdot p_1\,\varepsilon_1 \cdot p_2. \tag{7.23}$$

We use gauge freedom within the Lorentz gauge to use a “transverse gauge”, such that $\varepsilon_1^0 = 0$ and $\varepsilon_2^0 = 0$.
Then $\varepsilon_1^\mu = [0, \boldsymbol{\epsilon}\,(\mathbf{k}_1, \boldsymbol{\lambda}_1)$ and $\varepsilon_2^\mu = [0, \boldsymbol{\epsilon}\,(\mathbf{k}_2, \boldsymbol{\lambda}_2)$.

We are working in the lab frame of the initial pion, for which $p_1^\mu = [m, 0, 0, 0]$, so $p_1 \cdot \varepsilon_1 = 0$ and $p_1 \cdot \varepsilon_2^* = 0$, so only term M_3 contributes, and

$$M = -2\,e^2\,\varepsilon_2^* \cdot \varepsilon_1. \tag{7.24}$$

Using explicit circular polarization vectors worked out in **`photon2.wxm`**, we can use this to evaluate the four possible photon helicity amplitudes for the cases $R \to R$, $R \to L$, $L \to R$, and $L \to L$.

8 Electroweak Unification Examples

8.1 W-boson Decay

Our reference for this section is Mark Thomson, Modern Particle Physics, Sec. 15.1.1, and the details of the calculation are given in the wxMaxima worksheet **`W-decay.wxm`**. Quoting Thomson (with some minor editing and additions):

> The calculation of the W-boson decay rate provides a good illustration of the use of polarization four-vectors in matrix element calculations. The lowest-order Feynman diagram for the $W^- \to e^- \,\overline{\nu}_e$ decay is shown in Figure 15.1.

The figure referred to consists of a single vertex diagram with three external legs.

> The matrix element for the decay is obtained using the appropriate Feynman rules. The final-state electron and antineutrino are written respectively as the adjoint particle spinor $\overline{u}\,(p_3,\,\sigma_3)$ and the antiparticle spinor $v\,(p_4,\,\sigma_4)$. The polarization state of the initial W^- is $\epsilon^\mu\,(p_1,\,\lambda)$, in which $\lambda = 0, \pm 1$. If $\vec{p}_1 = |\vec{p}_1|\,\hat{z}$, then $\epsilon^\mu\,(p_1,\,\pm 1)$ describes the states $|S, S_z >= |1, \pm 1>$, and $\epsilon^\mu\,(p_1,\,0)$ describes the state $|S, S_z >= |1, 0>$. Explicit 4-vector expressions for $\epsilon^\mu\,(p_1,\,\lambda)$ for this case are derived in the worksheet **`massive-spin1-pol.wxm`**.
>
> Finally, the vertex factor for the weak charged-current is the usual V - A interaction
>
> $$-i\,\frac{g_{\rm W}}{\sqrt{2}}\,\frac{1}{2}\gamma^\mu\left(1-\gamma^5\right). \tag{8.1}$$
>
> Using these Feynman rules, the matrix element for $W^- \to e^- \,\overline{\nu}_e$ is given by
>
> $$-i\,\mathcal{M}_{f\,i} = \epsilon_\mu\,(p_1,\,\lambda)\;\overline{u}\,(p_3,\,\sigma_3)\left[-i\,\frac{g_{\rm W}}{\sqrt{2}}\,\frac{1}{2}\gamma^\mu\left(1-\gamma^5\right)\right]\,v\,(p_4,\,\sigma_4)\,, \tag{8.2}$$
>
> and therefore
>
> $$\mathcal{M}_{f\,i} = \frac{g_{\rm W}}{\sqrt{2}}\,\epsilon_\mu\,(p_1,\,\lambda)\;\overline{u}\,(p_3,\,\sigma_3)\;\gamma^\mu\,S(-1)\,v\,(p_4,\,\sigma_4)\,, \tag{8.3}$$
>
> where the “chiral projection operator” S used here is defined as
>
> $$S(\sigma) = \frac{1}{2}\,\left(1+\sigma\,\gamma^5\right). \tag{8.4}$$

A neutrino is left-handed (negative helicity) (and massless), and an antineutrino is right-handed (positive helicity), so in this calculation we must take $\sigma_4 = 1$.

Quoting Thomson, Sec. 6.4,

> Any Dirac spinor (whether representing a massive or massless particle) can be decomposed into left- and right-handed chiral components using the “chiral projection operators”, P_L and P_R, defined by
>
> $$P_R = S(1)$$
> $$P_L = S(-1)$$
>
> Using the properties of the γ^5 matrix, it is straightforward to show that P_R and P_L behave as needed for quantum mechanical projection operators:
>
> $$P_R + P_L = 1, \quad P_R\,P_R = P_R, \quad P_L\,P_L = P_L, \quad P_L\,P_R = 0. \tag{8.5}$$

9 Dirac Gamma Matrices γ^μ and γ^5

We use the same notation conventions as P/S (Peskin and Schroeder) and Maggiore, the diagonal elements of the metric tensor $g^{\mu\nu}$ are $(1, -1, -1, -1)$,

$$g^{\mu\nu} = g_{\mu\nu} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{9.1}$$

with Greek indices running over $0, 1, 2, 3$. We also have

$$g^\mu_{\ \nu} = \delta^\mu_{\ \nu} = \begin{cases} 1 & \text{if } \mu = \nu \\ 0 & \text{otherwise}, \end{cases} \tag{9.2}$$

$\boldsymbol{\epsilon^{0123}} = +1$, the electromagnetic units are Heaviside-Lorentz, and natural units are used, so $\mathbf{e^2} = \boldsymbol{4\pi\alpha}$.

The only exception is that we interpret the Feynman rules as an expression for the quantity $-i\,M$, following the conventions of Griffiths, and Halzen/Martin.

The helicity (chiral) representation is used for explicit matrix methods; in this representation, the Dirac gamma matrices (in 2×2 block form) are

$$\gamma^0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \gamma^j = \begin{pmatrix} 0 & \sigma^j \\ -\sigma^j & 0 \end{pmatrix}, \quad \gamma^5 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{9.3}$$

in which σ^j for $j = 1, 2, 3$ are the 2×2 Pauli matrices

$$\sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{9.4}$$

The above form for $\boldsymbol{\gamma^5}$ is equivalent to the matrix product

$$\boldsymbol{\gamma^5 = i\,\gamma^0\,\gamma^1\,\gamma^2\,\gamma^3}. \tag{9.5}$$

Again using 2×2 block form, the 4×4 unit matrix is denoted by

$$\mathbb{1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{9.6}$$

The curly bracket denotes the anticommutator:

$$\{A, B\} \equiv A\,B + B\,A. \tag{9.7}$$

The gamma matrices satisfy the anticommutation relations

$$\{\gamma^\mu, \gamma^\nu\} = 2\,g^{\mu\nu}\,\mathbb{1}, \quad \{\gamma^5, \gamma^\mu\} = \mathbb{0}, \quad \mu = 0, 1, 2, 3, \tag{9.8}$$

in which $\mathbb{0}$ is the 4×4 zero matrix.

The second anticommutator relation above implies

$$\gamma^5\,\gamma^\mu = -\gamma^\mu\,\gamma^5, \quad \text{for} \quad \mu = 0, 1, 2, 3. \tag{9.9}$$

$A^\dagger$ is the hermitian conjugate of A, A^{T} is the transpose of A, and A^* is the complex conjugate of A.

$$A^\dagger \equiv (A^{\mathrm{T}})^*. \tag{9.10}$$

γ^0 is both real and symmetric and hence hermitian.

$$\left(\gamma^0\right)^* = \gamma^0, \quad \left(\gamma^0\right)^{\mathrm{T}} = \gamma^0, \tag{9.11}$$

$$\left(\gamma^0\right)^\dagger = \gamma^0, \quad \gamma^0\,\gamma^0 = \mathbb{1}, \tag{9.12}$$

$$(\gamma^r)^\dagger = -\gamma^r, \quad \gamma^r\,\gamma^r = -\mathbb{1}, \quad r = 1, 2, 3, \tag{9.13}$$

$$\left(\gamma^5\right)^\dagger = \gamma^5, \quad \gamma^5\,\gamma^5 = \mathbb{1}. \tag{9.14}$$

10 Lepton Spinor $u(p, \sigma)$ Conventions

If p refers to the particle momentum four-vector p^μ, the lepton Dirac spinor is the four element column matrix $u(p, \sigma)$ where $\sigma = +1$ picks out positive helicity $h = +1/2$ and $\sigma = -1$ picks out negative helicity $h = -1/2$. We also have the basic property

$$\not{p}\, u(p, \sigma) = m\, u(p, \sigma), \tag{10.1}$$

where the Feynman slashed momentum symbol is (using the summation convention)

$$\not{p} \equiv \gamma_\mu\, p^\mu. \tag{10.2}$$

The corresponding barred particle spinor is a four element row matrix

$$\bar{u}(p, \sigma) \equiv u(p, \sigma)^\dagger\, \gamma^0. \tag{10.3}$$

The component $\bar{u}_a$ is then defined by (using the summation convention for repeated matrix indices)

$$\bar{u}_a = u_b^*\, \gamma^0_{ba}. \tag{10.4}$$

The lepton Dirac spinor normalization is

$$\bar{u}(p, \sigma)\, u(p, \sigma') = 2\, m\, \delta_{\sigma\,\sigma'}, \quad \sigma,\, \sigma' = \pm 1, \tag{10.5}$$

and we also have the “projection matrix” relation

$$\sum_{\sigma = \pm 1} u(p, \sigma)\, \bar{u}(p, \sigma) = \not{p} + m, \tag{10.6}$$

in which the 4×4 identity matrix $\mathbb{1}$ is understood to be multiplying the scalar mass symbol m, and again $\not{p} = p_\mu\, \gamma^\mu$.

11 Anti-Lepton Spinor $v(p, \sigma)$ Conventions

The **antilepton** Dirac spinor is the four element column matrix $v(p, \sigma)$ in which $\sigma = +1$ picks out the positive helicity $h = +1/2$ antiparticle state and $\sigma = -1$ picks out negative helicity $h = -1/2$ antiparticle state, and is defined (with P/S phase choice) via

$$v(p, \sigma) = -\gamma^5\, u(p, -\sigma) \tag{11.1}$$

with the basic property

$$\not{p}\, v(p, \sigma) = -m\, v(p, \sigma). \tag{11.2}$$

The corresponding barred antiparticle spinor is a four element row matrix

$$\bar{v}(p, \sigma) \equiv v(p, \sigma)^\dagger\, \gamma^0 \tag{11.3}$$

and the normalization is

$$\bar{v}(p, \sigma)\, v(p, \sigma') = -2\, m\, \delta_{\sigma\,\sigma'} \quad \sigma,\, \sigma' = \pm 1. \tag{11.4}$$

We also have the “projection matrix” relation

$$\sum_{\sigma} v(p, \sigma)\, \bar{v}(p, \sigma) = \not{p} - m \tag{11.5}$$

12 Polarized → Unpolarized, Traces

Calculation of unpolarized cross sections requires summing the absolute value squared of a polarized amplitude over the helicities of the final particles and averaging over the helicities of the initial particles. The result can be written as the matrix trace of a 4×4 matrix. Here is an example. We use a compressed notation in which, for example

$$u_1 \equiv u(p_1, \sigma_1), \quad u_2 \equiv u(p_2, \sigma_2). \tag{12.1}$$

Recall that A^* is the complex conjugate of A, and $\bar{u} \equiv u^\dagger \gamma^0$. If u is a column matrix, then both $u^\dagger$ and u^{T} are row matrices. Let Γ be an **arbitrary** 4×4 matrix.

One can show that

$$(\bar{u}_2 \, \Gamma \, u_1)^* = \bar{u}_1 \, \bar{\Gamma} \, u_2 \tag{12.2}$$

in which

$$\bar{\Gamma} \equiv \gamma^0 \, \Gamma^\dagger \, \gamma^0. \tag{12.3}$$

Thus, making use of the reality and symmetry of γ^0, and displaying matrix indices and using the summation convention for those matrix indices)

$$\begin{aligned}
(\bar{u}_2 \, \Gamma \, u_1)^* &= \left(\bar{u}_2 \, \Gamma \, \gamma^0 \, \gamma^0 \, u_1\right)^* \\
&= \left\{\bar{u}_{2a} \, \left(\Gamma \, \gamma^0 \, \gamma^0\right)_{ab} \, u_{1b}\right\}^* \\
&= \left\{\left(u^*_{2c} \, \gamma^0_{ca}\right) \, \left(\Gamma \, \gamma^0 \, \gamma^0\right)_{ab} \, u_{1b}\right\}^* \\
&= \left\{u^*_{2c} \, \gamma^0_{ca} \, \Gamma_{ad} \, \gamma^0_{de} \, \gamma^0_{eb} \, u_{1b}\right\}^* \\
&= u^*_{1b} \, \gamma^0_{be} \, \gamma^0_{ca} \, \Gamma^*_{ad} \, \gamma^0_{de} \, u_{2c} \\
&= \bar{u}_{1e} \, \gamma^0_{ed} \, \Gamma^\dagger_{da} \, \gamma^0_{ac} \, u_{2c} \\
&= \bar{u}_{1e} \, \bar{\Gamma}_{ec} \, u_{2c} \\
&= \bar{u}_1 \, \bar{\Gamma} \, u_2.
\end{aligned}$$

We made no use of the properties of the lepton spinors in the above manipulation, so a similar identity holds for any pair w_1 and w_2 of four-element column vectors.

We can now use this result as follows:

$$\sum_{\sigma_1 \, \sigma_2} |\bar{u}_2 \, \Gamma \, u_1|^2 = \sum_{\sigma_1 \, \sigma_2} \bar{u}_2 \, \Gamma \, u_1 \, \bar{u}_1 \, \bar{\Gamma} \, u_2 \tag{12.4}$$

We use the sum over σ_1 to replace $u_1 \, \bar{u}_1$ by $(\not{p}_1 + m)$, and are left with

$$\sum_{\sigma_2} \bar{u}_2 \, \hat{\Gamma} \, u_2, \quad \hat{\Gamma} \doteq \Gamma \, (\not{p}_1 + m) \, \bar{\Gamma}. \tag{12.5}$$

Writing out the matrix indices explicitly, the sum becomes (using the summation convention for repeated matrix indices)

$$\sum_{\sigma_2} \bar{u}_{2\,a} \, \hat{\Gamma}_{a\,b} \, u_{2\,b} = \hat{\Gamma}_{a\,b} \sum_{\sigma_2} u_{2\,b} \, \bar{u}_{2\,a} = \hat{\Gamma}_{a\,b} \, (\not{p}_2 + m)_{b\,a} = \textbf{Trace} \, \{\hat{\Gamma} \, (\not{p}_2 + m)\} \tag{12.6}$$

Note that **Trace** $(A\,B) =$ **Trace** $(B\,A)$. We then have

$$\sum_{\sigma_1\,\sigma_2} |\bar{u}_2\,\Gamma\,u_1|^2 = \textbf{Trace}\,\{(\not{p}_2 + m)\,\Gamma\,(\not{p}_1 + m)\,\bar{\Gamma}\}. \tag{12.7}$$

in which (to repeat ourselves)

$$\bar{\Gamma} = \gamma^0\,\Gamma^\dagger\,\gamma^0. \tag{12.8}$$

13 Symbolic Methods vs. Explicit Matrix Methods

Loading the Dirac package is accomplished by the command **`load("dirac3.mac")`** (or just **`load(dirac3)`** if you have set up your init file as recommended in Maxima by Example, Chapter 1). The **`dirac3.mac`** file itself then loads five package files: **`simplifying-new.lisp`**, **`dgtrace3.mac`**, **`dgcon3.mac`**, **`dgeval3.mac`**, and **`dgmatrix3.mac`**.

Here is the information displayed when you first load **`dirac3.mac`**. (You can comment out the sections, in **`dirac3.mac`**, which display this information, to avoid having to deal with it each time you load the package.)

For interactive examples in this pdf document, we prefer to use the Xmaxima interface to Maxima, since production of a pdf via Xmaxima input and output (Tex to dvi to ps to pdf) is so much easier than if we were to show sections of the wxMaxima input and output (requiring a painful Tex editing process). We have restored the **`display2d:true`** setting for better matrix displays.

```
Maxima 5.36.1 http://maxima.sourceforge.net
using Lisp SBCL 1.2.7
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
                           maxima_userdir = c:/work5

(%i1) load(dirac3);
dirac3.mac
simplifying-new.lisp
dgtrace3.mac
dgcon3.mac
dgeval3.mac
dgmatrix3.mac
current symbol assignments

  scalarL =    [c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]
  indexL =    [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,la,mu,nu,rh,si,ta,al,be,ga,de,ep]
 massL =    [m,M]
  Nlast =    0
 reserved program capital letter name use:
 Chi, Con, D, Eps, EpsL,  G, G(1), G5, G5p
 Gam, Gm, Gtr, KD,  LI, Nlast, UI,  P,  S
 Sig, Sigb, SIG, UU, VP, VV
 I2, Z2, CZ2, I4, Z4, CZ4, RZ4, N1,N2,...
 reserved array names: gmet, eps4, eps4L

  invar_flag =    true
  stu_flag =    false
(%o1)                         c:/work5/dirac3.mac
```

Note from the last two printouts, that **`invar_flag = true`** and **`stu_flag = false`** (our defaults).

The fact that **`invar_flag = true`** means that the symbolic trace function **`tr`** will automatically make use of the list **`invarR`** (produced by the function **`set_invarR`**) which defines the replacements for **`D(pa,pb)`** in terms of the Mandelstam variables **`s`**, **`t`**, **`u`**. (After the package loads, **`invarR`** has the default value **`[]`**, an empty list.)

The fact that **`stu_flag = false`** means that the function **`noncov`** will not automatically replace the Mandelstam variables **`s, t, u`**, by **`s_th, t_th, and u_th`**.

The file **`dgfunctions3.txt`** has an alphabetical listing of the **`Dirac3`** package symbols with the name of the file (**`*.mac`**) in which the definition or Maxima code for that symbol will be found.

One should then go to that file (load into your text editor: I use **`notepad++`**) and use a name search for the symbol. Near the top and/or bottom of many of the Maxima function codes will be notes on syntax, purpose, functions which call it, and (for the most important functions) some explicit examples of use and output. Once the program is loaded into either Xmaxima or wxMaxima, you can experiment with calls to any of these functions, using the interactive mode.

Symbols for Lorentz indices need to be declared using the function **`Mindex`**, such as

```
Mindex(n11,n12);
```

However, a set of default Lorentz index symbols has been defined in **`dirac3.mac`** with the command:
`Mindex (n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,la,mu,nu,rh,si,ta,al,be,ga,de,ep)$`,
and the package list **`indexL`** will show you the current Lorentz index symbols recognised. There is a function **`unindex`** which allows you to remove recognised Lorentz index symbols from the list **`indexL`**. The Lorentz indices such as **`n1`** or **`mu`**, **`nu`**,... take values **`{0,1,2,3}`**.

`dirac3.mac` also assigns two default symbols to be recognised as symbols for **particle masses**, using the command: **`Mmass (m,M)$`** (for “make mass”). The package list **`massL`** will show you the current list of recognised mass symbols. For example, if you want to also use the symbols **`m1`** and **`m2`** for masses, you use the command **`Mmass (m1,m2);`** (we show this example below). The list **`massL`** is then updated to contain four recognised mass symbols.

The launch file **`dirac3.mac`** also defines a default set of scalars, using the command:
`Mscalar (c1,c2,c3,c4,c5,c6,c7,c8,c9,c10)$` (for “make scalars”). You can see the current list of recognised scalars in the package list **`scalarL`**. (There is a function **`unscalar`** which both removes symbols from the list **`scalarL`** and also removes the **`scalar`** property).

An **`atom`** (like **`p1`** or **`a`**) which is not a recognised Lorentz index symbol, not a recognised mass symbol, and not a recognised declared scalar, is treated as a 4-momentum (or Feynman slashed momentum, depending on the context) symbol.

Purely symbolic entities include the symbolic metric tensor **`Gm (mu,nu)`** corresponding to $\mathbf{g}^{\mu\,\nu}$ (or $\mathbf{g}_{\mu\,\nu}$ depending on the context) , the symbolic dot product of four vectors **`D(p1,p2)`** corresponding to $\mathbf{p_1}\cdot\mathbf{p_2}$, the symbolic four dimensional “antisymmetric” tensor **`Eps(mu,nu,rh,si)`** corresponding to $\epsilon^{\mu\,\nu\,\rho\,\sigma}$. (**`Eps`** however is not treated as antisymmetric until conversion (using **`noncov`**) to the numerical array **`eps4[n1,n2,n3,n4]`** occurs).

A new feature of version 3 is the symbol **`EpsL(mu,nu,rh,si)`** corresponding to $\epsilon_{\mu\,\nu\,\rho\,\sigma}$. (**`EpsL`** however is not treated as antisymmetric until conversion (using **`noncov`**) to the numerical array **`eps4L[n1,n2,n3,n4]`** occurs).

A symbolic contravariant four vector component is described with the notation **`UI(p,mu)`**, corresponding to $\mathbf{p}^{\mu}$, and is converted to the array component **`p[mu]`** by **`noncov`** (see below). The covariant component $\mathbf{p}_{\mu}$ is described by **`LI(p,mu)`**. With our metric convention, $\mathbf{p_0} = \mathbf{p^0}$.

Our metric convention (more details later) for lowering indices, etc., is

$$g_{\mu\nu} = g^{\mu\nu} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{13.1}$$

with Lorentz indices running over $0, 1, 2, 3$.

We use the same notation conventions as P/S (Peskin and Schroeder) and Maggiore, $\epsilon^{0123} = +1$, the electromagnetic units are Heaviside-Lorentz, and natural units are used, so $e^2 = 4\pi\alpha$.

The only exception is that we interpret the Feynman rules as an expression for the quantity $-i\,M$, following the conventions of Griffiths, and Halzen/Martin.

13.1 Symbolic Contractions of Symbolic Products of Dirac matrices

We use **`G (A,B,C,...)`** for the symbolic representation of a product of a combination of Dirac gamma matrices and slashed 4-momentum matrices.

The expression **`G(mu)`** represents γ^{μ}. The expression **`G(1)`** represents the 4×4 unit matrix $\mathbb{1}$. Don't confuse the product of two dirac matrices **`G(mu,nu)`** with the symbol for the metric tensor **`Gm(mu,nu)`** representing $g^{\mu\nu}$.

The expression **`G(mu,p,nu,q)`**, for example, represents the purely symbolic product of four matrices $\gamma^{\mu}\, \not{p}\, \gamma^{\nu}\, \not{q}$, in which the Feynman slashed matrix $\not{p} = p_{\alpha}\, \gamma^{\alpha}$ (using the summation convention).

`G5` is the symbol that stands for γ^5, itself a product of the fundamental Dirac matrices (we discuss our conventions in more detail later).

$$\gamma^5 = i\, \gamma^0\, \gamma^1\, \gamma^2\, \gamma^3, \tag{13.2}$$

although our program doesn't explicitly recognize this equation if we use **`(is(equal..))`**

```
(%i2) is(equal(G5,%i*G(0)*G(1)*G(2)*G(3)));
(%o2)                               unknown
```

and the symbols are just that for now (they don't evaluate to anything):

```
(%i3) G(1);
(%o3)                                G(1)
(%i4) G(mu);
(%o4)                                G(mu)
(%i5) G5;
(%o5)                                 G5
(%i6) G(mu,p,nu,q);
(%o6)                           G(mu, p, nu, q)
(%i7) Gm(mu,nu);
(%o7)                             Gm(mu, nu)
(%i8) Gm(0,0);
(%o8)                              Gm(0, 0)
```

However, the file **`dirac3.mac`** declares that the symbolic four-vector dot product **`D(p,q)`** is **symmetric**, as is the symbolic metric tensor **`Gm(mu,nu)`**, as an aid in automatic simplification of expressions.

```
(%i9) D(p,q);
(%o9)                               D(p, q)
(%i10) D(q,p);
(%o10)                              D(p, q)
(%i11) Gm(mu,nu);
(%o11)                            Gm(mu, nu)
(%i12) Gm(nu,mu);
(%o12)                            Gm(mu, nu)
```

Contraction of an expression with contravariant indices, two of which are equal, involves lowering one of the indices and summing the resulting expression over the four values of the repeated index.

"Product rules" or "contraction rules" for the Dirac matrices are (using the summation convention)

$$\gamma_\mu \gamma^\mu = 4 \cdot \mathbb{1}$$
$$\gamma_\mu \gamma^\nu \gamma^\mu = -2\gamma^\nu$$
$$\gamma_\mu \not{a} \gamma^\mu = -2\not{a}$$
$$\gamma_\mu \not{a} \gamma^\mu \not{b} = -2\not{a}\not{b}$$
$$\gamma_\mu \not{a} \gamma^\mu \gamma^\nu = -2\not{a}\gamma^\nu$$

Our contraction function **Con** has the syntax forms: **Con(expr)** or **Con(expr,index)**, or **Con(expr,index1,index2)**, etc. In the following five examples, we use the first form, letting the program find the sole contraction index on its own.

```
(%i13) Con (G(mu,mu));
(%o13)                           4 G(1)
(%i14) Con (G(mu,nu,mu));
(%o14)                          - 2 G(nu)
(%i15) Con (G(mu,p1,mu));
(%o15)                          - 2 G(p1)
(%i16) Con( G(mu,p1,mu,p2));
(%o16)                        - 2 G(p1, p2)
(%i17) Con( G(mu,p1,mu,nu));
(%o17)                        - 2 G(p1, nu)
```

Likewise, we have

$$\gamma_\mu \gamma^\nu \gamma^\lambda \gamma^\mu = 4\, g^{\nu\lambda}\, \mathbb{1}$$
$$\gamma_\mu \not{a} \not{b} \gamma^\mu = 4(a \cdot b)\, \mathbb{1}$$

Recall that the symbol for the metric tensor is **Gm(mu,nu)** representing $g^{\mu\nu}$.

```
(%i18) Con(G(mu,nu,la,mu));
(%o18)                       4 G(1) Gm(la, nu)
(%i19) grind(%)$
4*G(1)*Gm(la,nu)$
(%i20) Con(G(mu,a,b,mu));
(%o20)                        4 G(1) D(a, b)
(%i21) grind(%)$
4*G(1)*D(a,b)$
```

in which **D(p1,p2)** is the symbol we use for the four-vector dot product (scalar product) of the four-momentum vectors **p1** and **p2**.

```
(%i22) noncov(D(a,b));
(%o22)                 (- a  b ) - a  b  - a  b  + a  b
                            3  3     2  2    1  1    0  0
(%i23) grind(%)$
(-a[3]*b[3])-a[2]*b[2]-a[1]*b[1]+a[0]*b[0]$
```

The package function **noncov** assumes we have defined reference frame dependent four vector arrays to represent the 4-momenta (most easily done with the function **comp_def**, more about this later). Note that you **don't** need to tell Maxima that **a** and **b** are symbols for 4-vectors. The **explicit** expression for the invariant dot product of two 4-vectors can be immediately achieved by using **VP** ("vector product"):

```
(%i24) VP(a,b);
(%o24)                 (- a  b ) - a  b  - a  b  + a  b
                            3  3     2  2    1  1    0  0
(%i25) grind(%)$
(-a[3]*b[3])-a[2]*b[2]-a[1]*b[1]+a[0]*b[0]$
```

The superfluous parentheses on the first term of the output here is an artifact of a minor bug in the version of Maxima we are using, which doesn't interfere with the mathematical correctness of an expression.

Continuing with contraction of products of Dirac matrices, we have

$$\gamma_\mu \gamma^\nu \gamma^\lambda \gamma^\sigma \gamma^\mu = -2\gamma^\sigma \gamma^\lambda \gamma^\nu$$
$$\gamma_\mu \not{a} \not{b} \not{c} \gamma^\mu = -2 \not{c} \not{b} \not{a}$$

```
(%i26) Con( G(mu,nu,la,si,mu));
(%o26)                          - 2 G(si, la, nu)
(%i27) Con( G(mu,a,b,c,mu));
(%o27)                           - 2 G(c, b, a)
```

The Dirac gamma matrices should satisfy the anticommutation relations (note $\{A, B\} = AB + BA$):

$$\{\gamma^\mu, \gamma^\nu\} = 2\, g^{\mu\nu} \mathbb{1} \quad \{\gamma^5, \gamma^\mu\} = \mathbb{0}. \tag{13.3}$$

Our symbolic products do not exhibit the latter property:

```
(%i28) is(equal(G(G5,mu),-G(mu,G5)));
(%o28)                               unknown
```

but our explicit matrix products (see next section) do.

13.2 Explicit Dirac Matrices in the Helicity (Chiral) Representation and mcon

In addition to using symbolic Dirac matrix products, as shown above, we also use **explicit matrix** definitions (defined in **`dgmatrix3.mac`**) in a chosen representation.

The helicity (chiral) representation is used for explicit matrix methods; in this representation, the Dirac gamma matrices (in 2 × 2 block form) are

$$\gamma^0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \gamma^j = \begin{pmatrix} 0 & \sigma^j \\ -\sigma^j & 0 \end{pmatrix}, \quad \gamma^5 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{13.4}$$

in which σ^j for $j = 1, 2, 3$ are the 2 × 2 Pauli matrices

$$\sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{13.5}$$

The above form for $\boldsymbol{\gamma^5}$ is equivalent to the matrix product

$$\boldsymbol{\gamma^5 = i\,\gamma^0\,\gamma^1\,\gamma^2\,\gamma^3}. \tag{13.6}$$

The elements of the array **`Gam[mu]`**, defined for **`mu = 0,1,2,3,5`**, are **`4 x 4`** Maxima matrices:

```
(%i29) Gam[0];
                                [ 0  0  1  0 ]
                                [            ]
                                [ 0  0  0  1 ]
(%o29)                          [            ]
                                [ 1  0  0  0 ]
                                [            ]
                                [ 0  1  0  0 ]
(%i30) Gam[1];
                              [  0    0   0  1 ]
                              [                ]
                              [  0    0   1  0 ]
(%o30)                        [                ]
                              [  0   - 1  0  0 ]
                              [                ]
                              [ - 1   0   0  0 ]
```

```
(%i31) Gam[2];
                            [  0    0   0   - %i ]
                            [                    ]
                            [  0    0   %i   0   ]
(%o31)                      [                    ]
                            [  0    %i  0    0   ]
                            [                    ]
                            [ - %i  0   0    0   ]
(%i32) Gam[3];
                              [  0   0  1   0  ]
                              [                ]
                              [  0   0  0  - 1 ]
(%o32)                        [                ]
                              [ - 1  0  0   0  ]
                              [                ]
                              [  0   1  0   0  ]
(%i33) Gam[5];
                              [ - 1   0   0  0 ]
                              [                ]
                              [  0   - 1  0  0 ]
(%o33)                        [                ]
                              [  0    0   1  0 ]
                              [                ]
                              [  0    0   0  1 ]
(%i34) Gam[4];
(%o34)                             Gam
                                      4
(%i35) grind(%)$
Gam[4]$
```

The explicit **Pauli Matrices** are available as the matrices `Sig[i]`, defined in `dgmatrix3.mac`.

```
(%i36) Sig[1];
                                     [ 0  1 ]
(%o36)                               [      ]
                                     [ 1  0 ]
(%i37) grind(%)$
matrix([0,1],[1,0])$
(%i38) Sig[2];
                                   [ 0   - %i ]
(%o38)                             [          ]
                                   [ %i   0   ]
(%i39) Sig[3];
                                    [ 1   0  ]
(%o39)                              [        ]
                                    [ 0  - 1 ]
(%i40) Sig[1] . Sig[1];
                                     [ 1  0 ]
(%o40)                               [      ]
                                     [ 0  1 ]
(%i41) is(equal(Sig[1] . Sig[2], %i*Sig[3]));
(%o41)                                 true
```

The gamma matrices satisfy the anticommutation relations

$$\{\gamma^\mu, \gamma^\nu\} = 2\, g^{\mu\nu} \quad \{\gamma^5, \gamma^\mu\} = 0, \tag{13.7}$$

in which the unit 4×4 matrix is understood to be multiplying the right hand side of both equations. We can check this using our explicit gamma matrices `Gam[mu]`, and using our explicit matrix anticommutator function `acomm(A,B)`, and our explicit 4×4 zero matrix `Z4` and our explicit 4×4 unit matrix `I4`.

```
(%i2) acomm(A,B);
(%o2)                        B . A + A . B
(%i3) test(i,j) := acomm(Gam[i],Gam[j]) - 2*gmet[i,j]*I4$
```

```
(%i4) test(0,0);
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%o4)                           [            ]
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%i5) Z4;
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%o5)                           [            ]
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%i6) for i:0 thru 3 do
          for j:i thru 3 do
             print(i,j," ",is(equal(test(i,j),Z4)))$
0 0   true
0 1   true
0 2   true
0 3   true
1 1   true
1 2   true
1 3   true
2 2   true
2 3   true
3 3   true
(%i7) I4;
                                [ 1  0  0  0 ]
                                [            ]
                                [ 0  1  0  0 ]
(%o7)                           [            ]
                                [ 0  0  1  0 ]
                                [            ]
                                [ 0  0  0  1 ]
```

The second anticommutator relation implies

$$\gamma^5\,\gamma^\mu = -\gamma^\mu\,\gamma^5, \quad \text{for} \quad \mu = 0, 1, 2, 3, \tag{13.8}$$

and we again use **acomm** as a check of this property

```
(%i8) for mu:0 thru 3 do
           print (mu,"  ",is(equal(acomm (Gam[5], Gam[mu]),Z4)))$
0    true
1    true
2    true
3    true
```

In the file **dgmatrix3.mac**, you can also find the definitions of the explicit Dirac particle spinor (4-component column vector) **UU**, and the explicit Dirac anti-particle spinor (also a 4-component column vector) **VV**, (both defined using Maxima's **matrix** function) and having a concrete form which is generated from the specified 4-momentum and helicity of the particle or anti-particle. Many examples of explicit matrix methods (based on **dgmatrix3.mac** code) will be found in the wxMaxima worksheets.

The file **dirac3.mac** includes some definitions of useful utility functions, and also defines the explicit numerical metric tensor array **gmet[mu,nu]**

```
(%i2) arrayinfo (gmet);
(%o2)                       [complete, 2, [3, 3]]
(%i3) listarray (gmet);
(%o3)       [1, 0, 0, 0, 0, - 1, 0, 0, 0, 0, - 1, 0, 0, 0, 0, - 1]
(%i4) gmet[0,0];
(%o4)                                  1
```

and the explicit four dimensional completely antisymmetric tensor array components for **eps4[mu,nu,rh,si]** and **eps4L[mu,nu,rh,si]**. (The Lorentz indices **mu**, **nu**,... take values **0,1,2,3**.)

13.2.1 Use of noncov

The function **noncov** converts **Eps(mu,nu,rh,si)** to **eps4[mu,nu,rh,si]** (with the convention that $\epsilon^{0\,1\,2\,3} = 1$), and converts **EpsL(mu,nu,rh,si)** to **eps4L[mu,nu,rh,si]** (with the convention that $\epsilon_{0\,1\,2\,3} = -1$).

```
(%i5) noncov(Eps(mu,nu,la,si));
(%o5)                         eps4
                                  la, mu, nu, si
(%i6) grind(%)$
eps4[la,mu,nu,si]$
(%i7) noncov(EpsL(mu,nu,la,si));
(%o7)                         eps4L
                                   la, mu, nu, si
(%i8) eps4[0,1,2,3];
(%o8)                                1
(%i9) eps4L[0,1,2,3];
(%o9)                              - 1
```

Likewise, **noncov** converts **Gm(mu,nu)** to **gmet[mu,nu]**, converts **UI(p,mu)** to **p[mu]**, and converts **LI(p,mu)** to **sum (gmet[mu,nu]*p[nu],nu,0,3)**. For example **LI (p,0) --> p[0], LI (p,1) --> -p[1]**.

```
(%i10) noncov(Gm(mu,nu));
(%o10)                            gmet
                                      mu, nu
(%i11) grind(%)$
gmet[mu,nu]$
(%i12) noncov(UI(p,mu));
(%o12)                                p
                                       mu
(%i13) grind(%)$
p[mu]$
```

Notice that with **display2d = true**, the position of the index **mu** is technically incorrect. We have not embedded our program into Maxima's tensor packages, which would have provided the correct placement of contravariant indices. We must simply remember that **p[mu]** is a contravariant component of **p**.

The conversion of a covariant component of the 4-vector **p**, using **noncov**, is:

```
(%i14) noncov(LI(p,mu));
(%o14)     p  gmet      + p  gmet      + p  gmet      + p  gmet
            3     3, mu    2     2, mu    1     1, mu    0     0, mu
(%i15) grind(%)$
p[3]*gmet[3,mu]+p[2]*gmet[2,mu]+p[1]*gmet[1,mu]+p[0]*gmet[0,mu]$
(%i17) noncov( LI(p,0));
(%o17)                                p
                                       0
(%i18) noncov( LI(p,1));
(%o18)                              - p
                                       1
```

The function **noncov** also converts **D(p,q)** into **sum (sum (gmet[mu,nu]*p[mu]*q[nu],nu,0,3),mu,0,3)**, which reduces to the single sum **sum (gmet[mu,mu]*p[mu]*q[mu],mu,0,3)**, which becomes the explicit sum **p[0]*q[0] - p[1]*q[1] - p[2]*q[2] - p[3]*q[3]**.

```
(%i19) noncov(D(p,q));
(%o19)                 (- p  q ) - p  q  - p  q  + p  q
                            3  3      2  2     1  1     0  0
(%i20) grind(%)$
(-p[3]*q[3])-p[2]*q[2]-p[1]*q[1]+p[0]*q[0]$
```

The function **noncov** does **nothing** to **G(...)**.

```
(%i21) noncov(G(mu,nu,al,be));
(%o21)                        G(mu, nu, al, be)
```

13.2.2 The Use of comp_def

The reference frame dependent function **comp_def (p(Ep,px,py,pz))** creates an **array** with the name **p** whose elements can be viewed using **listarray (p)** and whose elements can be accessed by using **p[0]** to get the value **Ep**, using **p[1]** to get the value **px**, etc.

```
(%i22) comp_def(p1(E,p*sin(th),0,p*cos(th)));
(%o22)                              done
(%i23) listarray(p1);
(%o23)                  [E, p sin(th), 0, p cos(th)]
(%i24) p1[0];
(%o24)                                E
```

The symbolic trace of a product of gamma matrices is usually carried out using the **tr** function. Thus the syntax **tr(mu,nu,rh,si)** carries out the trace of a product of four Dirac gamma matrices.

13.2.3 The Use of mat_trace and m_tr

The same calculation can be carried out using explicit Maxima matrices using the syntax:
mat_trace (Gam[mu] . Gam[nu] . Gam[rh] . Gam[si]);
(although one must supply definite values for the Lorentz index symbols **mu, nu, rh, si** to get a simple result from the explicit matrix method).

A simpler-to-use function for the trace of explicit matrix expressions is **m_tr**. For example, **m_tr (mu,p,q,nu)** uses the exact same syntax as **tr** but internally translates this into
mat_trace (Gam[mu] . sL(p) . sL(q) . Gam[nu]).

13.2.4 The Use of mcon for Contraction of Explicit Matrix Expressions

To obtain useful frame-dependent expressions from the explicit matrix methods, one should first use **comp_def** to define the frame dependent components of relevant 4-momenta, then use either the **m_tr** or **mat_trace** syntax for the trace, and then use **mcon** (not **Con**) for contraction. The correct syntax for **mcon** ("m" for matrix) **always** includes the contraction indices to be summed over, as in **mcon(expr, mu, nu,...)**, otherwise no contraction will occur.

13.2.5 Symbolic Traces of Expressions which include γ^5

S A symbolic trace involving a gamma5 matrix, such as **tr (G5,p,nu,rh,si)** generates expressions proportional to (for example) **LI(p,N1)*Eps (N1,nu,rh,si)**, in which **N1** is an unsummed "dummy index" created by the code. (The sum over that dummy index is done by the later use of **noncov**.) The global symbol **Nlast** will always tell you what the last dummy symbol created is.

```
(%i25) mytr : tr (G5,p,nu,rh,si);
(%o25)              - 4 %i LI(p, N1) Eps(N1, nu, rh, si)
(%i26) Nlast;
(%o26)                                1
```

The symbolic **tr** code creates these dummy indices in the order **N1,N2,...** and establishes **indexp** and **dummyp** attributes, so that, for example, **indexp (N1) --> true** and **dummyp (N1) --> true**.

```
(%i27) indexp (N1);
(%o27)                              true
(%i28) dummyp (N1);
(%o28)                              true
```

13.2.6 Using Gtr to Convert G(a,b,c) into tr(a,b,c)

An expression or sum of terms involving **`G(a,b,c,..)'s`** can be converted into an expression (or sum of such) in which each **`G(...)`** is replace by **`tr (...)`** by using the symbolic function **`Gtr`**.
Thus **`Gtr (G(a,b,c,d)) -->  tr (a,b,c,d)`**.

The symbolic trace function **`tr`** **automatically contracts** on repeated Lorentz indices, such as in **`tr (mu,p,q,mu)`**, before using the actual trace algorithms. The package function **`Gtr`** performs a symbolic trace on an expression containing a factor **`G(a,b,...)`** by replacing **`G`** by **`tr`** (see below).

Symbolic contraction (without the trace) on repeated indices in a symbolic product of gamma matrices can be done using **`Con`**, which recognises a symbolic case and then calls our package function **`scon`**. Examples in which **`scon`** ("s" for "symbolic") is called by **`Con`** are: **`Con (G (mu,p,q,mu))`** (equivalent to **`scon(G(mu,p,q,mu))`**)
and
`Con (G (mu,G5,mu))` (equivalent to **`scon(G(mu,G5,mu))`**).

```
(%i2) mytr : tr (mu, p, q, mu);
(%o2)                          16 D(p, q)
(%i3) Con (G (mu,p,q,mu));
(%o3)                        4 G(1) D(p, q)
(%i4) Con (G (mu,p,q,mu), mu);
(%o4)                        4 G(1) D(p, q)
(%i5) Gtr (%);
(%o5)                          16 D(p, q)
(%i6) mytr : tr (mu,G5,mu);
(%o6)                               0
(%i7) Con (G(mu,G5,mu));
(%o7)                           - 4 G(G5)
(%i8) Gtr(%);
(%o8)                               0
(%i9) fundef(Gtr);
(%o9)       Gtr(expr) := (subst('tr, G, expr), ev(%%), expand(%%))
```

13.2.7 Comparison of Con, scon and mcon

`Con` (and **`scon`**) automatically contract on all repeated Lorentz indices unless the user explicitly supplies the desired contraction indices.

For example, **`Con (G(mu,nu,p,q,nu,mu),mu,nu)`** (or the same with **`Con --> scon`**) will carry out symbolic contraction on both **`mu`** and **`nu`** and produce the same answer as **`Con (G(mu,nu,p,q,nu,mu))`**.
But **`Con (G(mu,nu,p,q,nu,mu),nu)`** will only contract on **`nu`**.

When using *explicit* Dirac gamma matrices and Maxima **matrix** methods, it is preferable to use
`mcon (expr, index1, index2, ...)` for contraction (i.e., you must always supply the desired contraction indices), although calling **`Con`** instead should in principle recognise the matrix character of the expression and call **`mcon`**. You must always supply the desired contraction indices to **`Con`** (which will recognise the explicit matrix case and call **`mcon`** with the desired contraction indices). For example,
`Con (mat_trace (Gam[mu] . sL(p) . sL(q) . Gam[mu]), mu)` or the same with **`Con --> mcon`** .

```
(%i10) mcon (mat_trace (Gam[mu] . sL(p) . sL(q) . Gam[mu]), mu);
(%o10)           (- 16 p  q ) - 16 p  q  - 16 p  q  + 16 p  q
                        3  3        2  2       1  1       0  0
(%i11) Con (mat_trace (Gam[mu] . sL(p) . sL(q) . Gam[mu]), mu);
(%o11)           (- 16 p  q ) - 16 p  q  - 16 p  q  + 16 p  q
                        3  3        2  2       1  1       0  0
```

A frequent case met with in the examples worked out is the symbolic contraction of a product of symbolic traces each produced by **tr**. The most efficient symbolic method is to first convert each of the symbolic trace expressions into expressions depending on **eps4**, **gmet**, and 4-momentum components like **p[1]** by using **noncov before** the contraction.

13.2.8 Use of nc_tr

This is such a common approach, that the package function **nc_tr** is defined to immediately apply **noncov** to each trace result returned from the function **tr1** (controlled by **simp_tr1** via **simplifying-new.lisp**) called by **TR1**, which is called by **tr**.

One then needs the contraction of products of such expressions, which should be carried out using **mcon**.

The **explicit** metric tensor, represented by the array **gmet[mu,nu]**, has, of course, definite values for the components. Here we use both **mcon** and **Con** for contaction of the metric tensor in the explicit array form **gmet[mu,nu]**.

```
(%i2) listarray (gmet);
(%o2)       [1, 0, 0, 0, 0, - 1, 0, 0, 0, 0, - 1, 0, 0, 0, 0, - 1]
(%i3) gmet[0,0];
(%o3)                                  1
(%i4) mcon( gmet[mu,mu], mu);
(%o4)                                  4
(%i5) Con ( gmet[mu,mu], mu);
(%o5)                                  4
```

We can also use the Maxima function **sum**. Note that

$$g^{\mu}_{\ \mu} = \sum_{\nu} g_{\mu\nu} g^{\mu\nu}, \quad \left(\cancel{\sum_{\mu}}\right), \tag{13.9}$$

so

$$\sum_{\mu} g^{\mu}_{\ \mu} = \sum_{\mu\nu} g_{\mu\nu} g^{\mu\nu} = 4, \tag{13.10}$$

which can be calculated in Maxima as

```
(%i6) sum ( sum (gmet[i,j]*gmet[i,j],j,0,3),i,0,3);
(%o6)                                  4
```

13.3 Some Symbolic Contraction Examples

The symbolic contraction functions **scon** or **Con** call the function **scon11(expr, index)**, which can be called directly. The function **scon11(expr,index)** assumes **expr** is a single term, and performs symbolic contraction with respect to the supplied index symbol.

As an example of symbolic contraction of the expression $-p^{\mu} q^{\mu} r^{\nu}/3$ with respect to the Lorentz index μ,

$$-\frac{1}{3} p^{\mu} q^{\mu} r^{\nu} \rightarrow -\frac{1}{3} r^{\nu} \sum_{\mu=0}^{\mu=3} p^{\mu} q_{\mu}, \tag{13.11}$$

we use first **scon11** with the required contraction index supplied, and then **scon**, with and without the contraction index, and then **Con**, with and without the contraction index.

We expect the answer

$$-\frac{1}{3} r^\nu p \cdot q. \tag{13.12}$$

13.3.1 UI(p,mu) and p^μ

The symbol **UI(p,mu)** ("upper index") is used to stand for p^μ, and the symbol **D(p,q)** is used to stand for the four-vector product $p \cdot q$.

If the contraction index is not explicitly supplied, when using either **Con** or **scon**, the program assumes there is one repeated Lorentz index in each term of the supplied expression, and automatically contracts with respect to the single repeated Lorentz index found in each term.

```
(%i2) scon11(-UI(p,mu)*UI(q,mu)*UI(r,nu)/3,mu);
                                D(p, q) UI(r, nu)
(%o2)                         - -----------------
                                        3
(%i3) scon(-UI(p,mu)*UI(q,mu)*UI(r,nu)/3,mu);
                                D(p, q) UI(r, nu)
(%o3)                         - -----------------
                                        3
(%i4) scon(-UI(p,mu)*UI(q,mu)*UI(r,nu)/3);
                                D(p, q) UI(r, nu)
(%o4)                         - -----------------
                                        3
(%i5) Con(-UI(p,mu)*UI(q,mu)*UI(r,nu)/3,mu);
                                D(p, q) UI(r, nu)
(%o5)                         - -----------------
                                        3
(%i6) Con(-UI(p,mu)*UI(q,mu)*UI(r,nu)/3);
                                D(p, q) UI(r, nu)
(%o6)                         - -----------------
                                        3
```

As an example of using **scon** or **Con** for an expression containing more than one term, we consider the contraction

$$p^\nu q^\nu r^\mu + p^\mu q^\nu r^\nu \rightarrow r^\mu p \cdot q + p^\mu q \cdot r. \tag{13.13}$$

```
(%i7) scon(UI(p,nu)*UI(q,nu)*UI(r,mu) + UI(p,mu)*UI(q,nu)*UI(r,nu));
(%o7)                D(p, q) UI(r, mu) + UI(p, mu) D(q, r)
(%i8) Con(UI(p,nu)*UI(q,nu)*UI(r,mu) + UI(p,mu)*UI(q,nu)*UI(r,nu));
(%o8)                D(p, q) UI(r, mu) + UI(p, mu) D(q, r)
```

As an example of using **scon** or **Con** for an expression containing more than one contraction index, we consider the contraction

$$p^\mu q^\mu r^\nu s^\nu \rightarrow p \cdot q \; r \cdot s. \tag{13.14}$$

```
(%i9) scon(UI(p,mu)*UI(q,mu)*UI(r,nu)*UI(s,nu),mu,nu);
(%o9)                           D(p, q) D(r, s)
(%i10) scon(UI(p,mu)*UI(q,mu)*UI(r,nu)*UI(s,nu));
(%o10)                          D(p, q) D(r, s)
(%i11) Con(UI(p,mu)*UI(q,mu)*UI(r,nu)*UI(s,nu),mu,nu);
(%o11)                          D(p, q) D(r, s)
(%i12) Con(UI(p,mu)*UI(q,mu)*UI(r,nu)*UI(s,nu));
(%o12)                          D(p, q) D(r, s)
```

13.3.2 LI(p,mu) and p_μ

The *covariant* momentum four-vector p_μ is represented by the symbol **LI(p,mu)** ("lower index"). Some contraction examples are:

$$-\frac{1}{3} p_\mu q_\mu r^\nu \rightarrow -\frac{1}{3} r^\nu \sum_{\mu=0}^{\mu=3} p^\mu q_\mu . \tag{13.15}$$

```
(%i13) Con(-LI(p,mu)*LI(q,mu)*UI(r,nu)/3);
                                D(p, q) UI(r, nu)
(%o13)                        - -----------------
                                        3
(%i14) Con(LI(p,mu)*LI(q,mu));
(%o14)                              D(p, q)
```

$$p_\mu q_\mu r_\nu s_\nu \rightarrow p \cdot q \; r \cdot s. \tag{13.16}$$

```
(%i15) Con(LI(p,mu)*LI(q,mu)*LI(r,nu)*LI(s,nu));
(%o15)                          D(p, q) D(r, s)
```

and

$$p_\nu q_\nu r^\mu + p^\mu q_\nu r_\nu \rightarrow r^\mu p \cdot q + p^\mu q \cdot r. \tag{13.17}$$

```
(%i16) Con(LI(p,nu)*LI(q,nu)*UI(r,mu) + UI(p,mu)*LI(q,nu)*LI(r,nu));
(%o16)               D(p, q) UI(r, mu) + UI(p, mu) D(q, r)
```

An example of contraction of our symbolic metric tensor can be carried out in our package using either **Con** or **scon** ("s" for symbolic), either specifying the contraction index, or not:

```
(%i17) Con (Gm(mu,mu));
(%o17)                             4
(%i18) Con (Gm(mu,mu),mu);
(%o18)                             4
(%i19) scon (Gm(mu,mu));
(%o19)                             4
(%i20) scon (Gm(mu,mu),mu);
(%o20)                             4
```

Our symbolic quantity **D(pa,pb)** stands for the 4-momentum invariant product $p_a \cdot p_b$. Our **UI(p,mu)** stands for p^μ. Likewise, **LI(p,mu)** stands for p_μ

```
(%i21) Con(UI(p1,mu)*UI(p1,mu));
(%o21)                              D(p1, p1)
(%i22) Con(UI(p1,mu)*UI(p2,mu));
(%o22)                              D(p1, p2)
(%i23) Con(UI(p1,mu)*Gm(mu,nu));
(%o23)                             UI(p1, nu)
(%i24) Con(LI(p1,mu)*Gm(mu,nu));
(%o24)                             LI(p1, nu)
(%i25) Con(LI(p1,mu)*Gm(mu,nu),mu);
(%o25)                             LI(p1, nu)
```

Contraction of **symbolic** expressions does not require specifying the contraction index; the repeated contraction index **mu** in the expression is understood to mean summing over that index.

14 Survey of Some Dirac Package Functions

14.1 Unit Four Tensor δ^{μ}_{ν}, KD, kron_delta, Con

The unit four tensor δ^{μ}_{ν} satisfies the condition

$$\delta^{\mu}_{\nu}\, A^{\nu} = A^{\mu}, \tag{14.1}$$

for any four vector A^{ν} and has the definition

$$\delta^{\mu}_{\nu} = g^{\mu}_{\ \nu} = \begin{cases} 1 & \text{if } \mu = \nu \\ 0 & \text{otherwise.} \end{cases} \tag{14.2}$$

Its trace is

$$\delta^{\mu}_{\mu} = 4. \tag{14.3}$$

The symbol **`KD(n1,n2)`** is used to execute contractions which involve the unit tensor.

```
(%i2) Con(UI(p,n1)*KD(n1,n2));
(%o2)                            UI(p, n2)
(%i3) Con(UI(p,n1)*KD(n1,n2),n1);
(%o3)                            UI(p, n2)
(%i4) Con(KD(n1,n5)*Eps(n1,n2,n3,n4));
(%o4)                       Eps(n5, n2, n3, n4)
(%i5) Con(KD(n1,n5)*Eps(n1,n2,n3,n4),n1);
(%o5)                       Eps(n5, n2, n3, n4)
```

The symbol **`KD`** has no numerical values or properties, and can be converted into the core Maxima constant **`kron_delta`** using our package function **`convertKD`**.

```
(%i6) kron_delta(1,0);
(%o6)                                0
(%i7) kron_delta(1,1);
(%o7)                                1
(%i8) KD(1,0);
(%o8)                            KD(1, 0)
(%i9) convertKD(%);
(%o9)                                0
(%i10) KD(1,1);
(%o10)                           KD(1, 1)
(%i11) convertKD(%);
(%o11)                               1
(%i12) KD(n,m);
(%o12)                           KD(n, m)
(%i13) convertKD(%);
(%o13)                       kron_delta(m, n)
```

14.2 $\not{p}_1 = \gamma_{\mu}\, p^{\mu}_1$, sL(p1)

Using our explicit gamma matrices, we can construct an explicit 4×4 matrix representing $\not{p}_1 \equiv \gamma_{\mu}\, p^{\mu}_1$ using our package function **`sL(p1)`** (**`sL`** for “slash”). Here, **`p1`** stands for a four-vector, and **`sL`** assumes that the components of that four-vector have been defined either as elements of a list or elements of a hashed array, so that p^0_1 is given by **`p1[0]`**, and p^1_1 is given by **`p1[1]`**, etc.

Since we have not defined these quantities, they are just written as symbols here:

```
(%i9) sL(p1);
        [      0             0           p1  - p1       %i p1  - p1  ]
        [                                  0     3           2     1 ]
        [                                                            ]
        [      0             0        (- %i p1 ) - p1    p1  + p1    ]
        [                                     2      1     3     0   ]
(%o9)   [                                                            ]
        [  p1  + p1     p1  - %i p1          0               0       ]
        [    3     0      1        2                                 ]
        [                                                            ]
        [ %i p1  + p1    p1  - p1            0               0       ]
        [      2     1     0     3                                   ]
(%i10) grind(%)$
matrix([0,0,p1[0]-p1[3],%i*p1[2]-p1[1]],[0,0,(-%i*p1[2])-p1[1],p1[3]+p1[0]],
       [p1[3]+p1[0],p1[1]-%i*p1[2],0,0],[%i*p1[2]+p1[1],p1[0]-p1[3],0,0])$
```

We can use our function **comp_def** ("components-definition") to create one or more hashed (zero-based) arrays containing the components of the four-momentum p_1^μ, and other four vectors. The syntax

```
   comp_def ( p1(E1, p1x,p1y,p1z));
```

is used, or for more than one 4-vector, for example two:

```
   comp_def ( p1(E1, p1x,p1y,p1z),  p2(E2, p2x,p2y,p2z) );
```

You can have an arbitrary number of arguments to **comp_def**.

Here we assume the relativistic energy of a free particle with 4-momentum p_1^μ is **E**, the magnitude of the 3-momentum of the particle is represented by the symbol **p**, and the 3-momentum vector $\vec{p}_1$ lies in the z-x plane, making an angle θ (represented by the symbol **th**) with the z-axis. The symbol **I4** is a predefined 4×4 unit matrix.

```
(%i11) comp_def (p1 (E,p*sin(th),0,p*cos(th)))$
(%i12) listarray(p1);
(%o12)                 [E, p sin(th), 0, p cos(th)]
(%i13) p1[0];
(%o13)                                E
```

We can now show the result of the invariant 4-vector dot product functions, symbolic **D** and explicit **VP**, when the args are 4-vectors with explicitly defined components:

```
(%i14) D(p1,p1);
(%o14)                            D(p1, p1)
(%i15) noncov(%);
                                   2    2
(%o15)                            E  - p
(%i16) VP(p1,p1);
                                   2    2
(%o16)                            E  - p
(%i17) sL(p1);
        [       0              0        E - p cos(th)   - p sin(th)  ]
        [                                                            ]
        [       0              0         - p sin(th)   E + p cos(th) ]
(%o17)  [                                                            ]
        [ E + p cos(th)    p sin(th)          0              0       ]
        [                                                            ]
        [   p sin(th)    E - p cos(th)        0              0       ]
(%i18) sL(p1) + m*I4;
        [       m              0        E - p cos(th)   - p sin(th)  ]
        [                                                            ]
        [       0              m         - p sin(th)   E + p cos(th) ]
(%o18)  [                                                            ]
        [ E + p cos(th)    p sin(th)          m              0       ]
        [                                                            ]
        [   p sin(th)    E - p cos(th)        0              m       ]
```

14.3 Dirac Spinors and Helicity Operators

14.3.1 Lepton Spinors, $u(p,\sigma)$, UU

If we let p temporarily stand for the particle momentum four-vector p^μ, the lepton Dirac spinor is the four element column matrix $u(p,\sigma)$ where $\sigma = +1$ picks out positive helicity $h = +1/2$ and $\sigma = -1$ picks out negative helicity $h = -1/2$. We also have the basic property

$$\not p\, u(p,\sigma) = m\, u(p,\sigma). \tag{14.4}$$

The corresponding barred particle spinor is a four element row matrix

$$\bar u(p,\sigma) = u(p,\sigma)^\dagger\, \gamma^0 \tag{14.5}$$

in which the dagger symbol † indicates the hermitian conjugate operation, and the lepton Dirac spinor normalization is

$$\bar u(p,\sigma)\, u(p,\sigma') = 2\, m\, \delta_{\sigma\,\sigma'}, \quad \sigma,\, \sigma' = \pm 1, \tag{14.6}$$

and we also have the "projection matrix" relation

$$\sum_{\sigma=\pm 1} u(p,\sigma)\, \bar u(p,\sigma) = \not p + m, \tag{14.7}$$

in which the 4×4 identity matrix is understood to be multiplying the scalar mass symbol m, and $\not p = p_\mu\, \gamma^\mu$ with use of the summation convention.

We generate an **explicit** Dirac "spinor", a four element column **matrix**, denoted symbolically by $u(p_1,\sigma)$, corresponding to a lepton with **four momentum** $p_1^\mu = (E, \vec p)$, using the function `UU(E,p,theta,phi,sv)` which descibes a spin `1/2` particle with relativistic energy `E`, 3-momentum magnitude `p`, whose 3-vector momentum direction is described by spherical polar angles `(theta,phi)`, where `0 <= theta <= %pi` and `0 <= phi <= 2*%pi`, and whose helicity quantum number `sv` is `1` for helicity `+1/2` and `sv` is `-1` for helicity `-1/2`.

```
(%i2) UU(E,p,th,phi,1);
                       [         th                ]
                       [     cos(--) sqrt(E - p)   ]
                       [         2                 ]
                       [                           ]
                       [   %i phi     th           ]
                       [ %e       sin(--) sqrt(E - p) ]
                       [              2            ]
(%o2)                  [                           ]
                       [         th                ]
                       [     cos(--) sqrt(E + p)   ]
                       [         2                 ]
                       [                           ]
                       [   %i phi     th           ]
                       [ %e       sin(--) sqrt(E + p) ]
                       [              2            ]
(%i3) UU(E,p,th,phi,-1);
                     [     - %i phi     th                ]
                     [ - %e         sin(--) sqrt(E + p)   ]
                     [                  2                 ]
                     [                                    ]
                     [           th                       ]
                     [       cos(--) sqrt(E + p)          ]
                     [           2                        ]
(%o3)                [                                    ]
                     [     - %i phi     th                ]
                     [ - %e         sin(--) sqrt(E - p)   ]
                     [                  2                 ]
                     [                                    ]
                     [           th                       ]
                     [       cos(--) sqrt(E - p)          ]
                     [           2                        ]
```

Thus for arbitrary angles

$$u(E,p,\theta,\phi,\sigma=1) = \begin{bmatrix} \sqrt{E-p}\,\cos(\theta/2) \\ \sqrt{E-p}\,\sin(\theta/2)\,e^{i\phi} \\ \sqrt{E+p}\,\cos(\theta/2) \\ \sqrt{E+p}\,\sin(\theta/2)\,e^{i\phi} \end{bmatrix} \tag{14.8}$$

and

$$u(E,p,\theta,\phi,\sigma=-1) = \begin{bmatrix} -\sqrt{E+p}\,\sin(\theta/2)\,e^{-i\phi} \\ \sqrt{E+p}\,\cos(\theta/2) \\ -\sqrt{E-p}\,\sin(\theta/2)\,e^{-i\phi} \\ \sqrt{E-p}\,\cos(\theta/2) \end{bmatrix} \tag{14.9}$$

We can use our explicit spinors to show the property

$$\not{p}\,u(p,\sigma) = m\,u(p,\sigma) \tag{14.10}$$

or, expressed differently,

$$(\not{p} - m\,\mathbb{1})\,u(p,\sigma) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14.11}$$

To show an explicit example of this property, using our explicit spinors, we make use of our package function **`trigcrunch(expr,angle)`** as well as our package function **`psq_to_Esq(expr,p,E,m)`**, first for the row 1 element, and then for the matrix column all at once. The latter method makes use of the core Maxima function **`matrixmap`**, as well as the anonymous **`lambda`** function.

```
(%i4) comp_def(p1(E,p*sin(th)*cos(phi),p*sin(th)*sin(phi),p*cos(th)))$
(%i5) listarray (p1);
(%o5)       [E, p cos(phi) sin(th), p sin(phi) sin(th), p cos(th)]
(%i6) up1_plus : UU(E,p,th,phi,1)$
(%i7) mprod : (sL(p1) - m*I4) . up1_plus;
                %i phi     th
(%o7) matrix([%e       sin(--) (%i p sin(phi) sin(th) - p cos(phi) sin(th))
                           2
                     th                     th
 sqrt(E + p) - m cos(--) sqrt(E - p) + cos(--) sqrt(E + p) (E - p cos(th))],
                     2                      2
    th
[cos(--) ((- %i p sin(phi) sin(th)) - p cos(phi) sin(th)) sqrt(E + p)
    2
      %i phi     th                   %i phi     th
 - m %e       sin(--) sqrt(E - p) + %e       sin(--) sqrt(E + p)
                 2                              2
                                 th
 (E + p cos(th))], [(- m cos(--) sqrt(E + p))
                                 2
     %i phi     th
 + %e       sin(--) (p cos(phi) sin(th) - %i p sin(phi) sin(th)) sqrt(E - p)
               2
       th
 + cos(--) sqrt(E - p) (E + p cos(th))],
       2
       %i phi     th                      th
[(- m %e       sin(--) sqrt(E + p)) + cos(--)
                  2                       2
 (%i p sin(phi) sin(th) + p cos(phi) sin(th)) sqrt(E - p)
     %i phi     th
 + %e       sin(--) sqrt(E - p) (E - p cos(th))])
               2
```

```
(%i8) grind(mprod)$
matrix([%e^(%i*phi)*sin(th/2)*(%i*p*sin(phi)*sin(th)-p*cos(phi)*sin(th))
                   *sqrt(E+p)
         -m*cos(th/2)*sqrt(E-p)+cos(th/2)*sqrt(E+p)*(E-p*cos(th))],
       [cos(th/2)*((-%i*p*sin(phi)*sin(th))-p*cos(phi)*sin(th))*sqrt(E+p)
         -m*%e^(%i*phi)*sin(th/2)*sqrt(E-p)
         +%e^(%i*phi)*sin(th/2)*sqrt(E+p)*(E+p*cos(th))],
       [(-m*cos(th/2)*sqrt(E+p))+%e^(%i*phi)*sin(th/2)
                                            *(p*cos(phi)*sin(th)
                                             -%i*p*sin(phi)*sin(th))*sqrt(E-p)
                                +cos(th/2)*sqrt(E-p)*(E+p*cos(th))],
       [(-m*%e^(%i*phi)*sin(th/2)*sqrt(E+p))
         +cos(th/2)*(%i*p*sin(phi)*sin(th)+p*cos(phi)*sin(th))*sqrt(E-p)
         +%e^(%i*phi)*sin(th/2)*sqrt(E-p)*(E-p*cos(th))])$
```

This is a four element column matrix. (Note that Xmaxima displays the output **`(%o7)`** using **`matrix ([row one],[row two],...)`** despite **`display2d = true`** for this large structure.) Let's concentrate on the top element of this column matrix (row one, column one) formally, and call it **`mp1`**.

14.3.2 Use of trigcrunch, rootcrunch and matrixmap

We first use **`trigcrunch`** and then **`psq_to_Esq`** on this element.

```
(%i2) fundef (trigcrunch);
(%o2) trigcrunch(expr, x) := (trigsimp(expand(demoivre(expr))), to_ao2(%%, x),
                                                                rootcrunch(%%))
(%i3) fundef (rootcrunch);
(%o3) rootcrunch(_x%) := (expand(_x%), expand(%% conj(%%)), rootscontract(%%),
                                                                    factor(%%))
```

```
(%i9) mp1 : mprod[1,1];
        %i phi     th
(%o9) %e       sin(--) (%i p sin(phi) sin(th) - p cos(phi) sin(th)) sqrt(E + p)
                   2
                          th                     th
                  - m cos(--) sqrt(E - p) + cos(--) sqrt(E + p) (E - p cos(th))
                          2                      2
(%i10) trigcrunch(mp1,th);
                  2 th                       2    2     2    2    2
(%o10)       - cos (--) (E - p) (2 m sqrt(E  - p ) - E  + p  - m )
                    2
(%i11) psq_to_Esq(%,p,E,m);
(%o11)                              0
```

We can now work on the whole column matrix at once, using **`matrixmap`**. Here is the basic behavior we use:

```
(%i12) M : matrix([a1],[a2],[a3],[a4]);
                                    [ a1 ]
                                    [    ]
                                    [ a2 ]
(%o12)                              [    ]
                                    [ a3 ]
                                    [    ]
                                    [ a4 ]
(%i13) matrixmap('f,M);
                                   [ f(a1) ]
                                   [       ]
                                   [ f(a2) ]
(%o13)                             [       ]
                                   [ f(a3) ]
                                   [       ]
                                   [ f(a4) ]
```

We first map **`trigcrunch`** on all four elements, via the **`lambda`** function method, and then map **`psq_to_Esq`** on all four elements via the same method, which yields a column vector all of whose elements are zero.

```
(%i14) mprodA : matrixmap(lambda([x],trigcrunch (x,th)), mprod);
                    2 th                     2    2     2    2    2
(%o14) matrix([- cos (--) (E - p) (2 m sqrt(E  - p ) - E  + p  - m )],
                      2
       2           2          2 th
[- (sin (phi) + cos (phi)) sin (--) (E - p)
                                2
            2    2     2    2    2          2 th
 (2 m sqrt(E  - p ) - E  + p  - m )], [- cos (--) (E + p)
                                              2
            2    2     2    2    2           2           2          2 th
 (2 m sqrt(E  - p ) - E  + p  - m )], [- (sin (phi) + cos (phi)) sin (--)
                                                                      2
                    2    2     2    2    2
 (E + p) (2 m sqrt(E  - p ) - E  + p  - m )])
(%i15) mprodB : matrixmap(lambda([x],psq_to_Esq(x,p,E,m)),mprodA);
                                     [ 0 ]
                                     [   ]
                                     [ 0 ]
(%o15)                               [   ]
                                     [ 0 ]
                                     [   ]
                                     [ 0 ]
```

We normally describe scattering in the **`z-x`** plane, and if the 3-vector $\vec{p}$ makes an angle $0 \leq \theta \leq \pi$ with the positive z axis and we are describing a "positive" helicity (**`sv = 1`**) (or "righthanded, R") case, we would use the syntax **`up1 : UU(E,p,th,0,1)`**.

```
(%i16) up1 : UU(E,p,th,0,1);
                            [     th              ]
                            [ cos(--) sqrt(E - p) ]
                            [     2               ]
                            [                     ]
                            [     th              ]
                            [ sin(--) sqrt(E - p) ]
                            [     2               ]
(%o16)                      [                     ]
                            [     th              ]
                            [ cos(--) sqrt(E + p) ]
                            [     2               ]
                            [                     ]
                            [     th              ]
                            [ sin(--) sqrt(E + p) ]
                            [     2               ]
```

If the same lepton were instead moving in the **`z-x`** plane in the **opposite direction** and with the same energy **`E`**, momentum magnitude **`p`**, and positive helicity, we would use the syntax **`up1 : UU (E,p,%pi - th, %pi, 1)`**.

```
(%i17) up1 : UU (E,p,%pi - th, %pi, 1);
                           [      th               ]
                           [  sin(--) sqrt(E - p)  ]
                           [      2                ]
                           [                       ]
                           [       th              ]
                           [ - cos(--) sqrt(E - p) ]
                           [       2               ]
(%o17)                     [                       ]
                           [      th               ]
                           [  sin(--) sqrt(E + p)  ]
                           [      2                ]
                           [                       ]
                           [       th              ]
                           [ - cos(--) sqrt(E + p) ]
                           [       2               ]
```

Note that we have set **`trigexpand:true`** in **`dirac3.mac`**, so that we have the behavior:

```
(%i18) cos((%pi-th)/2);
                                     %pi - th
(%o18)                           cos(--------)
                                        2
(%i19) expand(%);
                                        th
(%o19)                              sin(--)
                                        2
```

and the last step in the definition of **UU** is to apply **expand** to the column matrix.

For a **negative helicity** lepton, in both cases, we would use **sv = -1**.
For the case of a lepton with four-momentum p_1, moving in the **z-x** plane, using some temporary notation, let **up1_plus** be the positive helicity case and **up1_minus** be the **same** lepton with the same energy and the **same** direction of 3-momentum, but with **negative** helicity.

14.3.3 Lepton Helicity Operator H, H, Σ, SIG

If we arbitrarily define the **helicity operator** applicable to a **lepton** H as

$$H\,u(p,\sigma) = \sigma\,u(p,\sigma), \quad \text{with} \quad \sigma = \pm 1 \tag{14.12}$$

then we use

$$H = \Sigma \cdot \hat{p}, \quad \text{where} \quad \hat{p} = \frac{\vec{p}}{|\vec{p}|}, \tag{14.13}$$

and for $k = 1, 2, 3$

$$\Sigma^k = \begin{bmatrix} \sigma_k & 0 \\ 0 & \sigma_k \end{bmatrix} \tag{14.14}$$

are available as our 4×4 matrices **SIG[k]**, defined in terms of the Pauli matrices. In the following calculations, **to_ao2(expr, angle)** is our own function which uses trig identies to replace an expression involving **angle** with an equivalent expression involving **angle/2**.

```
(%i2) up1_plus : UU(E,p,th,0,1);
                            [     th              ]
                            [ cos(--) sqrt(E - p) ]
                            [     2               ]
                            [                     ]
                            [     th              ]
                            [ sin(--) sqrt(E - p) ]
                            [     2               ]
(%o2)                       [                     ]
                            [     th              ]
                            [ cos(--) sqrt(E + p) ]
                            [     2               ]
                            [                     ]
                            [     th              ]
                            [ sin(--) sqrt(E + p) ]
                            [     2               ]
(%i3) H : sin(th)*SIG[1] + cos(th)*SIG[3];
                  [ cos(th)   sin(th)      0         0     ]
                  [                                        ]
                  [ sin(th)  - cos(th)     0         0     ]
(%o3)             [                                        ]
                  [    0         0      cos(th)   sin(th)  ]
                  [                                        ]
                  [    0         0      sin(th)  - cos(th) ]
```

```
(%i4) H_up1_plus : H . up1_plus;
         [     th                            th                    ]
         [ sin(--) sin(th) sqrt(E - p) + cos(--) cos(th) sqrt(E - p) ]
         [     2                             2                     ]
         [                                                         ]
         [     th                            th                    ]
         [ cos(--) sin(th) sqrt(E - p) - sin(--) cos(th) sqrt(E - p) ]
         [     2                             2                     ]
(%o4)    [                                                         ]
         [     th                            th                    ]
         [ sin(--) sin(th) sqrt(E + p) + cos(--) cos(th) sqrt(E + p) ]
         [     2                             2                     ]
         [                                                         ]
         [     th                            th                    ]
         [ cos(--) sin(th) sqrt(E + p) - sin(--) cos(th) sqrt(E + p) ]
         [     2                             2                     ]
(%i5) H_up1_plus : to_ao2 (%, th);
                            [     th             ]
                            [ cos(--) sqrt(E - p) ]
                            [     2              ]
                            [                    ]
                            [     th             ]
                            [ sin(--) sqrt(E - p) ]
                            [     2              ]
(%o5)                       [                    ]
                            [     th             ]
                            [ cos(--) sqrt(E + p) ]
                            [     2              ]
                            [                    ]
                            [     th             ]
                            [ sin(--) sqrt(E + p) ]
                            [     2              ]
(%i6) is(equal(H_up1_plus,up1_plus));
(%o6)                                true
(%i7) up1_minus : UU(E,p,th,0,-1);
                           [       th             ]
                           [ - sin(--) sqrt(E + p) ]
                           [       2              ]
                           [                      ]
                           [      th              ]
                           [  cos(--) sqrt(E + p)  ]
                           [      2               ]
(%o7)                      [                      ]
                           [       th             ]
                           [ - sin(--) sqrt(E - p) ]
                           [       2              ]
                           [                      ]
                           [      th              ]
                           [  cos(--) sqrt(E - p)  ]
                           [      2               ]
(%i8) H_up1_minus : to_ao2( H . up1_minus, th);
                           [      th              ]
                           [  sin(--) sqrt(E + p)  ]
                           [      2               ]
                           [                      ]
                           [       th             ]
                           [ - cos(--) sqrt(E + p) ]
                           [       2              ]
(%o8)                      [                      ]
                           [      th              ]
                           [  sin(--) sqrt(E - p)  ]
                           [      2               ]
                           [                      ]
                           [       th             ]
                           [ - cos(--) sqrt(E - p) ]
                           [       2              ]
(%i9) is(equal(H_up1_minus,-up1_minus));
(%o9)                                true
```

A **barred** lepton spinor (a row matrix), corresponding to the column matrix $u(p,\sigma)$ has the definition

$$\bar{u}(p,\sigma) = u(p,\sigma)^{\dagger}\,\gamma^0. \tag{14.15}$$

Using our package function **sbar**, which has the Maxima code definition

```
   sbar (_uu%) := hc (_uu%) . Gam[0]$
```

in which **hc** is the package defined hermitian conjugate function, we can use either of the equivalent routes

```
   up1_plus_bar : sbar (UU(E,p,th,0,1))
```

or

```
   up1_plus_bar : sbar (up1_plus)
```

to generate the explicit **barred** spinor corresponding to an explicit given Dirac spinor.

We can check the **normalization** (14.6) of our lepton Dirac spinor definition here. We first show that

$$\bar{u}(p,+1)\,u(p,+1) = 2\,m, \tag{14.16}$$

using several different paths to arrive at the same answer $2\,m$. We use **up1_plus** and **up1_minus** already defined above.

```
(%i10) up1_plus_bar : sbar(up1_plus);
                [     th               ]         [     th               ]
(%o10)  Col 1 = [ cos(--) sqrt(E + p) ] Col 2 = [ sin(--) sqrt(E + p) ]
                [     2                ]         [     2                ]
                        [     th               ]        [     th               ]
                Col 3 = [ cos(--) sqrt(E - p) ] Col 4 = [ sin(--) sqrt(E - p) ]
                        [     2                ]        [     2                ]
(%i11) grind(%)$
matrix([cos(th/2)*sqrt(E+p),sin(th/2)*sqrt(E+p),cos(th/2)*sqrt(E-p),
        sin(th/2)*sqrt(E-p)])$
(%i12) ubu : up1_plus_bar . up1_plus;
(%o12) 2*sin(th/2)^2*sqrt(E-p)*sqrt(E+p)+2*cos(th/2)^2*sqrt(E-p)*sqrt(E+p)
```

14.3.4 Use of uv_simp

The simplest way to simplify this expression is to use our function **uv_simp**, defined in **dirac3.mac** with the code

```
(%i13) fundef (uv_simp);
(%o13) uv_simp(%expr, %p, %E, %m) := trigsimp(uv_simp1(%expr, %p, %E, %m))
(%i14) fundef (uv_simp1);
(%o14) uv_simp1(%expr, %p, %E, %m) := (trigsimp(%expr), rootscontract(%%),
                                                  psq_to_Esq(%%, %p, %E, %m))
```

which gives us our desired answer:

```
(%i15) uv_simp(ubu,p,E,m);
(%o15)                                2 m
```

Instead of that fast method, it may be more educational to do this massaging one step at a time, for example:

```
(%i16) ubu1 : trigsimp(ubu);
(%o16)                    2 sqrt(E - p) sqrt(E + p)
(%i17) ubu2 : rootscontract(ubu1);
                                             2    2
(%o17)                         2 sqrt(E  - p )
```

At this stage there are several ways one can proceed:

```
(%i18) ubu2, E = sqrt(p^2 + m^2);
(%o18)                            2 m
(%i19) ratsubst (m, sqrt(E^2 - p^2), ubu2);
(%o19)                            2 m
(%i20) psq_to_Esq (ubu2,p,E,m);
(%o20)                            2 m
(%i21) Esq_to_psq (ubu2,p,E,m);
(%o21)                            2 m
```

In the last two steps, we used the package function **psq_to_Esq** which replaces the symbol for the square of 3-momentum in terms of the energy symbol and the mass symbol, and then the package function **Esq_to_psq** which replaces the symbol for the square of the energy in terms of the momentum symbol and the mass symbol. The definitions in **dirac3.mac** are:

```
  psq_to_Esq (expr,%p,%E,%m) :=
      expand (ratsubst (%E^2 - %m^2, %p^2, expr))$
  Esq_to_psq (expr,%p,%E,%m) :=
      expand (ratsubst (%m^2 + %p^2 , %E^2, expr))$
```

Recall that **ratsubst (a,b,c)** substitutes **a** for **b** in **c**.

We next show that

$$\bar{u}(p,+1)\,u(p,-1) = 0. \tag{14.17}$$

using **up1_plus_bar** and **up1_minus** defined above:

```
(%i22) up1_plus_bar . up1_minus;
(%o22)                             0
```

We next check the "projection matrix" relation in Eq. (14.7). In order to calculate the right hand side, we need to use **comp_def** to assign values to the array components **p1[j]** for **j = 0, 1,2,3**.

```
(%i23) comp_def(p1(E,p*sin(th),0,p*cos(th)))$
(%i24) listarray (p1);
(%o24)                 [E, p sin(th), 0, p cos(th)]
(%i25) p1[0];
(%o25)                             E
```

We can then create the explicit matrix **sL(p1)** which corresponds to $\not{p}_1$, and then let **RHS** stand for the righthand side of Eq (14.7), namely: $\not{p}_1 + m$. In the following, **I4** is a predefined 4×4 unit matrix.

```
(%i26) RHS : sL(p1) + m*I4;
        [       m              0        E - p cos(th)    - p sin(th)  ]
        [                                                             ]
        [       0              m         - p sin(th)   E + p cos(th)  ]
(%o26)  [                                                             ]
        [ E + p cos(th)    p sin(th)          m              0        ]
        [                                                             ]
        [   p sin(th)    E - p cos(th)        0              m        ]
```

We now construct the sum **uub_sum** on the lefthand side of Eq (14.7). The expression **RHS** depends on **cos(th)** and **sin(th)**, and the Dirac spinors depend on **cos(th/2)** and **sin(th/2)**.

14.3.5 Use of fr_ao2

Our package function **fr_ao2(expr, th)** uses trig identities to convert trig functions of **th/2** to equivalent functions of **th**.

```
(%i27) up1_minus_bar : sbar (up1_minus);
                [       th              ]          [     th              ]
(%o27)  Col 1 = [ - sin(--) sqrt(E - p) ] Col 2 = [ cos(--) sqrt(E - p) ]
                [       2               ]          [     2               ]
                      [       th              ]          [     th              ]
              Col 3 = [ - sin(--) sqrt(E + p) ] Col 4 = [ cos(--) sqrt(E + p) ]
                      [       2               ]          [     2               ]
```

```
(%i28) grind(%)$
matrix([-sin(th/2)*sqrt(E-p),cos(th/2)*sqrt(E-p),-sin(th/2)*sqrt(E+p),
        cos(th/2)*sqrt(E+p)])$
(%i29) uub_sum :
         uv_simp (up1_plus . up1_plus_bar + up1_minus . up1_minus_bar, p,E,m);
                [          m           ]         [          0           ]
                [                      ]         [                      ]
                [          0           ]         [          m           ]
                [                      ]         [                      ]
                [            2 th      ]         [          th      th  ]
(%o29)  Col 1 = [ E + 2 p cos (--) - p ] Col 2 = [ 2 p cos(--) sin(--)  ]
                [             2        ]         [         2       2    ]
                [                      ]         [                      ]
                [         th      th   ]         [            2 th      ]
                [ 2 p cos(--) sin(--)  ]         [ E - 2 p cos (--) + p ]
                [        2       2     ]         [             2        ]
                    [           2 th       ]          [           th      th  ]
                    [ E - 2 p cos (--) + p  ]         [ - 2 p cos(--) sin(--) ]
                    [              2        ]         [          2       2    ]
                    [                       ]         [                       ]
                    [           th      th  ]         [            2 th       ]
            Col 3 = [ - 2 p cos(--) sin(--) ] Col 4 = [ E + 2 p cos (--) - p  ]
                    [          2       2    ]         [             2         ]
                    [                       ]         [                       ]
                    [          m            ]         [           0           ]
                    [                       ]         [                       ]
                    [          0            ]         [           m           ]
(%i30) uub_sum1 : fr_ao2 (uub_sum, th);
        [       m             0        E - p cos(th)   - p sin(th)  ]
        [                                                           ]
        [       0             m         - p sin(th)   E + p cos(th) ]
(%o30)  [                                                           ]
        [ E + p cos(th)    p sin(th)          m              0      ]
        [                                                           ]
        [   p sin(th)    E - p cos(th)        0              m      ]
(%i31) is(equal(uub_sum1,RHS));
(%o31)                               true
```

14.3.6 Anti-Lepton Spinors, $v(p,\sigma)$, VV

The **antilepton** Dirac spinor is the four element column matrix $v(p,\sigma)$ in which $\sigma = +1$ picks out the positive helicity $h = +1/2$ antiparticle state and $\sigma = -1$ picks out negative helicity $h = -1/2$ antiparticle state, and is defined (with P/S phase choice) via

$$v(p,\sigma) = -\gamma^5\, u(p,-\sigma) \tag{14.18}$$

with the basic property

$$\not{p}\, v(p,\sigma) = -m\, v(p,\sigma). \tag{14.19}$$

The corresponding barred antiparticle spinor is a four element row matrix

$$\bar{v}(p,\sigma) = v(p,\sigma)^\dagger\, \gamma^0 \tag{14.20}$$

and the normalization is

$$\bar{v}(p,\sigma)\, v(p,\sigma') = -2\, m\, \delta_{\sigma\,\sigma'} \quad \sigma,\, \sigma' = \pm 1. \tag{14.21}$$

We also have the "projection matrix" relation

$$\sum_\sigma v(p,\sigma)\, \bar{v}(p,\sigma) = \not{p} - m \tag{14.22}$$

For the case of an antilepton with four-momentum p_1, moving in the **`z-x`** plane, using some temporary notation, let **`vp1_plus`** be the positive helicity case and **`vp1_minus`** be the **same** antilepton with the same energy and the **same** direction of 3-momentum, but with **negative** helicity.

14.3.7 Antilepton Helicity Operator $\mathrm{H_v}$, Hv

If we arbitrarily define the **helicity operator** applicable to an **anti-lepton** H_v as

$$H_v\, v(p,\sigma) = \sigma\, v(p,\sigma), \quad \text{with} \quad \sigma = \pm 1 \tag{14.23}$$

then we need to use the definition

$$H_v = -\Sigma \cdot \hat{p}, \quad \text{where} \quad \hat{p} = \frac{\vec{p}}{|\,\vec{p}\,|}, \tag{14.24}$$

and for $k = 1, 2, 3$

$$\Sigma^k = \begin{bmatrix} \sigma_k & 0 \\ 0 & \sigma_k \end{bmatrix} \tag{14.25}$$

are available as our 4×4 matrices **`SIG[k]`**, defined in terms of the Pauli matrices. In the following calculations, **`to_ao2(expr, angle)`** is our own function which uses trig identies to replace an expression involving **`angle`** with an equivalent expression involving **`angle/2`**.

We generate an **explicit** antilepton Dirac "spinor", a four element column **matrix**, denoted symbolically by $v(p_1, \sigma)$, corresponding to an antilepton with **four momentum** $p_1^\mu = (E, \vec{p})$, using the function **`VV(E,p,theta,phi,sv)`** which descibes a spin **`1/2`** antilepton with relativistic energy **`E`**, 3-momentum magnitude **`p`**, whose 3-vector momentum direction is described by spherical polar angles **`(theta,phi)`**, where **`0 <= theta <= %pi`** and **`0 <= phi <= 2*%pi`**, and whose helicity quantum number **`sv`** is **`1`** for helicity **`+1/2`** and **`sv`** is **`-1`** for helicity **`-1/2`**.

For arbitrary angles (subject to the above), our conventions give

```
(%i2) VV(E,p,th,phi,1);
                          [     - %i phi     th               ]
                          [ - %e          sin(--) sqrt(E + p) ]
                          [                  2                ]
                          [                                   ]
                          [           th                      ]
                          [       cos(--) sqrt(E + p)         ]
                          [           2                       ]
(%o2)                     [                                   ]
                          [    - %i phi     th                ]
                          [  %e          sin(--) sqrt(E - p)  ]
                          [                 2                 ]
                          [                                   ]
                          [            th                     ]
                          [      - cos(--) sqrt(E - p)        ]
                          [            2                      ]
(%i3) VV(E,p,th,phi,-1);
                           [          th                    ]
                           [      cos(--) sqrt(E - p)       ]
                           [          2                     ]
                           [                                ]
                           [    %i phi     th               ]
                           [  %e       sin(--) sqrt(E - p)  ]
                           [               2                ]
(%o3)                      [                                ]
                           [           th                   ]
                           [     - cos(--) sqrt(E + p)      ]
                           [           2                    ]
                           [                                ]
                           [     %i phi     th              ]
                           [ - %e       sin(--) sqrt(E + p) ]
                           [                2               ]
```

Thus for arbitrary angles

$$v(E,p,\theta,\phi,\sigma=1)=\begin{bmatrix}-\sqrt{E+p}\,\sin(\theta/2)\,e^{-i\phi}\\ \sqrt{E+p}\,\cos(\theta/2)\\ \sqrt{E-p}\,\sin(\theta/2)\,e^{-i\phi}\\ -\sqrt{E-p}\,\cos(\theta/2)\end{bmatrix} \tag{14.26}$$

and

$$v(E,p,\theta,\phi,\sigma=-1)=\begin{bmatrix}\sqrt{E-p}\,\cos(\theta/2)\\ \sqrt{E-p}\,\sin(\theta/2)\,e^{i\phi}\\ -\sqrt{E+p}\,\cos(\theta/2)\\ -\sqrt{E+p}\,\sin(\theta/2)\,e^{i\phi}\end{bmatrix} \tag{14.27}$$

We normally describe scattering in the **z-x** plane, and if the 3-vector $\vec{p}$ makes an angle $\theta \le \pi$ with the positive z axis and we are describing a "positive" helicity (**sv = 1**) (or "righthanded, R") case, we would use the syntax **vp1 : VV(E,p,th,0,1)**.

```
(%i4) vp1 : VV(E,p,th,0,1);
                           [       th            ]
                           [ - sin(--) sqrt(E + p) ]
                           [       2             ]
                           [                     ]
                           [      th             ]
                           [  cos(--) sqrt(E + p)  ]
                           [      2              ]
(%o4)                      [                     ]
                           [      th             ]
                           [  sin(--) sqrt(E - p)  ]
                           [      2              ]
                           [                     ]
                           [       th            ]
                           [ - cos(--) sqrt(E - p) ]
                           [       2             ]
```

If the same particle were instead moving in the **z-x** plane in the **opposite direction** and with the same energy, momentum magnitude, and positive helicity, we would use the syntax **vp1 : VV (E,p,%pi - th, %pi, 1)**.

```
(%i5) vp1 : VV (E,p,%pi - th, %pi, 1);
                           [      th             ]
                           [  cos(--) sqrt(E + p)  ]
                           [      2              ]
                           [                     ]
                           [      th             ]
                           [  sin(--) sqrt(E + p)  ]
                           [      2              ]
(%o5)                      [                     ]
                           [       th            ]
                           [ - cos(--) sqrt(E - p) ]
                           [       2             ]
                           [                     ]
                           [       th            ]
                           [ - sin(--) sqrt(E - p) ]
                           [       2             ]
```

For a negative helicity antilepton, in both cases, we would use **sv = -1**.

For the case of an antilepton with four-momentum p_1, moving in the **z-x** plane, using some temporary notation, let **vp1_plus** be the positive helicity case and **vp1_minus** be the **same** particle with the same energy and the **same** direction of 3-momentum, but with **negative** helicity. We let the symbol **Hv** stand for the antilepton helicity operator H_v, bearing in mind that $\hat{p} = \hat{x}\,\sin(th) + \hat{z}\,\cos(th)$.

```
(%i6) vp1_plus : VV(E,p,th,0,1);
                           [       th              ]
                           [ - sin(--) sqrt(E + p) ]
                           [       2               ]
                           [                       ]
                           [      th               ]
                           [  cos(--) sqrt(E + p)  ]
                           [      2                ]
(%o6)                      [                       ]
                           [      th               ]
                           [  sin(--) sqrt(E - p)  ]
                           [      2                ]
                           [                       ]
                           [       th              ]
                           [ - cos(--) sqrt(E - p) ]
                           [       2               ]
(%i7) Hv : -sin(th)*SIG[1] - cos(th)*SIG[3];
                [ - cos(th)  - sin(th)      0          0     ]
                [                                            ]
                [ - sin(th)   cos(th)       0          0     ]
(%o7)           [                                            ]
                [     0          0      - cos(th)  - sin(th) ]
                [                                            ]
                [     0          0      - sin(th)   cos(th)  ]
(%i8) Hv_vp1_plus :  to_ao2( Hv . vp1_plus, th );
                           [       th              ]
                           [ - sin(--) sqrt(E + p) ]
                           [       2               ]
                           [                       ]
                           [      th               ]
                           [  cos(--) sqrt(E + p)  ]
                           [      2                ]
(%o8)                      [                       ]
                           [      th               ]
                           [  sin(--) sqrt(E - p)  ]
                           [      2                ]
                           [                       ]
                           [       th              ]
                           [ - cos(--) sqrt(E - p) ]
                           [       2               ]
(%i9) is(equal(Hv_vp1_plus, vp1_plus));
(%o9)                               true
(%i10) vp1_minus : VV(E,p,th,0,-1);
                           [      th               ]
                           [  cos(--) sqrt(E - p)  ]
                           [      2                ]
                           [                       ]
                           [      th               ]
                           [  sin(--) sqrt(E - p)  ]
                           [      2                ]
(%o10)                     [                       ]
                           [       th              ]
                           [ - cos(--) sqrt(E + p) ]
                           [       2               ]
                           [                       ]
                           [       th              ]
                           [ - sin(--) sqrt(E + p) ]
                           [       2               ]
```

```
(%i11) Hv_vp1_minus :  to_ao2( Hv . vp1_minus, th );
                           [        th              ]
                           [ - cos(--) sqrt(E - p) ]
                           [        2              ]
                           [                        ]
                           [        th              ]
                           [ - sin(--) sqrt(E - p) ]
                           [        2              ]
(%o11)                     [                        ]
                           [       th               ]
                           [  cos(--) sqrt(E + p)  ]
                           [       2                ]
                           [                        ]
                           [       th               ]
                           [  sin(--) sqrt(E + p)  ]
                           [       2                ]
(%i12) is(equal(Hv_vp1_minus, -vp1_minus));
(%o12)                               true
```

Here we check the normalization of the antilepton spinors (simplifying the result using **`uv_simp (expr, p, E, m)`**) as well as checking the orthogonality of antilepton spinors of opposite helicity.

```
(%i13) vp1_plus_bar : sbar(vp1_plus);
                [      th              ]          [        th              ]
(%o13)  Col 1 = [ sin(--) sqrt(E - p) ] Col 2 = [ - cos(--) sqrt(E - p) ]
                [      2               ]          [        2               ]
                      [        th              ]        [      th              ]
              Col 3 = [ - sin(--) sqrt(E + p) ] Col 4 = [ cos(--) sqrt(E + p) ]
                      [        2               ]        [      2               ]
(%i14) grind(%)$
matrix([sin(th/2)*sqrt(E-p),-cos(th/2)*sqrt(E-p),-sin(th/2)*sqrt(E+p),
        cos(th/2)*sqrt(E+p)])$
(%i15) vbv : vp1_plus_bar . vp1_plus;
                2 th
(%o15) (- 2 sin (--) sqrt(E - p) sqrt(E + p))
                  2
                                                  2 th
                                           - 2 cos (--) sqrt(E - p) sqrt(E + p)
                                                    2
(%i16) uv_simp(vbv,p,E,m);
(%o16)                               - 2 m
(%i17) vp1_plus_bar . vp1_minus;
(%o17)                                 0
```

We next check the projection matrix relation (14.22). After using **`comp_def`** to define the array **`p1[j]`**, we can then create the explicit matrix **`sL(p1)`** which corresponds to $\not{p}_1$ and **`RHS`** which correponds to the righthand side of (14.22) $\not{p}_1 - m$. In the following, **`I4`** is a predefined 4×4 unit matrix.

```
(%i18) comp_def(p1(E,p*sin(th),0,p*cos(th)))$
(%i19) listarray(p1);
(%o19)                  [E, p sin(th), 0, p cos(th)]
(%i20) RHS : sL(p1) - m*I4;
        [      - m             0        E - p cos(th)   - p sin(th)  ]
        [                                                            ]
        [       0             - m       - p sin(th)   E + p cos(th) ]
(%o20)  [                                                            ]
        [ E + p cos(th)    p sin(th)         - m             0       ]
        [                                                            ]
        [   p sin(th)    E - p cos(th)        0             - m      ]
```

We now construct the sum **`vvb_sum`** on the lefthand side of (14.22). Our package function **`fr_ao2(expr, angle)`** uses trig identities to convert explicit functions of $\theta/2$ to equivalent functions of θ.

```
(%i21) vp1_minus_bar : sbar (vp1_minus);
                [       th             ]         [       th             ]
(%o21)  Col 1 = [ - cos(--) sqrt(E + p) ] Col 2 = [ - sin(--) sqrt(E + p) ]
                [       2              ]         [       2              ]
                        [     th              ]         [     th              ]
                Col 3 = [ cos(--) sqrt(E - p) ] Col 4 = [ sin(--) sqrt(E - p) ]
                        [     2               ]         [     2               ]
(%i22) grind(%)$
matrix([-cos(th/2)*sqrt(E+p),-sin(th/2)*sqrt(E+p),cos(th/2)*sqrt(E-p),
        sin(th/2)*sqrt(E-p)])$
(%i23) vvb_sum :
          uv_simp (vp1_plus . vp1_plus_bar + vp1_minus . vp1_minus_bar, p,E,m);
                [         - m          ]         [          0           ]
                [                      ]         [                      ]
                [          0           ]         [         - m          ]
                [                      ]         [                      ]
                [            2 th      ]         [          th      th   ]
(%o23)  Col 1 = [ E + 2 p cos (--) - p ] Col 2 = [ 2 p cos(--) sin(--)  ]
                [              2       ]         [          2       2   ]
                [                      ]         [                      ]
                [         th      th   ]         [             2 th     ]
                [ 2 p cos(--) sin(--)  ]         [ E - 2 p cos (--) + p ]
                [         2       2    ]         [               2      ]
                    [            2 th        ]         [           th      th  ]
                    [ E - 2 p cos (--) + p   ]         [ - 2 p cos(--) sin(--) ]
                    [              2         ]         [           2       2   ]
                    [                        ]         [                       ]
                    [           th      th   ]         [            2 th       ]
            Col 3 = [ - 2 p cos(--) sin(--) ] Col 4 = [ E + 2 p cos (--) - p  ]
                    [           2       2    ]         [               2       ]
                    [                        ]         [                       ]
                    [          - m           ]         [           0           ]
                    [                        ]         [                       ]
                    [           0            ]         [          - m          ]
(%i24) vvb_sum1 : fr_ao2(vvb_sum, th);
        [      - m             0        E - p cos(th)   - p sin(th)  ]
        [                                                            ]
        [       0             - m       - p sin(th)   E + p cos(th) ]
(%o24)  [                                                            ]
        [ E + p cos(th)    p sin(th)        - m             0        ]
        [                                                            ]
        [   p sin(th)    E - p cos(th)       0             - m       ]
(%i25) is(equal(vvb_sum1,RHS));
(%o25)                               true
```

14.4 Spin Projection Matrix $\mathbb{S}(p,\sigma)$, P(sv), P(sv, Sp)

14.4.1 Right-Handed Spin 4-Vector $\mathbf{s^\mu_R(p)}$, Sp

We define the 4×4 spin projection matrix $\mathbb{S}(p,\sigma)$.

$$\mathbb{S}(p,\sigma) = \frac{1}{2}\left(\mathbb{1} + \sigma\,\gamma^5\,\not{s}_R(p)\right) \quad \text{for} \quad \sigma = \pm 1, \tag{14.28}$$

in which $\mathbb{1}$ stands for the 4×4 unit matrix and in which $\not{s}_R(p) = s_{R\mu}\,\gamma^\mu$, and the right handed spin 4-vector s^μ_R generated by the 4-vector p has the components

$$s^\mu_R(p) = \left(\frac{|\vec{p}|}{m}, \frac{E}{m}\,\hat{\mathbf{p}}\right) \tag{14.29}$$

in which $E = \sqrt{m^2 + \mathbf{p}^2}$.

The spin 4-vector s^μ_R has the properties

$$s_R(p) \cdot s_R(p) = -1, \quad \text{and} \quad s_R(p) \cdot p = 0. \tag{14.30}$$

In our code, we usually use a symbol like **Sp1** for the spin vector associated with the 4-vector **p1**. Here is an example of the above constraints (in addition to $p_1 \cdot p_1 = m^2$) being satisfied:

```
(%i2) comp_def (p1(E,p*sin(th),0,p*cos(th)), Sp1(p/m,E*sin(th)/m,0,E*cos(th)/m))$
(%i3) listarray (p1);
(%o3)                    [E, p sin(th), 0, p cos(th)]
(%i4) VP (p1,p1);
                                     2    2
(%o4)                               E  - p
(%i5) psq_to_Esq(%,p,E,m);
                                       2
(%o5)                                 m
(%i6) listarray (Sp1);
                          p  sin(th) E     cos(th) E
(%o6)                    [-, ---------, 0, ---------]
                          m      m             m
(%i7) VP (Sp1, Sp1);
                                     2    2
                                    p    E
(%o7)                               -- - --
                                     2    2
                                    m    m
(%i8) psq_to_Esq(%,p,E,m);
(%o8)                                 - 1
(%i9) VP (Sp1, p1);
(%o9)                                  0
```

and we can also show that

$$\not{p}\,\gamma^5\,\not{s}_R(p) = \gamma^5\,\not{s}_R(p)\,\not{p}. \tag{14.31}$$

```
(%i10) is(equal(trigsimp (sL(p1) . Gam[5] . sL(Sp1)), trigsimp( Gam[5] . sL(Sp1) . sL(p1))));
(%o10)                               true
```

The relation (14.31) implies that $\not{p}$ commutes with $\mathbb{S}(p,\sigma)$. To show this directly we can use the explicit spin projection matrix **P**. The **explicit** spin projection matrix has the syntax **P(sv)** or **P(sv,Sp)**, for the massless and massive case respectively.

```
(%i11) A : sL(p1) . P(1,Sp1)$
(%i12) B : P(1,Sp1) . sL(p1)$
(%i13) is(equal(uv_simp(A,p,E,m), uv_simp(B,p,E,m)));
(%o13)                               true
(%i14) A : sL(p1) . P(-1,Sp1)$
(%i15) B : P(-1,Sp1) . sL(p1)$
(%i16) is(equal(uv_simp(A,p,E,m), uv_simp(B,p,E,m)));
(%o16)                               true
```

In **trace calculations**, the spin projection matrix $\mathbb{S}(p,\sigma)$ is represented symbolically by the symbol **S(sv)** or **S(sv,Sp)**, for the massless and massive case respectively, where **sv** takes on values **+/- 1**, and **Sp** represents the right-handed spin 4-vector defined by the 4-momentum of the particle. Note the order of the arguments (of our code symbols) in the massive case.

The **explicit** spin projection matrix is given by **P(sv)** or **P(sv,Sp)**, for the massless and massive case respectively.

For massive $m > 0$ leptons, we have the identity

$$u(p,\sigma)\,\bar{u}(p,\sigma) = \mathbb{S}(p,\sigma)\,(\not{p} + m) = (\not{p} + m)\,\mathbb{S}(p,\sigma). \tag{14.32}$$

We can use explicit matrices to check this equation for $m > 0$ leptons:

```
(%i17) up1_plus : UU(E,p,th,0,1)$
(%i18) up1_minus : UU(E,p,th,0,-1)$
(%i19) up1_plus_bar : sbar(up1_plus)$
(%i20) up1_minus_bar : sbar (up1_minus)$
```

First for $\sigma = 1$,

```
(%i21) A : up1_plus . up1_plus_bar$
(%i22) A : uv_simp(A,p,E,m)$
(%i23) A : fr_ao2(A,th)$
(%i24) display2d:false$
(%i25) A : trigsimp(A);
(%o25) matrix([(m*cos(th)+m)/2,(m*sin(th))/2,((cos(th)+1)*E-p*cos(th)-p)/2,
                (sin(th)*E-p*sin(th))/2],
               [(m*sin(th))/2,-(m*cos(th)-m)/2,(sin(th)*E-p*sin(th))/2,
                -((cos(th)-1)*E-p*cos(th)+p)/2],
               [((cos(th)+1)*E+p*cos(th)+p)/2,(sin(th)*E+p*sin(th))/2,
                (m*cos(th)+m)/2,(m*sin(th))/2],
               [(sin(th)*E+p*sin(th))/2,-((cos(th)-1)*E+p*cos(th)-p)/2,
                (m*sin(th))/2,-(m*cos(th)-m)/2])
(%i26) B : P(1,Sp1) . (sL(p1) + m*I4)$
(%i27) B : uv_simp(B,p,E,m)$
(%i28) B : trigsimp(B);
(%o28) matrix([(m*cos(th)+m)/2,(m*sin(th))/2,((cos(th)+1)*E-p*cos(th)-p)/2,
                (sin(th)*E-p*sin(th))/2],
               [(m*sin(th))/2,-(m*cos(th)-m)/2,(sin(th)*E-p*sin(th))/2,
                -((cos(th)-1)*E-p*cos(th)+p)/2],
               [((cos(th)+1)*E+p*cos(th)+p)/2,(sin(th)*E+p*sin(th))/2,
                (m*cos(th)+m)/2,(m*sin(th))/2],
               [(sin(th)*E+p*sin(th))/2,-((cos(th)-1)*E+p*cos(th)-p)/2,
                (m*sin(th))/2,-(m*cos(th)-m)/2])
(%i29) is(equal(A,B));
(%o29) true
```

and next for $\sigma = -1$,

```
(%i30) A : up1_minus . up1_minus_bar$
(%i31) A : uv_simp(A,p,E,m)$
(%i32) A : fr_ao2(A,th)$
(%i33) A : trigsimp(A);
(%o33) matrix([-(m*cos(th)-m)/2,-(m*sin(th))/2,-((cos(th)-1)*E+p*cos(th)-p)/2,
                -(sin(th)*E+p*sin(th))/2],
               [-(m*sin(th))/2,(m*cos(th)+m)/2,-(sin(th)*E+p*sin(th))/2,
                ((cos(th)+1)*E+p*cos(th)+p)/2],
               [-((cos(th)-1)*E-p*cos(th)+p)/2,-(sin(th)*E-p*sin(th))/2,
                -(m*cos(th)-m)/2,-(m*sin(th))/2],
               [-(sin(th)*E-p*sin(th))/2,((cos(th)+1)*E-p*cos(th)-p)/2,
                -(m*sin(th))/2,(m*cos(th)+m)/2])
(%i34) B : P(-1,Sp1) . (sL(p1) + m*I4)$
(%i35) B : uv_simp(B,p,E,m)$
(%i36) B : trigsimp(B);
(%o36) matrix([-(m*cos(th)-m)/2,-(m*sin(th))/2,-((cos(th)-1)*E+p*cos(th)-p)/2,
                -(sin(th)*E+p*sin(th))/2],
               [-(m*sin(th))/2,(m*cos(th)+m)/2,-(sin(th)*E+p*sin(th))/2,
                ((cos(th)+1)*E+p*cos(th)+p)/2],
               [-((cos(th)-1)*E-p*cos(th)+p)/2,-(sin(th)*E-p*sin(th))/2,
                -(m*cos(th)-m)/2,-(m*sin(th))/2],
               [-(sin(th)*E-p*sin(th))/2,((cos(th)+1)*E-p*cos(th)-p)/2,
                -(m*sin(th))/2,(m*cos(th)+m)/2])
(%i37) is(equal(A,B));
(%o37) true
```

For massive $m > 0$ antileptons, we have the identity

$$v(p,\sigma)\,\bar{v}(p,\sigma) = \mathbb{S}(p,\sigma)\,(\not{p} - m) = (\not{p} - m)\,\mathbb{S}(p,\sigma) \tag{14.33}$$

In the case of zero mass leptons

$$u(p,\sigma)\,\bar{u}(p,\sigma) = \frac{1}{2}\,(1 + \sigma\,\gamma^5)\,\not{p} \tag{14.34}$$

and for zero mass antileptons

$$v(p,\sigma)\,\bar{v}(p,\sigma) = \frac{1}{2}\,(1 - \sigma\,\gamma^5)\,\not{p}. \tag{14.35}$$

14.4.2 Unpolarized Cross Section using the Trace Operation

Calculation of unpolarized cross sections requires summing the absolute value squared of a polarized amplitude over the helicity quantum numbers of the particles involved, and the result can be written as the matrix trace of a 4×4 matrix. A simple example follows. We use a compressed notation in which, for example, u_1 stands for $u(p_1, \sigma_1)$ and u_2 stands for $u(p_2, \sigma_2)$. Let Γ be an arbitrary 4×4 matrix.

One can show that

$$(\bar{u}_2\,\Gamma\,u_1)^* = \bar{u}_1\,\bar{\Gamma}\,u_2 \tag{14.36}$$

in which

$$\bar{\Gamma} = \gamma^0\,\Gamma^\dagger\,\gamma^0. \tag{14.37}$$

Then

$$\sum_{\sigma_1\,\sigma_2} |\bar{u}_2\,\Gamma\,u_1|^2 = \sum_{\sigma_1\,\sigma_2} \bar{u}_2\,\Gamma\,u_1\,\bar{u}_1\,\bar{\Gamma}\,u_2 \tag{14.38}$$

We use the sum over σ_1 to replace $u_1\,\bar{u}_1$ by $(\not{p}_1 + m)$, and are left with

$$\sum_{\sigma_2} \bar{u}_2\,\hat{\Gamma}\,u_2, \quad \hat{\Gamma} \doteq \Gamma\,(\not{p}_1 + m)\,\bar{\Gamma}. \tag{14.39}$$

Writing out the matrix indices explicitly, the sum becomes (using the summation convention for repeated matrix indices)

$$\sum_{\sigma_2} \bar{u}_{2\,a}\,\hat{\Gamma}_{a\,b}\,u_{2\,b} = \hat{\Gamma}_{a\,b}\sum_{\sigma_2} u_{2\,b}\,\bar{u}_{2\,a} = \hat{\Gamma}_{a\,b}\,(\not{p}_2 + m)_{b\,a} = \textbf{Trace}\,\{\hat{\Gamma}\,(\not{p}_2 + m)\} \tag{14.40}$$

Note that **Trace** $(A\,B) =$ **Trace** $(B\,A)$. We then have

$$\sum_{\sigma_1\,\sigma_2} |\bar{u}_2\,\Gamma\,u_1|^2 = \textbf{Trace}\,\{(\not{p}_2 + m)\,\Gamma\,(\not{p}_1 + m)\,\bar{\Gamma}\}. \tag{14.41}$$

14.5 The Levi-Civita Tensor

14.5.1 $\epsilon^{\lambda\mu\nu\sigma}$, Eps(la,mu,nu,si), eps4[la,mu,nu,si]

The Levi-Civita tensor $\epsilon^{\lambda\mu\nu\sigma}$ is the totally antisymmetric tensor, using the convention that $\epsilon^{0123} = +1$ and this implies that

$$\epsilon^{1230} = -1, \quad \epsilon_{0123} = -1 \tag{14.42}$$

and

$$\epsilon_{\mu\nu\lambda\sigma} = -\epsilon^{\mu\nu\lambda\sigma}. \tag{14.43}$$

See Ch. 1, Sect. 6, "Four Vectors", of The Classical Theory of Fields, 4th Rev. Ed., Landau and Lifshitz, for the definitions and properties used here.

We let **Eps[la,mu,nu,si]** and **eps4[la,mu,nu,si]** stand for $\epsilon^{\lambda\mu\nu\sigma}$, and in the file **dirac3.mac** we have the code

```
declare (eps4,antisymmetric)$
tellsimpafter ( eps4[0,1,2,3], 1 )$
```

which results in the behavior

```
(%i2) eps4[0,1,2,3];
(%o2)                               1
(%i3) eps4[1,0,2,3];
(%o3)                              - 1
```

There is no corresponding definition for **Eps**, which is just a symbolic place holder. The function **noncov** can be used to convert **Eps(...)** into **eps4[....]**.

```
(%i4) Eps(0,1,2,3);
(%o4)                          Eps(0, 1, 2, 3)
(%i5) Eps(1,0,2,3);
(%o5)                          Eps(1, 0, 2, 3)
(%i6) noncov (Eps(0,1,2,3));
(%o6)                                 1
(%i7) noncov (Eps(1,0,2,3));
(%o7)                                - 1
(%i8) noncov (Eps(n1,n2,n3,n4));
(%o8)                        eps4
                                 n1, n2, n3, n4
(%i9) grind(%)$
eps4[n1,n2,n3,n4]$
```

14.5.2 $\epsilon_{\lambda\mu\nu\sigma}$, EpsL(la,mu,nu,si), eps4L[la,mu,nu,si]

We let **EpsL[la,mu,nu,si]** and **eps4L[la,mu,nu,si]** stand for $\epsilon_{\lambda\mu\nu\sigma}$, and in the file **dirac3.mac** we have the code

```
declare (eps4L,antisymmetric)$
tellsimpafter ( eps4L[0,1,2,3], -1 )$
```

which results in the behavior

```
(%i10) eps4L[0,1,2,3];
(%o10)                               - 1
(%i11) eps4L[1,0,2,3];
(%o11)                                1
```

The is no corresponding definition for **EpsL**, which is just a symbolic place holder. The function **noncov** can be used to convert **EpsL(...)** into **eps4L[....]**.

```
(%i12) EpsL(0,1,2,3);
(%o12)                        EpsL(0, 1, 2, 3)
(%i13) EpsL(1,0,2,3);
(%o13)                        EpsL(1, 0, 2, 3)
(%i14) noncov (EpsL(0,1,2,3));
(%o14)                               - 1
(%i15) noncov (EpsL(1,0,2,3));
(%o15)                                1
(%i16) noncov (EpsL(n1,n2,n3,n4));
(%o16)                       eps4L
                                  n1, n2, n3, n4
(%i17) grind(%)$
eps4L[n1,n2,n3,n4]$
```

14.5.3 Contraction of a Product of Two Levi-Civita Tensors, Con, mcon

Four index contraction

Four index contraction of common indices using the summation convention, of a product of two Levi-Civita tensors in the usual notation is written

$$\epsilon^{\mu\nu\lambda\sigma}\,\epsilon_{\mu\nu\lambda\sigma} = -24, \qquad (14.44)$$

and is carried out with the dirac package using **Con**, **scon**, and **mcon**, **provided** you supply the four contraction indices at the end of the command.

Con, scon, mcon

The package contraction functions **Con**, **scon**, and **mcon** can be used for contraction of indices, with **Con** being a general starting point, but **mcon** being usually faster if provided the right args. The general contraction function **Con** calls **scon** if the elements to be contracted are all known symbolic entities, calls **mcon** if all the elements are explicit arrays or matrices, and calls **econ** for intermediate cases.

Contraction of **KD(n,n)** should produce the number **4**:

```
(%i18) Con(KD(n,n),n);
(%o18)                              4
(%i19) Con(KD(n1,n1));
(%o19)                              4
(%i20) scon(KD(n,n),n);
(%o20)                              4
(%i21) mcon(KD(n,n),n);
(%o21)                              4
```

More examples of contraction involving **KD**.

14.5.4 indexL

For symbolic contraction, **Con** and **scon** can find repeated indices if the index symbols appear on the list **indexL**. However, you can supply any explicit contraction index for contraction and the package then doesn't care about the **indexL** list.

```
(%i22) indexL;
(%o22) [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, la, mu, nu, rh, si, ta, al,
                                                                  be, ga, de, ep]
(%i23) Con(KD(n1,n1));
(%o23)                              4
(%i24) Con(KD(n,n));
              scon: no repeated index symbols supplied or found

(%o24)                          KD(n, n)
(%i25) Con(KD(n,n),n);
(%o25)                              4
(%i26) Con(Gm(n1,n2)*KD(n1,n3));
(%o26)                           Gm(n2, n3)
(%i27) Con(Gm(n1,n2)*KD(n1,n3),n1);
(%o27)                           Gm(n2, n3)
(%i28) scon(Gm(n1,n2)*KD(n1,n3));
(%o28)                           Gm(n2, n3)
(%i29) scon(Gm(n1,n2)*KD(n1,n3),n1);
(%o29)                           Gm(n2, n3)
```

But when using **mcon**, you must supply the contraction indices.

```
(%i30) mcon(gmet[n1,n2]*KD(n1,n3),n1);
(%o30)                           gmet
                                     n3, n2
(%i31) grind(%)$
gmet[n3,n2]$
(%i32) mcon(gmet[n1,n2]*KD(n1,n3));
find_eps4_prod:  no symbolic contraction indices found
 mcon: fatal error
(%o32)                              done
```

Returning to the four index contraction of a product of antisymmetric tensors, if **Con** detects a term which includes both **Eps** and **EpsL**, or both **eps4** and **eps4L**, **Con** will transfer the contraction job to **mcon**, after being sure the right arguments are in place.

Contraction of a product of Levi-Civita tensors is then ultimately done by **mcon**. You **must** supply the contraction indices with the call to **Con** to get **mcon** to do the job.

```
(%i33) trace(mcon);
(%o33)                             [mcon]
(%i34) Con(Eps(n1,n2,n3,n4)*EpsL(n1,n2,n3,n4),n1,n2,n3,n4);
1 Enter mcon [eps4              eps4L              , n1, n2, n3, n4]
                  n1, n2, n3, n4      n1, n2, n3, n4
1 Exit  mcon - 24
(%o34)                              - 24
(%i35) Con(eps4[n1,n2,n3,n4]*eps4L[n1,n2,n3,n4],n1,n2,n3,n4);
1 Enter mcon [eps4              eps4L              , n1, n2, n3, n4]
                  n1, n2, n3, n4      n1, n2, n3, n4
1 Exit  mcon - 24
(%o35)                              - 24
(%i36) Con(Eps(n1,n2,n3,n4)*EpsL(n1,n2,n3,n4));
1 Enter mcon [eps4              eps4L              ]
                  n1, n2, n3, n4      n1, n2, n3, n4
find_eps4_prod:  no symbolic contraction indices found
 mcon: fatal error
1 Exit  mcon done
(%o36)                              done
(%i37) untrace();
(%o37)                             [mcon]
(%i38) mcon(eps4[n1,n2,n3,n4]*eps4L[n1,n2,n3,n4],n1,n2,n3,n4);
(%o38)                              - 24
(%i39) econ(eps4[n1,n2,n3,n4]*eps4L[n1,n2,n3,n4],n1,n2,n3,n4);
(%o39)                              - 24
```

A straightforward multiple use of **sum** reproduces the value returned by **Con** (although our code does not use this brute force method):

```
(%i40) sum( sum( sum( sum(eps4[n1,n2,n3,n4]*eps4L[n1,n2,n3,n4],n1,0,3),n2,0,3),n3,0,3),n4,0,3);
(%o40)                                      - 24
```

Three index contraction of common indices, using the summation convention, of a product of two Levi-Civita tensors obeys

$$\epsilon^{\mu\nu\lambda\sigma}\,\epsilon_{\mu\nu\lambda\rho} = -6\,\delta^{\sigma}_{\rho}. \tag{14.45}$$

The coefficient -6 can be checked by contracting on σ and ρ, using (14.3), and finding agreement with the four index contraction result (14.44).

Using our package functions:

```
(%i41) Con(Eps(n1,n2,n3,n4)*EpsL(n1,n2,n3,m4),n1,n2,n3);
(%o41)                             - 6 kron_delta(m4, n4)
(%i42) Con(eps4[n1,n2,n3,n4]*eps4L[n1,n2,n3,m4],n1,n2,n3);
(%o42)                             - 6 kron_delta(m4, n4)
(%i43) mcon(eps4[n1,n2,n3,n4]*eps4L[n1,n2,n3,m4],n1,n2,n3);
(%o43)                             - 6 kron_delta(m4, n4)
```

Two index contraction of common indices, using the summation convention, of a product of two Levi-Civita tensors obeys

$$\epsilon^{\mu\nu\rho\sigma}\,\epsilon_{\mu\nu\gamma\delta} = -2\left(\delta^{\rho}_{\gamma}\,\delta^{\sigma}_{\delta} - \delta^{\rho}_{\delta}\,\delta^{\sigma}_{\gamma}\right) \tag{14.46}$$

Using our package functions: Using **mcon** with the needed product form **eps4*eps4L**:

```
(%i44) Con(Eps(n1,n2,n3,n4)*EpsL(n1,n2,m3,m4),n1,n2);
(%o44) 2 kron_delta(m3, n4) kron_delta(m4, n3) - 2 kron_delta(m3, n3) kron_delta(m4, n4)
(%i45) Con(eps4[n1,n2,n3,n4]*eps4L[n1,n2,m3,m4],n1,n2);
(%o45) 2 kron_delta(m3, n4) kron_delta(m4, n3) - 2 kron_delta(m3, n3) kron_delta(m4, n4)
(%i46) mcon(eps4[n1,n2,n3,n4]*eps4L[n1,n2,m3,m4],n1,n2);
(%o46) 2 kron_delta(m3, n4) kron_delta(m4, n3) - 2 kron_delta(m3, n3) kron_delta(m4, n4)
```

14.6 Explicit Gamma Matrix Algebra and Contraction Theorems: Con, mcon

The basic gamma matrix algebra relations are

$$\gamma^{\mu}\,\gamma^{\nu} = -\gamma^{\nu}\,\gamma^{\mu} + 2\,g^{\mu\,\nu}\,\mathbb{1} \tag{14.47}$$

for Lorentz indices μ and ν which can have values $(0, 1, 2, 3)$ and

$$\gamma^{5}\,\gamma^{\mu} = -\gamma^{\mu}\,\gamma^{5}. \tag{14.48}$$

14.6.1 Anti-commutator Function acomm

We can use our explicit matrix definitions to check these two relations (and to check our definitions). We use our anti-commutator function **acomm**, which can be used with matrices. We first confirm that the symbols **A** and **B** are not bound to anything:

```
(%i2) [A,B];
(%o2)                              [A, B]
```

and then apply **acomm**:

```
(%i3) acomm(A,B);
(%o3)                           B . A + A . B
```

We use the notation **Gam[mu]** (for **mu** equal to $0, 1, 2, 3$) for the matrix representation of γ^{μ} and **I4** for the explicit 4×4 unit matrix, and the array **gmet[mu,nu]** for the matrix representation of the metric tensor. We define explicit matrix functions **f** and **g** here:

```
(%i4) f (mu,nu) := acomm (Gam[mu],Gam[nu])$
(%i5) g (mu,nu) := 2*gmet[mu,nu]*I4$
(%i6) f(0,0);
                                [ 2  0  0  0 ]
                                [            ]
                                [ 0  2  0  0 ]
(%o6)                           [            ]
                                [ 0  0  2  0 ]
                                [            ]
                                [ 0  0  0  2 ]
(%i7) g(0,0);
                                [ 2  0  0  0 ]
                                [            ]
                                [ 0  2  0  0 ]
(%o7)                           [            ]
                                [ 0  0  2  0 ]
                                [            ]
                                [ 0  0  0  2 ]
(%i8) for n1:0 thru 3 do
           for n2:0 thru 3 do if f(n1,n2) # g(n1,n2) then print("false")$
(%i9)
```

To check the second relation, we can use the explicit matrix **Gam[5]**, representing γ^{5} and **Z4**, the 4×4 zero matrix.

```
(%i9) Gam[5];
                              [ - 1   0    0  0 ]
                              [                 ]
                              [  0   - 1   0  0 ]
(%o9)                         [                 ]
                              [  0    0    1  0 ]
                              [                 ]
                              [  0    0    0  1 ]
```

```
(%i10) Z4;
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%o10)                          [            ]
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%i11) for n1:0 thru 3 do
         if acomm(Gam[5], Gam[n1]) # Z4 then print ("false")$
(%i12)
```

Contraction Theorems

$$\gamma_\mu \, \gamma^\mu = 4 \, \mathbb{1} \tag{14.49}$$

Using either **Con** or **mcon**, we get (note that **G(1)** is the **symbolic** unit matrix):

```
(%i12) Con(G(mu,mu));
(%o12)                              4 G(1)
(%i13) Con(G(mu,mu),mu);
(%o13)                              4 G(1)
(%i14) Con(Gam[mu]^^2,mu);
                                [ 4  0  0  0 ]
                                [            ]
                                [ 0  4  0  0 ]
(%o14)                          [            ]
                                [ 0  0  4  0 ]
                                [            ]
                                [ 0  0  0  4 ]
(%i15) Con(Gam[mu] . Gam[mu],mu);
                                [ 4  0  0  0 ]
                                [            ]
                                [ 0  4  0  0 ]
(%o15)                          [            ]
                                [ 0  0  4  0 ]
                                [            ]
                                [ 0  0  0  4 ]
(%i16) mcon(Gam[mu]^^2,mu);
                                [ 4  0  0  0 ]
                                [            ]
                                [ 0  4  0  0 ]
(%o16)                          [            ]
                                [ 0  0  4  0 ]
                                [            ]
                                [ 0  0  0  4 ]
(%i17) mcon(Gam[mu] . Gam[mu],mu);
                                [ 4  0  0  0 ]
                                [            ]
                                [ 0  4  0  0 ]
(%o17)                          [            ]
                                [ 0  0  4  0 ]
                                [            ]
                                [ 0  0  0  4 ]
```

$$\gamma_\mu \, \gamma^\nu \, \gamma^\mu = -2 \, \gamma^\nu \tag{14.50}$$

We can again use either **Con** or **mcon** contraction functions to check this relation. Note that in the definition of **f(n2)** below, **n1** is a dummy contraction variable.

```
(%i18) Con(G(mu,nu,mu));
(%o18)                             - 2 G(nu)
(%i19) Con(G(mu,nu,mu),mu);
(%o19)                             - 2 G(nu)
```

```
(%i20) f(n2) := mcon (Gam[n1] . Gam[n2] . Gam[n1], n1)$
(%i21) g(n2) := -2*Gam[n2]$
(%i22) is(equal(f(0),g(0)));
(%o22)                               true
(%i23) for mu:0 thru 3 do
          if f(mu) # g(mu) then print ("false")$
(%i24)
```

$$\gamma_\mu \gamma^\nu \gamma^\lambda \gamma^\mu = 4\, g^{\nu\,\lambda}\, \mathbb{1} \tag{14.51}$$

Note that **G(1)** is a symbolic representation of the unit 4×4 matrix and **I4** is the explicit unit matrix.

```
(%i24) Con (G (mu,nu,la,mu));
(%o24)                        4 G(1) Gm(la, nu)
(%i25) Con (G (mu,nu,la,mu),mu);
(%o25)                        4 G(1) Gm(la, nu)
(%i26) f(n2,n3) := mcon (Gam[n1] . Gam[n2] . Gam[n3] . Gam[n1],n1)$
(%i27) g(n2,n3) := 4*gmet[n2,n3]*I4$
(%i28) is(equal(f(0,0),g(0,0)));
(%o28)                               true
(%i29) for mu:0 thru 3 do
           for nu:0 thru 3 do
              if f(mu,nu) # g(mu,nu) then print ("false")$
(%i30)
```

A general contraction pattern for the case that the pair of contracting gamma matrices surround an odd number (3, 5, ...) of gamma matrices: we get the order reversed times (-2).

$$\gamma_\mu \gamma^\nu \gamma^\lambda \gamma^\sigma \gamma^\mu = -2\, \gamma^\sigma \gamma^\lambda \gamma^\nu \tag{14.52}$$

```
(%i30) Con (G(mu,nu,la,si,mu));
(%o30)                        - 2 G(si, la, nu)
(%i31) Con (G(mu,nu,la,si,mu),mu);
(%o31)                        - 2 G(si, la, nu)
(%i32) f(n2,n3,n4) := mcon (Gam[n1] . Gam[n2] . Gam[n3] . Gam[n4] . Gam[n1],n1)$
(%i33) g(n2,n3,n4) :=  -2*Gam[n4] . Gam[n3] . Gam[n2]$
(%i34) is(equal(f(0,0,0),g(0,0,0)));
(%o34)                               true
(%i35) for n2:0 thru 3 do
         for n3:0 thru 3 do
            for n4:0 thru 3 do
               if f(n2,n3,n4) # g(n2,n3,n4) then print ("false")$
(%i36)
```

A general contraction pattern for the case that the pair of contracting gamma matrices surround an even number (4, 6, ...) of gamma matrices: we get two terms with the first term bringing the last gamma to the front, and the second term being the new first term in reversed order, all multiplied by (2).

$$\gamma_\mu \gamma^\lambda \gamma^\nu \gamma^\rho \gamma^\sigma \gamma^\mu = 2\, (\gamma^\sigma \gamma^\lambda \gamma^\nu \gamma^\rho + \gamma^\rho \gamma^\nu \gamma^\lambda \gamma^\sigma) \tag{14.53}$$

```
(%i36) Con (G(mu,la,nu,rh,si,mu));
(%o36)            2 G(si, la, nu, rh) + 2 G(rh, nu, la, si)
```

14.7 Trace Theorems for Gamma Matrices: tr, Gtr, mat_trace, m_tr

The trace of the product of an **odd** number of gamma matrices is **zero**.

```
(%i2) tr(mu);
(%o2)                              0
(%i3) tr(mu,nu,la);
(%o3)                              0
(%i4) Gtr(G(mu));
(%o4)                              0
(%i5) Gtr(G(mu,nu,la));
(%o5)                              0
(%i6) f(mu) := mat_trace (Gam[mu])$
(%i7) map ('f,[0,1,2,3]);
(%o7)                        [0, 0, 0, 0]
```

For the trace of a product of an **even** number of gamma matrices (including zero) is based on a number of relations.

The trace of the unit matrix is 4.

$$Tr(\mathbb{1}) = 4 \tag{14.54}$$

```
(%i8) tr(1);
(%o8)                              4
(%i9) Gtr (G(1));
(%o9)                              4
(%i10) mat_trace(I4);
(%o10)                             4
(%i11) m_tr(1);
(%o11)                             4
```

The function **`m_tr`** (used above) allows one to write matrix and momenta arguments in the same form as when using **`tr`**, but translates the arguments into explicit matrices and then calls **`mat_trace`**.

The trace of a product of two gamma matrices is given by:

$$Tr(\gamma^{\mu}\,\gamma^{\nu}) = 4\,g^{\mu\,\nu} \tag{14.55}$$

Note that **`Gm(mu,nu)`** is the symbolic metric tensor and **`gmet[mu,nu]`** is the explicit array of metric tensor values.

```
(%i12) tr (mu,nu);
(%o12)                       4 Gm(mu, nu)
(%i13) Gtr (G(mu,nu));
(%o13)                       4 Gm(mu, nu)
(%i14) f(mu,nu) := mat_trace(Gam[mu] . Gam[nu])$
(%i15) f(0,0);
(%o15)                             4
(%i16) for n1:0 thru 3 do
          for n2:0 thru 3 do
            if f(n1,n2) # 4*gmet[n1,n2] then print ("false")$
(%i17)
```

$$Tr(\gamma^{\mu}\,\gamma^{\nu}\,\gamma^{\lambda}\,\gamma^{\sigma}) = 4\,(g^{\mu\,\nu}\,g^{\lambda\,\sigma} - g^{\lambda\,\mu}\,g^{\nu\,\sigma} + g^{\mu\,\sigma}\,g^{\lambda\,\nu}) \tag{14.56}$$

```
(%i17) tr(mu,nu,la,si);
(%o17) (- 4 Gm(la, mu) Gm(nu, si)) + 4 Gm(la, nu) Gm(mu, si)
                                               + 4 Gm(la, si) Gm(mu, nu)
```

For the trace of a product of an even number of gamma matrices, we can use the general formula for the reduction of a trace of a product of **`(2 N )`** matrices to a sum involving the traces of products of **`(2 N - 2)`** matrices.

Let's use a more efficient notation here:

$$tr(n1, n2, n3, ..., 2\,N) \leftrightarrow Tr(\gamma^{n_1}\,\gamma^{n_2}\,\gamma^{n_3}\,\ldots\gamma^{2\,N}) \tag{14.57}$$

The reduction formula is then (see, for example, Barger and Phillips, p. 550 or Kaku, p. 752):

$$\begin{aligned} tr(n1,n2,n3,...,2N) &= g^{n_1\,n_2}\,tr(n3,n4,n5,...,2N) \\ &- g^{n_1\,n_3}\,tr(n2,n4,n5,...,2N) \\ &+ g^{n_1\,n_4}\,tr(n2,n3,n5,...,2N) \\ &- \ldots + g^{n_1\,n_{2N}}\,tr(n3,n4,n5,...,2N-1) \end{aligned}$$

For example, if $2N = 6$ we have

$$\begin{aligned} tr(n1,n2,n3,n4,n5,n6) &= g^{n_1\,n_2}\,tr(n3,n4,n5,n6) \\ &- g^{n_1\,n_3}\,tr(n2,n4,n5,n6) + g^{n_1\,n_4}\,tr(n2,n3,n5,n6) \\ &- g^{n_1\,n_5}\,tr(n2,n3,n4,n6) + g^{n_1\,n_6}\,tr(n2,n3,n4,n5) \end{aligned}$$

Traces Involving γ^5

Since $\boldsymbol{\gamma}^{\mathbf{5}} = i\,\boldsymbol{\gamma}^{\mathbf{0}}\,\boldsymbol{\gamma}^{\mathbf{1}}\,\boldsymbol{\gamma}^{\mathbf{2}}\,\boldsymbol{\gamma}^{\mathbf{3}}$ is proportional to the product of an **even** number of gamma matrices, the trace of the product of gamma5 by an odd number of gamma matrices is zero. For example

$$Tr(\boldsymbol{\gamma}^{\mathbf{5}}\,\boldsymbol{\gamma}^{\boldsymbol{\mu}}) = 0 \tag{14.58}$$

```
(%i18) tr(G5,mu);
(%o18)                              0
(%i19) Gtr (G(G5,mu));
(%o19)                              0
(%i20) f(mu) := mat_trace (Gam[5] . Gam[mu])$
(%i21) map ('f,[0,1,2,3]);
(%o21)                        [0, 0, 0, 0]
```

and

$$Tr(\boldsymbol{\gamma}^{\mathbf{5}}\,\boldsymbol{\gamma}^{\boldsymbol{\mu}}\,\boldsymbol{\gamma}^{\boldsymbol{\nu}}\,\boldsymbol{\gamma}^{\boldsymbol{\lambda}}) = 0 \tag{14.59}$$

```
(%i22) tr(G5,mu,nu,la);
(%o22)                              0
(%i23) Gtr (G(G5,mu,nu,la));
(%o23)                              0
(%i24) f(mu,nu,la) := mat_trace(Gam[5] . Gam[mu] . Gam[nu] . Gam[la])$
(%i25) for n1:0 thru 3 do
          for n2:0 thru 3 do
             for n3:0 thru 3 do
                if f(n1,n2,n3) # 0 then print("false")$
(%i26)
```

On the other hand, when $\boldsymbol{\gamma}^{\mathbf{5}}$ is multiplied by an **even** number of gamma matrices, we have (first for multiplication by zero gamma matrices):

$$Tr(\boldsymbol{\gamma}^{\mathbf{5}}) = 0 \tag{14.60}$$

```
(%i26) tr(G5);
(%o26)                              0
(%i27) Gtr (G(G5));
(%o27)                              0
(%i28) mat_trace (Gam[5]);
(%o28)                              0
(%i29) m_tr(G5);
(%o29)                              0
```

and next for multiplication by two gamma matrices:

$$Tr(\gamma^5\,\gamma^\mu\,\gamma^\nu) = 0 \tag{14.61}$$

```
(%i30) tr(G5,mu,nu);
(%o30)                                 0
(%i31) Gtr(G(G5,mu,nu));
(%o31)                                 0
(%i32) f(mu,nu) := mat_trace(Gam[5] . Gam[mu] . Gam[nu])$
(%i33) for n1:0 thru 3 do
          for n2:0 thru 3 do
            if f(n1,n2) # 0 then print("false")$
(%i34)
```

and next for multiplication by four gamma matrices:

$$Tr(\gamma^5\,\gamma^\mu\,\gamma^\nu\,\gamma^\lambda\,\gamma^\sigma) = -4\,i\,\epsilon^{\mu\,\nu\,\lambda\,\sigma} \tag{14.62}$$

```
(%i34) tr(G5,mu,nu,la,si);
(%o34)                    - 4 %i Eps(mu, nu, la, si)
(%i35) noncov(%);
(%o35)                     - 4 %i eps4
                                      la, mu, nu, si
(%i36) grind(%)$
-4*%i*eps4[la,mu,nu,si]$
(%i37) Gtr(G(G5,mu,nu,la,si));
(%o37)                    - 4 %i Eps(mu, nu, la, si)
(%i38) f(mu,nu,la,si):= mat_trace(Gam[5] . Gam[mu] . Gam[nu] . Gam[la] . Gam[si]) +
          4*%i*eps4[mu,nu,la,si]$
(%i39) f(0,1,2,3);
(%o39)                                 0
(%i40) for n1:0 thru 3 do
          for n2:0 thru 3 do
             for n3:0 thru 3 do
               for n4:0 thru 3 do
                 if f(n1,n2,n3,n4) # 0 then print("false")$
(%i41)
```

For the cases that γ^5 is multiplied by six or eight gamma matrices, we have used the Chisholm-Kahane gamma matrix identity which replaces a product of three Dirac gamma matrices by a series of four terms. With our (Peskin/Schroeder) conventions, $\epsilon^{0123} = +1$ and $\gamma^5 = +i\,\gamma^0\,\gamma^1\,\gamma^2\,\gamma^3$, that identity is (using the summation convention for the repeated index λ):

$$\gamma^\mu\,\gamma^\nu\,\gamma^\rho = g^{\mu\,\nu}\,\gamma^\rho - g^{\mu\,\rho}\,\gamma^\nu + g^{\nu\,\rho}\,\gamma^\mu - i\,\epsilon^{\mu\nu\rho\lambda}\,\gamma^5\,\gamma_\lambda \tag{14.63}$$

Here is the symbolic calculation of the trace of the product of **G5** with six gamma matrices.

```
(%i41) tr6:tr(G5,n1,n2,n3,n4,n5,n6);
(%o41) (- 4 %i Eps(n1, n2, n3, n4) Gm(n5, n6))
 + 4 %i Eps(n1, n2, n3, n5) Gm(n4, n6) - 4 %i Eps(n1, n2, n3, n6) Gm(n4, n5)
 - 4 %i Gm(n1, n2) Eps(n3, n4, n5, n6) + 4 %i Gm(n1, n3) Eps(n2, n4, n5, n6)
 - 4 %i Eps(n1, n4, n5, n6) Gm(n2, n3)
(%i42) noncov(tr6);
(%o42) (- 4 %i eps4               gmet      )
                   n1, n2, n3, n4     n5, n6
 + 4 %i eps4               gmet      - 4 %i eps4               gmet
            n1, n2, n3, n5     n4, n6            n1, n2, n3, n6     n4, n5
 - 4 %i gmet       eps4                + 4 %i gmet       eps4
            n1, n2     n3, n4, n5, n6            n1, n3     n2, n4, n5, n6
 - 4 %i eps4               gmet
            n1, n4, n5, n6     n2, n3
```

Here we calculate a trace of gamma5 multiplied by eight gamma matrices:

```
(%i43) tr8:tr(G5,n1,n2,n3,n4,n5,n6,n7,n8);
(%o43) (- 4 %i Eps(n1, n2, n3, n4) Gm(n5, n6) Gm(n7, n8))
 + 4 %i Eps(n1, n2, n3, n5) Gm(n4, n6) Gm(n7, n8)
 - 4 %i Eps(n1, n2, n3, n6) Gm(n4, n5) Gm(n7, n8)
 - 4 %i Gm(n1, n2) Eps(n3, n4, n5, n6) Gm(n7, n8)
 + 4 %i Gm(n1, n3) Eps(n2, n4, n5, n6) Gm(n7, n8)
 - 4 %i Eps(n1, n4, n5, n6) Gm(n2, n3) Gm(n7, n8)
 + 4 %i Eps(n1, n2, n3, n4) Gm(n5, n7) Gm(n6, n8)
 - 4 %i Eps(n1, n2, n3, n5) Gm(n4, n7) Gm(n6, n8)
 + 4 %i Eps(n1, n2, n3, n7) Gm(n4, n5) Gm(n6, n8)
 + 4 %i Gm(n1, n2) Eps(n3, n4, n5, n7) Gm(n6, n8)
 - 4 %i Gm(n1, n3) Eps(n2, n4, n5, n7) Gm(n6, n8)
 + 4 %i Eps(n1, n4, n5, n7) Gm(n2, n3) Gm(n6, n8)
 - 4 %i Eps(n1, n2, n3, n4) Gm(n5, n8) Gm(n6, n7)
 + 4 %i Eps(n1, n2, n3, n5) Gm(n4, n8) Gm(n6, n7)
 - 4 %i Eps(n1, n2, n3, n8) Gm(n4, n5) Gm(n6, n7)
 - 4 %i Gm(n1, n2) Eps(n3, n4, n5, n8) Gm(n6, n7)
 + 4 %i Gm(n1, n3) Eps(n2, n4, n5, n8) Gm(n6, n7)
 - 4 %i Eps(n1, n4, n5, n8) Gm(n2, n3) Gm(n6, n7)
 + 4 %i Eps(n1, n2, n3, n6) Gm(n4, n7) Gm(n5, n8)
 - 4 %i Eps(n1, n2, n3, n7) Gm(n4, n6) Gm(n5, n8)
 - 4 %i Eps(n1, n2, n3, n6) Gm(n4, n8) Gm(n5, n7)
 + 4 %i Eps(n1, n2, n3, n8) Gm(n4, n6) Gm(n5, n7)
 - 4 %i Gm(n1, n2) Gm(n3, n4) Eps(n5, n6, n7, n8)
 + 4 %i Gm(n1, n3) Gm(n2, n4) Eps(n5, n6, n7, n8)
 - 4 %i Gm(n1, n4) Gm(n2, n3) Eps(n5, n6, n7, n8)
 + 4 %i Eps(n1, n2, n3, n7) Gm(n4, n8) Gm(n5, n6)
 - 4 %i Eps(n1, n2, n3, n8) Gm(n4, n7) Gm(n5, n6)
 + 4 %i Gm(n1, n2) Gm(n3, n5) Eps(n4, n6, n7, n8)
 - 4 %i Gm(n1, n3) Gm(n2, n5) Eps(n4, n6, n7, n8)
 + 4 %i Gm(n1, n5) Gm(n2, n3) Eps(n4, n6, n7, n8)
 - 4 %i Gm(n1, n2) Eps(n3, n6, n7, n8) Gm(n4, n5)
 + 4 %i Gm(n1, n3) Eps(n2, n6, n7, n8) Gm(n4, n5)
 - 4 %i Eps(n1, n6, n7, n8) Gm(n2, n3) Gm(n4, n5)
```

The present package function **simp_tr1** (which controls **tr1** via the lisp code in **simplifying-new.lisp**) does not return an evaluation for the case of an even number equal to ten or more gamma matrices multiplying **G5**. In the example, we use **Mindex** to add **n11** and **n12** to the list **indexL**, a list used by the package to recognise canonical Lorentz indices.

```
(%i44) tr10:tr(G5,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10);
    simp_tr1: present version of tr only returns nonzero gamma5 traces for

                     G5 times a product of 4,6,or 8 gammas

(%o44)                               0
(%i45) Mindex(n11,n12);
(%o45)                             done
(%i46) indexL;
(%o46) [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, la, mu, nu, rh, si, ta, al,
                                                      be, ga, de, ep, n11, n12]
(%i47) tr11:tr(G5,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11);
(%o47)                               0
(%i48) tr12:tr(G5,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12);
    simp_tr1: present version of tr only returns nonzero gamma5 traces for

                     G5 times a product of 4,6,or 8 gammas

(%o48)                               0
```

More compact expressions for gamma 5 traces (compared to the classical methods used here) can be found in the paper:

```
A new method for calculation of traces of Dirac gamma-matrices
  in Minkowski space, by Alexander L. Bondarev,
    https://arxiv.org/abs/hep-ph/0504223
```

15 Heaviside-Lorentz Units Compared with Gaussian Units

You can compare our discussion with the webpage
`https://en.wikipedia.org/wiki/Lorentz%E2%80%93Heaviside_units`.

15.1 Gaussian CGSE Units

In the CGSE system (absolute electrostatic system: gram, cm, sec, force in dynes, charge in esu) the magnitude of the electron charge in esu or "statcoulomb" (statC) is

$$|q_e| \approx 4.803 \times 10^{-10}\text{esu}. \tag{15.1}$$

We can replace the electron's electric charge (in Gaussian units) by the fine structure constant using

$$\alpha = \frac{e_G^2}{\hbar c} \approx \frac{1}{137}. \tag{15.2}$$

The electrostatic unit of charge is defined such that the magnitude of the force (in dynes) between charges e_1 and e_2 separated by a distance r is

$$|\vec{F}| = \frac{|e_1\, e_2|}{r^2} \tag{15.3}$$

The implies that

$$[e] = \left([Fr^2]\right)^{1/2} = \left(\frac{\text{g. cm}}{\text{sec}^2}\,\text{cm}^2\right)^{1/2} = \text{g}^{1/2}\,\text{cm}^{3/2}\,\text{sec}^{-1}. \tag{15.4}$$

The mutual potential energy (in ergs) of two electric charges e_1 and e_2 a distance r apart is

$$U = \frac{e_1 e_2}{r}, \tag{15.5}$$

under the assumption that the potential energy should be zero when the charges are at an infinite distance apart.

In the SI system of units, a much larger unit of charge is used, called the *coulomb* :

$$1 \text{ coulomb} = 1 \text{ C} \approx 3 \times 10^9 \text{ statC}. \tag{15.6}$$

We also have

$$|q_e| \approx 1.6 \times 10^{-19}\,\text{C} \approx 4.8 \times 10^{-10}\,\text{esu}, \tag{15.7}$$

and the dimensionless fine structure constant $\alpha \approx 1/137$ is, in terms of the electron charge q_e,

$$\alpha = \left(\frac{q_e^2}{\hbar c}\right)_{CGSE} = \left(\frac{q_e^2}{4\pi\epsilon_0\,\hbar c}\right)_{SI}. \tag{15.8}$$

Returning to the CGSE units description, the force exerted on a charge e by another charge e_1 can be written

$$\vec{F} = e\,\vec{E}, \tag{15.9}$$

where $\vec{E}$ is a vector independent of the charge e called the *electric field* due to the charge e_1. Its magnitude is

$$|\vec{E}| = \frac{|e_1|}{r^2} \tag{15.10}$$

and is directed along the line joining the two charges.

The mutual potential energy of the charges e and e_1 separated by a distance r is

$$U = \frac{e\,e_1}{r} = e\,\phi \tag{15.11}$$

where the *potential* ϕ of the electric field due to the charge e_1 is

$$\phi = \frac{e_1}{r}, \quad \text{such that} \quad \vec{E} = -\nabla\phi. \tag{15.12}$$

The unit of electric potential (called the *statvolt*) has dimensions

$$[\phi] = \left[\frac{U}{e}\right] = \mathrm{g}^{1/2}\,\mathrm{cm}^{1/2}\,\mathrm{sec}^{-1} \tag{15.13}$$

In the SI system a unit approximately 300 times smaller is used, called the *volt* .

$$1\ \mathrm{V} \approx 1/300\ \text{statvolt}. \tag{15.14}$$

If a charge of one coulomb moves between two points whose potentials differ by one volt, then the work done by the field forces is

$$\mathrm{W} = \Delta\mathrm{U} = \mathrm{q}\,\Delta\phi = \left(3\times10^9\mathrm{esu}\right)\left(\frac{1}{300}\text{statvolt}\right) = 10^7\ \text{ergs} = 1\ \text{Joule} \tag{15.15}$$

Thus

$$1\ \mathrm{C\ V} = 1\ \mathrm{J}. \tag{15.16}$$

Quoting Wikipedia's page under the name "statvolt,"

> The statvolt is a unit of voltage and electrical potential used in the esu-cgs and gaussian system of units. The conversion to the SI system is exactly 1 statvolt $= 299.792458$ volts. (The conversion factor 299.792458 is simply the numerical value of the speed of light in m/s divided by 10^6) The statvolt is also defined in the cgs system as 1 erg/esu. It is a useful unit for electromagnetism because, in a vacuum, an electric field of one statvolt/cm has the same energy density as a magnetic field of one gauss.

We will use a subscript G to denote quantities expressed in CGSE units. The electric field $\vec{E}_G$ has the same mechanical units (gram,cm, sec) as the magnetic field $\vec{B}_G$ although we use statvolt cm^{-1} to refer to the electric field and gauss to refer to the magnetic field. In the CGSE system of units, Maxwell's equations are

$$\begin{aligned}
&\nabla\cdot\vec{B}_G = 0\\
&\nabla\cdot\vec{E}_G = 4\,\pi\,\rho_G\\
&\nabla\times\vec{E}_G + \frac{1}{c}\,\frac{\partial\vec{B}_G}{\partial t} = 0\\
&\nabla\times\vec{B}_G - \frac{1}{c}\,\frac{\partial\vec{E}_G}{\partial t} = \frac{4\,\pi}{c}\,\vec{J}_G
\end{aligned}$$

The Lorentz force (dynes) on a particle of charge q_G (esu) is

$$\vec{F} = q_G\left(\vec{E}_G + \frac{1}{c}\,\vec{v}\times\vec{B}_G\right). \tag{15.17}$$

The electromagnetic field energy (ergs) is

$$U_{\text{fields}} = \frac{1}{8\pi}\int d^3x\left(\vec{E}_G^2 + \vec{B}_G^2\right). \tag{15.18}$$

The Poynting vector (ergs per centimeter squared per second) is

$$\vec{S} = \frac{c}{4\pi}\,\vec{E}_G\times\vec{B}_G. \tag{15.19}$$

The electromagnetic field momentum is

$$\vec{P}_{\text{fields}} = \frac{1}{4\pi c}\int d^3x\left(\vec{E}_G\times\vec{B}_G\right). \tag{15.20}$$

15.2 Heaviside-Lorentz Units

In Heaviside-Lorentz cgs units (cm., gram, sec, force in dynes) we hide factors of 4π by using the relations

$$q_{HL} = \sqrt{4\pi}\, q_G, \quad \rho_{HL} = \sqrt{4\pi}\, \rho_G, \quad \vec{J}_{HL} = \sqrt{4\pi}\, \vec{J}_G$$
$$\vec{E}_G = \sqrt{4\pi}\, \vec{E}_{HL}$$
$$\vec{B}_G = \sqrt{4\pi}\, \vec{B}_{HL}$$

We then can express the dimensionless fine structure contant α in terms of the square of the electron charge in HL units,

$$\alpha = \frac{e_{HL}^2}{4\pi\hbar c}. \tag{15.21}$$

Maxwell's equations become

$$\nabla\cdot\vec{B}_{HL} = 0$$
$$\nabla\cdot\vec{E}_{HL} = \rho_{HL}$$
$$\nabla\times\vec{E}_{HL} + \frac{1}{c}\frac{\partial\vec{B}_{HL}}{\partial t} = 0$$
$$\nabla\times\vec{B}_{HL} - \frac{1}{c}\frac{\partial\vec{E}_{HL}}{\partial t} = \frac{1}{c}\vec{J}_{HL}$$

The Lorentz force (dynes) on a particle of charge q_{HL} (esu) is

$$\vec{F} = q_{HL}\left(\vec{E}_{HL} + \frac{1}{c}\vec{v}\times\vec{B}_{HL}\right). \tag{15.22}$$

The electromagnetic field energy (ergs) is

$$U_{\text{fields}} = \frac{1}{2}\int d^3x\left(\vec{E}_{HL}^2 + \vec{B}_{HL}^2\right). \tag{15.23}$$

The Poynting vector (ergs per centimeter squared per second) is

$$\vec{S} = c\,\vec{E}_{HL}\times\vec{B}_{HL}. \tag{15.24}$$

The electromagnetic field momentum is

$$\vec{P}_{\text{fields}} = \frac{1}{c}\int d^3x\left(\vec{E}_{HL}\times\vec{B}_{HL}\right). \tag{15.25}$$

In **both** Gaussian and Heaviside-Lorentz systems we have the relations

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\vec{J} = 0$$
$$\vec{E} = -\nabla\phi - \frac{1}{c}\frac{\partial\vec{A}}{\partial t}$$
$$\vec{B} = \nabla\times\vec{A},$$

provided

$$\phi_G = \sqrt{4\pi}\,\phi_{HL}$$
$$\vec{A}_G = \sqrt{4\pi}\,\vec{A}_{HL}.$$

As a numerical example, suppose $\mathcal{E}_G = 10\,\text{statvolt}/\text{cm}$ and $q_G = 5\,\text{esu}$. Then the electric force is $F = q_G\,\mathcal{E}_G = 50\,\text{ergs}/\text{cm} = 50\,\text{dynes}$ Using HL units, $q_{HL} = \sqrt{4\pi}\,q_G = 17.725\,\text{esu}$, and $\mathcal{E}_{HL} = \frac{\mathcal{E}_G}{\sqrt{4\pi}} = 2.821\,\text{statvolt}/\text{cm}$. Using HL units, $F = q_{HL}\,\mathcal{E}_{HL} = 50\,\text{dynes}$.

16 Natural Units

In particle physics it is convenient (and conventional) to use a system of units (called "natural units") in which the formulas used are formally the same as if we set $\hbar = 1$ and $c = 1$ in the corresponding Heaviside-Lorentz (HL) formula.

Let $[A]$ stand for the units of A.

Velocity

With v a velocity in ordinary units and $\bar{v}$ the corresponding velocity in n.u., we define

$$v = \bar{v}c, \quad [\bar{v}] = 1. \tag{16.1}$$

In other words, " $\bar{v}$ is dimensionless."

Energy

The energy E of a particle of mass $m > 0$, which is at rest, is $m\,c^2$, so we set

$$E = \bar{E}\,c^2, \quad [\bar{E}] = [m]. \tag{16.2}$$

Linear Momentum

Since the units of linear momentum are those of the product of a mass by a velocity,

$$[p] = [m\,c], \tag{16.3}$$

we set

$$p = \bar{p}\,c \quad [\bar{p}] = [m]. \tag{16.4}$$

Angular Momentum

An angular momentum J has the same units as $\hbar$, so

$$J = \bar{J}\,\hbar, \quad [\bar{J}] = 1, \tag{16.5}$$

so $\bar{J}$ is "dimensionless."

Spatial Distance

A natural unit of length we can construct from the parameters $\hbar$, c, and m is the (reduced) Compton wavelenth of a particle of mass m

$$\lambda_c = \frac{\hbar}{mc}. \tag{16.6}$$

Let x be a length in conventional units and $\bar{x}$ be the corresponding length in natural units (n.u.). The position - linear momentum uncertainty relation in ordinary units (o.u.) is

$$\Delta x\,\Delta p \geq \frac{\hbar}{2}. \tag{16.7}$$

Hence

$$[x] = \left[\frac{\hbar}{p}\right] = \left[\frac{\hbar}{m\,c}\right], \tag{16.8}$$

so we define

$$x = \bar{x}\,\frac{\hbar}{c}, \quad [\bar{x}] = \left[\frac{1}{m}\right]. \tag{16.9}$$

Then in natural units the position - linear momentum relation becomes

$$\Delta\bar{x}\,\Delta\bar{p} \geq \frac{1}{2}. \tag{16.10}$$

Gradient

With $\overline{\nabla}$ the gradient operator in natural units

$$\nabla = \frac{c}{\hbar}\,\overline{\nabla}, \quad [\overline{\nabla}] = \left[\frac{1}{\bar{x}}\right] = [m]. \tag{16.11}$$

Time

A natural unit of time is the number of seconds required for a photon to traverse the Compton wavelength λ_c of a particle of mass m.

$$\Delta t = \frac{\lambda_c}{c} = \frac{\hbar}{mc^2}, \tag{16.12}$$

which motivates the definition

$$t = \bar{t}\frac{\hbar}{c^2}, \quad [\bar{t}] = \left[\frac{1}{m}\right] \tag{16.13}$$

consistent with $\bar{v} = \bar{x}/\bar{t}$ being dimensionless.

Kronecker Delta Function

Since

$$\int dw\,\delta(w) = \int d\bar{w}\,\delta(\bar{w}) = 1, \tag{16.14}$$

we have

$$[\delta(\bar{w})] = \left[\frac{1}{\bar{w}}\right]. \tag{16.15}$$

Hence

$$[\delta(\bar{x})] = \left[\frac{1}{\bar{x}}\right] = [m] \tag{16.16}$$

and

$$[\delta(\bar{p})] = \left[\frac{1}{\bar{p}}\right] = \left[\frac{1}{m}\right] \tag{16.17}$$

Force

Let U be a potential energy in o.u. and $\overline{U}$ be the corresponding potential energy in n.u. with $U = \overline{U}\,c^2$, (and, of course, $[\overline{U}] = [m]$). If F is a force in ordinary units and $\overline{F}$ is the corresponding force in n.u., then

$$F_x = -\frac{\partial U}{\partial x} = -\frac{\partial \bar{U}}{\partial \bar{x}}\,\frac{c^3}{\hbar} = \overline{F}_x\,\frac{c^3}{\hbar}. \tag{16.18}$$

$$[F] = [E/x] = \left[\frac{mc^2}{\frac{\hbar}{mc}}\right] = \left[\frac{m^2c^3}{\hbar}\right] \tag{16.19}$$

so

$$F = \overline{F}\,\frac{c^3}{\hbar}, \quad [\overline{F}] = [m^2]. \tag{16.20}$$

Number Density

Let n be the number density and V be a specified volume in ordinary units, with $\bar{n}$ and $\overline{V}$ the corresponding quantities in n.u.. With N particles in V,

$$n = \frac{N}{V} = \frac{N}{\overline{V}\,\left(\frac{\hbar}{c}\right)^3} = \bar{n}\,\left(\frac{c}{\hbar}\right)^3. \tag{16.21}$$

We have used

$$\bar{n} = \frac{N}{\overline{V}}, \quad \text{so} \quad [\bar{n}] = [m^3] \tag{16.22}$$

Mass Density

Let ρ_m be the mass density and V a given volume in o.u., and let $\bar{\rho}_m$ and $\overline{V}$ be the corresponding quantities in n.u. With mass M in V,

$$\rho_m = \frac{M}{V} = \frac{M}{\overline{V}\,\left(\frac{\hbar}{c}\right)^3} = \bar{\rho}_m\,\left(\frac{c}{\hbar}\right)^3 \tag{16.23}$$

We have used

$$\bar{\rho}_m = \frac{M}{\overline{V}}, \quad \text{so} \quad [\bar{\rho}_m] = [m^4] \tag{16.24}$$

Universal Constant of Gravitation

Using

$$U = \frac{Gm^2}{r}, \quad \overline{U} = \frac{\overline{G}m^2}{\bar{r}}, \quad U = \overline{U}c^2, \tag{16.25}$$

we get

$$G = \overline{G}\,\hbar\,c \tag{16.26}$$

and since $[\bar{r}\overline{U}] = 1$,

$$[\overline{G}] = \left[\frac{\bar{r}\,\overline{U}}{m^2}\right] = \left[\frac{1}{m^2}\right] \tag{16.27}$$

Electric Charge

With the electric charge in natural units being $\bar{q}$ and the electric charge in HL units being q,

$$q = \sqrt{\hbar c}\,\bar{q}, \quad [\bar{q}] = 1. \tag{16.28}$$

The relation between the fine structure constant α and the value of the electron charge in natural units then becomes

$$\alpha = \frac{e_{\text{nu}}^2}{4\pi}. \tag{16.29}$$

Electric Charge Density

With ρ the HL electric charge density and q the HL charge, let $\bar{\rho}$ be the electric charge density in natural units with $\bar{q}$ the electric charge in natural units. Then

$$\rho = \frac{q}{V} = \frac{\bar{q}\,\sqrt{\hbar c}}{\overline{V}\,(\hbar/c)^3} = \frac{\bar{q}}{\overline{V}}\sqrt{\frac{c^7}{\hbar^5}} \tag{16.30}$$

So

$$\rho = \bar{\rho}\sqrt{\frac{c^7}{\hbar^5}}, \quad [\bar{\rho}] = \left[\frac{\bar{q}}{\overline{V}}\right] = [m^3]. \tag{16.31}$$

Electric Current Density

With J the HL electric **current** density, ρ the HL electric **charge** density,

$$J = \rho\, v = \bar{\rho}\sqrt{\frac{c^7}{\hbar^5}}\bar{v}\, c = \bar{\rho}\,\bar{v}\,\sqrt{\frac{c^9}{\hbar^5}} \tag{16.32}$$

so

$$J = \overline{J}\,\sqrt{\frac{c^9}{\hbar^5}}, \quad [\overline{J}] = [\bar{\rho}\,\bar{v}] = [\bar{\rho}] = [m^3]. \tag{16.33}$$

Electric Field

With the electric field magnitude in natural units being $\overline{\mathcal{E}}$ and the electric field magnitude in HL units being $\mathcal{E}$, by considering the magnitude of the electric field force,

$$\mathcal{E} = \frac{F}{q} = \frac{\overline{F}\,c^3/\hbar}{\bar{q}\,\sqrt{\hbar c}} = \frac{\overline{F}}{\bar{q}}\frac{c^{5/2}}{\hbar^{3/2}}, \tag{16.34}$$

so we have

$$\mathcal{E} = \overline{\mathcal{E}}\,\sqrt{c^5/\hbar^3}, \quad [\overline{\mathcal{E}}] = \left[\frac{\overline{F}}{\bar{q}}\right] = [m^2]. \tag{16.35}$$

Magnetic Field

With the magnetic field magnitude in natural units being $\overline{\mathcal{B}}$ and the magnetic field magnitude in HL units being $\mathcal{B}$, by considering the magnitude of the magnetic force when $\vec{v}\cdot\vec{\mathcal{B}} = 0$,

$$\mathcal{B} = \frac{cF}{qv} = \frac{\overline{F}\,c^4/\hbar}{\bar{q}\,\sqrt{\hbar c}\,\bar{v}c} = \frac{\overline{F}}{\bar{q}\,\bar{v}}\frac{c^{5/2}}{\hbar^{3/2}}, \tag{16.36}$$

so we have

$$\mathcal{B} = \overline{\mathcal{B}}\,\sqrt{c^5/\hbar^3}, \quad [\overline{\mathcal{B}}] = \left[\frac{\overline{F}}{\bar{q}\,\bar{v}}\right] = [m^2]. \tag{16.37}$$

16.1 Summary of Mass Dependence

Using physical quantities in natural units, each quantity has dimensions given by some power of mass: m^d. A dimensionless quantity has the same dimensions as $m^0 = 1$. We omit the bar or overline from these symbols, each of which (other than mass m) represents a quantity in natural units.

quantity	dim	quantity	dim
v	1	n	m^3
E	m	ρ_m	m^4
p	m	G	m^{-2}
x	m^{-1}	q	1
∇	m	ρ_e	m^3
t	m^{-1}	J_e	m^3
$\vec{r}\times\vec{p}$	1	$\mathcal{E}$	m^2
F	m^2	$\mathcal{B}$	m^2
ϕ	m	$\vec{A}$	m
$\delta(x)$	m	$\delta(p)$	m^{-1}

This table can be used to check the dimensional consistency of any equation written down in natural units.

16.2 Electromagnetic Equations in Natural Units

We then can take over the set of electromagnetic equations displayed above in the section on HL units, and using natural units, the parameter c disappears. You can use the mass dimension of each quantity to check that each of these equations is dimensionally consistent. In natural units we have (naturally we omit any bar or overline notation here)

$$\begin{aligned}
&\nabla \cdot \vec{B} = 0 \\
&\nabla \cdot \vec{E} = \rho \\
&\nabla \times \vec{E} + \frac{\partial \vec{B}}{\partial t} = 0 \\
&\nabla \times \vec{B} - \frac{\partial \vec{E}}{\partial t} = \vec{J}
\end{aligned}$$

The Lorentz force on a particle of charge q is

$$\vec{F} = q\left(\vec{E} + \vec{v} \times \vec{B}\right). \tag{16.38}$$

The electromagnetic field energy is

$$U_{\text{fields}} = \frac{1}{2}\int d^3x \left(\vec{E}^2 + \vec{B}^2\right). \tag{16.39}$$

The Poynting vector is

$$\vec{S} = \vec{E} \times \vec{B}. \tag{16.40}$$

The electromagnetic field momentum is

$$\vec{P}_{\text{fields}} = \int d^3x \left(\vec{E} \times \vec{B}\right). \tag{16.41}$$

We also have

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{J} = 0 \\
&\vec{E} = -\nabla\phi - \frac{\partial \vec{A}}{\partial t} \\
&\vec{B} = \nabla \times \vec{A}.
\end{aligned}$$

Using GeV units

In particle physics it is common to measure quantities in units of GeV ($1\text{GeV} = 10^9$electron volts), a choice motivated by the fact that the rest energy of the proton is roughly 1GeV. Since in natural units, mass and energy have the same units, it is convenient to speak of either mass, energy, or momentum all in terms of GeV, and to measure length and time in units of GeV^{-1}.

Cross sections are often expressed in millibarns, where

$$1\text{mb} = 10^{-3}\,\text{b} = 10^{-27}\,\text{cm}^2, \tag{16.42}$$

and

$$1\,\text{GeV}^{-2} = 0.389\,\text{mb}. \tag{16.43}$$

Here is a table of conventional mass, length, time, and positron charge in terms of $\hbar = c = 1$ energy units.

Conversion Factor	Energy Units	Conventional Units
$1\text{kg} = 5.61 \times 10^{26}\text{GeV}$	GeV	$\frac{\text{GeV}}{c^2}$
$1\text{m} = 5.07 \times 10^{15}\text{GeV}^{-1}$	GeV^{-1}	$\frac{\hbar\, c}{\text{GeV}}$
$1\text{sec} = 1.52 \times 10^{24}\text{GeV}^{-1}$	GeV^{-1}	$\frac{\hbar}{\text{GeV}}$
$e = \sqrt{4\,\pi\,\alpha}$	–	$(\hbar\, c)^{1/2}$

17 QED Vertex Factors for Fermions and Spinless Bosons

The interactions between photons and fermions and between photons and spinless bosons are implied by the interaction Lagrangians. Total electrical charge must be conserved at each vertex. We assume here the particle (fermion or spinless boson) has an electrical charge Q in natural units (which may include a sign). Then the vertex factor for a fermion is $-iQ\gamma_\mu$: We can use our **qdraw** package (Ch. 13) to make the fermion vertex factor figure:

```
(%i1) load(draw);
(%o1) "C:/Program Files/Maxima-sbcl-5.36.1/share/maxima/5.36.1/share/draw/draw.lisp"
(%i2) load(qdraw);
" qdraw(...), qdensity(...), syntax: type qdraw(); "
(%o2) "c:/work5/qdraw.mac"
(%i25) qdraw(xr(-5,8),yr(-5,5),ex1(0.3*sin(2*%pi*x),x,-2,2,lc(black),lw(2)),
            line(-4,-4,-2,0,lw(4)),line(-2,0,-4,4,lw(4)),
              arrowhead(-3,-2,63.43,0.4),arrowhead(-3,2,116.57,0.4),
               label(["{/=18 P_{a}}",-4,-2],["{/=18 P_{b}}",-3.8,1.65],
                   ["{/Symbol=30 m}",2.2,0],
                   ["{/=18 - i Q} {/Symbol=40 g}_{{/Symbol=20 m}}",4,0]),cut(all))$
```

which produces the plot

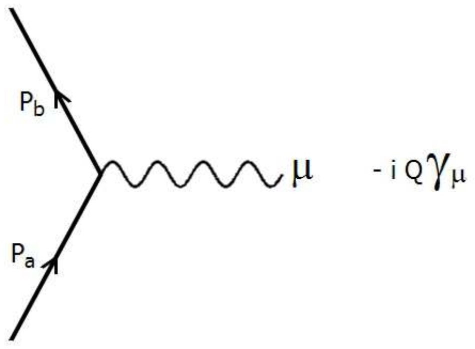

Figure 2: QED Vertex Factor for Fermions

The single photon vertex factor for a spinless boson is $-iQ\,(p_a + p_b)_\mu$.

```
(%i28) qdraw(xr(-5,10),yr(-5,5),ex1(0.3*sin(2*%pi*x),x,-2,2,lc(black),lw(2)),
            line(-4,-4,-2,0,lw(4)),line(-2,0,-4,4,lw(4)),
              arrowhead(-3,-2,63.43,0.4),arrowhead(-3,2,116.57,0.4),
               label(["{/=18 P_{a}}",-4,-2],["{/=18 P_{b}}",-3.8,1.65],
           ["{/Symbol=30 m}",2.2,0],
                   ["{/=20 - i Q} {/=18 (P_{a} + P_{b})}_{{/Symbol=20 m}}",4,0]),
                   cut(all))$
```

which produces the plot

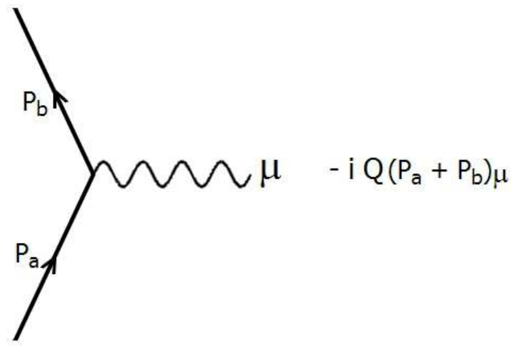

Figure 3: QED Single Photon Vertex Factor for Spinless Bosons

The double photon vertex factor for a spinless boson is $+2iQ^2 g_{\mu\nu}$.

```
(%i38) qdraw(xr(-5,10),yr(-5,5),ex1(2 + x + 0.3*sin(2*%pi*x),x,-2,2,lc(black),lw(2)),
           ex1(-2 - x + 0.3*sin(2*%pi*x),x,-2,2,lc(black),lw(2)),
                        line(-4,-4,-2,0,lw(4)),line(-2,0,-4,4,lw(4)),
                   arrowhead(-3,-2,63.43,0.4),arrowhead(-3,2,116.57,0.4),
                   label(["{/=18 P_{a}}",-4,-2],["{/=18 P_{b}}",-3.8,1.65],
                   ["{/=20 + 2 i Q^{2}} {/=22 g}_{{/Symbol=20 m n}}",4,0],
                   ["{/Symbol=20 m}",2.3,-4],["{/Symbol=20 n}",2.2,4]),cut(all))$
```

which produces the plot (the “sea-gull diagram”):

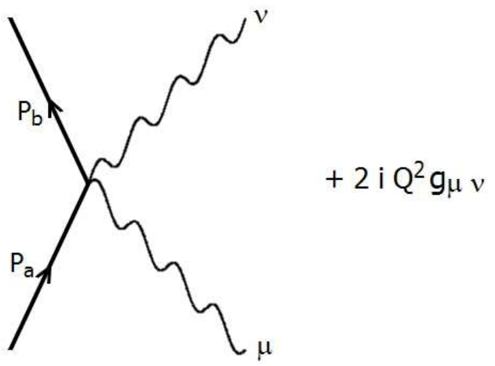

Figure 4: QED Double Photon Vertex Factor for Spinless Bosons

18 Some Code Details

18.1 Symbolic Expansions

18.1.1 Symbolic Expansions of D(a,b) using Dexpand

The package uses **`D(a,b)`** to represent the symbolic four-vector dot product of two four vectors **`a`** and **`b`**. The package function **`Dexpand`** will expand arguments of **`D`** which consist of two or more terms, and will do a mass expansion when declared mass symbols (such as the default **`m`** and **`M`** symbols) are found, and will also pull out scalars declared using **`Mscalar`** (the default set of scalars are called **`c1, c2, ...,c10`**).

Here are some interactive examples which use the package function **`Dexpand`**.

```
(%i2) Dexpand(D(a,b));
(%o2) D(a,b)
(%i3) Dexpand(D(a,b+c));
(%o3) D(a,c)+D(a,b)
(%i4) Dexpand(D(a,b+m));
(%o4) D(a)*m+D(a,b)
(%i5) Dexpand(D(a+m,b+M));
(%o5) m*M+D(a)*M+D(b)*m+D(a,b)
(%i6) Dexpand(D(2*a+m,b/3+M));
(%o6) m*M+2*D(a)*M+D(b)*m/3+2*D(a,b)/3
(%i7) Dexpand(D(2*c1*a+m,b/3+M));
(%o7) m*M+2*c1*D(a)*M+D(b)*m/3+2*c1*D(a,b)/3
(%i8) Dexpand ( D(a,b+c) + D(e,f+g) );
(%o8) D(e,g)+D(e,f)+D(a,c)+D(a,b)
(%i9) Dexpand ( r*D(a,b+c)/D(e,f) );
(%o9) D(a,c)*r/D(e,f)+D(a,b)*r/D(e,f)
(%i10) Dexpand ( r*D(a,b+c)/(s*D(e,f+g)) );
(%o10) D(a,c)*r/(D(e,g)*s+D(e,f)*s)+D(a,b)*r/(D(e,g)*s+D(e,f)*s)
```

18.1.2 Symbolic Expansion of G(a,b,c,...) Using Gexpand

The package uses **`G(a,b,c,d)`** to represent the product of four Dirac matrices, in which the symbols **`a,b,c,d`** represent Feynman slashed four-vectors unless they have been declared index symbols using **`Mindex`**. You can see the current list of index symbols available by looking at the contents of the list **`indexL`**.

```
(%i11) indexL;
(%o11) [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,la,mu,nu,rh,si,ta,al,be,ga,de,ep]
```

With this (default) list of recognised Lorentz indices, the expression **`G(mu,a,nu,b)`** would represent the matrix product **`Gam[mu] . sL(a) . Gam[nu] . sL(b)`**.

The package function **`Gexpand`** will expand symbolic products of Dirac gamma matrices, expanding arguments with multiple terms, pulling out numbers and declared scalars, and making mass expansions when finding a term which is a recognised mass symbol.

An additional expansion occurs (using **`Gexpand`**) if a slot contains the symbol **`S(sv)`** or **`S(sv,Sp)`** which are used to symbolically represent heliciy projection matrices (the simpler first case for particles treated as massless, and the more complicated two argument version for the more general case of massive particles).

The symbol **`sv`** is a stand-in for the helicity quantum number taken to have values **`sv = 1`** or **`sv = -1`**. The symbol **`Sp`** represents the positive helicity spin 4-vector implied by a particle's 4-momentum vector. Many examples of the use of helicity projection operator methods will be shown in the worked out examples. When **`Gexpand`** encounters such a helicity projection operator symbol an expansion is made using the symbolic gamma5 matrix **`G5`**.

```
(%i12) Gexpand (G (S(sv)));
(%o12) sv*G(G5)/2+G(1)/2
(%i13) Gexpand (G (S(sv,Sp)));
(%o13) sv*G(G5,Sp)/2+G(1)/2
```

Note especially that the symbol **`G(1)`** represents symbolically the unit **`4 x 4`** matrix, whose trace is 4.

The package trace functions **`tr`**, **`Gtr`**, and **`nc_tr`** all use the function **`Gexpand`** as the first step in calculating a trace of a product of Dirac gamma matrices symbolically.

Here are some examples of the use of **`Gexpand`**.

```
(%i14) Gexpand (G());
(%o14) G()
(%i15) Gexpand (G(1));
(%o15) G(1)
(%i16) Gexpand (G(-1));
(%o16) -G(1)
(%i17) Gexpand (G(-a));
(%o17) -G(a)
(%i18) Gexpand (G(m));
(%o18) G(1)*m
(%i19) Gexpand (G(a + m));
(%o19) G(1)*m+G(a)
(%i20) Gtr (%);
(%o20) 4*m
(%i21) Gexpand (G(a,b+c));
(%o21) G(a,c)+G(a,b)
(%i22) Gtr (%);
(%o22) 4*D(a,c)+4*D(a,b)
(%i23) Gexpand (G (a+m,b+M));
(%o23) G(1)*m*M+G(a)*M+G(b)*m+G(a,b)
(%i24) Gtr (%);
(%o24) 4*m*M+4*D(a,b)
(%i25) Gexpand (G (S(1),a,S(1),b));
(%o25) G(G5,a,G5,b)/4+G(G5,a,b)/4+G(a,G5,b)/4+G(a,b)/4
(%i26) Gexpand (G (2*c1*a,3*c2*b - c3*c/7));
(%o26) 6*c1*c2*G(a,b)-2*c1*c3*G(a,c)/7
(%i27) Gtr(%);
(%o27) 24*c1*c2*D(a,b)-8*c1*c3*D(a,c)/7
```

18.2 Using simplifying-new.lisp

The following functions are based on the use of **`simplifying-new.lisp`**:

- **`simp_tr1`**: used when **`tr1`** is called by **`TR1`** to symbolically evaluate the trace of a product of Dirac matrices,

- **simp_Gexpand1**: used when **Gexpand1** is called by **Gexpand**,
- **simp_Dexpand1**: used when **Dexpand1** is called by **Dexpand**,
- **simp_scon1**: used when **scon1** is called by **scon**,
- **simp_VP1**: used when **VP** calls **VP1**.

After the definition of, for example, **simp_VP1**, there is a line of code which uses a Maxima level function defined in the code file **simplifying-new.lisp**:

```
    simplifying ('VP1, 'simp_VP1)
```

This causes a call to **VP1** to be taken over by **simp_VP1**, which is an ordinary Maxima function which can call **VP1** itself, leading to a recursive process. Each new entry into **simp_VP1** is then able to apply different sorts of massaging of the supplied expression based on the current state of that simplification.

At the top of the file **simplifying-new.lisp** is a simple example of using this code.

I am grateful to Maxima developer Barton Willis for suggesting the use of this method and for providing valuable advice with details.

The Maxima code file **...share/contrib/simplifying.lisp** was written by Maxima developer Stavros Macrakis, and will eventually be updated to the form of our package file **simplifying-new.lisp**, according to my sources.

19 References

The following works are useful sources. Some abbreviations (which are used in the text) are defined here.

- Aitchison, I.J.R., Relativistic Quantum Mechanics, Barnes and Noble,1972
- A/H: Aitchison,I.J.R, and Hey, A.J.G., Gauge Theories in Particle Physics, Adam Hilger, 1989
- Barger, Vernon D., and Phillips, Roger J. N., Collider Physics, Addison-Wesley, 1987
- B/D: Bjorken, James D. and Drell, Sidney D, Relativistic Quantum Mechanics, McGraw Hill, 1964
- BLP: Berestetskii, V. B, Lifshitz, E. M., and Pitaevskii, L. P., Quantum Electrodynamics, Course of Theoretical Physics, Vol. 4, 2nd. Ed. Pergamon Press, 1982
- Berestetskii, V. B, Lifshitz, E. M., and Pitaevskii, L. P., Relativistic Quantum Theory, Part 1, Course of Theoretical Physics, Vol. 4, Pergamon Press, 1971
- De Wit, B. and Smith, J., Field Theory in Particle Physics, North-Holland, 1986
- Gastmans, R. and Wu, Tai Tsun, The Ubiquitous Photon: Helicity Method for QED and QCD, Clarendon Press, Oxford, 1990
- G/R: Greiner, Walter, and Reinhardt, Joachim, Quantum Electrodynamics, 4'th ed., Springer, 2009
- Griffiths, Introduction to Elementary Particles, Harper and Row, 1987
- Gross, Franz, Relativistic Quantum Mechanics and Field Theory, Wiley, 1986, 1993
- H/M: Halzen, Francis and Martin, Alan D., Quarks and Leptons, John Wiley, 1984
- I/Z: Itzykson, Claude and Zuber, Jean-Bernard, Quantum Field Theory, McGraw-Hill, 1980

- Jauch, J.M., and Rohrlich, F., The Theory of Photons and Electrons, Second Expanded Edition, Springer-Verlag, 1976
- Kaku, Michio, Quantum Field Theory, Oxford Univ. Press, 1993
- Maggiore, Michele, A Modern Introduction to Quantum Field Theory, Oxford Univ. Press, 2005
- P/S: Peskin, Michael E. and Schroeder, Daniel V., An Introduction to Quantum Field Theory, Addison-Wesley, 1995
- Quigg, Chris, Gauge Theories of the Strong, Weak, and Electromagnetic Interactions, Benjamin/Cummings, 1983
- Renton, Peter, Electroweak Interactions, Cambridge Univ. Press, 1990
- Schwinger, Julian, Particles, Sources, and Fields, Vol. 1, Addison-Wesley, 1970
- Sterman, George, An Introduction to Quantum Field Theory, Cambridge Univ. Press, 1993
- Thomson, Mark, Modern Particle Physics, Cambridge Univ. Press, 2013
- Weinberg, Steven, The Quantum Theory of Fields, Vol.I, Cambridge Univ. Press, 1995