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

Edwin L. Woollett

April 15, 2011

Contents

*This version uses **Maxima 5.23.2** Check `http://www.csulb.edu/~woollett/` for the latest version of these notes. Send comments and suggestions to `woollett@charter.net`

Preface

```
COPYING AND DISTRIBUTION POLICY
This document is part of a series of notes titled
"Maxima by Example" and is made available
via the author's webpage http://www.csulb.edu/~woollett/
to aid new users of the Maxima computer algebra system.

NON-PROFIT PRINTING AND DISTRIBUTION IS PERMITTED.

You may make copies of this document and distribute them
to others as long as you charge no more than the costs of printing.

These notes (with some modifications) will be published in book form
eventually via Lulu.com in an arrangement which will continue
to allow unlimited free download of the pdf files as well as the option
of ordering a low cost paperbound version of these notes.
```

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(fft) to load in the
Maxima file fft.lisp.
To load in your own file, such as qxxx.mac
using the brief version load(qxxx), you either need to place
qxxx.mac in one of the folders Maxima searches by default, or
else put a line like:

file_search_maxima : append(["c:/work2/###.{mac,mc}"],file_search_maxima )$

in your personal startup file maxima-init.mac (see later in this chapter
for more information about this).

Otherwise you need to provide a complete path in double quotes,
as in load("c:/work2/qxxx.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 5.23.2
 (2011). 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**.

12.1 References

The following works are useful sources. Some abbreviations (which will be used in the following sections) 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
- 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
- G/R: Greiner, Walter, and Reinhardt, Joachim, Quantum Electrodynamics, fourth ed., Springer, 2009
- Griffiths, Introduction to Elementary Particles, Harper and Row, 1987
- 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
- Weinberg, Steven, The Quantum Theory of Fields, Vol.I, Cambridge Univ. Press, 1995

12.2 High Energy Physics Notation Conventions

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

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 diagonal elements of the metric tensor are $(1, -1, -1, -1)$, and the helicity (chiral) representation used for explicit matrix methods with the Dirac gamma matrices is (in 2×2 block form)

$$\gamma^0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \gamma^j = \begin{pmatrix} 0 & \sigma^j \\ -\sigma^j & 0 \end{pmatrix}, \quad \gamma^5 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{12.1}$$

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

$$\sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{12.2}$$

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

$$\boldsymbol{\gamma^5 = i\,\gamma^0\,\gamma^1\,\gamma^2\,\gamma^3}. \tag{12.3}$$

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 $h = -1/2$. The corresponding barred particle spinor is a four element row matrix

$$\bar{u}(p, \sigma) = u^\dagger\,\gamma^0 \tag{12.4}$$

and the particle Dirac spinor normalization is

$$\bar{u}(p, \sigma)\,u(p, \sigma') = 2\,m\,\delta_{\sigma\,\sigma'}, \quad \sigma,\,\sigma' = \pm 1, \tag{12.5}$$

and we also have

$$\sum_\sigma u(p, \sigma)\,\bar{u}(p, \sigma) = \not{p} + m, \tag{12.6}$$

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

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, is defined (with P/S phase choice) via

$$v(p, \sigma) = -\gamma^5\,u(p, -\sigma) \tag{12.7}$$

The corresponding barred antiparticle spinor is a four element row matrix

$$\bar{v}(p, \sigma) = v^\dagger\,\gamma^0 \tag{12.8}$$

and the normalization is

$$\bar{v}(p, \sigma)\,v(p, \sigma') = -2\,m\,\delta_{\sigma\,\sigma'} \quad \sigma,\,\sigma' = \pm 1. \tag{12.9}$$

We also have

$$\sum_\sigma v(p, \sigma)\,\bar{v}(p, \sigma) = \not{p} - m \tag{12.10}$$

We define the 4×4 the spin projection matrix

$$S(p, \sigma) = \frac{1}{2}\left(1 + \sigma\,\gamma^5\,\not{s}_R(p)\right) \tag{12.11}$$

in which the 1 is interpreted as 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_R^μ generated by the 4-vector p has the components

$$s_r^\mu(p) = \left(\frac{|\mathbf{p}|}{m},\,\frac{E}{m}\,\hat{\mathbf{p}}\right) \tag{12.12}$$

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

For finite (non-zero) mass m we have

$$u(p,\sigma)\,\bar{u}(p,\sigma) = S(p,\sigma)\,(\not{p} + m) = (\not{p} + m)\,S(p,\sigma) \tag{12.13}$$

and again for finite mass,

$$v(p,\sigma)\,\bar{v}(p,\sigma) = S(p,\sigma)\,(\not{p} - m) = (\not{p} - m)\,S(p,\sigma) \tag{12.14}$$

In the case of zero mass leptons

$$u(p,\sigma)\,\bar{u}(p,\sigma) = \frac{1}{2}\,(1 + \sigma\,\gamma^5)\,\not{p} \tag{12.15}$$

and for zero mass antiparticles

$$v(p,\sigma)\,\bar{v}(p,\sigma) = \frac{1}{2}\,(1 - \sigma\,\gamma^5)\,\not{p}. \tag{12.16}$$

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

$$(\bar{u}_2\,\Gamma\,u_1)^* = \bar{u}_1\,\bar{\Gamma}\,u_2 \tag{12.17}$$

in which

$$\bar{\Gamma} = \gamma^0\,\Gamma^\dagger\,\gamma^0. \tag{12.18}$$

Then

$$\sum_{\sigma_1\,\sigma_2} |\bar{u}_2\,\Gamma\,u_1|^2 = \sum_{\sigma_1\,\sigma_2} \bar{u}_2\,\Gamma\,u_1\,\bar{u}_1\,\bar{\Gamma}\,u_2 \tag{12.19}$$

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

$$\sum_{\sigma_2} \bar{u}_2\,\hat{\Gamma}\,u_2, \quad \hat{\Gamma} \doteq \Gamma\,(\not{p}_1 + m)\,\bar{\Gamma}. \tag{12.20}$$

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

$$\sum_{\sigma_2} \bar{u}_{2\,a}\,\hat{\Gamma}_{a\,b}\,u_{2\,b} = \hat{\Gamma}_{a\,b} \sum_{\sigma_2} u_{2\,b}\,\bar{u}_{2\,a} = \hat{\Gamma}_{a\,b}\,(\not{p}_2 + m)_{b\,a} = \textbf{Trace}\,\{\hat{\Gamma}\,(\not{p}_2 + m)\} \tag{12.21}$$

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

$$\sum_{\sigma_1\,\sigma_2} |\bar{u}_2\,\Gamma\,u_1|^2 = \textbf{Trace}\,\{(\not{p}_2 + m)\,\Gamma\,(\not{p}_1 + m)\,\bar{\Gamma}\}. \tag{12.22}$$

12.2.1 Contraction and Trace Theorems

The basic gamma matrix algebra relations are

$$\boldsymbol{\gamma^\mu\,\gamma^\nu} = -\boldsymbol{\gamma^\nu\,\gamma^\mu} + 2\,g^{\mu\,\nu}\,I_4 \tag{12.23}$$

for Lorentz indices $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ which can have values $(0, 1, 2, 3)$ and

$$\boldsymbol{\gamma^5\,\gamma^\mu} = -\boldsymbol{\gamma^\mu\,\gamma^5}. \tag{12.24}$$

Contraction Theorems

$$\boldsymbol{\gamma_\mu}\,\boldsymbol{\gamma^\mu} = 4\,I_4 \tag{12.25}$$

$$\boldsymbol{\gamma_\mu}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\mu} = -2\,\gamma^\nu \tag{12.26}$$

$$\boldsymbol{\gamma_\mu}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\lambda}\boldsymbol{\gamma^\mu} = 4\,g^{\nu\,\lambda}\,I_4 \tag{12.27}$$

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).

$$\boldsymbol{\gamma_\mu}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\lambda}\,\boldsymbol{\gamma^\sigma}\,\boldsymbol{\gamma^\mu} = -2\,\boldsymbol{\gamma^\sigma}\,\boldsymbol{\gamma^\lambda}\,\boldsymbol{\gamma^\nu} \tag{12.28}$$

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 first term in reversed order, all multiplied by (2).

$$\boldsymbol{\gamma_\mu}\,\boldsymbol{\gamma^\lambda}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\rho}\,\boldsymbol{\gamma^\sigma}\,\boldsymbol{\gamma^\mu} = 2\,(\boldsymbol{\gamma^\sigma}\,\boldsymbol{\gamma^\lambda}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\rho} + \boldsymbol{\gamma^\rho}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\lambda}\,\boldsymbol{\gamma^\sigma}) \tag{12.29}$$

Trace Theorems

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

$$Tr(I_4) = 4 \tag{12.30}$$

$$Tr(\boldsymbol{\gamma^\mu}\,\boldsymbol{\gamma^\nu}) = 4\,g^{\mu\,\nu} \tag{12.31}$$

$$Tr(\boldsymbol{\gamma^\mu}\,\boldsymbol{\gamma^\nu}\,\boldsymbol{\gamma^\lambda}\,\boldsymbol{\gamma^\sigma}) = 4\,(g^{\mu\,\nu}\,g^{\lambda\,\sigma} - g^{\mu\,\lambda}\,g^{\nu\,\sigma} + g^{\mu\,\sigma}\,g^{\nu\,\lambda}) \tag{12.32}$$

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 **(N = even)** matrices to a sum involving the traces of products of **(N - 2)** matrices.

Let's use a more efficient notation here:

$$tr(n1, n2, n3, ..., nN) \leftrightarrow Tr(\boldsymbol{\gamma^{n_1}}\,\boldsymbol{\gamma^{n_2}}\,\boldsymbol{\gamma^{n_3}}\,\boldsymbol{...}\,\boldsymbol{\gamma^{n_N}}) \tag{12.33}$$

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

$$\begin{aligned} tr(n1, n2, n3, ..., nNm1, nN) = {} & g^{n_1\,n_2}\,tr(n3, n4, n5, ..., nN) \\ & - g^{n_1\,n_3}\,tr(n2, n4, n5, ..., nN) \\ & + g^{n_1\,n_4}\,tr(n2, n3, n5, ..., nN) \\ & - \ldots + g^{n_1\,n_N}\,tr(n3, n4, n5, ..., nNm1) \end{aligned}$$

For example, if **N = 6** we have

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

Gamma 5 Traces

Since $\gamma^5 = i\,\gamma^0\,\gamma^1\,\gamma^2\,\gamma^3$ 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. Hence $Tr(\gamma^5\,\gamma^\mu) = 0$ and $Tr(\gamma^5\,\gamma^\mu\,\gamma^\nu\,\gamma^\lambda) = 0$

When γ^5 is multiplied by an **even** number of gamma matrices, we have

$$Tr(\gamma^5) = 0 \tag{12.34}$$

$$Tr(\gamma^5\,\gamma^\mu\,\gamma^\nu) = 0 \tag{12.35}$$

$$Tr(\gamma^5\,\gamma^\mu\,\gamma^\nu\,\gamma^\lambda\,\gamma^\sigma) = -4\,i\,\epsilon^{\mu\,\nu\,\lambda\,\sigma} \tag{12.36}$$

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):

$$\gamma^\mu\,\gamma^\nu\,\gamma^\rho = g^{\mu\,\nu}\,\gamma^\rho - g^{\mu\,\rho}\,\gamma^\nu + g^{\nu\,\rho}\,\gamma^\mu - i\,\epsilon^{\mu\nu\rho\lambda}\,\gamma^5\,\gamma_\lambda \tag{12.37}$$

Here is the symbolic calculation of the trace of the product of **G5** with six gamma matrices.

```
(%i81) tr6:tr(G5,n1,n2,n3,n4,n5,n6);
(%o81) -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)
```

Version 1 of the Dirac package produces a result we will call **tr6comp** (copied from a separate work session using version 1):

```
(%i82) tr6comp : 4*%i*Eps(n3,n2,n1,n4)*Gm(n5,n6)-4*%i*Eps(n3,n2,n1,n5)*Gm(n4,n6)
                                       +4*%i*Eps(n3,n2,n1,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)
```

If we take the difference, we do not get zero, because we are using a different algorithm in version 2, and the symbol **Eps** is not declared antisymmetric by the package.

```
(%i83) tr6 - tr6comp;
(%o83) -4*%i*Eps(n3,n2,n1,n4)*Gm(n5,n6)-4*%i*Eps(n1,n2,n3,n4)*Gm(n5,n6)
                                       +4*%i*Eps(n3,n2,n1,n5)*Gm(n4,n6)
                                       +4*%i*Eps(n1,n2,n3,n5)*Gm(n4,n6)
                                       -4*%i*Eps(n3,n2,n1,n6)*Gm(n4,n5)
                                       -4*%i*Eps(n1,n2,n3,n6)*Gm(n4,n5)
```

However, use of **noncov** introduces **eps4**, which is declared antisymmetic, and shows the results are equivalent when the antisymmetry is taken into account.

```
(%i84) noncov(%);
(%o84) 0
```

Here we calculate a trace of gamma5 multiplied by eight gamma matrices:

```
(%i85) tr8:tr(G5,n1,n2,n3,n4,n5,n6,n7,n8);
(%o85) -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)
(%i86) length(%);
(%o86) 33
```

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**.

```
(%i2) 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"
(%o2) Gamma5trace(G5,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10)
(%i93) Mindex(n11,n12);
(%o93) done
(%i94) indexL;
(%o94) [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,la,mu,nu,rh,si,ta,al,be,ga,de,ep,n11,
        n12]
(%i95) tr11:tr(G5,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11);
(%o95) 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,
http://arxiv.org/abs/hep-ph/0504223

12.3 Introduction to the Dirac Package, Version 2

Three independent Dirac algebra methods are available in our package. One can gain confidence in an unexpected result by doing the same calculation in multiple ways. The explicit 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 Dirac matrix methods using the Maxima function **mat_trace** 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 usually use the fastest and most convenient method for a given problem. An example of using the **matrix** trace and contraction methods, and the **symbolic** trace and contraction methods to generate the square of polarized amplitudes is provided in the section dealing with the batch file **moller3.mac**.

Each physics calculation is written in the form of a batch file with an appropriate name, such as **moller1.mac**. One can start a problem calculation in two steps by first loading the Dirac package with **load("dirac2.mac")** (which kills all definitions and arrays and resets options to the default) and then, for example, launching the batch file using **batch ("moller1.mac")**.

The batch files use code in the the launch file **dirac2.mac** to print out the year-month-day of the batch file run, and also to print the Maxima version number. This code is:

```
load("new-timedate.lisp")$
mydate : apply ('sconcat, rest (charlist (timedate(+7)),-15))$
_binfo% : "Maxima 5.23.2"$
```

This code also is in the author's **maxima-init** file, to allow a print out the year-month-day at the start of a Maxima session screen, using

```
disp (mydate)$
```

as a line in **maxima-init**.

However the launch file **dirac2.mac** starts with the lines

```
kill(all);
reset();
if not lfreeof (rules, eps4rule1) then apply ('kill, [eps4] )$
   /* remove all global arrays currently existing */
if length (arrays) > 0 then apply ('remarray, arrays)$
```

(which is probably overkill), but removes the **mydate** dependent code in **maxima-init** also, hence that code again appears in **dirac2.mac**.

The lisp file **new-timedat.lisp**, available on the author's webpage, was written by Maxima developer Leo Butler, and allows Windows users to overcome a bug to get a year-month-day stamp using Maxima's **timedate** function.

In our example batch files is the line

```
print ("      ver: ",_binfo%,"  date: ",mydate )$
```

which prints out the Maxima version number and year-month-day as:

```
ver:  Maxima 5.23.2   date:  2011-02-28
```

12.4 Overview and Examples of the Dirac Package Tools

Loading the Dirac package is accomplished by the command **load("dirac2.mac")** (or just **load(dirac2)** if you have set up your init file as recommended in Chapter 1). The **dirac2.mac** file itself then loads five package files: **simplifying-new.lisp**, **dgtrace2.mac**, **dgcon2.mac**, **dgeval2.mac**, and **dgmatrix2.mac**.

This package uses symbols for both purely symbolic entities and also for purely numerical and matrix entities. The most used (in the twelve example batch files) "symbolic route" package functions are **tr**, **nc_tr**, **Con**, **comp_def**, and **noncov**.

The file **dgfunctions2.txt** has an alphabetical listing of the Dirac package functions with the name the program file in which the code for that function will be found. One should then go to that file and use a name search for the name of the function. Near the top and/or the bottom of each function code 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, you can experiment with calls to any of these functions, using the interactive mode.

Symbols for Lorentz indices need to be declared using Mindex. However, a set of default Lorentz index symbols is defined in **dirac2.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}**.

dirac2.mac also assigns two default symbols to be recognised as symbols for particle masses, using the command: **Mmass (m,M)$**. The package list **massL** will show you the current list of recognised mass symbols. 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 **dirac2.mac** also defines a default set of scalars with the command:
Mscalar (c1,c2,c3,c4,c5,c6,c7,c8,c9,c10)$. 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 $\mathrm{g}^{\mu\,\nu}$ (or $\mathrm{g}_{\mu\,\nu}$ 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 to the numerical array **eps4[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, $\mathbf{p}_0 = \mathbf{p}^0$.

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 and use **G(1)** to symbolically denote the unit **4 x 4** matrix.

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). Using explicit Maxima matrix notation, this matrix product is **Gam[mu] . sL(p) . Gam[nu] . sL(q)** in which **Gam[mu]** is one of the four Dirac gamma matrices and **sL(p)** is the Feynman slash matrix $\not{p}$ formed from the explicit gamma matrices and the explicit

4-momentum array **p[mu]**.

A symbolic product which includes the gamma5 matrix γ^5, represented symbolically by **G5**, is **G(G5,mu,p1,nu,p2)**, for example, and corresponds to: **Gam[5] . Gam[mu] . sL(p1) . Gam[nu] . sL(p2)**. The arrays **Gam[mu]** and **Gam[5]** are **4 x 4** matrices, as is the matrix function **sL(p)**, and they are defined in **dgmatrix2.mac**, where you can also find the definitions of the Dirac particle spinor (4-component column vector) **UU**, and the 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 **dgmatrix2.mac** code) will be found in the twelve batch file problems worked out.

The file **dirac2.mac** includes some definitions of useful utility functions, and also defines the explicit numerical metric tensor array **gmet[mu,nu]** and the explicit four dimensional completely antisymmetric tensor array **eps4[mu,nu,rh,si]** components (and antisymmetry property). (The Lorentz indices **mu**, **nu**,... take values **0,1,2,3**.) **dirac2.mac** also 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.

The function **noncov** converts **Eps(mu,nu,rh,si)** to **eps4[mu,nu,rh,si]** (with the convention that $\epsilon^{0123} = 1$), converts **Gm(mu,nu)** to **gmet[mu,nu]**, converts **UI(p,mu)** to **p[mu]**, 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]**. 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]**. The function **noncov** does **nothing** to **G(...)**.

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.

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, and 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]).

To obtain useful 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 **Con** (or **mcon**) for contraction.

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. 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**.

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.

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 **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))**.

Con (and **scon**) automatically contract on all repeated Lorentz indices unless the user 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, 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** .

A frequent case met with in the twelve examples 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. 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 is provided by using either the contraction function **econ** ("e" for "explicit") or (usually quicker in operation) **mcon**. You could simply use **Con** (which should recognise the post-**noncov** case and call **econ** with the supplied contraction indices). As in the explicit matrix case, you must supply the desired contraction indices (in this third case) to accomplish the desired contraction.

In the following introductory examples, carried out in an interactive context, you will see examples of all of the above for both traces and contractions.

12.4.1 Some Basic Package Symbols

In the initial examples, no values have been assigned to the 4-momenta array components referred to (this will eventually be done using the package function **comp_def**), so we will see things like **p[1]** for example. The last line of **dirac2.mac** is **display2d:false$**, which allows more information per screen, especially when displaying matrices.

```
(%i1) load(dirac2);
                                    dirac2.mac
                                simplifying-new.lisp
                                    dgtrace2.mac
                                     dgcon2.mac
                                     dgeval2.mac
                                   dgmatrix2.mac
               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, G, G(1), G5, G5p, Gam, Gm, Gtr, LI, Nlast, UI, P, S, Sig
               UU, VP, VV, I2, Z2, CZ2, I4, Z4, CZ4, RZ4, N1,N2,...
                           reserved array names: gmet, eps4
(%o1) "c:/work5/dirac2.mac"
(%i2) noncov(Eps(mu,nu,rh,si));
(%o2) eps4[mu,nu,rh,si]
(%i3) noncov (Eps(mu,nu,al,be));
(%o3) eps4[al,be,mu,nu]
(%i4) Eps(0,1,2,3);
(%o4) Eps(0,1,2,3)
(%i5) eps4 [0,1,2,3];
(%o5) 1
(%i6) eps4 [1,0,2,3];
(%o6) -1
(%i7) noncov (Gm (mu,nu));
(%o7) gmet[mu,nu]
(%i8) Gm (0,0);
(%o8) Gm(0,0)
(%i9) gmet[0,0];
(%o9) 1
(%i10) gmet[1,1];
(%o10) -1
(%i11) Gm (be,al);
(%o11) Gm(al,be)
(%i12) noncov (D(p,q));
(%o12) -p[3]*q[3]-p[2]*q[2]-p[1]*q[1]+p[0]*q[0]
(%i13) D (q,p);
(%o13) D(p,q)
(%i14) D(b,a);
(%o14) D(a,b)
(%i15) noncov (UI(p,0));
(%o15) p[0]
(%i16) noncov (UI(p,1));
(%o16) p[1]
(%i17) noncov (LI (p,0));
(%o17) p[0]
(%i18) noncov (LI (p,1));
(%o18) -p[1]
(%i19) massL;
(%o19) [m,M]
(%i20) Mmass (m1,m2);
(%o20) done
```

```
(%i21) massL;
(%o21) [m,M,m1,m2]
(%i22) map ('massp,%);
(%o22) [true,true,true,true]
(%i23) indexL;
(%o23) [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,la,mu,nu,rh,si,ta,al,be,ga,de,ep]
(%i24) map ('indexp,%);
(%o24) [true,true,true,true,true,true,true,true,true,true,true,true,true,true,
        true,true,true,true,true,true,true]
(%i25) scalarL;
(%o25) [c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]
(%i26) map ('scalarp,%);
(%o26) [true,true,true,true,true,true,true,true,true,true]
(%i27) noncov (G (mu,p1,nu,p2) );
(%o27) G(mu,p1,nu,p2)
```

12.4.2 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)
```

12.4.3 Some Examples of the 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 standin of 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 reaction 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
```

12.4.4 Symbolic and Explicit Matrix Trace Examples: tr, Gtr, mat_trace, m_tr

Some Symbolic Trace Examples

We next display some examples of symbolic traces.

The symbol **G5** symbolically represents γ^5.
The symbol **Gm(mu,nu)** whose arguments are both recognised Lorentz index symbols, represents symbolically either $g^{\mu\nu}$ or $g_{\mu\nu}$, depending on the context.

The symbol **UI(p,mu)** ("upper index") symbolically represents p^{μ}.

The symbol **LI(p,mu)** ("lower index") symbolically represents p_{μ}.

The symbol **Eps (la,mu,nu,rh)** represents symbolically $\epsilon^{\lambda\mu\nu\rho}$.

```
(%i2) tr(1);
(%o2) 4
(%i3) Gtr (G(1));
(%o3) 4
(%i4) tr(mu);
(%o4) 0
(%i5) tr(mu,nu);
(%o5) 4*Gm(mu,nu)
(%i6) Gtr (G (mu,nu));
(%o6) 4*Gm(mu,nu)
(%i7) tr(p,q);
(%o7) 4*D(p,q)
(%i8) tr(p,mu);
(%o8) 4*UI(p,mu)
(%i9) noncov(%);
(%o9) 4*p[mu]
(%i10) tr(mu,p);
(%o10) 4*UI(p,mu)
(%i11) tr(G5);
(%o11) 0
(%i12) tr(G5,mu);
(%o12) 0
(%i13) tr (G5,mu,nu);
(%o13) 0
(%i14) tr (G5,mu,nu,rh);
(%o14) 0
(%i15) tr (G5,al,be,la,mu);
(%o15) -4*%i*Eps(al,be,la,mu)
(%i16) noncov(%);
(%o16) -4*%i*eps4[al,be,la,mu]
(%i17) tr(la,mu,nu,rh);
(%o17) 4*Gm(la,mu)*Gm(nu,rh)-4*Gm(la,nu)*Gm(mu,rh)+4*Gm(la,rh)*Gm(mu,nu)
(%i18) tr (a,b,c,d);
(%o18) 4*D(a,b)*D(c,d)-4*D(a,c)*D(b,d)+4*D(a,d)*D(b,c)
(%i19) tr(G5,p,be,la,mu);
(%o19) -4*%i*LI(p,N1)*Eps(N1,be,la,mu)
(%i136) tr(G5,mu,nu,rh,al);
(%o136) -4*%i*Eps(mu,nu,rh,al)
(%i137) tr(mu,G5,nu,rh,al);
(%o137) 4*%i*Eps(mu,nu,rh,al)
(%i138) tr(mu,nu,G5,rh,al);
(%o138) -4*%i*Eps(mu,nu,rh,al)
(%i139) tr(mu,nu,rh,G5,al);
(%o139) 4*%i*Eps(mu,nu,rh,al)
(%i140) tr(mu,nu,rh,al,G5);
(%o140) -4*%i*Eps(mu,nu,rh,al)
```

Some Simple Explicit Matrix Trace Examples

Next are some very simple examples of the use of explicit matrix trace methods. **I4** is an explicit unit matrix in 4 dimensions, and **Z4** is an explicit zero matrix in 4 dimensions.

```
(%i21) I4;
(%o21) matrix([1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1])
(%i22) mat_trace(%);
(%o22) 4
(%i23) Z4;
(%o23) matrix([0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0])
(%i24) mat_trace (%);
(%o24) 0
(%i25) a(mu,nu):=mat_trace(Gam[mu] . Gam[nu])$
(%i26) b(mu,nu):=4*gmet[mu,nu]$
(%i27) c(mu,nu):=is(equal(a(mu,nu),b(mu,nu)))$
(%i28) c(0,0);
(%o28) true
(%i30) for mu : 0 thru 3 do
           for nu : 0 thru 3 do
                if not c(mu,nu) then print(mu,nu,"  false")$
```

In the last step, we have shown that **a(mu,nu) = b(mu,nu)** for all 16 **mu,nu** pairs, since there were no **false** messages printed to the screen.

More Symbolic Trace Examples

Returning to the symbolic trace with **tr**, recognised mass symbols get special treatment.

```
(%i31) tr(m+p1,mu,m+p2,mu);
(%o31) 16*m^2-8*D(p1,p2)
(%i32) tr(m,m);
(%o32) 4*m^2
(%i33) tr(m,M);
(%o33) 4*m*M
```

The arguments of **tr(m+p1,mu,m+p2,mu)** are examined to look for "mass terms", and if any are found those arguments are expanded and the mass symbols are factored out of the **tr** function. An argument like **p + m** in **tr** is treated like the matrix sum **sL(p) + m*I4** and a parallel treatment using explicit matrix methods needs to have the mass symbol multiplied by **I4**.

The last two simple examples can be reproduced using **mat_trace**, but we have to be careful to use parentheses around the quantity **m*I4**, since the Maxima function **mat_trace** does not know anything about special mass symbols.

```
(%i35) mat_trace((m*I4) . (m*I4));
(%o35) 4*m^2
```

Likewise, numbers and recognised scalars which multiply and/or divide momentum symbols are factored out of **tr** arguments.

```
(%i36) scalarL;
(%o36) [c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]
(%i37) tr(c1*p,q);
(%o37) 4*c1*D(p,q)
(%i38) tr(-c1*p,q/c2);
(%o38) -4*c1*D(p,q)/c2
(%i39) tr(-c1*%i*p,q/c2);
(%o39) -4*%i*c1*D(p,q)/c2
(%i40) tr(-7*c1*%i*p,q/c2/10);
(%o40) -14*%i*c1*D(p,q)/(5*c2)
```

Factoring out scalars, etc., occurs after multiple term arguments are first expanded.

```
(%i41) tr(a,b);
(%o41) 4*D(a,b)
(%i42) tr(a,b+c);
(%o42) 4*D(a,c)+4*D(a,b)
(%i43) tr(a,c1*b+c2*c+c3*d);
(%o43) 4*c3*D(a,d)+4*c2*D(a,c)+4*c1*D(a,b)
```

12.4.5 Symbolic and Explicit Matrix Contraction Examples Using Con

Examples of Symbolic Contraction of Repeated Lorentz Indices

We next show some examples of contractions on repeated Lorentz indices. Recall that **G(1)** symbolically represents the unit **4 x 4** matrix and **G5** represents the matrix γ^5.

```
(%i4) Con (G (mu,p,q,mu));
(%o4) 4*G(1)*D(p,q)
(%i5) Gtr (%);
(%o5) 16*D(p,q)
(%i6) tr (mu,p,q,mu);
(%o6) 16*D(p,q)
(%i7) Con (G (mu,G5,mu));
(%o7) -4*G(G5)
(%i8) Con (G(n1,p,n2,n1),n1);
(%o8) 4*G(1)*UI(p,n2)
(%i9) Con (G(n1,p,n2,n1));
(%o9) 4*G(1)*UI(p,n2)
(%i10) Con (G(n1,n2,n2,n1),n1);
(%o10) 4*G(1)*Gm(n2,n2)
(%i11) Con (%);
(%o11) 16*G(1)
(%i12) Gtr (%);
(%o12) 64
(%i13) Con (UI(p,mu)*UI(q,mu));
(%o13) D(p,q)
(%i14) Con (UI(p,mu)*Gm(mu,nu));
(%o14) UI(p,nu)
(%i15) Con (G(a,mu,b,c,mu,a));
(%o15) 4*G(1)*D(a,a)*D(b,c)
(%i16) Con (G(n1,p+m,q+M,n1),n1);
(%o16) -2*G(p)*M+4*G(1)*m*M-2*m*G(q)+4*G(1)*D(p,q)
(%i17) Gtr (%);
(%o17) 16*m*M+16*D(p,q)
(%i18) tr(n1,p+m,q+M,n1);
(%o18) 16*m*M+16*D(p,q)
(%i19) Con (G(n1,2*a,b/3,n1),n1);
(%o19) 8*G(1)*D(a,b)/3
(%i20) Con  (G(n5,n4,n3,n2,n1,n5,n1,n4,n3,n2));
(%o20) 256*G(1)
(%i21) Gtr (%);
(%o21) 1024
(%i22) tr (n5,n4,n3,n2,n1,n5,n1,n4,n3,n2);
(%o22) 1024
(%i25) Con (G (G5,mu,nu,nu,mu));
(%o25) 16*G(G5)
(%i2) scon (Gm (n1,n2)*Eps (n1,n3,n4,n5));
(%o2) Eps(n2,n3,n4,n5)
```

The last example shows a contraction between an index which appears in a **Gm** and a multiplying **Eps**. **scon** calls **get_rind** to find repeated indices and then **simp_scon1** to automatically contract on repeated indices, which it will

do for a product of **Gm(..)** and **Eps(..)**. looks for repeated indices.

If we try the same contraction with **Con** we get

```
(%i13) Con (Gm (n1,n2)*Eps (n1,n3,n4,n5));
(%o13) gmet[n1,n2]*eps4[n1,n3,n4,n5]
```

which occurs because **Con** calls **Eps_facp**, (which returns **true**), thus detecting the presence of an **Eps** factor in the expression, and as a consequence, calls **noncov** immediately, followed by a call to **econ**. **noncov** converts **Gm( )** to **gmet[ ]** and **Eps( )** to **eps4[ ]**. **econ** will not automatically contract; you must supply the contraction index.

```
(%i14) Con (Gm (n1,n2)*Eps (n1,n3,n4,n5),n1);
(%o14) eps4[n2,n3,n4,n5]
```

Note that the present version of **scon** will not simplify a product of two **Eps** factors.

```
(%i17) scon(Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7));
(%o17) Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7)
(%i18) scon(Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7),n1);
(%o18) Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7)
```

However, you can use **Con** (supplying the contraction index as well) which again will detect the **Eps** factors, then will call **noncov** and **econ** to do the summatation over the contraction index.

```
(%i16) Con(Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7),n1);
(%o16) -eps4[3,n2,n3,n4]*eps4[3,n5,n6,n7]-eps4[2,n2,n3,n4]*eps4[2,n5,n6,n7]
                                         -eps4[1,n2,n3,n4]*eps4[1,n5,n6,n7]
                                         +eps4[0,n2,n3,n4]*eps4[0,n5,n6,n7]
```

Alternatively, you can use **noncov** and then **econ**:

```
(%i19) noncov (Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7));
(%o19) eps4[n1,n2,n3,n4]*eps4[n1,n5,n6,n7]
(%i20) econ (%,n1);
(%o20) -eps4[3,n2,n3,n4]*eps4[3,n5,n6,n7]-eps4[2,n2,n3,n4]*eps4[2,n5,n6,n7]
                                         -eps4[1,n2,n3,n4]*eps4[1,n5,n6,n7]
                                         +eps4[0,n2,n3,n4]*eps4[0,n5,n6,n7]
```

The present version of the Dirac package does not simplify the implied contraction presented by **Con (Eps (n1,n2,n3,n4)*Eps (n1,n5,n6,n7),n1)**. Using the corresponding expression with **Eps(...)** replaced by **eps4[...]** and then using **econ** (or **Con**) will return a contracted expression in terms of **eps4's**.

```
(%i34) Con (Eps (n1,n2,n3,n4)*Eps (n1,n5,n6,n7));
(%o34) Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7)
(%i37) Con (Eps (n1,n2,n3,n4)*Eps (n1,n5,n6,n7),n1);
(%o37) Eps(n1,n2,n3,n4)*Eps(n1,n5,n6,n7)
(%i35) noncov(%);
(%o35) eps4[n1,n2,n3,n4]*eps4[n1,n5,n6,n7]
(%i36) econ(%,n1);
(%o36) -eps4[3,n2,n3,n4]*eps4[3,n5,n6,n7]-eps4[2,n2,n3,n4]*eps4[2,n5,n6,n7]
                                         -eps4[1,n2,n3,n4]*eps4[1,n5,n6,n7]
                                         +eps4[0,n2,n3,n4]*eps4[0,n5,n6,n7]
(%i38) Con (eps4[n1,n2,n3,n4]*eps4[n1,n5,n6,n7],n1);
(%o38) -eps4[3,n2,n3,n4]*eps4[3,n5,n6,n7]-eps4[2,n2,n3,n4]*eps4[2,n5,n6,n7]
                                         -eps4[1,n2,n3,n4]*eps4[1,n5,n6,n7]
                                         +eps4[0,n2,n3,n4]*eps4[0,n5,n6,n7]
```

Examples of Explicit Matrix Contractions on Repeated Lorentz Indices Using Con

```
(%i26) Con (Gam[n1] . Gam[n1],n1);
(%o26) matrix([4,0,0,0],[0,4,0,0],[0,0,4,0],[0,0,0,4])
(%i27) is (equal (%,4*I4));
(%o27) true
(%i28) Con (G(n1,n1));
(%o28) 4*G(1)
(%i29) Con (Gam[mu] . Gam[5] . Gam[mu],mu);
(%o29) matrix([4,0,0,0],[0,4,0,0],[0,0,-4,0],[0,0,0,-4])
(%i30) is (equal (%,-4*Gam[5]));
(%o30) true
(%i32) Con (G(mu,G5,mu));
(%o32) -4*G(G5)
(%i23) Con(Gam[5] . Gam[mu] . Gam[nu] . Gam[nu] . Gam[mu],mu,nu);
(%o23) matrix([-16,0,0,0],[0,-16,0,0],[0,0,16,0],[0,0,0,16])
(%i24) is (equal (%,16*Gam[5]));
(%o24) true
```

12.4.6 Examples of noncov, comp_def, VP, and econ

```
(%i2) noncov(D(p,q));
(%o2) -p[3]*q[3]-p[2]*q[2]-p[1]*q[1]+p[0]*q[0]
(%i3) VP (p,q);
(%o3) -p[3]*q[3]-p[2]*q[2]-p[1]*q[1]+p[0]*q[0]
(%i4) noncov(Gm(mu,nu));
(%o4) gmet[mu,nu]
(%i5) gmet[0,0];
(%o5) 1
(%i6) gmet[1,1];
(%o6) -1
(%i7) noncov(UI(p,mu));
(%o7) p[mu]
(%i8) noncov(LI(p,mu));
(%o8) p[3]*gmet[3,mu]+p[2]*gmet[2,mu]+p[1]*gmet[1,mu]+p[0]*gmet[0,mu]
(%i9) noncov(LI(p,0));
(%o9) p[0]
(%i10) noncov(LI(p,1));
(%o10) -p[1]
(%i11) noncov(Eps(mu,nu,rh,la));
(%o11) -eps4[la,mu,nu,rh]
(%i12) eps4[0,1,2,3];
(%o12) 1
(%i13) Eps(0,1,2,3);
(%o13) Eps(0,1,2,3)
(%i14) eps4[1,0,2,3];
(%o14) -1
```

PRODUCTS OF eps4's WITH econ AND noncov: COMPARISONS

To get summation of dummy indices **N1,N2,..** etc from **noncov**, the code for **sum_eps1** automatically recognises that these symbols represent dummy summation variables.

```
(%i11) map ('dummyp,[N1,N2,N3,N4,N5]);
(%o11) [true,true,true,true,true]
```

The use of **noncov** will automatically sum over all dummy indices.

```
(%i12) noncov(Eps(N1,N2,N3,N4)*Eps(N1,N2,N3,N4));
(%o12) 24
```

However, the above result is NOT what is meant by contraction. True contraction of indices includes both lowering one of each pair of indices and then summing, and this operation is correctly carried out by **econ**.

```
(%i13) econ(eps4[n1,n2,n3,n4]*eps4[n1,n2,n3,n4],n1,n2,n3,n4);
(%o13) -24
```

Here are a few more comparisons between **noncov** and **econ** in this **eps4** product context:

```
(%i14) noncov(Eps(N1,N2,N3,0)*Eps(N1,N2,N3,0));
(%o14) 6
(%i15) econ(eps4[n1,n2,n3,0]*eps4[n1,n2,n3,0],n1,n2,n3);
(%o15) -6
(%i16) noncov(Eps(N1,N2,N3,0)*Eps(N1,N2,N3,1));
(%o16) 0
(%i17) econ(eps4[n1,n2,n3,0]*eps4[n1,n2,n3,1],n1,n2,n3);
(%o17) 0
(%i18) noncov(Eps(N1,N2,0,1)*Eps(N1,N2,0,1));
(%o18) 2
(%i19) econ(eps4[n1,n2,0,1]*eps4[n1,n2,0,1],n1,n2);
(%o19) 2
```

EXAMPLE OF comp_def WITH 2 --> 2 SCATTERING KINEMATICS

Assume high energy scattering in the **z-x** plane in the center of momentum frame. Then the masses can be ignored, and the magnitude of the 3-momentum is the same as the relativistic energy (outside the range of the force field felt by each particle). Remember that we are using natural units in which momentum, energy and mass all have the same units. The incident particles move along the positive and negative **z** axis and the particle initially moving along the positive **z** axis is scattered into the direction in the **z-x** plane which makes the angle **th** radians with the **z** axis.

The momenta of the incoming particles are **(p1, p2)** and the momenta of the outgoing particles are **(p3, p4)**. We use **comp_def** to define relativistic energy and cartesian components of 3-momentum with the syntax **p(E,px,py,pz)**. The kinematics are described in terms of the symbols **E, th**.

```
(%i15) assume(E > 0,th >= 0,th <= %pi)$
(%i16) comp_def(p1(E,0,0,E),p2(E,0,0,-E),p3(E,E*sin(th),0,E*cos(th)),
                p4(E,-E*sin(th),0,-E*cos(th)))$
(%i17) map ('listarray, [p1,p2,p3,p4]);
(%o17) [[E,0,0,E],[E,0,0,-E],[E,sin(th)*E,0,cos(th)*E],
        [E,-sin(th)*E,0,-cos(th)*E]]
(%i18) p1[0];
(%o18) E
(%i19) p3[1];
(%o19) sin(th)*E
(%i20) noncov(D(p1,p2));
(%o20) 2*E^2
(%i21) noncov(D(p2+p1,p2+p1));
(%o21) 4*E^2
(%i22) VP(p2+p1,p2+p1);
(%o22) 4*E^2
(%i23) factor(VP(p1-p3,p1-p3));
(%o23) 2*(cos(th)-1)*E^2
```

12.4.7 Contraction of a Product of Traces: Con, scon, mcon, econ

In the example of the high energy scattering of electrons in the center of momentum frame, worked out in the batch file **moller1.mac** later, we need the contraction of the product of two traces, followed by a reduction to the kinematic variables.

We will obtain a correct answer three different ways here.

METHOD1: We first find the symbolic traces, and then the symbolic contractions, followed by use of **noncov**. We call this **method1**.

```
(%i1) load(dirac2);
 [launch data edited out here]
(%o1) "c:/work5/dirac2.mac"
(%i2) assume(E > 0,th >= 0,th <= %pi)$
(%i3) comp_def(p1(E,0,0,E),p2(E,0,0,-E),p3(E,E*sin(th),0,E*cos(th)),
                p4(E,-E*sin(th),0,-E*cos(th)))$
(%i4) tr1 : tr(p3,mu,p1,nu);
(%o4) 4*UI(p1,mu)*UI(p3,nu)+4*UI(p1,nu)*UI(p3,mu)-4*Gm(mu,nu)*D(p1,p3)
(%i5) tr2 : tr(p4,mu,p2,nu);
(%o5) 4*UI(p2,mu)*UI(p4,nu)+4*UI(p2,nu)*UI(p4,mu)-4*Gm(mu,nu)*D(p2,p4)
(%i6) method1a : scon (tr1*tr2);
(%o6) 32*D(p1,p2)*D(p3,p4)+32*D(p1,p4)*D(p2,p3)
(%i7) method1b : Con (tr1*tr2);
(%o7) 32*D(p1,p2)*D(p3,p4)+32*D(p1,p4)*D(p2,p3)
(%i8) method1c : Con (tr1*tr2,mu,nu);
(%o8) 32*D(p1,p2)*D(p3,p4)+32*D(p1,p4)*D(p2,p3)
(%i9) method1 : noncov (method1a);
(%o9) 32*cos(th)^2*E^4+64*cos(th)*E^4+160*E^4
```

METHOD2: We use **nc_tr**. This is a special function which is not just **tr** followed by **noncov**, but rather as **TR1** generates the trace of each term of the initial argument expansion, the function **noncov** is immediately applied to the trace result, and the output is accumulated in a sum which is eventually returned by **nc_tr**.

The contraction of uncontracted repeated Lorentz indices can then be effected using **mcon**, or **econ** (which **Con** should call automatically by recognising the typical after-effects of **noncov**).

We will call this **method2**. We will see that method2 produces an apparently different trigonometric result, but use of **trigsimp**, followed by **expand**, results finally in the same answer as found for method1.

```
(%i10) nctr1 : nc_tr(p3,mu,p1,nu);
(%o10) 4*gmet[mu,nu]*cos(th)*E^2-4*gmet[mu,nu]*E^2+4*p1[mu]*p3[nu]
                                +4*p3[mu]*p1[nu]
(%i11) nctr2 : nc_tr(p4,mu,p2,nu);
(%o11) 4*gmet[mu,nu]*cos(th)*E^2-4*gmet[mu,nu]*E^2+4*p2[mu]*p4[nu]
                                +4*p4[mu]*p2[nu]
(%i12) method2a : mcon (nctr1*nctr2,mu,nu);
(%o12) 64*sin(th)^2*E^4+96*cos(th)^2*E^4+64*cos(th)*E^4+96*E^4
(%i13) method2b : econ (nctr1*nctr2,mu,nu);
(%o13) 64*sin(th)^2*E^4+96*cos(th)^2*E^4+64*cos(th)*E^4+96*E^4
(%i14) method2c : Con (nctr1*nctr2,mu,nu);
(%o14) 64*sin(th)^2*E^4+96*cos(th)^2*E^4+64*cos(th)*E^4+96*E^4
(%i15) method2 : expand (trigsimp (method2a));
(%o15) 32*cos(th)^2*E^4+64*cos(th)*E^4+160*E^4
(%i16) method2 - method1;
(%o16) 0
```

We now obtain the same answer using the explicit matrix route, calling this **method3**. In this case, **Con** will call **mcon** for the contraction, and again, we must supply the desired contraction indices. Using the more compact (but still explicit matrix) syntax of **m_tr**, we call the result **method3a**:

```
(%i17) method3a : Con(m_tr(p3,mu,p1,nu)*m_tr(p4,mu,p2,nu),mu,nu);
(%o17) 64*sin(th)^2*E^4+96*cos(th)^2*E^4+64*cos(th)*E^4+96*E^4
(%i18) method3a : expand (trigsimp (%));
(%o18) 32*cos(th)^2*E^4+64*cos(th)*E^4+160*E^4
(%i19) method3a - method1;
(%o19) 0
```

Now, using the transparently explicit matrix route with **mat_trace**, we call the result **method3b**.

```
(%i20) method3b : Con(mat_trace(sL(p3) . Gam[mu] . sL(p1) . Gam[nu])
                      *mat_trace(sL(p4) . Gam[mu] . sL(p2) . Gam[nu]),mu,nu);
(%o20) 64*sin(th)^2*E^4+96*cos(th)^2*E^4+64*cos(th)*E^4+96*E^4
(%i21) method3b : expand (trigsimp (%));
(%o21) 32*cos(th)^2*E^4+64*cos(th)*E^4+160*E^4
(%i22) method3b - method1;
(%o22) 0
```

As a passing remark, it is easy to verify two properties of **Gam[5]** using explicit matrices. First, **Gam[5]** has zero trace, and then that the matrix product of **Gam[5]** with itself results in the unit **4 x 4** matrix:

```
(%i23) mat_trace(Gam[5]);
(%o23) 0
(%i24) is (equal (Gam[5] . Gam[5],I4));
(%o24) true
```

12.4.8 Contraction Choices: Timing Comparisons for Polarized Squared Amplitudes

In the examples above, as well as in the section dealing with **moller3.mac**, which looks for consistency in different methods of calculation of the square of polarized amplitudes in Moller scattering of finite mass polarized leptons, we have examples which compare matrix methods (**m_tr**) with symbolic methods (**nc_tr**).

Here we present an example which shows that the symbolic route will take about **3 - 7** times as long as the explicit matrix route, depending on the contraction function chosen.

The quickest symbolic contraction route is to force the use of **mcon** as the contraction function (in the context of the contraction of a product of **nc_tr's**).

Forcing the use of **econ** almost doubles the time required (as compared with using **mcon**), whereas using **Con** (which interposes some intermediate poking at the expression to be contracted) increases the length of time over the forced use of **econ** (which **Con** eventually calls).

We have commented out most of **moller3.mac** and used just the top section to set up the invariants and kinematics. The work starting with **(%i27)** is interactive work.

```
(%i3) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch("moller3.mac");
read and interpret file: #pc:/work5/moller3.mac
(%i3) " moller3.mac "
(%i4) "*****************************************"
(%i5) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-09

(%i6) " Maxima by Example, Ch. 12 "
(%i7) " Dirac Algebra and Quantum Electrodynamics "
(%i8) " Edwin L. Woollett "
```

```
(%i9) " http://www.csulb.edu/~woollett  "
(%i10) " woollett@charter.net "
(%i11) "SQUARED POLARIZED AMPLITUDES VIA SYMBOLIC AND MATRIX TRACE METHODS"
(%i12) "  FOR ARBITRARY ENERGY MOLLER SCATTERING  "
(%i13) "  e(-,p1,sv1) + e(-,p2,sv2) --> e(-,p3,sv3) + e(-,p4,sv4) "
(%i14) " ------------------------------------------------------"
(%i15) invar(D(p1,p1) = m^2,D(p2,p2) = m^2,D(p3,p3) = m^2,D(p4,p4) = m^2,
             D(p1,Sp1) = 0,D(Sp1,Sp1) = -1,D(p2,Sp2) = 0,D(Sp2,Sp2) = -1,
             D(p3,Sp3) = 0,D(Sp3,Sp3) = -1,D(p4,Sp4) = 0,D(Sp4,Sp4) = -1)
(%i16) comp_def(p1(E,0,0,p),Sp1(p/m,0,0,E/m),p2(E,0,0,-p),Sp2(p/m,0,0,(-E)/m),
                p3(E,p*sin(th),0,p*cos(th)),
                Sp3(p/m,E*sin(th)/m,0,E*cos(th)/m),
                p4(E,-p*sin(th),0,-p*cos(th)),
                Sp4(p/m,(-E*sin(th))/m,0,(-E*cos(th))/m))
(%i17) p_Em(expr):=expand(ratsubst(E^2-m^2,p^2,expr))
(%i18) E_pm(expr):=expand(ratsubst(m^2+p^2,E^2,expr))
(%i19) s_th:VP(p2+p1,p2+p1)
(%o19) 4*E^2
(%i20) t_th:VP(p1-p3,p1-p3)
(%o20) 2*p^2*cos(th)-2*p^2
(%i21) u_th:VP(p1-p4,p1-p4)
(%o21) -2*p^2*cos(th)-2*p^2
(%i22) t_th2:to_ao2(t_th,th)
(%o22) -4*p^2*sin(th/2)^2
(%i23) u_th2:to_ao2(u_th,th)
(%o23) 4*p^2*sin(th/2)^2-4*p^2
(%i24) t_thE:p_Em(t_th)
(%o24) 2*cos(th)*E^2-2*E^2-2*m^2*cos(th)+2*m^2
(%i25) u_thE:p_Em(u_th)
(%o25) -2*cos(th)*E^2-2*E^2+2*m^2*cos(th)+2*m^2
(%o26) "moller3.mac"
(%i27) /* case RR --> LL  */
         [sv1,sv2,sv3,sv4]:[1,1,-1,-1]$
(%i28) /* explicit matrix method always faster */
        M1n_m : trigsimp (E_pm (mcon ( m_tr (S(sv3,Sp3),p3+m,mu,S(sv1,Sp1),p1+m,nu)*
                 m_tr (S(sv4,Sp4),p4+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu)));
(%o28) -4*m^4*sin(th)^2-8*m^4*cos(th)+8*m^4
(%i29) time(%);
(%o29) [1.11]
(%i30) /* forcing use of econ with symbolic method --> six times
                 as long as explicit matrix method  */
       M1n_s : trigsimp (E_pm (econ ( nc_tr (S(sv3,Sp3),p3+m,mu,S(sv1,Sp1),p1+m,nu)*
                 nc_tr (S(sv4,Sp4),p4+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu)));
(%o30) -4*m^4*sin(th)^2-8*m^4*cos(th)+8*m^4
(%i31) time(%);
(%o31) [6.25]
(%i32)  /* forcing use of mcon here is the fastest symbolic method */
         M1n_sm : trigsimp (E_pm (mcon ( nc_tr (S(sv3,Sp3),
            p3+m,mu,S(sv1,Sp1),p1+m,nu)*nc_tr (S(sv4,Sp4),
              p4+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu)));
(%o32) -4*m^4*sin(th)^2-8*m^4*cos(th)+8*m^4
(%i33) time(%);
(%o33) [3.03]
(%i34) /* letting Con choose method here ends up with choosing econ,
           and Con's overhead results in the slowest symbolic method */
        M1n_sc : trigsimp (E_pm (Con ( nc_tr (S(sv3,Sp3),
            p3+m,mu,S(sv1,Sp1),p1+m,nu)*nc_tr (S(sv4,Sp4),
              p4+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu)));
(%o34) -4*m^4*sin(th)^2-8*m^4*cos(th)+8*m^4
(%i35) time(%);
(%o35) [7.22]
```

```
(%i36) M1n_s - M1n_m;
(%o36) 0
(%i37) M1n_sm - M1n_m;
(%o37) 0
(%i38) M1n_sc - M1n_m;
(%o38) 0
```

12.4.9 Dirac Spinors and Helicity Amplitudes

Finally, we present an example of an expression constructed from **Dirac spinors** and matrices for given helicity quantum numbers.

We generate an explicit Dirac "spinor", a four element column matrix, denoted symbolically by $\mathbf{u}(\mathbf{p},\boldsymbol{\sigma})$ 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 is described by 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**.

We normally describe scattering in the **z−x** plane, and if $\mathbf{p}_1$ makes an angle **th <= %pi** with the positive **z** axis and we are describing "positive" helicity (**sv = 1**) (or "righthanded, R") we would use the syntax
up1 : UU(E1,p1,th,0,1).

If a different particle is moving in the **z−x** plane in a direction opposite to **p1** with 3-momentum **p2** and energy **E2**, we would use the syntax (taking **sv2 = +1**)
up2 : UU (E2,p2,%pi − th, %pi, 1). For a negative helicity particle, we would use **sv = −1**.

A barred particle spinor (a row matrix) is generated by using **sbar** which has the definition
sbar (_uu%) := hc (_uu%) . Gam[0]$

in which **hc** is the package defined hermitian conjugate function.

We would use the syntax **up1b : sbar (UU(E1,p1,th,0,1))** for example.

To generate the antiparticle spinor, denoted symbolically by $\mathbf{v}(\mathbf{p},\boldsymbol{\sigma})$, for an antiparticle of energy **E** and 3-momentum **p** we use the function **VV(E,p,theta,phi,sv)**.

We normally describe scattering in the **z−x** plane, and if $\mathbf{p}_3$ makes an angle **th <= %pi** with the positive **z** axis and we are describing "positive" helicity (**sv = 1**) (or "righthanded, R") we would use the syntax
vp3 : VV(E3,p3,th,0,1). For a negative helicity antiparticle, we would use **sv = −1**.

If a different antiparticle is moving in the **z−x** plane in a direction opposite to $\mathbf{p}_3$ with 3-momentum **p4** and energy **E4**, we would use the syntax (taking for example **sv4 = +1**)
vp4 : VV (E4,p4,%pi − th, %pi, 1).

A barred antiparticle spinor (a row matrix) is generated by using **sbar** with the syntax
vp3b : sbar (VV (E3,p3,th,0,1)) for example.

12.4.10 Polarized Amplitudes for (electron,positron) → (muon, antimuon)

High Energy Limit Dirac Spinor Polarized Amplitudes

In the following example, we construct spinors needed to calculate the matrix element for the **RL --> RL** process of the process (electron, positron) **-->** (muon, antimuon),
e(-,p1,1) + e(+,p2,-1) --> mu(-,p3,1) + mu(+,p4,-1), in the high energy limit in which we neglect the mass of the particles compared with their relativistic energy. (See Peskin/Schroeder, p.131 ff for details.)

The incident electron has attributes **p1**,**sv1 = 1** and is represented by the spinor **up1** (a column vector), the incident positron has attributes **p2**, **sv2 = -1** and is represented by the barred antiparticle spinor **vp2b** (a row vector). The outgoing muon has attributes **p3** and **sv3 = 1** and is represented by the barred particle spinor **up3b** (a row vector), the outgoing antimuon has attributes **p4** and **sv4 = -1** and is represented by the antiparticle spinor **vp4** (a column vector).

Since in the high energy limit the energy and 3-momentum magnitude are the same (ignoring mass), we replace **p** by **E**, and we are working in the center of momentum frame, so each "particle" has the same energy.

```
(%i1) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) comp_def (p1(E,0,0,E),
          p2(E,0,0,-E),
          p3(E,E*sin(th),0,E*cos(th)),
          p4(E,-E*sin(th),0,-E*cos(th)))$
(%i3) listarray(p3);
(%o3) [E,sin(th)*E,0,cos(th)*E]
(%i4) up1 : UU(E,E,0,0,1);
(%o4) matrix([0],[0],[sqrt(2*E)],[0])
(%i5) vp2b : sbar (VV (E,E,%pi,0,-1));
(%o5) matrix([0,-sqrt(2*E),0,0])
(%i6) a12 : vp2b.Gam[mu].up1;
(%o6) matrix([0,-sqrt(2*E),0,0]) . Gam[mu] . matrix([0],[0],[sqrt(2*E)],[0])
(%i7) up3b : sbar (UU (E,E,th,0,1));
(%o7) matrix([cos(th/2)*sqrt(2*E),sin(th/2)*sqrt(2*E),0,0])
(%i8) vp4 : VV (E,E,%pi-th,%pi,-1);
(%o8) matrix([0],[0],[-cos((%pi-th)/2)*sqrt(2*E)],[sin((%pi-th)/2)*sqrt(2*E)])
(%i9) a34 : up3b . Gam[mu] . vp4;
(%o9) matrix([cos(th/2)*sqrt(2*E),sin(th/2)*sqrt(2*E),0,0])
        . Gam[mu]
        . matrix([0],[0],[-cos((%pi-th)/2)*sqrt(2*E)],
                 [sin((%pi-th)/2)*sqrt(2*E)])
```

Both **a12** and **a34** are (in general) complex numbers. Ignoring the factor **-e^2**, the **numerator Mn** of the transition amplitude is given by the contraction of the product **a12*a34** on the index **mu**.

```
(%i10) Mn : mcon (a12*a34,mu);
(%o10) 8*cos(th/2)^2*E^2
```

The **denominator** of the transition amplitude is called **s_th** and is the 4-vector dot product of the sum of the incident 4-momenta:

```
(%i11) s_th : VP (p1 + p2,p1 + p2);
(%o11) 4*E^2
```

The amplitude **Amp_RL_RL** (again, we are ignoring **-e^2**) is then

```
(%i12) Amp_RL_RL : Mn/s_th;
(%o12) 2*cos(th/2)^2
```

Note that this amplitude is real.

As an aside, in the high energy limit the spinor normalizations are zero. For example:

```
(%i13) sbar(up1) . up1;
(%o13) 0
```

If we redefine **up1** for the arbitrary energy case, in which mass cannot be neglected, then the normalization becomes **2*m**, where **m** is the mass of the particle.

```
(%i14) up1 : UU(E,p,0,0,1);
(%o14) matrix([sqrt(E-p)],[0],[sqrt(E+p)],[0])
(%i15) sbar (up1) . up1;
(%o15) 2*sqrt(E-p)*sqrt(E+p)
(%i16) rootscontract (%);
(%o16) 2*sqrt(E^2-p^2)
(%i17) ratsubst (m, sqrt(E^2-p^2), %);
(%o17) 2*m
```

which goes to zero as **m** goes to zero.

Trace Methods and High Energy Limit Helicity Projection Matrices and Operators

We can check the **square** of the polarized amplitude calculated above (using Dirac spinors for given helicity assignments) by using either the **mat_trace** method, the **m_tr** method, or the purely symbolic method. The matrix method using **m_tr** syntax is designed to have arguments which look the same as the purely symbolic case.

We first use the **mat_trace** method using explicit Dirac matrices. For the helicity dependent matrix element squared via trace methods, helicity projection operators must be inserted into the traces. We use **P** for the explicit helicity projection matrix, which takes two forms, depending on whether massive or (formally) massless particles are considered. In the following, **P(sv)** is the **massless case** helicity projection matrix which becomes **(I4 + sv*Gam[5])/2, and \mv**sv—takes values **1, -1**.

```
(%i18) is (equal (P(1), (I4 + Gam[5])/2 ));
(%o18) true
(%i19) is (equal (P(-1), (I4 - Gam[5])/2 ));
(%o19) true
```

In the **massless limit P(+1)** picks out **negative** physical helicity case (**sv = -1**) if we are dealing with an **antiparticle**.

```
(%i20) Mn_sq:(a12:mat_trace(P(1) . sL(p2)
                                . Gam[mu] . P(1) . sL(p1) . Gam[nu]),
              a34:mat_trace(P(1) . sL(p3)
                                . Gam[mu] . P(1) . sL(p4) . Gam[nu]),
              mcon (a12*a34,mu,nu),factor(%%));
(%o20) 16*(cos(th)+1)^2*E^4
(%i21) time(%);
(%o21) [0.03]
```

To compare with the absolute value squared of the transition matrix element calculated above using Dirac spinors, we convert the a function of **th/2** using the Dirac package function **to_ao2 (expr,a)**, where the symbol for the angle must be correctly inserted.

```
(%i22) Mn_sq : to_ao2 (Mn_sq, th);
(%o22) 64*cos(th/2)^4*E^4
```

Dividing this "numerator squared" expression by **s_th^2**, we get what should be the same as **Amp_RL_RL^2** (since **Amp_RL_RL** is real):

```
(%i23) M_sq : Mn_sq/s_th^2;
(%o23) 4*cos(th/2)^4
(%i24) M_sq - Amp_RL_RL^2;
(%o24) 0
```

which shows agreement.

We next use the **m_tr** method, which is translated into an explicit **mat_trace** expression by **m_tr**, including the replacement of **S** by **P**. The virtue of the **m_tr** method is simply that the notation is formally the same as is used in the symbolic **tr** case.

```
(%i25) Mn_sq:(a12 : m_tr (S(1),p2,mu,S(1),p1,nu),
              a34 : m_tr (S(1),p3,mu,S(1),p4,nu),
                mcon (a12*a34,mu,nu), factor(%%));
(%o25) 16*(cos(th)+1)^2*E^4
(%i26) time(%);
(%o26) [0.02]
(%i27) Mn_sq : to_ao2 (Mn_sq, th);
(%o27) 64*cos(th/2)^4*E^4
(%i28) M_sq : Mn_sq/s_th^2;
(%o28) 4*cos(th/2)^4
(%i29) M_sq-Amp_RL_RL^2;
(%o29) 0
```

which again shows agreement.

For a purely symbolic check, the use of a **nc_tr** method is recommended, since the helicity projection operators **S(1)** and **S(-1)** eventually cause the introdution of two **Eps** factors which the present version of the Dirac package does not symbolically reduce (ie **scon** does not effect the contraction of a product of **Eps** containing contraction indices). For example,

```
(%i30) scon (LI(p1,N4)*LI(p2,N3)*Eps(N3,mu,N4,nu)*
               LI(p3,N7)*LI(p4,N8)*Eps(N7,mu,N8,nu), mu,nu);
(%o30) LI(p1,N4)*LI(p2,N3)*LI(p3,N7)*LI(p4,N8)*Eps(N3,mu,N4,nu)
                *Eps(N7,mu,N8,nu)
```

Here is the recommended symbolic approach, in which the immediate use of **nc_tr** allows **mcon** to do the contraction. The factor **S(sv) --> (1 + sv*G5)/2**, effectively, is the massless helicity projection operator for the symbolic method.

```
(%i31) Mn_sq:(a12 : nc_tr (S(1),p2,mu,S(1),p1,nu),
              a34 : nc_tr (S(1),p3,mu,S(1),p4,nu),
              mcon (a12*a34,mu,nu),
              factor(%%));
(%o31) -16*(sin(th)^2-2*cos(th)-2)*E^4
(%i32) time(%);
(%o32) [0.34]
(%i33) Mn_sq : to_ao2 (Mn_sq,th);
(%o33) 64*cos(th/2)^4*E^4
(%i34) M_sq : Mn_sq/s_th^2;
(%o34) 4*cos(th/2)^4
(%i35) M_sq - Amp_RL_RL^2;
(%o35) 0
```

In terms of time, the explicit matrix approach (**m_tr**) to these polarized squared amplitudes is about five times faster than the **nc_tr** symbolic approach.

Just as an experiment, the **non-recommended and slow** calculation, in which we apply **noncov** to the product of two expressions produced by **tr**, and then use **mcon** to contract, looks like this:

```
(%i36) Mn_sq : (a12 : tr(S(1),p2,mu,S(1),p1,nu),
         a34 : tr(S(1),p3,mu,S(1),p4,nu),
                 noncov (a12*a34),
                 mcon (%%,mu,nu),factor(%%));
(%o36) -16*(sin(th)^2-2*cos(th)-2)*E^4
(%i37) time(%);
(%o37) [3.89]
```

which took about eleven times as long as the recommended **nc_tr** method. Slightly longer times arise with the explicit use of **econ** or **Con**:

```
(%i38) Mn_sq : (a12 : tr(S(1),p2,mu,S(1),p1,nu),
         a34 : tr(S(1),p3,mu,S(1),p4,nu),
                 noncov (a12*a34),
                 econ (%%,mu,nu),factor(%%));
(%o38) -16*(sin(th)^2-2*cos(th)-2)*E^4
(%i39) time(%);
(%o39) [3.98]
(%i40) Mn_sq : (a12 : tr(S(1),p2,mu,S(1),p1,nu),
         a34 : tr(S(1),p3,mu,S(1),p4,nu),
                 noncov (a12*a34),
                 Con (%%,mu,nu),factor(%%));
(%o40) -16*(sin(th)^2-2*cos(th)-2)*E^4
(%i41) time(%);
(%o41) [4.01]
```

Arbitrary Energy Case (e e → mu mu) of Polarized Amplitudes via Dirac Spinors

We continue to examine the same physical process (**e(-) e(+) --> mu(-) mu(+)**) but for arbitary center of momentum frame energy, in which case we must take into account the mass of the electron **m** and the mass of the muon **M**.

We use (as before) the symbols **p3,p4** for the final muon 4-momenta, but now the symbol **k** for the magnitude of a final muon 3-momentum, and the symbol **p** for the magnitude of an initial electron 3-momentum. The overall process description is:
e(-,p1,sv1) + e(+,p2,sv2) --> mu(-,p3,sv3) + mu(+,p4,sv4).

The total relativistic energy of the initial electron and positron, **2*E**, is shared equally (center of momentum frame) and passed on to the the final muon and antimuon by energy conservation, and again shared equally, so each "particle" has relativistic energy **E**.

Thus **E = sqrt(p^2+m^2) = Emu = sqrt(k^2 + M^2)**.

```
(%i42) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) comp_def(p1(E,0,0,p),
               p2(E,0,0,-p),
               p3(E,k*sin(th),0,k*cos(th)),
               p4(E,-k*sin(th),0,-k*cos(th)))$
```

The **denominator** of the transition amplitude is called **s_th** (the square of the total CMS energy) and is the 4-vector dot product of the sum of the incident 4-momenta:

```
(%i3) s_th : VP (p1 + p2,p1 + p2);
(%o3) 4*E^2
```

As in the high energy case, the incident electron has attributes **p1**,**sv1 = 1** and is represented by the spinor **up1** (a column vector), the incident positron has attributes **p2**, **sv2 = -1** and is represented by the barred antiparticle spinor **vp2b** (a row vector). The outgoing muon has attributes **p3** and **sv3 = 1** and is represented by the barred particle spinor **up3b** (a row vector), the outgoing antimuon has attributes **p4** and **sv4 = -1** and is represented by the antiparticle spinor **vp4** (a column vector).

```
(%i4) up1 : UU(E,p,0,0,1);
(%o4) matrix([sqrt(E-p)],[0],[sqrt(E+p)],[0])
(%i5) vp2b : sbar(VV(E,p,%pi,0,-1));
(%o5) matrix([0,-sqrt(E+p),0,sqrt(E-p)])
(%i6) a12 : vp2b . Gam[mu] . up1$
(%i7) up3b : sbar(UU(E,k,th,0,1));
(%o7) matrix([cos(th/2)*sqrt(E+k),sin(th/2)*sqrt(E+k),cos(th/2)*sqrt(E-k),
              sin(th/2)*sqrt(E-k)])
(%i8) vp4 : VV(E,k,%pi-th,%pi,-1);
(%o8) matrix([cos((%pi-th)/2)*sqrt(E-k)],[-sin((%pi-th)/2)*sqrt(E-k)],
             [-cos((%pi-th)/2)*sqrt(E+k)],[sin((%pi-th)/2)*sqrt(E+k)])
(%i9) a34 : up3b . Gam[mu] . vp4$
```

Both **a12** and **a34** are (in general) complex numbers. Ignoring the factor **-e^2**, the **numerator Mn** of the transition amplitude is given by the contraction of the product **a12*a34** on the index **mu**, and, as before, the amplitude **Amp_RL_RL** (again, we are ignoring **-e^2**) is **Mn/s_th**.

```
(%i10) Mn : mcon (a12*a34,mu);
(%o10) 8*cos(th/2)^2*E^2
(%i11) Amp_RL_RL : Mn/s_th;
(%o11) 2*cos(th/2)^2
```

which is the same as the high energy limit previously discussed for this process.

Trace Methods and Arbitrary Energy Case Helicity Projection Matrices and Operators

The symbol **Sp1** is the positive helicity spin 4-vector defined by the incident electron 4-momentum vector **p1**. The symbol **Sp2** is the positive helicity spin 4-vector defined by the incident positron 4-momentum vector **p2**. The symbol **Sp3** is the positive helicity spin 4-vector defined by the final muon 4-momentum vector **p3**. The symbol **Sp4** is the positive helicity spin 4-vector defined by the final antimuon 4-momentum vector **p4**.

```
(%i12) tr(S(1,Sp1));
(%o12) 2
```

```
(%i13) comp_def(Sp1(p/m,0,0,E/m),
                Sp2(p/m,0,0,(-E)/m),
                Sp3(k/M,E*sin(th)/M,0,E*cos(th)/M),
                Sp4(k/M,(-E*sin(th))/M,0,(-E*cos(th))/M))$
(%i14) p_Em(expr) := expand (ratsubst (E^2-m^2,p^2,expr))$
(%i15) k_EM(expr) := expand (ratsubst (E^2-M^2,k^2,expr))$
```

In the contraction result, we can simplify by replacing **k^2** and **p^2** in terms of the equivalent quantities depending on **E** and either **m** or **M**.

First the **mat_trace** method. In the explicit matrix method, a matrix factor **P(sv1,Sp1)**, turns into the matrix **(I4 + sv1*Gam[5].sL(Sp1))/2**, in which **Sp1** is the positive helicity spin 4-vector implied by the particle's 4-momentum **p1**.

```
(%i16) is (equal (P(1,Sp1),(I4 + Gam[5].sL(Sp1))/2));
(%o16) true
(%i17) is (equal (P(-1,Sp1),(I4 - Gam[5].sL(Sp1))/2));
(%o17) true
```

The matrix **P(sv1,Sp1)** is the matrix version of the finite mass spin projection operator, and requires a frame dependent definition (using **comp_def**) of the components of the positive helicity spin 4-vector corresponding to a particle's 4-momentum.

```
(%i18) a12 : mat_trace(Gam[mu] . P(1,Sp1) . (sL(p1) + m*I4)
                       . Gam[nu] . P(-1,Sp2) . (sL(p2) - m*I4))$
(%i19) a34 : mat_trace (Gam[mu] . P(-1,Sp4) . (sL(p4) - M*I4)
                    . Gam[nu] . P(1,Sp3) . (sL(p3) + M*I4))$
(%i20) Mn_sq : (mcon (a12*a34,mu,nu),
                  k_EM(%%),p_Em(%%),
                  expand(trigsimp(%%)),factor(%%));
(%o20) 16*(cos(th)+1)^2*E^4
(%i21) time(%);
(%o21) [0.58]
(%i22) Mn_sq : to_ao2 (Mn_sq, th);
(%o22) 64*cos(th/2)^4*E^4
(%i23) M_sq : Mn_sq/s_th^2;
(%o23) 4*cos(th/2)^4
```

The **mat_trace** method requires careful attention to parentheses, and it is simpler (and also faster) to use the equivalent **m_tr** method, which uses the same operator notation as **tr**, but actually translates everything into a **mat_trace** expression.

```
(%i24) a12 : m_tr (mu,S(1,Sp1),p1 + m,nu,S(-1,Sp2),p2 - m)$
(%i25) a34 : m_tr (mu,S(-1,Sp4),p4 - M,nu,S(1,Sp3),p3 + M)$
(%i26) Mn_sq : (mcon (a12*a34,mu,nu),
                  k_EM(%%),p_Em(%%),
                  expand(trigsimp(%%)),factor(%%));
(%o26) 16*(cos(th)+1)^2*E^4
(%i27) time(%);
(%o27) [0.59]
(%i28) Mn_sq : to_ao2 (Mn_sq, th);
(%o28) 64*cos(th/2)^4*E^4
(%i29) M_sq : Mn_sq/s_th^2;
(%o29) 4*cos(th/2)^4
```

Finally we use the symbolic **nc_tr** method, which is a specially designed function which immediately takes **noncov** of the trace of each term of the general expansion of the starting expression.

The symbolic method uses the helicity projection operators such as **S(sv1,Sp1)**, used for a particle with helicity quantum number **sv1 = +/- 1** and 4-momentum **p1** and corresponding positive helicity spin vector **Sp1** defined by **p1**. In the symbolic method, a factor **S(sv1,Sp1)**, turns into the symbolic factor
(1 + sv1*G5*Sp1)/2, in which **Sp1** is the positive helicity spin 4-vector implied by the particle's 4-momentum **p1**.

```
(%i36) tr(S(1,Sp1));
(%o36) 2
(%i37) tr(S(-1,Sp1));
(%o37) 2
(%i38) tr(mu,S(1,Sp1));
(%o38) 0
(%i39) tr(mu,nu,rh,G5,Sp1);
(%o39) 4*%i*Eps(mu,nu,rh,N4)*LI(Sp1,N4)
(%i40) tr(mu,nu,rh,S(1,Sp1));
(%o40) 2*%i*Eps(mu,nu,rh,N5)*LI(Sp1,N5)
(%i41) tr(mu,nu,rh,S(-1,Sp1));
(%o41) -2*%i*Eps(mu,nu,rh,N6)*LI(Sp1,N6)
```

We first populate the list **invarR** used by the symbolic method for the immediate evaluation of 4-vector dot products such as **D(p1,p1)**, which is done by **tr**.

```
(%i42) invarR;
(%o42) []
(%i43) invar(D(p1,p1) = m^2,
             D(p2,p2) = m^2,
             D(p3,p3) = M^2,
             D(p4,p4) = M^2,
             D(p1,Sp1) = 0,
             D(Sp1,Sp1) = -1,
             D(p2,Sp2) = 0,
             D(Sp2,Sp2) = -1,
             D(p3,Sp3) = 0,
             D(Sp3,Sp3) = -1,
             D(p4,Sp4) = 0,
             D(Sp4,Sp4) = -1)$
(%i44) invarR;
(%o44) [D(Sp4,Sp4) = -1,D(p4,Sp4) = 0,D(Sp3,Sp3) = -1,D(p3,Sp3) = 0,
        D(Sp2,Sp2) = -1,D(p2,Sp2) = 0,D(Sp1,Sp1) = -1,D(p1,Sp1) = 0,
        D(p4,p4) = M^2,D(p3,p3) = M^2,D(p2,p2) = m^2,D(p1,p1) = m^2]
(%i45) Mn_sq:(nc_tr(mu,S(1,Sp1),m+p1,nu,S(-1,Sp2),p2-m),a12:p_Em(%%),
              nc_tr(mu,S(-1,Sp4),p4-M,nu,S(1,Sp3),M+p3),a34:k_EM(%%),
              mcon(a12*a34,mu,nu),k_EM(%%),p_Em(%%),trigsimp(%%),factor(%%));
(%o45) -16*(sin(th)^2-2*cos(th)-2)*E^4
(%i46) time(%);
(%o46) [1.43]
(%i47) Mn_sq:to_ao2(Mn_sq,th);
(%o47) 64*cos(th/2)^4*E^4
(%i48) M_sq:Mn_sq/s_th^2;
(%o48) 4*cos(th/2)^4
```

We now proceed to use a batch file method to examine a series of basic simple examples of quantum electrodynamics using our tools.

12.5 moller0.mac: Scattering of Identical Scalar (Spin 0) Charged Particles

The batch file **moller0.mac** treats the scattering event

$$\pi^+(p_1) + \pi^+(p_2) \rightarrow \pi^+(p_3) + \pi^+(p_4) \tag{12.38}$$

With the definitions

$$s = (p_1 + p_2)^2, \quad t = (p_1 - p_3)^2, \quad u = (p_1 - p_4)^2, \tag{12.39}$$

The symbolic method is used to derive the unpolarized differential cross section, starting with the invariant amplitude $M = M_1 + M_2$, in which

$$M_1 = -\frac{e^2\,(p_1 + p_3)\cdot(p_2 + p_4)}{t}, \tag{12.40}$$

and

$$M_2 = -\frac{e^2\,(p_1 + p_4)\cdot(p_2 + p_3)}{u}. \tag{12.41}$$

The Feynman rules for scalar electrodynamics are discussed in: 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.

The list **invarR** (rules for invariant dot products) is first populated using the Dirac package function **invar**, which will be used by **tr** to replace invariant dot products such as **D(p1,p1)**, **D(p1,p2)**, etc by expressions in terms of the mass **m** of the scalar particle, and the Mandelstam variables **s, t, u**.

We then use **comp_def** to define the components of the 4-vectors in the center of momentum (CM) frame, and evaluate the Mandelstam variables in that frame.
Some other Dirac package defined functions used in **moller0.mac** are **take_parts**, **ev_Ds**, **pullfac**, **sub_stu**, and **fr_ao2**.

The batch file **moller0.mac** is used after loading in the Dirac package.

```
(%i1) load(dirac2);
                                  dirac2.mac
                              simplifying-new.lisp
                                  dgtrace2.mac
                                   dgcon2.mac
                                   dgeval2.mac
                                 dgmatrix2.mac
              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, G, G(1), G5, G5p, Gam, Gm, Gtr, LI, Nlast, UI, P, S, Sig
              UU, VP, VV, I2, Z2, CZ2, I4, Z4, CZ4, RZ4, N1,N2,...
                         reserved array names: gmet, eps4
                                 invar_flag = true
                                 stu_flag = false
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("moller0.mac");
read and interpret file: #pc:/work5/moller0.mac
(%i3) " ======================================================"
(%i4) "  file moller0.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) " ======================================================"
(%i11) " ELASTIC SCATTERING OF SPIN 0 PARTICLES (SAME CHARGE)"
(%i12) " FOR EXAMPLE: "
(%i13) "      PI(+,p1) PI(+,p2)  --->  PI(+,p3) PI(+,p4)       "
(%i14) " POPULATE THE LIST invarR OF 4-VEC DOT PRODUCT VALUES, "
(%i15) " Using p1 + p2 = p3 + p4, s = (p1+p2)^2 = (p3+p4)^2 ,"
(%i16) "  t = (p1-p3)^2 = (p2-p4)^2,"
(%i17) "   and u = (p1-p4)^2 = (p2-p3)^2 "
(%i18) " ---------------------------------------------"
(%i19) invar(D(p1,p1) = m^2,D(p1,p2) = s/2-m^2,D(p1,p3) = m^2-t/2,
             D(p1,p4) = m^2-u/2,D(p2,p2) = m^2,D(p2,p3) = m^2-u/2,
             D(p2,p4) = m^2-t/2,D(p3,p3) = m^2,D(p3,p4) = s/2-m^2,
             D(p4,p4) = m^2)
(%i20) "-----------------------------------------------------"
(%i21) " factor out -e^2 from Mfi, leaving Mfi = M1 + M2 "
(%i22) M1:D(p3+p1,p4+p2)/D(p1-p3,p1-p3)
(%i23) M2:D(p4+p1,p3+p2)/D(p1-p4,p1-p4)
(%i24) M1:ev_Ds(M1)
(%o24) s/t-u/t
```

```
(%i25) M1:pullfac(M1,1/t)
(%o25) (s-u)/t
(%i26) M2:ev_Ds(M2)
(%o26) s/u-t/u
(%i27) M2:pullfac(M2,1/u)
(%o27) (s-t)/u
(%i28) Mfi:M2+M1
(%o28) (s-u)/t+(s-t)/u
(%i29) " we get the result Mfi = (s-u)/t + (s-t)/u  "
(%i30) "CM FRAME EVALUATION "
(%i31) assume(p > 0,th >= 0,th <= %pi)
(%i32) comp_def(p1(E,0,0,p),p2(E,0,0,-p),p3(E,p*sin(th),0,p*cos(th)),
               p4(E,-p*sin(th),0,-p*cos(th)))
(%i33) s_th:noncov(D(p2+p1,p2+p1))
(%o33) 4*E^2
(%i34) t_th:factor(noncov(D(p1-p3,p1-p3)))
(%o34) 2*p^2*(cos(th)-1)
(%i35) u_th:factor(noncov(D(p1-p4,p1-p4)))
(%o35) -2*p^2*(cos(th)+1)
(%i36) Mfi:sub_stu(Mfi)
(%o36) -4*E^2/(p^2*sin(th)^2)-4/sin(th)^2+2
(%i37) " We want to combine the first two terms "
(%i38) " and replace p^2 by E^2 - m^2 in the numerator"
(%i39) Mfi_12:ratsimp(take_parts(Mfi,1,2))
(%o39) -(4*E^2+4*p^2)/(p^2*sin(th)^2)
(%i40) Mfi_12n:num(Mfi_12)
(%o40) -4*E^2-4*p^2
(%i41) Mfi_12n:factor(expand(subst(p^2 = E^2-m^2,Mfi_12n)))
(%o41) -4*(2*E^2-m^2)
(%i42) Mfi_12:Mfi_12n/denom(Mfi_12)
(%o42) -4*(2*E^2-m^2)/(p^2*sin(th)^2)
(%i43) " We now add in the third term and extract -2 "
(%i44) Mfi:pullfac(part(Mfi,3)+Mfi_12,-2)
(%o44) -2*(2*(2*E^2-m^2)/(p^2*sin(th)^2)-1)
(%i45) " having absorbed e^4 into a factor A which multiplies"
(%i46) " |Mfi|^2, and using e^2 = 4*%pi*alpha,"
(%i47) " in general, A = alpha^2*(pf/pi)/(4*Ecm^2) "
(%i48) " but here, pf = pi, and Ecm = 2*E, so "
(%i49) A:alpha^2/(16*E^2)
(%i50) " We can now write down the differential scattering cross section:"
(%i51) dsigdo:A*Mfi^2
(%i52) (display2d:true,display(dsigdo),display2d:false)
                                        2    2
                                   2  2 (2 E  - m )     2
                              alpha  (------------- - 1)
                                        2    2
                                       p  sin (th)
                     dsigdo = ----------------------------
                                          2
                                       4 E

(%i53) " which agrees with Itzykson and Zuber, p. 286."
(%i54) " The factor 1/sin(th)^2 can be displayed in terms of th/2"
(%i55) " using: "
(%i56) fr_ao2(1/cos(th/2)^2+1/sin(th/2)^2,th)
(%o56) 4/sin(th)^2
(%i57) " and, of course p^2 = E^2 - m^2 "
(%o57) "moller0.mac"
```

Since this is the first example of using a batch file to program the calculations, we show here the appearance of the batch file **moller0.mac**.

To save space, we will only show the batch file run output for the remaining examples, even though the batch file output is rather compressed (minimum white space) and symbols are rearranged in the way Maxima likes to organize printed output.

Of course you can edit your own batch file run output to include more white space which would make the results more readable and easier to follow.

```
/* file moller0.mac
   pi(+) pi(+) --> pi(+) pi(+)
   in scalar electrodynamics, ie,
   ignoring structure of pions */

   /* references:
      Renton p.182
      I/Z p. 286
      Schwinger, pp. 285 - 289
      */
 " ======================================================"$
 "  file moller0.mac "$
 "  Maxima by Example, Ch. 12 "$
 "  Dirac Algebra and Quantum Electrodynamics "$
 "  Edwin L Woollett, woollett@charter.net "$
 "  http://www.csulb.edu/~woollett  "$
    print ("      ver: ",_binfo%,"  date: ",mydate )$
 " ======================================================"$
 " ELASTIC SCATTERING OF SPIN 0 PARTICLES (SAME CHARGE)"$
 " FOR EXAMPLE: "$
"      PI(+,p1) PI(+,p2)  --->  PI(+,p3) PI(+,p4)       "$

  " POPULATE THE LIST invarR OF 4-VEC DOT PRODUCT VALUES, "$
  " Using p1 + p2 = p3 + p4, s = (p1+p2)^2 = (p3+p4)^2 ,"$
  "  t = (p1-p3)^2 = (p2-p4)^2,"$
  "   and u = (p1-p4)^2 = (p2-p3)^2 "$
  " ------------------------------------------"$
  invar (D(p1,p1) = m^2,
         D(p1,p2) = s/2 - m^2,
         D(p1,p3) = m^2 - t/2,
         D(p1,p4) = m^2 - u/2,
         D(p2,p2) = m^2,
         D(p2,p3) = m^2 - u/2,
         D(p2,p4) = m^2 - t/2,
         D(p3,p3) = m^2,
         D(p3,p4) = s/2 - m^2,
         D(p4,p4) = m^2)$
  "------------------------------------------------------"$
  " factor out -e^2 from Mfi, leaving Mfi = M1 + M2 "$

  M1 : D(p1+p3,p2+p4)/D(p1-p3,p1-p3)$

  M2 :  D(p1+p4,p2+p3)/D(p1-p4,p1-p4)$
```

```
M1 : ev_Ds (M1);

M1 : pullfac (M1,1/t);

M2 : ev_Ds (M2);

M2 : pullfac (M2,1/u);

Mfi : M1 + M2;

" we get the result Mfi = (s-u)/t + (s-t)/u  "$

"CM FRAME EVALUATION "$

assume ( p > 0, th >= 0, th <= %pi )$

comp_def ( p1( E,0,0,p),
           p2( E,0,0,-p),
           p3 (E,p*sin(th),0,p*cos(th)),
           p4 (E,-p*sin(th),0,-p*cos(th)) )$

s_th : noncov (D(p1+p2,p1+p2));

t_th : factor (noncov (D(p1-p3,p1-p3)));

u_th : factor (noncov (D(p1-p4,p1-p4)));

Mfi : sub_stu(Mfi);

" We want to combine the first two terms "$
" and replace p^2 by E^2 - m^2 in the numerator"$

Mfi_12 : ratsimp (take_parts (Mfi,1,2));

Mfi_12n : num (Mfi_12);

Mfi_12n : factor (expand (subst (p^2 = E^2 - m^2, Mfi_12n)));

Mfi_12 : Mfi_12n/denom(Mfi_12);

" We now add in the third term and extract -2 "$

Mfi : pullfac (Mfi_12 + part(Mfi,3),-2);

" having absorbed e^4 into a factor A which multiplies"$
" |Mfi|^2, and using e^2 = 4*%pi*alpha,"$
" in general, A = alpha^2*(pf/pi)/(4*Ecm^2) "$
" but here, pf = pi, and Ecm = 2*E, so "$

A : alpha^2/(16*E^2)$
```

```
" We can now write down the differential scattering cross section:"$

dsigdo : A*Mfi^2$

(display2d:true, display (dsigdo), display2d:false)$
" which agrees with Itzykson and Zuber, p. 286."$
" The factor 1/sin(th)^2 can be displayed in terms of th/2"$
" using: "$

fr_ao2(1/sin(th/2)^2 + 1/cos(th/2)^2, th );
" and, of course p^2 = E^2 - m^2 "$
```

12.6 moller1.mac: High Energy Elastic Scattering of Two Electrons

The batch file **moller1.mac** treats the high energy limit of the scattering event

$$e^-(p_1,\sigma_1) + e^-(p_2,\sigma_2) \rightarrow e^-(p_3,\sigma_3) + e^-(p_4,\sigma_4) \tag{12.42}$$

We consider the unpolarized ultra-relativistic limit in which the masses of the electrons can be ignored ($|\mathbf{p}| >> m$).

Some references are: C. Moller, 1932, Schwinger, PSF I, p. 300 - 305, Greiner-Reinhardt, QED, 4'th, Sec. 3.3, Griffith Sec. 7.6, Jauch and Rohrlich, p. 252-257, Renton Sec. 4.3.1, B/D Sec. 7.9, BLP, Sec. 81.

(Note that BLP's symbol r_e should be replaced by α/m to compare with our results - they use (along with Greiner/Reinhardt) Gaussian units for electromagnetic equations, whereas we use (along with Peskin/Schroeder and Bjorken/Drell) rationalized Heaviside-Lorentz electromagnetic units. $e^2_{HL} = 4\,\pi\,e^2_G$ See Greiner/Reinhardt, Sec 4.2.).

The invariant amplitude $M = M_1 - M_2$, has two terms. Using the definitions

$$s = (p_1 + p_2)^2, \quad t = (p_1 - p_3)^2, \quad u = (p_1 - p_4)^2, \tag{12.43}$$

and the abbreviations

$$u_1 = u(p_1,\sigma_1), \quad u_2 = u(p_2,\sigma_2), \quad \bar{u}_3 = \bar{u}(p_3,\sigma_3), \quad \bar{u}_4 = \bar{u}(p_4,\sigma_4) \tag{12.44}$$

we have

$$M_1 = -\frac{e^2\,\bar{u}_3\,\gamma^\mu\,u_1\,\bar{u}_4\gamma_\mu\,u_2}{t}, \tag{12.45}$$

and

$$M_2 = -\frac{e^2\,\bar{u}_4\,\gamma^\mu\,u_1\,\bar{u}_3\gamma_\mu\,u_2}{u}. \tag{12.46}$$

The unpolarized differential cross section in the CM frame is first found using symbolic methods, first in terms of **s, t, u** and then in terms of the scattering angle **th**.

The polarized amplitudes corresponding to definite electron helicity assignments are then calculated using explicit Dirac spinors and matrices, and the sum of the squares of these amplitudes reproduces the unpolarized symbolic calculation.

Finally one polarized squared amplitude is calculated using the symbolic methods. We will surpress the information output which **dirac2.mac** puts on the screen after loading in the package files.

```
(%i58) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("moller1.mac");
read and interpret file: #pc:/work5/moller1.mac
(%i3) " ======================================================"
(%i4) "  file moller1.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) " ======================================================"
(%i11) "              MOLLER SCATTERING                       "
(%i12) " HIGH ENERGY LIMIT, CENTER OF MOMENTUM FRAME, NEGLECT MASSES "
(%i13) "  e(-,p1,sv1) + e(-,p2,sv2) --> e(-,p3,sv3) + e(-,p4,sv4) "
(%i14) " m = electron mass is set to zero."
(%i15) " ---------------------------------------- "
(%i16) "NON-POLARIZED DIFFERENTIAL CROSS SECTION: SYMBOLIC METHODS"
(%i17) " POPULATE THE LIST invarR OF 4-VEC DOT PRODUCT VALUES, "
(%i18) " Using p1 + p2 = p3 + p4, s = (p1+p2)^2 = (p3+p4)^2 ,"
(%i19) "  t = (p1-p3)^2 = (p2-p4)^2,"
(%i20) "   and u = (p1-p4)^2 = (p2-p3)^2 "
(%i21) " CASE HIGH ENERGY (HE) LIMIT E >> m  "
(%i22) " ---------------------------------------------"
(%i23) invar(D(p1,p1) = 0,D(p1,p2) = s/2,D(p1,p3) = (-t)/2,D(p1,p4) = (-u)/2,
             D(p2,p2) = 0,D(p2,p3) = (-u)/2,D(p2,p4) = (-t)/2,D(p3,p3) = 0,
             D(p3,p4) = s/2,D(p4,p4) = 0)
(%i24) "-----------------------------------------------------"
(%i25) " factor out -e^2 from Mfi, leaving Mfi = M1 - M2 "
(%i26) " With a sum over all helicities implied,"
(%i27) " |Mfi|^2 = M1n/t^2 + M2n/u^2 -M12n/(t*u) - M21n/(t*u) "
(%i28) " M1n = t^2 * M1*conj(M1), M2n = u^2 * M2*conj(M2) "
(%i29) " M12n = (t*u)*M1*conj(M2), M21n = (t*u)*M2*conj(M1), and: "
(%i30) M1n:factor(Con(tr(p3,mu,p1,nu)*tr(p4,mu,p2,nu),mu,nu))
(%o30) 8*(u^2+s^2)
(%i31) M2n:factor(Con(tr(p4,mu,p1,nu)*tr(p3,mu,p2,nu),mu,nu))
(%o31) 8*(t^2+s^2)
(%i32) " NOTE AUTOMATIC PRETRACE CONTRACTION OF REPEATED "
(%i33) " LORENTZ INDICES WITHIN A SINGLE TRACE OCCURS USING tr."
(%i34) M12n:factor(tr(p3,mu,p1,nu,p4,mu,p2,nu))
(%o34) -8*s^2
(%i35) M21n:factor(tr(p4,mu,p1,nu,p3,mu,p2,nu))
(%o35) -8*s^2
(%i36) MfiSQ:pullfac((-M21n)/(t*u)+(-M12n)/(t*u)+M2n/u^2+M1n/t^2,8)
(%o36) 8*((u^2+s^2)/t^2+2*s^2/(t*u)+(t^2+s^2)/u^2)
(%i37) " We have absorbed e^4 into A, with e^2 = 4*%pi*alpha "
(%i38) " Averaging over initial spins means we need to divide A by 4"
(%i39) " to get the unpolarized differential cross section (CM, HE)"
(%i40) A:alpha^2/(4*s)
(%i41) dsigdo_unpol_CM_HE:A*MfiSQ/4
(%o41) alpha^2*((u^2+s^2)/t^2+2*s^2/(t*u)+(t^2+s^2)/u^2)/(2*s)
(%i42) (display2d:true,display(dsigdo_unpol_CM_HE),display2d:false)
                                      2    2      2    2    2
                                   2  u  + s    2 s    t  + s
                             alpha  (------- + ---- + -------)
                                        2      t u       2
                                       t                u
           dsigdo_unpol_CM_HE = ---------------------------------
                                              2 s
```

```
(%i43) " which agrees with Renton function of s,t, and u "
(%i44) " on page 159, Eq.(4.54) "
(%i45) " CONVERSION TO EXPLICIT FUNCTION OF SCATTERING ANGLE "
(%i46) assume(p > 0,th >= 0,th <= %pi)
(%i47) comp_def(p1(E,0,0,E),p2(E,0,0,-E),p3(E,E*sin(th),0,E*cos(th)),
               p4(E,-E*sin(th),0,-E*cos(th)))
(%i48) s_th:VP(p2+p1,p2+p1)
(%o48) 4*E^2
(%i49) t_th:factor(VP(p1-p3,p1-p3))
(%o49) 2*(cos(th)-1)*E^2
(%i50) u_th:factor(VP(p1-p4,p1-p4))
(%o50) -2*(cos(th)+1)*E^2
(%i51) " To get an expression we can compare with Renton and G/R, "
(%i52) " work with one term at a time "
(%i53) " sub_stu replaces s by s_th, t by t_th  "
(%i54) "  and u by u_th "
(%i55) M1SQ:sub_stu(M1n/t^2)
(%o55) 8*cos(th)^2/(cos(th)^2-2*cos(th)+1)+16*cos(th)/(cos(th)^2-2*cos(th)+1)
                                          +40/(cos(th)^2-2*cos(th)+1)
(%i56) " convert to (th/2) form "
(%i57) M1SQ:factor(ratsimp(to_ao2(M1SQ,th)))
(%o57) 8*(cos(th/2)^4+1)/sin(th/2)^4
(%i58) " ---------------- "
(%i59) M2SQ:sub_stu(M2n/u^2)
(%o59) 8*cos(th)^2/(cos(th)^2+2*cos(th)+1)-16*cos(th)/(cos(th)^2+2*cos(th)+1)
                                          +40/(cos(th)^2+2*cos(th)+1)
(%i60) M2SQ:ratsimp(to_ao2(M2SQ,th))
(%o60) (8*cos(th/2)^4-16*cos(th/2)^2+16)/cos(th/2)^4
(%i61) M2SQ:factor(ratsubst(1-sin(th/2)^2,cos(th/2)^2,num(M2SQ)))/denom(M2SQ)
(%o61) 8*(sin(th/2)^4+1)/cos(th/2)^4
(%i62) " ---------------------- "
(%i63) M12INT:-2*sub_stu(M12n/(t*u))
(%o63) -64/(cos(th)^2-1)
(%i64) M12INT:factor(M12INT)
(%o64) -64/((cos(th)-1)*(cos(th)+1))
(%i65) M12INT:subst([cos(th)-1 = -2*sin(th/2)^2,1+cos(th) = 2*cos(th/2)^2],
                    M12INT)
(%o65) 16/(cos(th/2)^2*sin(th/2)^2)
(%i66) " pull out a factor of 8 from each term. "
(%i67) MfiSQ:pullfac(M12INT+M2SQ+M1SQ,8)
(%o67) 8*((sin(th/2)^4+1)/cos(th/2)^4+2/(cos(th/2)^2*sin(th/2)^2)
                                      +(cos(th/2)^4+1)/sin(th/2)^4)
(%i68) dsigdo_unpol_CM_HE:A*MfiSQ/4
(%o68) alpha^2*((sin(th/2)^4+1)/cos(th/2)^4+2/(cos(th/2)^2*sin(th/2)^2)
                                           +(cos(th/2)^4+1)/sin(th/2)^4)
        /(2*s)
(%i69) (display2d:true,display(dsigdo_unpol_CM_HE),display2d:false)
                                  4 th                                  4 th
                              sin (--) + 1                          cos (--) + 1
                          2       2                 2                 2
                    alpha  (------------ + ----------------- + ------------)
                                  4 th          2 th     2 th          4 th
                             cos (--)      cos (--) sin (--)      sin (--)
                                  2             2        2             2
 dsigdo_unpol_CM_HE = ------------------------------------------------------
                                           2 s

(%i70) " which agrees with Renton, page 159, Eq.(4.55) "
(%i71) " as well as G/R, page 138, Eq. (3-139) "
(%i72) "-------------------------------------------------"
```

```
(%i73) " Convert to simple function of th displayed by G/R "
(%i74) MfiSQ:fr_ao2(MfiSQ,th)
(%o74) 16*cos(th)^4/sin(th)^4+96*cos(th)^2/sin(th)^4+144/sin(th)^4
(%i75) MfiSQ:factor(ratsimp(MfiSQ))
(%o75) 16*(cos(th)^2+3)^2/sin(th)^4
(%i76) dsigdo_unpol_CM_HE:A*MfiSQ/4
(%o76) alpha^2*(cos(th)^2+3)^2/(s*sin(th)^4)
(%i77) dsigdo_unpol_CM_HE:subst(s = s_th,dsigdo_unpol_CM_HE)
(%o77) alpha^2*(cos(th)^2+3)^2/(4*sin(th)^4*E^2)
(%i78) (display2d:true,display(dsigdo_unpol_CM_HE),display2d:false)
                                          2     2         2
                                    alpha  (cos (th) + 3)
                 dsigdo_unpol_CM_HE = ----------------------
                                            4      2
                                       4 sin (th) E

(%i79) " which agrees with G/R, page 139, top. "
```

The next section of **moller1.mac** uses explicit Dirac spinors and matrices to calculate polarized amplitudes.

```
(%i80) " ------------------------------------------------"
(%i81) " HE POLARIZED AMPLITUDES USING EXPLICIT DIRAC SPINORS "
(%i82) " ------------------------------------------------ "
(%i83) " case HE RR --> RR, ie, (+1,+1) --> (+1,+1) polarized amplitude "
(%i84) " Define the needed Dirac spinors and barred spinors."
(%i85) (up1:UU(E,E,0,0,1),up3b:sbar(UU(E,E,th,0,1)),up2:UU(E,E,%pi,0,1),
        up4b:sbar(UU(E,E,%pi-th,%pi,1)))
(%i86) " For example, following Halzen/Martin p. 120 ff :"
(%i87) " For helicity quantum numbers RR -> RR,  the amplitude Mfi"
(%i88) " (pulling out -e^2) has the form Mfi = M1 - M2 "
(%i89) " where M1 =  Mt/t and M2 =  Mu/u,  "
(%i90) " where t = (p1-p3)^2 and u = (p1-p4)^2, and "
(%i91) Mt:(a13:up3b . Gam[mu] . up1,a24:up4b . Gam[mu] . up2,mcon(a13*a24,mu),
          trigsimp(%%))
(%o91) -8*E^2
(%i92) Mu:(a14:up4b . Gam[mu] . up1,a23:up3b . Gam[mu] . up2,mcon(a14*a23,mu),
          trigsimp(%%))
(%o92) 8*E^2
(%i93) Mfi:Mt/t-Mu/u
(%o93) -8*E^2/u-8*E^2/t
(%i94) " Now replace s, t, and u by explicit functions of th "
(%i95) Mfi:sub_stu(Mfi)
(%o95) -8/(cos(th)^2-1)
(%i96) Mfi_RR_RR:ts(Mfi,th)
(%o96) 8/sin(th)^2
(%i97) " Now automate production of HE polarized amplitudes "
(%i98) " as functions of t and u "
(%i99) " Our definition of the amplitude does not include a factor of -e^2."
(%i100) he_me(sp1v,sp2v,sp3v,sp4v):=block([up1,up2,up3b,up4b,_mu%,Mt,Mu,temp],
              up1:UU(E,E,0,0,sp1v),up3b:sbar(UU(E,E,th,0,sp3v)),
              up2:UU(E,E,%pi,0,sp2v),up4b:sbar(UU(E,E,%pi-th,%pi,sp4v)),
              a13:up3b . Gam[_mu%] . up1,a24:up4b . Gam[_mu%] . up2,
              Mt:(Con(a13*a24,_mu%),expand(trigsimp(%%)))/t,
              a14:up4b . Gam[_mu%] . up1,a23:up3b . Gam[_mu%] . up2,
              Mu:(Con(a14*a23,_mu%),expand(trigsimp(%%)))/u,temp:Mt-Mu,
              if temp # 0 then temp:pullfac(temp,-8*E^2),temp)
(%i101) " test he_me for the case already worked out above "
(%i102) he_me(1,1,1,1)
(%o102) -8*(1/u+1/t)*E^2
(%i103) sub_stu(%)
(%o103) -8/(cos(th)^2-1)
```

```
(%i104) ts(%,th)
(%o104) 8/sin(th)^2
(%i105) " which agrees with our previous work."
(%i106) " ---------------------------------------- "
(%i107) " AN ALTERNATIVE PATH TO THE UNPOLARIZED CROSS SECTION "
(%i108) "  IS TO SUM THE ABSOLUTE VALUE SQUARED OF EACH "
(%i109) " OF THE POLARIZED AMPLITUDES, WHICH WE NOW DO."
(%i110) " The function Avsq(expr) is defined in dgmatrix.mac "
(%i111) " and computes the absolute value squared."
(%i112) "  (We could also use here abs(expr)^2 )"
(%i113) " Each nonzero amplitude is proportional to 8*E^2*e^2"
(%i114) " here since we have not replaced t or u in the denominators"
(%i115) " by a function of th."
(%i116) " We have already pulled out -e^2 from our 'amplitude' def"
(%i117) " Pull out also the factor 8*E^2 temporarily here."
(%i118) block([sL,s1,s2,s3,s4,temp],sL:[1,-1],mssq:0,print("  "),
              print(" svp1 svp2  svp3  svp4     amplitude     "),print("  "),
              for s1 in sL do
                  for s2 in sL do
                      for s3 in sL do
                          for s4 in sL do
                              (temp:expand(he_me(s1,s2,s3,s4)/(8*E^2)),
                               mssq:Avsq(temp)+mssq,print("  "),
                               print(s1,s2,s3,s4," ",temp)),
              mssq:expand(fr_ao2(mssq,th)))

 svp1 svp2  svp3  svp4     amplitude


1 1 1 1   -1/u-1/t

1 1 1 -1   0

1 1 -1 1   0

1 1 -1 -1   0

1 -1 1 1   0

1 -1 1 -1   cos(th/2)^2/t

1 -1 -1 1   sin(th/2)^2/u

1 -1 -1 -1   0

-1 1 1 1   0

-1 1 1 -1   sin(th/2)^2/u

-1 1 -1 1   cos(th/2)^2/t

-1 1 -1 -1   0

-1 -1 1 1   0

-1 -1 1 -1   0

-1 -1 -1 1   0

-1 -1 -1 -1   -1/u-1/t
(%i119) " ------------------------------------ "
```

```
(%i120) mssq
(%o120) 4/(t*u)+cos(th)^2/(2*u^2)-cos(th)/u^2+5/(2*u^2)+cos(th)^2/(2*t^2)
              +cos(th)/t^2+5/(2*t^2)
(%i121) mssq:sub_stu(mssq)
(%o121) cos(th)^4/(4*sin(th)^4*E^4)+3*cos(th)^2/(2*sin(th)^4*E^4)
                                   +9/(4*sin(th)^4*E^4)
(%i122) mssq:factor(ratsimp(mssq))
(%o122) (cos(th)^2+3)^2/(4*sin(th)^4*E^4)
(%i123) " restore (8*E^2)^2 = 64*E^4 to mssq "
(%i124) mssq:64*E^4*mssq
(%o124) 16*(cos(th)^2+3)^2/sin(th)^4
(%i125) dsigdo_unpol_CM_HE:A*mssq/4
(%o125) alpha^2*(cos(th)^2+3)^2/(s*sin(th)^4)
(%i126) dsigdo_unpol_CM_HE:subst(s = s_th,%)
(%o126) alpha^2*(cos(th)^2+3)^2/(4*sin(th)^4*E^2)
(%i127) (display2d:true,display(dsigdo_unpol_CM_HE),display2d:false)
                                          2     2          2
                                    alpha  (cos (th) + 3)
                 dsigdo_unpol_CM_HE = ----------------------
                                             4      2
                                        4 sin (th) E

(%i128) " which agrees with the symbolic calculation."
```

The last section of **moller1.mac** uses symbolic methods to find the square of polarized amplitudes. An argument **S(1)** or **S(-1)** inside **tr** is effectively interpreted as $\frac{1}{2}\left(1+\sigma\,\gamma^5\right)$, where $\sigma=\pm1$ is the argument of **S**.

```
(%i129) " ================================  "
(%i130) " SYMBOLIC POLARIZED AMPLITUDE SQUARED "
(%i131) " ----------------------------------------"
(%i132) " Instead of summing over helicities, insert helicity "
(%i133) " projection operators appropriate to the zero mass limit "
(%i134) " CASE RR --> RR "
(%i135) M1n:trigsimp(Con(tr(S(1),p3,mu,S(1),p1,nu)*tr(S(1),p4,mu,S(1),p2,nu),
                         mu,nu))
(%o135) 64*E^4+(8*t-8*t*cos(th))*E^2+4*t^2
(%i136) M2n:trigsimp(Con(tr(S(1),p4,mu,S(1),p1,nu)*tr(S(1),p3,mu,S(1),p2,nu),
                         mu,nu))
(%o136) 64*E^4+(8*cos(th)+8)*u*E^2+4*u^2
(%i137) " NOTE AUTOMATIC PRETRACE CONTRACTION OF REPEATED "
(%i138) " LORENTZ INDICES WITHIN A SINGLE TRACE OCCURS USING tr."
(%i139) M12n:tr(S(1),p3,mu,S(1),p1,nu,S(1),p4,mu,S(1),p2,nu)
(%o139) -4*s^2
(%i140) M21n:tr(S(1),p4,mu,S(1),p1,nu,S(1),p3,mu,S(1),p2,nu)
(%o140) -4*s^2
(%i141) MfiSQ:(-M21n)/(t*u)+(-M12n)/(t*u)+M2n/u^2+M1n/t^2
(%o141) (64*E^4+(8*cos(th)+8)*u*E^2+4*u^2)/u^2
         +(64*E^4+(8*t-8*t*cos(th))*E^2+4*t^2)/t^2+8*s^2/(t*u)
(%i142) MfiSQ:sub_stu(MfiSQ)
(%o142) 64/sin(th)^4
(%i143) "  which is the square of the Dirac spinor amplitude found above "
(%o144) "moller1.mac"
```

12.7 moller2.mac: Arbitrary Energy Moller Scattering

The batch file **moller2.mac** works out the case of arbitrary energy particles in the CM frame. See the references and kinematic notation in the previous section dealing with **moller1.mac**.

The first section of **moller2.mac** works out the unpolarized differential cross section for arbitrary energy, first in an arbitrary frame in terms of s, t, u, and then in the CM frame, using symbolic methods.

```
(%i145) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("moller2.mac");
read and interpret file: #pc:/work5/moller2.mac
(%i3) " ======================================================"
(%i4) "  file moller2.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) "             MOLLER SCATTERING                    "
(%i11) "       ARBITRARY ENERGY, CENTER OF MOMENTUM FRAME "
(%i12) "  e(-,p1,sv1) + e(-,p2,sv2) --> e(-,p3,sv3) + e(-,p4,sv4) "
(%i13) " ------------------------------------------ "
(%i14) " SYMBOLIC TRACES FOR UNPOLARIZED DIFFERENTIAL CROSS SECTION "
(%i15) " Supply s, t, and u expressions for dot products "
(%i16) " ------------------------------------------- "
(%i17) invar(D(p1,p1) = m^2,D(p1,p2) = s/2-m^2,D(p1,p3) = m^2-t/2,
             D(p1,p4) = m^2-u/2,D(p2,p2) = m^2,D(p2,p3) = m^2-u/2,
             D(p2,p4) = m^2-t/2,D(p3,p3) = m^2,D(p3,p4) = s/2-m^2,
             D(p4,p4) = m^2)
(%i18) M1n:Con(tr(m+p3,mu,m+p1,nu)*tr(m+p4,mu,m+p2,nu),mu,nu)
(%o18) 8*u^2-32*m^2*u+32*m^2*t+8*s^2-32*m^2*s+64*m^4
(%i19) M2n:Con(tr(m+p4,mu,m+p1,nu)*tr(m+p3,mu,m+p2,nu),mu,nu)
(%o19) 32*m^2*u+8*t^2-32*m^2*t+8*s^2-32*m^2*s+64*m^4
(%i20) " NOTE AUTOMATIC PRETRACE CONTRACTION OF REPEATED "
(%i21) " LORENTZ INDICES WITHIN A SINGLE TRACE OCCURS USING tr."
(%i22) M12n:tr(m+p3,mu,m+p1,nu,m+p4,mu,m+p2,nu)
(%o22) -16*m^2*u-16*m^2*t-8*s^2+48*m^2*s-32*m^4
(%i23) M21n:tr(m+p4,mu,m+p1,nu,m+p3,mu,m+p2,nu)
(%o23) -16*m^2*u-16*m^2*t-8*s^2+48*m^2*s-32*m^4
(%i24) " expressed as a function of s, t, and u "
(%i25) MfiSQ:(-M21n)/(t*u)+(-M12n)/(t*u)+M2n/u^2+M1n/t^2
(%o25) (8*u^2-32*m^2*u+32*m^2*t+8*s^2-32*m^2*s+64*m^4)/t^2
        +(32*m^2*u+8*t^2-32*m^2*t+8*s^2-32*m^2*s+64*m^4)/u^2
        +2*(16*m^2*u+16*m^2*t+8*s^2-48*m^2*s+32*m^4)/(t*u)
(%i26) " REPLACE s, t, and u WITH EXPLICIT FUNCTIONS OF th "
(%i27) " ------------------------------------------------ "
(%i28) assume(E > 0,p > 0,th >= 0,th <= %pi)
(%i29) comp_def(p1(E,0,0,p),p2(E,0,0,-p),p3(E,p*sin(th),0,p*cos(th)),
                p4(E,-p*sin(th),0,-p*cos(th)))
(%i30) s_th:VP(p2+p1,p2+p1)
(%o30) 4*E^2
(%i31) t_th:factor(VP(p1-p3,p1-p3))
(%o31) 2*p^2*(cos(th)-1)
(%i32) u_th:factor(VP(p1-p4,p1-p4))
(%o32) -2*p^2*(cos(th)+1)
(%i33) " ------------------------------------------------ "
(%i34) MfiSQ_th:factor(trigsimp(sub_stu(MfiSQ)))
(%o34) 16*(8*E^4+2*m^2*cos(th)^2*E^2-10*m^2*E^2+p^4*cos(th)^4+6*p^4*cos(th)^2
                +10*m^2*p^2*cos(th)^2+m^4*cos(th)^2+p^4-2*m^2*p^2+3*m^4)
        /(p^4*sin(th)^4)
```

```
(%i35) " show this equals BLP's expression in Eq. (81.10)"
(%i36) MfiSQ_cmp:16*(p^2+E^2)^2*((p^2/(p^2+E^2))^2*(4/sin(th)^2+1)
                                 -3/sin(th)^2+4/sin(th)^4)
                  /p^4
(%i37) trigsimp(expand(MfiSQ_th-MfiSQ_cmp))
(%o37) ((48*sin(th)^2+64)*E^4+((96*p^2-32*m^2)*sin(th)^2-128*p^2-128*m^2)*E^2
                             +(-144*p^4-160*m^2*p^2-16*m^4)*sin(th)^2+64*p^4
                             +128*m^2*p^2+64*m^4)
        /(p^4*sin(th)^4)
(%i38) ratsubst(p^2+m^2,E^2,%)
(%o38) 0
(%i39) " which confirms equality."
(%i40) A:alpha^2/(4*s)
(%i41) A:sub_stu(A)
(%o41) alpha^2/(16*E^2)
(%i42) " Averaging over initial spins means we need to divide A by 4"
(%i43) " to get the unpolarized differential cross section "
(%i44) dsigdo_unpol_CM:A*MfiSQ_cmp/4
(%o44) alpha^2*(E^2+p^2)^2
              *(p^4*(4/sin(th)^2+1)/(E^2+p^2)^2-3/sin(th)^2+4/sin(th)^4)
        /(4*p^4*E^2)
(%i45) (display2d:true,display(dsigdo_unpol_CM),display2d:false)
                                      4     4
                                     p  (-------- + 1)
                                           2
                         2   2    2 2      sin (th)           3          4
                    alpha  (E  + p )  (----------------- - -------- + --------)
                                            2    2 2          2          4
                                           (E  + p )        sin (th)   sin (th)
 dsigdo_unpol_CM = ------------------------------------------------------------
                                              4  2
                                             4 p  E

(%i46) " which agrees with BLP, p. 323, Eq (81.10). "
(%i46) " ============================================================"
```

The next section of **`moller2.mac`** uses explicit Dirac spinors and matrices to calculate polarized amplitudes.

```
(%i47) " ============================================================"
(%i48) " POLARIZED DIRAC SPINOR AMPLITUDES                           "
(%i49) " ------------------------------------------------------------"
(%i50) E_pm(expr):=expand(ratsubst(m^2+p^2,E^2,expr))
(%i51) p_Em(expr):=expand(ratsubst(E^2-m^2,p^2,expr))
(%i52) Ep_m(expr):=expand(ratsubst(m,sqrt(E-p)*sqrt(p+E),expr))
(%i53) Ep_Mm(expr):=(expand(ratsubst(M^2/4-m^2,p^2,expr)),
             expand(ratsubst(M/2,E,%%)))
(%i54) " ------------------------------------------------"
(%i55) " convert t_th and u_th  to th/2 forms "
(%i56) t_th2:to_ao2(t_th,th)
(%o56) -4*p^2*sin(th/2)^2
(%i57) u_th2:to_ao2(u_th,th)
(%o57) -4*p^2*cos(th/2)^2
(%i58) " dirac spinor amplitude given global s1,s2,s3,s4 "
(%i59) dA():=(
          (up1:UU(E,p,0,0,s1),up3b:sbar(UU(E,p,th,0,s3)),up2:UU(E,p,%pi,0,s2),
           up4b:sbar(UU(E,p,%pi-th,%pi,s4))),
          Mt:(a13:up3b . Gam[mu] . up1,a24:up4b . Gam[mu] . up2,
              mcon(a13*a24,mu)),Mt:Ep_m(Mt),M1:Mt/t_th2,
          Mu:(a14:up4b . Gam[mu] . up1,a23:up3b . Gam[mu] . up2,
              mcon(a14*a23,mu)),Mu:Ep_m(Mu),M2:Mu/u_th2,Mfi:M1-M2)
(%i60) " Print a table of polarized amplitudes."
```

```
(%i61) " Accumulate sum of squares mssq."
(%i62) block([sL,sv1,sv2,sv3,sv4,temp],sL:[1,-1],mssq:0,print("  "),
             print(" svp1 svp2  svp3  svp4     amplitude     "),print("  "),
             for sv1 in sL do
                 for sv2 in sL do
                     for sv3 in sL do
                         for sv4 in sL do
                             ([s1,s2,s3,s4]:[sv1,sv2,sv3,sv4],temp:dA(),
                              temp:E_pm(temp),mssq:Avsq(temp)+mssq,
                              print("  "),print(s1,s2,s3,s4," ",temp)),
             mssq:E_pm(mssq),mssq:expand(fr_ao2(mssq,th)))

 svp1 svp2  svp3  svp4     amplitude


1 1 1 1   m^2*sin(th/2)^2/(p^2*cos(th/2)^2)+2*sin(th/2)^2/cos(th/2)^2
                                +m^2*cos(th/2)^2/(p^2*sin(th/2)^2)
                                +2*cos(th/2)^2/sin(th/2)^2+4

1 1 1 -1   m*sin(th/2)*E/(p^2*cos(th/2))-m*cos(th/2)*E/(p^2*sin(th/2))

1 1 -1 1   m*sin(th/2)*E/(p^2*cos(th/2))-m*cos(th/2)*E/(p^2*sin(th/2))

1 1 -1 -1   2*m^2/p^2

1 -1 1 1   m*cos(th/2)*E/(p^2*sin(th/2))-m*sin(th/2)*E/(p^2*cos(th/2))

1 -1 1 -1   m^2*cos(th/2)^2/(p^2*sin(th/2)^2)+2*cos(th/2)^2/sin(th/2)^2-m^2/p^2

1 -1 -1 1   m^2*sin(th/2)^2/(p^2*cos(th/2)^2)+2*sin(th/2)^2/cos(th/2)^2-m^2/p^2

1 -1 -1 -1   m*sin(th/2)*E/(p^2*cos(th/2))-m*cos(th/2)*E/(p^2*sin(th/2))

-1 1 1 1   m*cos(th/2)*E/(p^2*sin(th/2))-m*sin(th/2)*E/(p^2*cos(th/2))

-1 1 1 -1   m^2*sin(th/2)^2/(p^2*cos(th/2)^2)+2*sin(th/2)^2/cos(th/2)^2-m^2/p^2

-1 1 -1 1   m^2*cos(th/2)^2/(p^2*sin(th/2)^2)+2*cos(th/2)^2/sin(th/2)^2-m^2/p^2

-1 1 -1 -1   m*sin(th/2)*E/(p^2*cos(th/2))-m*cos(th/2)*E/(p^2*sin(th/2))

-1 -1 1 1   2*m^2/p^2

-1 -1 1 -1   m*cos(th/2)*E/(p^2*sin(th/2))-m*sin(th/2)*E/(p^2*cos(th/2))

-1 -1 -1 1   m*cos(th/2)*E/(p^2*sin(th/2))-m*sin(th/2)*E/(p^2*cos(th/2))

-1 -1 -1 -1   m^2*sin(th/2)^2/(p^2*cos(th/2)^2)+2*sin(th/2)^2/cos(th/2)^2
                                +m^2*cos(th/2)^2/(p^2*sin(th/2)^2)
                                +2*cos(th/2)^2/sin(th/2)^2+4
(%i63) mssq
(%o63) -192*m^2/(p^2*sin(th)^2)-48*m^4/(p^4*sin(th)^2)-128/sin(th)^2
                               +256*m^2/(p^2*sin(th)^4)+64*m^4/(p^4*sin(th)^4)
                               +256/sin(th)^4+16
(%i64) " SHOW THIS IS THE SAME AS MfiSQ_th computed above with traces "
(%i65) MfiSQ_p:E_pm(MfiSQ_th)
(%o65) 16*cos(th)^4/sin(th)^4+192*m^2*cos(th)^2/(p^2*sin(th)^4)
                             +48*m^4*cos(th)^2/(p^4*sin(th)^4)
                             +96*cos(th)^2/sin(th)^4+64*m^2/(p^2*sin(th)^4)
                             +16*m^4/(p^4*sin(th)^4)+144/sin(th)^4
(%i66) trigsimp(mssq-MfiSQ_p)
(%o66) 0
```

```
(%i67) " which shows equality."
(%i68) " ============================================================"
(%o68) "moller2.mac"
```

12.8 bhabha1.mac: High Energy Limit of Bhabha Scattering

This batch file treats the high energy limit of the scattering event

$$e^-(p_1,\sigma_1) + e^+(p_2,\sigma_2) \rightarrow e^-(p_3,\sigma_3) + e^+(p_4,\sigma_4). \tag{12.47}$$

This limit is that in which the masses of the particles can be ignored ($|\mathbf{p}| >> m$). In practice this means treating the leptons as massless particles.

Some references are: H.J. Bhabha, 1936, BLP QED, Sec. 81, B/D RQM, Sec. 7.9, G/R QED, Sec. 3.4, Renton EI, Sec. 4.3, and Schwinger, PSF I, p. 306 - 309.

The invariant amplitude $M = M_1 - M_2$ has two terms. Using the definitions

$$s = (p_1 + p_2)^2, \quad t = (p_1 - p_3)^2, \quad u = (p_1 - p_4)^2, \tag{12.48}$$

and the abbreviations

$$u_1 = u(p_1,\sigma_1), \quad v_4 = v(p_4,\sigma_4), \quad \bar{u}_3 = \bar{u}(p_3,\sigma_3), \quad \bar{v}_2 = \bar{v}(p_2,\sigma_2) \tag{12.49}$$

we have

$$M_1 = -\frac{e^2\,\bar{u}_3\,\gamma^\mu\,u_1\,\bar{v}_2\gamma_\mu\,v_4}{t}, \tag{12.50}$$

and

$$M_2 = -\frac{e^2\,\bar{v}_2\,\gamma^\mu\,u_1\,\bar{u}_3\gamma_\mu\,v_4}{s}. \tag{12.51}$$

The batch file **bhabha1.mac** uses the Dirac package functions **pullfac**, **VP**, **to_ao2**, **Avsq**, and **fr_ao2**.

The first section of **bhabha1.mac** derives the unpolarized differential cross section, first in an arbitrary frame in terms of s, t, u, and then in the CM frame in terms of the scattering angle **th**, using symbolic methods.

```
(%i69) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("bhabha1.mac");
read and interpret file: #pc:/work5/bhabha1.mac
(%i3) " ============================================================"
(%i4) "  file bhabha1.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) "             BHABHA SCATTERING                    "
(%i11) " HIGH ENERGY LIMIT, CENTER OF MOMENTUM FRAME, NEGLECT MASSES "
(%i12) "  e(-,p1,s1) + e(+,p2,s2) --> e(-,p3,s3) + e(+,p4,s4) "
(%i13) " m = electron and positron mass is set to zero."
(%i14) " ------------------------------------------ "
```

```
(%i15) "NON-POLARIZED DIFFERENTIAL CROSS SECTION: SYMBOLIC METHODS"
(%i16) " POPULATE THE LIST invarR OF 4-VEC DOT PRODUCT VALUES, "
(%i17) " Using p1 + p2 = p3 + p4, s = (p1+p2)^2 = (p3+p4)^2 ,"
(%i18) "  t = (p1-p3)^2 = (p2-p4)^2,"
(%i19) "   and u = (p1-p4)^2 = (p2-p3)^2 "
(%i20) " CASE HIGH ENERGY (HE) LIMIT E >> m  "
(%i21) " ---------------------------------------------"
(%i22) invar(D(p1,p1) = 0,D(p1,p2) = s/2,D(p1,p3) = (-t)/2,D(p1,p4) = (-u)/2,
             D(p2,p2) = 0,D(p2,p3) = (-u)/2,D(p2,p4) = (-t)/2,D(p3,p3) = 0,
             D(p3,p4) = s/2,D(p4,p4) = 0)
(%i23) "-----------------------------------------------------"
(%i24) " With a sum over all helicities implied,"
(%i25) " |Mfi|^2 = M1n/t^2 + M2n/s^2 -M12n/(t*s) - M21n/(t*s) "
(%i26) " M1n = t^2 * M1*conj(M1), M2n = s^2 * M2*conj(M2) "
(%i27) " M12n = (t*s)*M1*conj(M2), M21n = (t*s)*M2*conj(M1), and: "
(%i28) M1n:factor(Con(tr(p3,mu,p1,nu)*tr(p2,mu,p4,nu),mu,nu))
(%o28) 8*(u^2+s^2)
(%i29) M2n:factor(Con(tr(p2,mu,p1,nu)*tr(p3,mu,p4,nu),mu,nu))
(%o29) 8*(u^2+t^2)
(%i30) " NOTE AUTOMATIC PRETRACE CONTRACTION OF REPEATED "
(%i31) " LORENTZ INDICES WITHIN A SINGLE TRACE OCCURS USING tr."
(%i32) M12n:factor(tr(p3,mu,p1,nu,p2,mu,p4,nu))
(%o32) -8*u^2
(%i33) M21n:factor(tr(p2,mu,p1,nu,p3,mu,p4,nu))
(%o33) -8*u^2
(%i34) MfiSQ:pullfac((-M21n)/(t*s)+(-M12n)/(t*s)+M2n/s^2+M1n/t^2,8)
(%o34) 8*((u^2+t^2)/s^2+(u^2+s^2)/t^2+2*u^2/(s*t))
(%i35) " We have absorbed e^4 into A, with e^2 = 4*%pi*alpha "
(%i36) " Averaging over initial spins means we need to divide A by 4"
(%i37) " to get the unpolarized differential cross section (CM, HE)"
(%i38) A:alpha^2/(4*s)
(%i39) dsigdo_unpol_CM_HE:A*MfiSQ/4
(%o39) alpha^2*((u^2+t^2)/s^2+(u^2+s^2)/t^2+2*u^2/(s*t))/(2*s)
(%i40) (display2d:true,display(dsigdo_unpol_CM_HE),display2d:false)
                                        2    2    2    2      2
                                     2  u  + t    u  + s    2 u
                                alpha  (------- + ------- + ----)
                                          2         2      s t
                                         s         t
            dsigdo_unpol_CM_HE = ---------------------------------
                                                2 s

(%i41) " which agrees with Renton's function of s,t, and u "
(%i42) " on page 159, Eq.(4.54) "
(%i43) " CONVERSION TO EXPLICIT FUNCTION OF SCATTERING ANGLE "
(%i44) assume(E > 0,th >= 0,th <= %pi)
(%i45) comp_def(p1(E,0,0,E),p2(E,0,0,-E),p3(E,E*sin(th),0,E*cos(th)),
                p4(E,-E*sin(th),0,-E*cos(th)))
(%i46) s_th:VP(p2+p1,p2+p1)
(%o46) 4*E^2
(%i47) t_th:factor(VP(p1-p3,p1-p3))
(%o47) 2*(cos(th)-1)*E^2
(%i48) u_th:factor(VP(p1-p4,p1-p4))
(%o48) -2*(cos(th)+1)*E^2
(%i49) "  CONVERT FROM s, t, u TO th form  "
(%i50) "-----------------------------------------------------"
(%i51) MfiSQ_th:(sub_stu(MfiSQ),factor(%%))
(%o51) 4*(cos(th)^2+3)^2/(cos(th)-1)^2
(%i52) A_th:subst(s = s_th,A)
(%o52) alpha^2/(16*E^2)
(%i53) dsigdo_unpol_th:A_th*MfiSQ_th/4
(%o53) alpha^2*(cos(th)^2+3)^2/(16*(cos(th)-1)^2*E^2)
```

```
(%i54) (display2d:true,display(dsigdo_unpol_th),display2d:false)
                                        2     2          2
                                   alpha  (cos (th) + 3)
                  dsigdo_unpol_th = ----------------------
                                                  2  2
                                    16 (cos(th) - 1)  E

(%i55) "------------------------------------------------------------"
(%i56) " which agrees with Renton, p. 160 "
```

The next section of **bhabha1.mac** uses explicit Dirac spinors and matrices to compute the polarized amplitudes.

```
(%i57) " HIGH ENERGY POLARIZED AMPLITUDES USING EXPLICIT DIRAC SPINORS "
(%i58) " ------------------------------------------------ "
(%i59) t_th2:to_ao2(t_th,th)
(%o59) -4*sin(th/2)^2*E^2
(%i60) " dirac spinor amplitude given global s1,s2,s3,s4 "
(%i61) dA():=(
          (up1:UU(E,E,0,0,s1),up3b:sbar(UU(E,E,th,0,s3)),
           vp2b:sbar(VV(E,E,%pi,0,s2)),vp4:VV(E,E,%pi-th,%pi,s4)),
          Mt:(a13:up3b . Gam[_mu%] . up1,a42:vp2b . Gam[_mu%] . vp4,
              mcon(a13*a42,_mu%),expand(trigsimp(%%))),M1:Mt/t_th2,
          Ms:(a12:vp2b . Gam[_mu%] . up1,a43:up3b . Gam[_mu%] . vp4,
              mcon(a12*a43,_mu%),expand(trigsimp(%%))),M2:Ms/s_th,M1-M2)
(%i62) " example: RR --> RR "
(%i63) [s1,s2,s3,s4]:[1,1,1,1]
(%i64) dA()
(%o64) 2/sin(th/2)^2
(%i65) " Make table of polarized amplitudes. "
(%i66) " Accumulate sum mssq of square of amplitudes."
(%i67) block([sL,sv1,sv2,sv3,sv4,temp],sL:[1,-1],mssq:0,print("  "),
             print(" s1 s2  s3  s4     amplitude     "),print("  "),
             for sv1 in sL do
                 for sv2 in sL do
                     for sv3 in sL do
                         for sv4 in sL do
                             ([s1,s2,s3,s4]:[sv1,sv2,sv3,sv4],temp:dA(),
                              mssq:Avsq(temp)+mssq,print("  "),
                              print(s1,s2,s3,s4,"        ",temp)),
             mssq:expand(fr_ao2(mssq,th)))

 s1 s2  s3  s4     amplitude


1 1 1 1          2/sin(th/2)^2

1 1 1 -1          0

1 1 -1 1          0

1 1 -1 -1          0

1 -1 1 1          0

1 -1 1 -1          2*cos(th/2)^2/sin(th/2)^2-2*cos(th/2)^2

1 -1 -1 1          -2*sin(th/2)^2

1 -1 -1 -1          0

-1 1 1 1          0
```

```
-1 1 1 -1          -2*sin(th/2)^2

-1 1 -1 1          2*cos(th/2)^2/sin(th/2)^2-2*cos(th/2)^2

-1 1 -1 -1          0

-1 -1 1 1          0

-1 -1 1 -1          0

-1 -1 -1 1          0

-1 -1 -1 -1          2/sin(th/2)^2
(%i68) " Sum of squares of polarized amplitudes:"
(%i69) mssq
(%o69) 4*cos(th)^4/(cos(th)^2-2*cos(th)+1)
        +24*cos(th)^2/(cos(th)^2-2*cos(th)+1)+36/(cos(th)^2-2*cos(th)+1)
(%i70) " COMPARE WITH SYMBOLIC RESULT MfiSQ_th CALCULATED ABOVE"
(%i71) trigsimp(mssq-MfiSQ_th)
(%o71) 0
(%i72) " WHICH SHOWS THEY ARE EQUIVALENT."
(%o72) "bhabha1.mac"
```

12.9 bhabha2.mac: Arbitrary Energy Bhabha Scattering

The batch file bhabha2.mac works out bhabha scattering for arbitrary particle energy in the CM frame. See the references and kinematic notation in the previous section dealing with **bhabha1.mac**. The batch file **bhabha1.mac** uses the Dirac package functions **pullfac**, **VP**, **sub_stu**, **to_ao2**, **Avsq**, and **fr_ao2**.

In addition, some kinematic substitution functions are defined in this batch file

- The function **E_pm(expr)** makes the replacement **E^2 --> m^2 + p^2**,
- The function **p_Em (expr)** makes the replacement **p^2 --> E^2 - m^2\mv**,
- The function **Ep_m (expr)** makes the replacement **sqrt (E-p)*sqrt (E+p) --> m**,
- The function **Ep_Mm (expr)** makes the two replacements: 1. **E --> M/2**, 2. **p^2 --> M^2/4 - m^2**, where **M** is the total center of momentum frame energy.

The first section of **bhabha2.mac** calculates the differential scattering cross section in the CM frame using symbolic methods.

```
(%i73) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("bhabha2.mac");
read and interpret file: #pc:/work5/bhabha2.mac
(%i3) " ======================================================"
(%i4) "  file bhabha2.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) "             BHABHA SCATTERING                    "
(%i11) " Finite mass, Arbitrary Energy, CENTER OF MOMENTUM FRAME  "
(%i12) "  e(-,p1,s1) + e(+,p2,s2) --> e(-,p3,s3) + e(+,p4,s4) "
```

```
(%i13) " ---------------------------------------- "
(%i14) " SYMBOLIC TRACES FOR UNPOLARIZED DIFFERENTIAL CROSS SECTION "
(%i15) " Supply s, t, and u expressions for dot products "
(%i16) " ---------------------------------------------- "
(%i17) invar(D(p1,p1) = m^2,D(p1,p2) = s/2-m^2,D(p1,p3) = m^2-t/2,
             D(p1,p4) = m^2-u/2,D(p2,p2) = m^2,D(p2,p3) = m^2-u/2,
             D(p2,p4) = m^2-t/2,D(p3,p3) = m^2,D(p3,p4) = s/2-m^2,
             D(p4,p4) = m^2)
(%i18) " With a sum over all helicities implied,"
(%i19) " |Mfi|^2 = M1n/t^2 + M2n/s^2 -M12n/(t*s) - M21n/(t*s) "
(%i20) " M1n = t^2 * M1*conj(M1), M2n = s^2 * M2*conj(M2) "
(%i21) " M12n = (t*s)*M1*conj(M2), M21n = (t*s)*M2*conj(M1), and: "
(%i22) M1n:Con(tr(m+p3,mu,m+p1,nu)*tr(p2-m,mu,p4-m,nu),mu,nu)
(%o22) 8*u^2-32*m^2*u+32*m^2*t+8*s^2-32*m^2*s+64*m^4
(%i23) M2n:Con(tr(p2-m,mu,m+p1,nu)*tr(m+p3,mu,p4-m,nu),mu,nu)
(%o23) 8*u^2-32*m^2*u+8*t^2-32*m^2*t+32*m^2*s+64*m^4
(%i24) " NOTE AUTOMATIC PRETRACE CONTRACTION OF REPEATED "
(%i25) "  LORENTZ INDICES WITHIN A SINGLE TRACE OCCURS USING tr."
(%i26) M12n:tr(m+p3,mu,m+p1,nu,p2-m,mu,p4-m,nu)
(%o26) -8*u^2+48*m^2*u-16*m^2*t-16*m^2*s-32*m^4
(%i27) M21n:tr(p2-m,mu,m+p1,nu,m+p3,mu,p4-m,nu)
(%o27) -8*u^2+48*m^2*u-16*m^2*t-16*m^2*s-32*m^4
(%i28) " sum over spins of |Mfi|^2 in terms of s,t,u "
(%i29) MfiSQ:(-M21n)/(t*s)+(-M12n)/(t*s)+M2n/s^2+M1n/t^2
(%o29) (8*u^2-32*m^2*u+8*t^2-32*m^2*t+32*m^2*s+64*m^4)/s^2
       +(8*u^2-32*m^2*u+32*m^2*t+8*s^2-32*m^2*s+64*m^4)/t^2
       +2*(8*u^2-48*m^2*u+16*m^2*t+16*m^2*s+32*m^4)/(s*t)
(%i30) " replace s, t, u in terms of scatt. angle th "
(%i31) assume(E > 0,p > 0,th >= 0,th <= %pi)
(%i32) comp_def(p1(E,0,0,p),p2(E,0,0,-p),p3(E,p*sin(th),0,p*cos(th)),
                p4(E,-p*sin(th),0,-p*cos(th)))
(%i33) E_pm(expr):=expand(ratsubst(m^2+p^2,E^2,expr))
(%i34) s_th:VP(p2+p1,p2+p1)
(%o34) 4*E^2
(%i35) t_th:factor(VP(p1-p3,p1-p3))
(%o35) 2*p^2*(cos(th)-1)
(%i36) u_th:factor(VP(p1-p4,p1-p4))
(%o36) -2*p^2*(cos(th)+1)
(%i37) " ------------------------------------------------ "
(%i38) MfiSQ_th:sub_stu(MfiSQ)
(%i39) " replace E^2 by m^2 + p^2 "
(%i40) MfiSQ_p:E_pm(MfiSQ_th)
(%i41) MfiSQ_p:trigsimp(MfiSQ_p)
(%o41) (4*p^8*sin(th)^4+(-8*m^2*p^6*cos(th)-32*p^8-72*m^2*p^6-52*m^4*p^4)
                       *sin(th)^2+(32*m^2*p^6+80*m^4*p^4+56*m^6*p^2)*cos(th)
                      +64*p^8+160*m^2*p^6+128*m^4*p^4+40*m^6*p^2+16*m^8)
       /((p^8+2*m^2*p^6+m^4*p^4)*cos(th)^2
         +(-2*p^8-4*m^2*p^6-2*m^4*p^4)*cos(th)+p^8+2*m^2*p^6+m^4*p^4)
(%i42) " ------------------------------------------------"
(%i43) " compare to Greiner, Reinhardt expression p. 148 "
(%i44) GR1:(2*p^4*(cos(th/2)^4+1)+4*p^2*m^2*cos(th/2)^2+m^4)/(p^4*sin(th/2)^4)
(%o44) (2*p^4*(cos(th/2)^4+1)+4*m^2*p^2*cos(th/2)^2+m^4)/(p^4*sin(th/2)^4)
(%i45) GR2:(p^4*(cos(th)^2+1)+4*p^2*m^2+3*m^4)/E^4
(%o45) (p^4*(cos(th)^2+1)+4*m^2*p^2+3*m^4)/E^4
(%i46) GR3:(-(4*p^4*cos(th/2)^4+8*p^2*m^2*cos(th/2)^2+3*m^4))
           /(E^2*p^2*sin(th/2)^2)
(%o46) (-4*p^4*cos(th/2)^4-8*m^2*p^2*cos(th/2)^2-3*m^4)/(p^2*sin(th/2)^2*E^2)
(%i47) " replace th/2 and E^2 for comparison "
(%i48) GR1_r:fr_ao2(GR1,th)
(%i49) GR2_r:E_pm(GR2)
(%i50) GR3_r:E_pm(fr_ao2(GR3,th))
(%i51) GR:trigsimp(expand(GR3_r+GR2_r+GR1_r))
```

```
(%i52) trigsimp(GR-MfiSQ_p/4)
(%o52) 0
(%i53) " which shows the equivalence we need "
(%i54) " dividing MfiSQ by 4 comes from averaging over"
(%i55) " the initial helicities "
(%i56) " We have absorbed e^4 into A, with e^2 = 4*%pi*alpha "
(%i57) " to get the unpolarized differential cross section (CM)"
(%i58) A:alpha^2/(4*s_th)
(%i59) GR_expr:GR3+GR2+GR1
(%i60) dsigdo_unpol_CM:A*GR_expr
(%i61) (display2d:true,display(dsigdo_unpol_CM),display2d:false)
                                 4    4 th       2  2    2 th       4
                           - 4 p  cos (--) - 8 m  p  cos (--) - 3 m
                        2             2                   2
dsigdo_unpol_CM = (alpha  (-----------------------------------------
                                        2    2 th   2
                                       p  sin (--) E
                                            2
    4     2               2  2      4
   p  (cos (th) + 1) + 4 m  p  + 3 m
 + ----------------------------------
                    4
                   E
      4     4 th            2  2    2 th     4
   2 p  (cos (--) + 1) + 4 m  p  cos (--) + m
             2                       2              2
 + ---------------------------------------))/(16 E )
                    4    4 th
                   p  sin (--)
                           2

(%i62) " which agrees with G/R, Exer. 3.8, eqn. 21, p. 148 "
```

The next section of **`bhabha2.mac`** uses explicit Dirac spinors and matrices to compute polarized amplitudes.

```
(%i63) " ============================================================ "
(%i64) " POLARIZED DIRAC SPINOR AMPLITUDES                           "
(%i65) " ------------------------------------------------------------"
(%i66) p_Em(expr):=expand(ratsubst(E^2-m^2,p^2,expr))
(%i67) Ep_m(expr):=expand(ratsubst(m,sqrt(E-p)*sqrt(p+E),expr))
(%i68) Ep_Mm(expr):=(expand(ratsubst(M^2/4-m^2,p^2,expr)),
             expand(ratsubst(M/2,E,%%)))
(%i69) " ---------------------------------------------"
(%i70) " convert t_th to th/2 form "
(%i71) t_th2:to_ao2(t_th,th)
(%o71) -4*p^2*sin(th/2)^2
(%i72) " dirac spinor amplitude given global s1,s2,s3,s4 "
(%i73) dA():=(
          (up1:UU(E,p,0,0,s1),up3b:sbar(UU(E,p,th,0,s3)),
           vp2b:sbar(VV(E,p,%pi,0,s2)),vp4:VV(E,p,%pi-th,%pi,s4)),
          Mt:(a13:up3b . Gam[mu] . up1,a42:vp2b . Gam[mu] . vp4,
              mcon(a13*a42,mu),Ep_m(%%),E_pm(%%),trigsimp(%%)),M1:Mt/t_th2,
          Ms:(a12:vp2b . Gam[mu] . up1,a43:up3b . Gam[mu] . vp4,
              mcon(a12*a43,mu),Ep_m(%%),E_pm(%%),trigsimp(%%)),M2:Ms/s_th,
          M2-M1)
(%i74) " example: RR --> RR  "
(%i75) [s1,s2,s3,s4]:[1,1,1,1]
(%o75) [1,1,1,1]
(%i76) dA()
(%o76) (4*m^2-8*m^2*sin(th/2)^2)/(4*E^2)
       +(-4*m^2*cos(th/2)^2-8*p^2)/(4*p^2*sin(th/2)^2)
```

```
(%i77) " ========================================================= "
(%i78) " table of polarized amplitudes generates global mssq "
(%i79) block([sL,sv1,sv2,sv3,sv4,temp],sL:[1,-1],mssq:0,print("  "),
            print(" s1  s2  s3  s4         amplitude     "),print("  "),
            for sv1 in sL do
                for sv2 in sL do
                    for sv3 in sL do
                        for sv4 in sL do
                            ([s1,s2,s3,s4]:[sv1,sv2,sv3,sv4],temp:dA(),
                             print("  "),print(s1,s2,s3,s4," ",temp),
                             temp:E_pm(temp),mssq:Avsq(temp)+mssq),
            mssq:E_pm(mssq),mssq:expand(fr_ao2(mssq,th)),mssq:trigsimp(mssq))

 s1  s2  s3  s4         amplitude


1 1 1 1   (4*m^2-8*m^2*sin(th/2)^2)/(4*E^2)+(-4*m^2*cos(th/2)^2-8*p^2)
                                /(4*p^2*sin(th/2)^2)

1 1 1 -1   2*m*cos(th/2)*sin(th/2)/E-m*cos(th/2)*E/(p^2*sin(th/2))

1 1 -1 1   m*cos(th/2)*E/(p^2*sin(th/2))-2*m*cos(th/2)*sin(th/2)/E

1 1 -1 -1   (4*m^2-8*m^2*sin(th/2)^2)/(4*E^2)+m^2/p^2

1 -1 1 1   m*cos(th/2)*E/(p^2*sin(th/2))-2*m*cos(th/2)*sin(th/2)/E

1 -1 1 -1   (8*p^2+8*m^2)*cos(th/2)^2/(4*E^2)+(-8*p^2-4*m^2)*cos(th/2)^2
                                /(4*p^2*sin(th/2)^2)

1 -1 -1 1   (8*p^2+8*m^2)*sin(th/2)^2/(4*E^2)-m^2/p^2

1 -1 -1 -1   m*cos(th/2)*E/(p^2*sin(th/2))-2*m*cos(th/2)*sin(th/2)/E

-1 1 1 1   2*m*cos(th/2)*sin(th/2)/E-m*cos(th/2)*E/(p^2*sin(th/2))

-1 1 1 -1   (8*p^2+8*m^2)*sin(th/2)^2/(4*E^2)-m^2/p^2

-1 1 -1 1   (8*p^2+8*m^2)*cos(th/2)^2/(4*E^2)+(-8*p^2-4*m^2)*cos(th/2)^2
                                /(4*p^2*sin(th/2)^2)

-1 1 -1 -1   2*m*cos(th/2)*sin(th/2)/E-m*cos(th/2)*E/(p^2*sin(th/2))

-1 -1 1 1   (4*m^2-8*m^2*sin(th/2)^2)/(4*E^2)+m^2/p^2

-1 -1 1 -1   2*m*cos(th/2)*sin(th/2)/E-m*cos(th/2)*E/(p^2*sin(th/2))

-1 -1 -1 1   m*cos(th/2)*E/(p^2*sin(th/2))-2*m*cos(th/2)*sin(th/2)/E

-1 -1 -1 -1   (4*m^2-8*m^2*sin(th/2)^2)/(4*E^2)+(-4*m^2*cos(th/2)^2-8*p^2)
                                /(4*p^2*sin(th/2)^2)
(%i80) "----------------------------------------"
(%i81) " sum of squares of polarized amplitudes:"
(%i82) mssq
(%o82) (4*p^8*sin(th)^4+(-8*m^2*p^6*cos(th)-32*p^8-72*m^2*p^6-52*m^4*p^4)
                        *sin(th)^2+(32*m^2*p^6+80*m^4*p^4+56*m^6*p^2)*cos(th)
                       +64*p^8+160*m^2*p^6+128*m^4*p^4+40*m^6*p^2+16*m^8)
        /((p^8+2*m^2*p^6+m^4*p^4)*cos(th)^2
          +(-2*p^8-4*m^2*p^6-2*m^4*p^4)*cos(th)+p^8+2*m^2*p^6+m^4*p^4)
(%i83) " SHOW THIS IS THE SAME AS MfiSQ_th computed above with traces "
(%i84) MfiSQ_p:E_pm(MfiSQ_th)
```

```
(%i85) trigsimp(mssq-MfiSQ_p)
(%o85) 0
(%i86) " which shows equality."
(%i87) " ============================================================ "
(%o88) "bhabha2.mac"
```

12.10 photon1.mac: Photon Transverse Polarization 3-Vector Sums

In this section we use Maxima to work out some well known properties of photon polarization 3-vectors corresponding to physical (external) photons, using the batch file **photon1.mac**. Some references to this subject are: G/R QED p. 187, Weinberg I p. 352, 360, 368, Renton p. 166, and Gross p. 315.

Let $\mathbf{e_{k\,s}}$ 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 $\mathbf{e_{k'\,r}}$ 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

$$\sum_{\mathbf{s=1}}^{\mathbf{2}} \left(\mathbf{e_{k\,s}} \cdot \mathbf{e_{k'\,r}}\right)^{\mathbf{2}} = \mathbf{1} - \left(\mathbf{\hat{k}} \cdot \mathbf{e_{k'\,r}}\right)^{\mathbf{2}} \tag{12.52}$$

$$\sum_{\mathbf{s,r=1}}^{\mathbf{2}} \left(\mathbf{e_{k\,s}} \cdot \mathbf{e_{k'\,r}}\right)^{\mathbf{2}} = \mathbf{1} + \mathbf{\cos^2\theta} \tag{12.53}$$

Here we are using real polarization 3-vectors to describe states of linear polarization.

```
(%i89) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("photon1.mac");
read and interpret file: #pc:/work5/photon1.mac
(%i3) display2d:false
(%i4) " ======================================================"
(%i5) "  file photon1.mac "
(%i6) "  Maxima by Example, Ch. 12 "
(%i7) "  Dirac Algebra and Quantum Electrodynamics "
(%i8) "  Edwin L Woollett, woollett@charter.net "
(%i9) "  http://www.csulb.edu/~woollett  "
(%i10) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i11) "------------------------------------------------------"
(%i12) "  PROPERTIES AND SUMS OF TRANSVERSE POLARIZATION VECTORS "
(%i13) " ------------------------------------------------- "
(%i14) " define 3-vectors as lists "
(%i15) " scattering plane is z-x plane "
(%i16) k_vec:[0,0,k]
(%i17) kp_vec:[kp*sin(th),0,kp*cos(th)]
(%i18) " e[1] X e[2] = k_vec/k "
(%i19) e[1]:[1,0,0]
(%i20) e[1] . e[1]
(%o20) 1
(%i21) e[1] . k_vec
(%o21) 0
(%i22) e[2]:[0,1,0]
```

```
(%i23) e[2] . e[2]
(%o23) 1
(%i24) e[2] . e[1]
(%o24) 0
(%i25) e[2] . k_vec
(%o25) 0
(%i26) " ep[1] X ep[2] = kp_vec/kp "
(%i27) " parallel to the scattering plane "
(%i28) ep[1]:[cos(th),0,-sin(th)]
(%i29) trigsimp(ep[1] . ep[1])
(%o29) 1
(%i30) ep[1] . kp_vec
(%o30) 0
(%i31) " perpendicular to the scattering plane "
(%i32) ep[2]:[0,1,0]
(%i33) ep[2] . ep[2]
(%o33) 1
(%i34) ep[2] . ep[1]
(%o34) 0
(%i35) ep[2] . kp_vec
(%o35) 0
(%i36) "============================================="
(%i37) " photon transverse polarization sum over both leads to "
(%i38) "  (1 + cos(th)^2 ) "
(%i39) sum(sum((e[s] . ep[r])^2,s,1,2),r,1,2)
(%o39) cos(th)^2+1
(%i40) "============================================="
(%i41) " Sum over only e[1] and e[2] values leads to "
(%i42) " sum ( (e[s] . ep[r])^2,s,1,2) = 1 - (ku . ep[r])^2 "
(%i43) " where ku is unit vector along k_vec "
(%i44) ku:k_vec/k
(%o44) [0,0,1]
(%i45) " first for r = 1 "
(%i46) sum((e[s] . ep[1])^2,s,1,2)
(%o46) cos(th)^2
(%i47) trigsimp(1-(ku . ep[1])^2)
(%o47) cos(th)^2
(%i48) " next for r = 2 "
(%i49) sum((e[s] . ep[2])^2,s,1,2)
(%o49) 1
(%i50) 1-(ku . ep[2])^2
(%o50) 1
(%i51) "============================================="
(%o51) "photon1.mac"
```

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

We will use the following relation 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 coresond to $\lambda = 1, 2$, and for $\mu,\ \nu = 0, 1, 2, 3$,

$$P^{\mu\,\nu}(k) = \sum_{\lambda=1,2} e^{\mu}_{k\,\lambda}\, e^{\nu *}_{k\,\lambda} = -g^{\mu\,\nu} + \frac{(k^{\mu}\,\bar{k}^{\nu} + k^{\nu}\,\bar{k}^{\mu})}{k\cdot\bar{k}}, \tag{12.54}$$

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

$$\bar{k}^{\,\mu} = -k^{\,\mu} + 2\,n^\mu\,(k\cdot n). \tag{12.55}$$

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

$$e^*_{k\,\lambda}\cdot e_{k\,\lambda'} = -\delta_{\lambda\,\lambda'}, \quad \lambda,\lambda' = 1,2 \tag{12.56}$$

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

$$P^{\mu\,\nu}(k) = \sum_{\lambda=1,2} e^\mu_{k\,\lambda}\,e^{\nu *}_{k\,\lambda} = -g^{\mu\,\nu} - \frac{k^\mu\,k^\nu}{(k\cdot n)^2} + \frac{(k^\mu\,n^\nu + k^\nu\,n^\mu)}{k\cdot n} \tag{12.57}$$

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: $\mathbf{e}^*_{k\,1}\cdot\mathbf{e}_{k\,2} = 0$.

12.12 compton0.mac: Compton Scattering by a Spin 0 Structureless Charged Particle

In this section we use the batch file **`compton0.mac`** to work out details of Compton scattering of scalar charged particles. Some references to this subject can be found in G/R QED, p 435, B/D RQM, p. 193, Schwinger PSF I, p290, Aitchison RQM, p87, De Wit/Smith, p 147, I/Z p. 286.

Three diagrams contribute to Compton scattering of a scalar charged particle which we can for convenience think of as

$$\pi^+(p_1 + \gamma(k_1,\lambda_1) \to \pi^+(p_2 + \gamma(k_2,\lambda_2) \tag{12.58}$$

Let

$$s = (k_1 + p_1)^2, \quad t = (k_1 - k_2)^2, \quad u = (k_1 - p_2)^2. \tag{12.59}$$

Four momentum conservation means $p_1 + k_1 = p_2 + k_2$. If we ignore a factor $-e^2$ and use real photon polarization 4-vectors, the invariant amplitude is

$$M = e^\mu_{k_2\,\lambda_2}\left(2\,g_{\mu\,\nu} - \frac{(2\,p_2 + k_2)_\mu\,(2\,p_1 + k_1)_\nu}{s - m^2} - \frac{(2\,p_1 - k_2)_\mu\,(2\,p_2 - k_1)_\nu}{u - m^2}\right) e^\nu_{k_1\,\lambda_1} \tag{12.60}$$

Use of $k_1\cdot e_{k_1\,\lambda_1} = 0$, $k_2\cdot e_{k_2\,\lambda_2} = 0$, $s - m^2 = 2\,p_1\cdot k_1$, and $u - m^2 = -2\,p_2\cdot k_1$ leads to

$$M = 2\,e^\mu_{k_2\,\lambda_2}\,T_{\mu\,\nu}\,e^\nu_{k_1\,\lambda_1} \tag{12.61}$$

where

$$T_{\mu\,\nu} = g_{\mu\,\nu} - \frac{p_{2\,\mu}\,p_{1\,\nu}}{k_1\cdot p_1} + \frac{p_{2\,\nu}\,p_{1\,\mu}}{k_1\cdot p_2} \tag{12.62}$$

We form the absolute square of M and sum over λ_2 first,

$$\sum_{\lambda_2}|M|^2 = 4\,T_{\mu\,\nu}\,e^\nu_{k_1\,\lambda_1}\,P^{\mu\,\alpha}(k_2)\,T_{\alpha\,\beta}\,e^\beta_{k_1\,\lambda_1} = -4\,T_{\mu\,\nu}\,T^\mu_{\ \beta}\,e^\nu_{k_1\,\lambda_1}\,e^\beta_{k_1\,\lambda_1} \tag{12.63}$$

Using the compact form of $P^{\mu\alpha}(k_2)$, only the first term contributes at this stage. The second term contribution is proportional to
$T_{\mu\nu}\, e^{\nu}_{k_1\,\lambda_1}\, k_2^{\mu}\, \bar{k}_2^{\alpha}\, T_{\alpha\beta}\, e^{\beta}_{k_1\,\lambda_1}$. Using $k_1 \cdot p_1 = k_2 \cdot p_2$ and $k_1 \cdot p_2 = k_2 \cdot p_1$,

$$k_2^{\mu}\, T_{\mu\nu} = (k_2 + p_2 - p_1)_{\nu} = k_{1\,\nu} \tag{12.64}$$

But then $k_{1\,\nu}\, e^{\nu}_{k_1\,\lambda_1} = 0$. Likewise the third term contribution is zero because $k_2^{\alpha}\, T_{\alpha\beta} = k_{1\,\beta}$, and $k_{1\,\beta}\, e^{\beta}_{k_1\,\lambda_1} = 0$.

We now sum over λ_1.

$$\sum_{\lambda_1,\,\lambda_2} |M|^2 = -4\, T_{\mu\nu}\, P^{\nu\beta}(k_1)\, T^{\mu}_{\ \beta} = 4\, \left(T_{\mu\nu}\, T^{\mu\nu} - 2\right), \tag{12.65}$$

in which all three terms of $P^{\nu\beta}(k_1)$ contribute.

For example, the second term is proportional to $\left(T_{\mu\nu}\, k_1^{\nu}\, \bar{k}_1^{\beta}\, T^{\mu}_{\ \beta}\right) / \left(k_1 \cdot \bar{k}_1\right)$. The numerator is equal to the denominator, since $k_1^{\nu}\, T_{\mu\nu} = (k_1 + p_1 - p_2)_{\mu} = k_{2\,\mu}$, and $k_{2\,\mu}\, T^{\mu}_{\ \beta} = k_{1\,\beta}$, as we saw above.

In the **`compton0.mac`** code, the expression $T^{\mu\nu}$ is represented by **`t_munu`**, written in terms of the constructs **`D`**, **`UI`**, and **`Gm`** which the Dirac package recognises. The contraction $T_{\mu\nu}\, T^{\mu\nu}$ becomes **`Con (t_munu^2)`**.

The batch file **`compton0.mac`** uses the Dirac package functions **`VP`** and **`ev_Ds`**.

```
(%i3) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("compton0.mac");
read and interpret file: #pc:/work5/compton0.mac
(%i3) " ======================================================"
(%i4) "  file compton0.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) " COMPTON SCATTERING OF STRUCTURELESS SPIN 0 PARTICLES "
(%i11) "  which we will call (pi,gamma) scattering "
(%i12) "  gamma(k1,s1) + pi(p1) --> gamma(k2,s2) + pi(p2) "
(%i13) " POPULATE THE LIST invarR OF 4-VEC DOT PRODUCT VALUES. "
(%i14) " Express 4-vector dot products in terms of s, t, u, and m^2 "
(%i15) " Using s = (p1+k1)^2 = (p2+k2)^2, t = (k2-k1)^2 = "
(%i16) " (p2-p1)^2, u = (k2-p1)^2 = (p2-k1)^2, s + t + u = 2*m^2 "
(%i17) invar(D(p1,p1) = m^2,D(p1,k1) = (s-m^2)/2,D(p1,k2) = (-(u-m^2))/2,
             D(p1,p2) = (u+s)/2,D(p2,p2) = m^2,D(p2,k1) = (-(u-m^2))/2,
             D(p2,k2) = (s-m^2)/2,D(k1,k1) = 0,D(k1,k2) = (u+s)/2-m^2,
             D(k2,k2) = 0)
(%i18) " 1. UNPOLARIZED RESULT IN ARBITRARY FRAME (s,t,u form) "
(%i19) " amplitude is M_fi "
(%i20) " M_fi = -2*e^2*( D(e2_cc,e1) - D(e2_cc,p2)*D(e1,p1)/D(k1,p1) "
(%i21) "               + D(e2_cc,p1)*D(e1,p2)/D(k1,p2) ) "
(%i22) " The parenthesis expression can be written as the "
(%i23) " contraction (assume here real pol. vectors ) "
(%i24) "  sum (sum (e2[mu]*T[mu,nu]*e1[mu],mu,0,3),nu,0,3) "
(%i25) " where T[mu,nu] is the expression t_munu : "
```

```
(%i26) " ------------------------------------------------"
(%i27) t_munu:UI(p1,mu)*UI(p2,nu)/D(k1,p2)-UI(p1,nu)*UI(p2,mu)/D(k1,p1)
                                        +Gm(mu,nu)
(%i28) "========================================================="
(%i29) "  Note Schwinger's expression (3-12.93) used here, "
(%i30) "  which he derives on page 73, and further discusses p.294"
(%i31) " The Lorentz gauge photon polarization 4-vector always has"
(%i32) " the properties: 1). D(k,e(k,s)) = 0, "
(%i33) "  2). D(n,e(k,s)) = 0, 3). D(kbar,e(k,s)) = 0 "
(%i34) " 4). D(e(k,s)_cc,e(k,r)) = -kroneker-delta(s,r) "
(%i35) " Within the Lorentz gauge one still has gauge freedom "
(%i36) " since one can choose the unit timelike vector n^mu  to"
(%i37) " have any components, with kbar^mu = -k^mu + 2*n^mu*D(k,n). "
(%i38) " In an frame in which n = (1,0,0,0) (which --> D(n,n) = 1)"
(%i39) " the spatial components of kbar are the opposite of those "
(%i40) " of k. One can identify n with any convenient 4-vector in "
(%i41) " the problem "
(%i42) "========================================================="
(%i43) " Evaluate Mssq = abs. value squared of M_fi."
(%i44) " pull out factor e^4 and sum over polarizations of both photons "
(%i45) " The result is : "
(%i46) Mssq:ratsimp(4*(Con(t_munu^2)-2))
(%o46) ((8*s^2+8*m^4)*u^2-32*m^6*u+8*m^4*s^2-32*m^6*s+40*m^8)
       /((s^2-2*m^2*s+m^4)*u^2+(-2*m^2*s^2+4*m^4*s-2*m^6)*u+m^4*s^2-2*m^6*s
                              +m^8)
(%i47) " LAB FRAME EVALUATION "
(%i48) " initial pion rest frame evaluation in terms of"
(%i49) " scattering angle of final photon relative to initial"
(%i50) " photon direction (z - x plane) "
(%i51) " kinematics with initial 'pion' p1 at rest. "
(%i52) " initial photon k1 moving along positive z axis. "
(%i53) assume(m > 0,k > 0,kp > 0,th >= 0,th <= %pi)
(%i54) comp_def(p1(m,0,0,0),k1(k,0,0,k),k2(kp,kp*sin(th),0,kp*cos(th)),
               p2(-kp+k+m,-kp*sin(th),0,k-kp*cos(th)))
(%i55) " ------------------------------------------------------"
(%i56) " conservation of relativistic energy gives "
(%i57) "  m + k = E_p2 + kp "
(%i58) " and conservation of 3 or 4-momentum implies "
(%i59) "  kp = m*k/(m + k*(1-cos(th)))    "
(%i60) " ---------------------------------------"
(%i61) " find replacement rule for kp using conservation of 4-mom"
(%i62) kp_rule:solve(VP(-k2+k1+p1,-k2+k1+p1) = ev_invar(D(p2,p2)),kp)
(%o62) [kp = -k*m/(k*cos(th)-m-k)]
(%i63) s_th:VP(k1+p1,k1+p1)
(%o63) m^2+2*k*m
(%i64) t_th:(VP(k1-k2,k1-k2),ev(%%,kp_rule))
(%o64) 2*k^2*m/(k*cos(th)-m-k)-2*k^2*m*cos(th)/(k*cos(th)-m-k)
(%i65) u_th:(VP(k2-p1,k2-p1),ev(%%,kp_rule))
(%o65) 2*k*m^2/(k*cos(th)-m-k)+m^2
(%i66) " replace s by s_th, etc "
(%i67) Mssq_lab:factor(sub_stu(Mssq))
(%o67) 4*(cos(th)^2+1)
(%i68) " having pulled out e^4 from Mssq "
(%i69) A:alpha^2*(kp/k)^2/(4*m^2)
(%o69) alpha^2*kp^2/(4*k^2*m^2)
(%i70) " ------------------------------------ "
(%i71) " To get unpolarized differential cross section, "
(%i72) " divide sum over spins Mssq_lab by 2 from average "
(%i73) " over polarization of the  initial photon. "
(%i74) dsigdo_lab_unpol:A*Mssq_lab/2
```

```
(%i75) (display2d:true,display(dsigdo_lab_unpol),display2d:false)
                                      2   2     2
                                 alpha  kp  (cos (th) + 1)
                dsigdo_lab_unpol = ------------------------
                                             2  2
                                          2 k  m

(%i76) " which agrees with G/R QED p. 437 Eq.(14)  "
(%i77) " since r_0^2 = alpha^2/m^2 "
(%i78) " ======================================="
(%i79) "-----------------------------------------------"
(%i80) " 2. LINEAR PHOTON POLARIZATION CASE     "
(%i81) " ------  lab frame, Coulomb gauge  --------------"
(%i82) " Leave real photon polarization 4-vectors e1,e2 in amplitude. "
(%i83) " replace p1 dot products using lab frame coulomb gauge."
(%i84) " replace p2 dot products with e1,e2 "
(%i85) "                 using 4-mom conservation relation. "
(%i86) " The D(p1,e1) and D(p1,e2) replacements depend on our "
(%i87) "               lab frame Coulomb gauge choices. "
(%i88) " Recall that D has property symmetric, so  "
(%i89) " D(a,b) = D(b,a). "
(%i90) " choose real polarization vectors => linear pol."
(%i91) invar(D(e1,e1) = -1,D(k1,e1) = 0,D(e2,e2) = -1,D(k2,e2) = 0,
             D(p1,e1) = 0,D(p1,e2) = 0,D(e1,p2) = -D(e1,k2),
             D(e2,p2) = D(e2,k1))
(%i92) Ampl:D(e2,c)*D(e1,d)/(u-m^2)+D(e2,a)*D(e1,b)/(s-m^2)-2*D(e1,e2)
(%o92) D(c,e2)*D(d,e1)/(u-m^2)+D(a,e2)*D(b,e1)/(s-m^2)-2*D(e1,e2)
(%i93) " the amplitude is real so just need Ampl^2 for cross section "
(%i94) Ampsq:expand(Ampl^2)
(%o94) D(c,e2)^2*D(d,e1)^2/(u^2-2*m^2*u+m^4)
        +2*D(a,e2)*D(b,e1)*D(c,e2)*D(d,e1)/(s*u-m^2*u-m^2*s+m^4)
        -4*D(c,e2)*D(d,e1)*D(e1,e2)/(u-m^2)
        +D(a,e2)^2*D(b,e1)^2/(s^2-2*m^2*s+m^4)
        -4*D(a,e2)*D(b,e1)*D(e1,e2)/(s-m^2)+4*D(e1,e2)^2
(%i95) "  replace a,b,c,d "
(%i96) Ampsq1:subst([a = k2+2*p2,b = k1+2*p1,c = 2*p1-k2,d = 2*p2-k1],Ampsq)
(%o96) D(e1,2*p2-k1)^2*D(e2,2*p1-k2)^2/(u^2-2*m^2*u+m^4)
        +2*D(e1,2*p1+k1)*D(e1,2*p2-k1)*D(e2,2*p1-k2)*D(e2,2*p2+k2)
         /(s*u-m^2*u-m^2*s+m^4)-4*D(e1,e2)*D(e1,2*p2-k1)*D(e2,2*p1-k2)/(u-m^2)
        +D(e1,2*p1+k1)^2*D(e2,2*p2+k2)^2/(s^2-2*m^2*s+m^4)
        -4*D(e1,e2)*D(e1,2*p1+k1)*D(e2,2*p2+k2)/(s-m^2)+4*D(e1,e2)^2
(%i97) " expand dot products and use invariants list invarR "
(%i98) Ampsq2:ev_Ds(Ampsq1)
(%o98) 4*D(e1,e2)^2
(%i99) " we see that there is no further reference to s and u in this result"
(%i100) " multiply by A to get polarized cross section "
(%i101) dsigdo_lab_pol:A*Ampsq2
(%i102) (display2d:true,display(dsigdo_lab_pol),display2d:false)
                                      2  2           2
                                 alpha  D (e1, e2) kp
                dsigdo_lab_pol = ----------------------
                                             2  2
                                          k  m

(%i103) " which agrees with G/R QED p. 437, Eq.(12)  "
(%i104) " since r_0^2 = alpha^2/m^2 "
(%i105) " In Coulomb gauge lab frame the four vector dot product square "
(%i106) " D(e2,e1)^2 reduces to dot(e2_vec,e1_vec)^2 "
(%i107) " We recover the unpolarized result above if we sum over both "
(%i108) " e1 and e2 values and divide by 2 since we are averaging over "
(%i109) " the e1 values (see above section on transverse pol. vector sums)"
```

```
(%i110) "-----------------------------------------------------------------"
(%i111) " If we only average over the polarization of the incident photon"
(%i112) " and leave the polarization of the final photon arbitrary, we "
(%i113) " replace (1+cos(th)^2) by ( 1 - dot(ku,e2_vec)^2) which gives "
(%i114) " 1 for e2_vec chosen perpendicular to the scattering plane, and "
(%i115) " gives sin(th)^2 for e2_vec chosen parallel to the scattering "
(%i116) " plane, leading to the scattered photons polarized preferentially"
(%i117) " in a direction perpendicular to the scattering plane."
(%o117) "compton0.mac"
```

12.13 compton1.mac: Lepton-photon scattering

The scattering event is

$$e^-(p_1,\sigma_1)+\gamma(k_1,\lambda_1)\rightarrow e^-(p_2,\sigma_2)+\gamma(k_2,\lambda_2). \tag{12.66}$$

The invariant amplitude is $M=M_1+M_2$, where, ignoring the charge squared factor e^2,

$$M_1=\frac{\bar{u}_2\,\Gamma_1\,u_1}{u-m^2},\quad \Gamma_1=\not{\epsilon}_1\,(\not{p}_1-\not{k}_2+m)\,\not{\epsilon}_2^*),\quad u=(p_1-k_2)^2 \tag{12.67}$$

and

$$M_2=\frac{\bar{u}_2\,\Gamma_2\,u_1}{s-m^2},\quad \Gamma_2=\not{\epsilon}_2^*\,(\not{p}_1+\not{k}_1+m)\,\not{\epsilon}_1,\quad s=(p_1+k_1)^2 \tag{12.68}$$

The first section of **`compton1.mac`** calculates the unpolarized differential cross section, proportional to

$$\sum_{\lambda_1,\,\lambda_2,\,\sigma_1,\,\sigma_2}|M_1+M_2|^2. \tag{12.69}$$

One the the four terms is $\sum|M_1|^2$. If we first sum on λ_1, the result is proportional to

$$\sum_{\lambda_1}\bar{u}_2\not{\epsilon}_1\,\Gamma_a\,u_1\,\bar{u}_1\,\Gamma_b\not{\epsilon}_1\,u_2=\left(\sum_{\lambda_1}e^{\mu}_{k_1\,\lambda_1}\,e^{\tau}_{k_1\,\lambda_1}\right)\,\bar{u}_2\,\gamma_\mu\,\Gamma_a\,u_1\,\bar{u}_1\,\Gamma_b\,\gamma_\tau\,u_2 \tag{12.70}$$

A convenient, simplifying feature of the lepton photon interaction is that only the first (metric tensor) term of $P^{\mu\,\tau}(k_1)$ need be retained, since the other terms produce zero contribution (as is discussed in quantum field theory texts). Thus for lepton photon interactions we effectively have

$$P^{\mu\,\nu}(k_1)\Rightarrow -g^{\mu\,\nu} \tag{12.71}$$

Thus the sum on λ_1 above reduces to

$$-\bar{u}_2\,\gamma_\mu\,\Gamma_a\,u_1\,\bar{u}_1\,\Gamma_b\,\gamma^\mu\,u_2. \tag{12.72}$$

After summing over $\lambda_2,\ \sigma_1,\ \sigma_2$ as well, and again ignoring the factor e^4,

$$\sum_{\lambda_1,\,\lambda_2,\,\sigma_1,\,\sigma_2}|M_1|^2=\frac{\text{Trace}\ \{(\not{p}_2+m)\gamma_\mu(\not{p}_1-\not{k}_2+m)\gamma_\nu(\not{p}_1+m)\gamma^\nu(\not{p}_1-\not{k}_2+m)\gamma^\mu\}}{(u-m^2)^2} \tag{12.73}$$

The **`compton1.mac`** code uses the symbol **`m1sq`** for the trace in the numerator, and uses the symbol **`Mssq_unpol`** for $\sum|M_1+M_2|^2$ (see also the run line **`(%i32)`**). The Dirac package functions **`VP`**, **`D_sub`**, **`take_parts`**,**`ts`**, and **`pullfac`** are used in this code.

```
(%i1) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch("compton1.mac");
file_search1: compton1.mac not found in file_search_maxima.
 -- an error. To debug this try: debugmode(true);
(%i3) batch("compton1.mac");
read and interpret file: #pc:/work5/compton1.mac
(%i4) " ======================================================"
(%i5) "  file compton1.mac "
(%i6) "  Maxima by Example, Ch. 12 "
(%i7) "  Dirac Algebra and Quantum Electrodynamics "
(%i8) "  Edwin L Woollett, woollett@charter.net "
(%i9) "  http://www.csulb.edu/~woollett  "
(%i10) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i11) " ---------------------------------------------"
(%i12) "   UNPOLARIZED LEPTON - PHOTON SCATTERING  "
(%i13) " for example : "
(%i14) " ---------------------------------------------"
(%i15) "  e(-,p1,m) + gamma(k1) --> e(-,p2,m) + gamma(k2) "
(%i16) " ---------------------------------------------"
(%i17) "    Arbitrary Frame Covariant Symbolic Methods  "
(%i18) " in an arb frame, D(k1,p1) = D(k2,p2) "
(%i19) "         and D(k1,p2) = D(k2,p1) "
(%i20) " In an arbitrary frame we can express everything "
(%i21) " in terms of m^2, D(k1,p1) and  D(k2,p1) "
(%i22) " with a view to then evaluating in the rest frame"
(%i23) " of the initial electron: p1 = (m,0,0,0) (lab frame) "
(%i24) " ---------------------------------------------"
(%i25) invar(D(p1,p1) = m^2,D(p2,p2) = m^2,D(k1,k1) = 0,D(k2,k2) = 0,
             D(p1,p2) = -D(k2,p1)+D(k1,p1)+m^2,D(k1,p2) = D(k2,p1),
             D(k2,p2) = D(k1,p1),D(k1,k2) = D(k1,p1)-D(k2,p1))
(%i26) m1sq:tr(m+p2,mu,m-k2+p1,nu,m+p1,nu,m-k2+p1,mu)
(%o26) 32*m^4-32*D(k2,p1)*m^2+32*D(k1,p1)*D(k2,p1)
(%i27) m2sq:tr(m+p2,mu,m+k1+p1,nu,m+p1,nu,m+k1+p1,mu)
(%o27) 32*m^4+32*D(k1,p1)*m^2+32*D(k1,p1)*D(k2,p1)
(%i28) m1m2sq:tr(m+p2,mu,m-k2+p1,nu,m+p1,mu,m+k1+p1,nu)
(%o28) 32*m^4-16*D(k2,p1)*m^2+16*D(k1,p1)*m^2
(%i29) m2m1sq:tr(m+p2,mu,m+k1+p1,nu,m+p1,mu,m-k2+p1,nu)
(%o29) 32*m^4-16*D(k2,p1)*m^2+16*D(k1,p1)*m^2
(%i30) m1m2sq-m2m1sq
(%o30) 0
(%i31) " The interference terms are equal, so  "
(%i32) " ---------------------------------------------"
(%i33) " Form the expression :"
(%i34) "  Mssq_unpol = m1sq/(u-m^2)^2 + m2sq/(s-m^2)^2 "
(%i35) "     + 2*m1m2sq/((u-m^2)*(s-m^2)),  "
(%i36) "     with u-m^2 = -2*D(k2,p1), s-m^2 = 2*D(k1,p1), "
(%i37) "        using u = (p1 - k2)^2, s = (p1 + k1)^2 "
(%i38) " ( we use sloppy notation p^2 for D(p,p) here )"
(%i39) " ---------------------------------------------"
(%i40) Mssq_unpol:expand((-2*m1m2sq)/(4*D(k2,p1)*D(k1,p1))
                         +m2sq/(4*D(k1,p1)^2)+m1sq/(4*D(k2,p1)^2))
(%o40) -16*m^4/(D(k1,p1)*D(k2,p1))+8*m^4/D(k2,p1)^2+8*m^4/D(k1,p1)^2
                                  -16*m^2/D(k2,p1)+16*m^2/D(k1,p1)
                                  +8*D(k2,p1)/D(k1,p1)+8*D(k1,p1)/D(k2,p1)
(%i41) " pull factor of 8 out of definition "
(%i42) Mssq_unpol8:expand(Mssq_unpol/8)
(%o42) -2*m^4/(D(k1,p1)*D(k2,p1))+m^4/D(k2,p1)^2+m^4/D(k1,p1)^2-2*m^2/D(k2,p1)
                                 +2*m^2/D(k1,p1)+D(k2,p1)/D(k1,p1)
                                 +D(k1,p1)/D(k2,p1)
```

```
(%i43) "------------------------------------------------------"
(%i44) "         LAB FRAME EVALUATION UNPOLARIZED             "
(%i45) "          rest frame of initial electron              "
(%i46) " -----------------------------------------------------"
(%i47) assume(m > 0,k > 0,kp > 0,th >= 0,th <= %pi)
(%i48) comp_def(p1(m,0,0,0),k1(k,0,0,k),k2(kp,kp*sin(th),0,kp*cos(th)),
                p2(-kp+k+m,-kp*sin(th),0,k-kp*cos(th)))
(%i49) " use conservation of 4-momentum to find kp "
(%i50) kp_rule:solve(VP(-k2+k1+p1,-k2+k1+p1) = ev_invar(D(p2,p2)),kp)
(%o50) [kp = -k*m/(k*cos(th)-m-k)]
(%i51) " To compare results, only replace kp in terms 1-5 "
(%i52) " leave kp alone in terms 6-7 "
(%i53) " D_sub (e,[D1,D2,..] uses noncov on each D(p1,p2) in list"
(%i54) M1t5:take_parts(Mssq_unpol8,1,5)
(%o54) -2*m^4/(D(k1,p1)*D(k2,p1))+m^4/D(k2,p1)^2+m^4/D(k1,p1)^2-2*m^2/D(k2,p1)
                                +2*m^2/D(k1,p1)
(%i55) M6t7:take_parts(Mssq_unpol8,6,7)
(%o55) D(k2,p1)/D(k1,p1)+D(k1,p1)/D(k2,p1)
(%i56) M6t7:D_sub(M6t7,[D(k1,p1),D(k2,p1)])
(%o56) kp/k+k/kp
(%i57) M1t5:D_sub(M1t5,[D(k1,p1),D(k2,p1)])
(%o57) -2*m^2/(k*kp)+m^2/kp^2+m^2/k^2-2*m/kp+2*m/k
(%i58) M1t5:trigsimp(ev(M1t5,kp_rule))
(%o58) cos(th)^2-1
(%i59) " ts is our alternative trigsimp function "
(%i60) M1t5:ts(M1t5,th)
(%o60) -sin(th)^2
(%i61) " restore overall factor of 8 "
(%i62) Mssq_unpol:8*(M6t7+M1t5)
(%o62) 8*(-sin(th)^2+kp/k+k/kp)
(%i63) A:alpha^2*(kp/k)^2/(4*m^2)
(%o63) alpha^2*kp^2/(4*k^2*m^2)
(%i64) " ------------------------------------ "
(%i65) " To get unpolarized differential cross section, "
(%i66) " divide sum over spins Mssq_unpol by 4 from average "
(%i67) " over spin of initial electron and initial photon. "
(%i68) dsigdo_lab_unpol:A*Mssq_unpol/4
(%i69) (display2d:true,display(dsigdo_lab_unpol),display2d:false)
                                  2   2       2       kp   k
                              alpha  kp  (- sin (th) + -- + --)
                                                     k    kp
            dsigdo_lab_unpol = ---------------------------------
                                          2  2
                                       2 k  m

(%i70) " which agrees with BLP (86.9) "
```

To calculate the differential cross section for given photon (linear) polarizations (but summing over the inital and final elecron helicities), we retain the photon polarization vector factors $\not{e}_{k_1\lambda_1}$ and $\not{e}_{k_2\lambda_2}$ inside the trace created by summing over the electron helicities.

We already have the basic defining inner products $e_{k_1\lambda_1} \cdot e_{k_1\lambda_1} = -1$, $k_1 \cdot e_{k_1\lambda_1} = 0$, $e_{k_2\lambda_2} \cdot e_{k_2\lambda_2} = -1$, and $k_2 \cdot e_{k_2\lambda_2} = 0$.

Since we want a result appropriate to the rest frame of the initial electron p_1, we identify $n^\mu = p_1^\mu/m$ so that the components of n^μ are $(1, 0, 0, 0)$.

Then $n \cdot e_{k_1\lambda_1} = 0$ implies that $e^0_{k_1\lambda_1} = 0$ and $n \cdot e_{k_2\lambda_2} = 0$ implies that $e^0_{k_2\lambda_2} = 0$.

Then $e_{k_1\lambda_1} = (0, \mathbf{e_{k_1\lambda_1}})$ and $e_{k_2\lambda_2} = (0, \mathbf{e_{k_2\lambda_2}})$.

Since the components of p_1^μ are $(m, 0, 0, 0)$, we then have $p_1 \cdot e_{k_1\lambda_1} = 0$ and $p_1 \cdot e_{k_2\lambda_2} = 0$. And using 4-momentum conservation, the former equation implies $0 = e_{k_1\lambda_1} \cdot (k_2 + p_2 - k_1)$, and hence $e_{k_1\lambda_1} \cdot p_2 = -e_{k_1\lambda_1} \cdot k_2$.

Finally we replace $e_{k_2\lambda_2} \cdot p_2 = e_{k_2\lambda_2} \cdot k_1$, using again 4-momentum conservation and the previous relations.

In our code, **e1** stands for $e_{k_1\lambda_1}$ and **e2** stands for $e_{k_2\lambda_2}$.

```
(%i71) "======================================================"
(%i72) "----UNPOLARIZED ELECTRONS, POLARIZED PHOTONS ------------"
(%i73) "------- rest frame of initial electron p1 ------------- "
(%i74) " Leave photon polarization 4-vectors e1,e2 in trace. "
(%i75) " replace p1 dot products using p1 rest frame choice."
(%i76) " replace p2 dot products with e1,e2 "
(%i77) "                  using 4-mom conservation relation. "
(%i78) " The D(p1,e1) and D(p1,e2) replacements depend on our "
(%i79) "               p1 rest frame choice."
(%i80) " Recall that D has property symmetric, so  "
(%i81) " D(a,b) = D(b,a). "
(%i82) invar(D(e1,e1) = -1,D(k1,e1) = 0,D(e2,e2) = -1,D(k2,e2) = 0,
             D(p1,e1) = 0,D(p1,e2) = 0,D(e1,p2) = -D(e1,k2),
             D(e2,p2) = D(e2,k1))
(%i83) M1sq:tr(m+p2,e1,m-k2+p1,e2,m+p1,e2,m-k2+p1,e1)
(%o83) 8*D(k1,p1)*D(k2,p1)-16*D(e1,k2)^2*D(k2,p1)
(%i84) M2sq:tr(m+p2,e2,m+k1+p1,e1,m+p1,e1,m+k1+p1,e2)
(%o84) 8*D(k1,p1)*D(k2,p1)+16*D(e2,k1)^2*D(k1,p1)
(%i85) M1m2sq:tr(m+p2,e1,m-k2+p1,e2,m+p1,e1,m+k1+p1,e2)
(%o85) -16*D(e1,e2)^2*D(k1,p1)*D(k2,p1)+8*D(k1,p1)*D(k2,p1)
                                       +8*D(e2,k1)^2*D(k2,p1)
                                       -8*D(e1,k2)^2*D(k1,p1)
(%i86) M2m1sq:tr(m+p2,e2,m+k1+p1,e1,m+p1,e2,m-k2+p1,e1)
(%o86) -16*D(e1,e2)^2*D(k1,p1)*D(k2,p1)+8*D(k1,p1)*D(k2,p1)
                                       +8*D(e2,k1)^2*D(k2,p1)
                                       -8*D(e1,k2)^2*D(k1,p1)
(%i87) M2m1sq-M1m2sq
(%o87) 0
(%i88) " The interference terms are equal, so  "
(%i89) Mssq_pol:expand((-2*M1m2sq)/(4*D(k2,p1)*D(k1,p1))
                       +M2sq/(4*D(k1,p1)^2)+M1sq/(4*D(k2,p1)^2))
(%o89) 2*D(k2,p1)/D(k1,p1)+2*D(k1,p1)/D(k2,p1)+8*D(e1,e2)^2-4
(%i90) " avoid use of noncov(D(e1,e2)) which would expand D(e1,e2)"
(%i91) " in terms of undefined components, by instead using ev"
(%i92) " or D_sub  for selected replacements. "
(%i93) Mssq_pol:D_sub(Mssq_pol,[D(k1,p1),D(k2,p1)])
(%o93) 2*kp/k+2*k/kp+8*D(e1,e2)^2-4
(%i94) Mssq_pol:pullfac(Mssq_pol,2)
(%o94) 2*(kp/k+k/kp+4*D(e1,e2)^2-2)
(%i95) "To get electron unpolarized, photon polarized differential cross section,"
(%i96) " divide  Mssq_pol by 2 from average over helicity of initial electron. "
(%i97) dsigdo_lab_pol:A*Mssq_pol/2
(%i98) (display2d:true,display(dsigdo_lab_pol),display2d:false)
                                2   2  kp   k       2
                           alpha  kp  (-- + -- + 4 D (e1, e2) - 2)
                                      k    kp
           dsigdo_lab_pol = ---------------------------------------
                                           2  2
                                       4 k  m

(%i99) " which agrees with the Klein-Nishina form quoted by Weinberg (8.7.39)"
```

```
(%i100) "============================================================"
(%i101) " Now assume unpolarized initial photon. Average  the "
(%i102) " polarized photon dsigdo(r,s) over the two possible values "
(%i103) " of r which specifies the transverse polarization "
(%i104) " vector of incident photon e1_vec. With n^mu = (1,0,0,0)"
(%i105) " D(e1,e2)^2 --> { dot3 (e1_vec,e2_vec) }^2 "
(%i106) " and sum (D(e1(r),e2(s))^2,r,1,2) --> 1 - {dot3(ku,e2_vec(s)}^2"
(%i107) " with ku = unit 3-vector along the z axis, dot3(a,b) = 3-vec dot prod. "
(%i108) " and dsigdo(s) = sum(dsigdo(r,s),r,1,2)/2 "
(%i109) " The overall result is recovered with: "
(%i110) dsigdo_s:subst(4*D(e1,e2)^2 = 2-2*dot3(ku,e2_vec)^2,dsigdo_lab_pol)
(%i111) (display2d:true,display(dsigdo_s),display2d:false)
                          2   2          2                kp   k
                      alpha  kp  (- 2 dot3 (ku, e2_vec) + -- + --)
                                                         k    kp
            dsigdo_s = ------------------------------------------
                                       2  2
                                    4 k  m

(%i112) "------------------------------------------------------------"
(%i113) "  which agrees with Weinberg (8.7.40) "
(%i114) " Summing the above over the two possible values of "
(%i115) " final photon polarization s then leads to our "
(%i116) " unpolarized result above "
(%o116) "compton1.mac"
```

12.14 pair1.mac: Unpolarized Two Photon Pair Annihilation

The pair annihilation to two photons event is

$$e^-(p_1,\sigma_1) + e^+(p_2,\sigma_2) \to \gamma(k_1,\lambda_1) + \gamma(k_2,\lambda_2). \tag{12.74}$$

With the definitions

$$s = (p_1+p_2)^2, \quad t = (p_1-k_1)^2, \quad u = (p_1-k_2)^2, \tag{12.75}$$

and the abbreviations

$$u_1 = u(p_1,\sigma_1), \quad \bar{v}_2 = \bar{v}(p_2,\sigma_2), \quad e_1^* = e^*_{k_1,\lambda_1}, \quad e_2^* = e^*_{k_2,\lambda_2}, \tag{12.76}$$

$M = M_1 + M_2$, with

$$M_1 = \frac{e^2\,\bar{v}_2\,\not{e}_2^*\,(\not{p}_1 - \not{k}_1 + m)\,\not{e}_1^*\,u_1}{t-m^2} \tag{12.77}$$

and

$$M_2 = \frac{e^2\,\bar{v}_2\,\not{e}_1^*\,(\not{p}_1 - \not{k}_2 + m)\,\not{e}_2^*\,u_1}{u-m^2}. \tag{12.78}$$

Some references for this calculation are Griffith p. 241, G/R QED: (p. 192 rest frame of electron, p. 198 CM frame), P/S p. 168, BLP p. 370, Schwinger p. 314, Kaku p. 170, I/Z p. 230, and B/D p. 132.

The batch file **`pair1.mac`** works out the unpolarized differential cross section first in an arbitrary frame, and then in the CM frame. To compare our result with G/R in the CM frame, we express the result in terms of **`(E,v)`**, where **`p = E*v, E^2 = m^2 + p^2`**, with **`v`** (the dimensionless velocity of either of the incident leptons relative to the CM frame) having values $0 \le v < 1$.

The Dirac package functions **`D_sub`** and **`ts`** are used.

```
(%i117) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("pair1.mac");
read and interpret file: #pc:/work5/pair1.mac
(%i3) " ================================================="
(%i4) "  file pair1.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) " -----------------------------------------"
(%i11) "      UNPOLARIZED PAIR ANNIHILATION          "
(%i12) " -----------------------------------------"
(%i13) "  e(-,p1) + e(+,p2) --> gamma(k1) + gamma(k2) "
(%i14) " -----------------------------------------"
(%i15) "   Arbitrary Frame Covariant Symbolic Methods  "
(%i16) " -----------------------------------------"
(%i17) " POPULATE THE LIST invarR OF 4-VEC DOT PRODUCT VALUES. "
(%i18) " Express 4-vector dot products in terms of t, u, and m^2 "
(%i19) " Using s = (p1+p2)^2 = (k1+k2)^2, t = (p1-k1)^2 = "
(%i20) " (k2-p2)^2, u = (p1-k2)^2 = (k1-p2)^2, s + t + u = 2*m^2 "
(%i21) " -------------------------------------------------"
(%i22) invar(D(p1,p1) = m^2,D(p1,k1) = (m^2-t)/2,D(p1,k2) = (m^2-u)/2,
             D(p1,p2) = (-(u+t))/2,D(p2,p2) = m^2,D(p2,k1) = (m^2-u)/2,
             D(p2,k2) = (m^2-t)/2,D(k1,k1) = 0,D(k1,k2) = m^2-(u+t)/2,
             D(k2,k2) = 0)
(%i23) "-------------------------------------------------"
(%i24) "  sum |M|^2 over spins of electrons and polarizations "
(%i25) "    of the  photons, which leads to the expression: "
(%i26) " -------------------------------------------------"
(%i27) "      m1sq/(t-m^2)^2 + m2sq/(u-m^2)^2 +           "
(%i28) "           (m1m2sq+m2m1sq)/((t-m^2)*(u-m^2))"
(%i29) " -------------------------------------------------"
(%i30) " --    where (ignoring e^4)  ---                  "
(%i31) m1sq:tr(p2-m,mu,m-k1+p1,nu,m+p1,nu,m-k1+p1,mu)
(%o31) 8*t*u-8*m^2*u-24*m^2*t-8*m^4
(%i32) m2sq:tr(p2-m,mu,m-k2+p1,nu,m+p1,nu,m-k2+p1,mu)
(%o32) 8*t*u-24*m^2*u-8*m^2*t-8*m^4
(%i33) m1m2sq:tr(p2-m,mu,m-k1+p1,nu,m+p1,mu,m-k2+p1,nu)
(%o33) -8*m^2*u-8*m^2*t-16*m^4
(%i34) m2m1sq:tr(p2-m,mu,m-k2+p1,nu,m+p1,mu,m-k1+p1,nu)
(%o34) -8*m^2*u-8*m^2*t-16*m^4
(%i35) m1m2sq-m2m1sq
(%o35) 0
(%i36) Mssq:ratsimp(2*m1m2sq/((t-m^2)*(u-m^2))+m2sq/(u-m^2)^2+m1sq/(t-m^2)^2)
(%o36) ((8*t-8*m^2)*u^3+(24*m^4-56*m^2*t)*u^2+(8*t^3-56*m^2*t^2+112*m^4*t)*u
                    -8*m^2*t^3+24*m^4*t^2-48*m^8)
       /((t^2-2*m^2*t+m^4)*u^2+(-2*m^2*t^2+4*m^4*t-2*m^6)*u+m^4*t^2-2*m^6*t
                           +m^8)
(%i37) "----- replace t and u in terms of dot products in arb frame  ------ "
(%i38) Mssq_dot:expand(subst([t = m^2-2*D(k1,p1),u = m^2-2*D(k2,p1)],Mssq))
(%o38) -16*m^4/(D(k1,p1)*D(k2,p1))-8*m^4/D(k2,p1)^2-8*m^4/D(k1,p1)^2
                                  +16*m^2/D(k2,p1)+16*m^2/D(k1,p1)
                                  +8*D(k2,p1)/D(k1,p1)+8*D(k1,p1)/D(k2,p1)
(%i39) "================================================="
```

```
(%i40) "  specialize to center of momentum frame  "
(%i41) "-------------------------------------------------"
(%i42) " kinematics in center of momentum frame "
(%i43) " p_vec_mag : E*v, v = velocity rel to CM frame "
(%i44) " E^2 = m^2 + p^2 = m^2/(1 - v^2) "
(%i45) " initial electron 3-mom in dir. of pos z axis"
(%i46) " e(-,p1) e(+,p2) --> gamma(k1) gamma(k2) "
(%i47) " CM energy = 2*E shared equally by incident"
(%i48) " fermions and outgoing photons "
(%i49) assume(m > 0,E > 0,th >= 0,th <= %pi)
(%i50) comp_def(p1(E,0,0,v*E),p2(E,0,0,-v*E),k1(E,E*sin(th),0,E*cos(th)),
               k2(E,-E*sin(th),0,-E*cos(th)))
(%i51) " pf/pi = E/p = E/(vE) = 1/v "
(%i52) " dsigdo_CM = A*|M|^2 for given pol, where "
(%i53) " having absorbed e^4 into A, e^2 = 4*%pi*alpha"
(%i54) A:alpha^2/(16*v*E^2)
(%o54) alpha^2/(16*v*E^2)
(%i55) " convert to a function of angle th between "
(%i56) " p1_vec and k1_vec "
(%i57) Mssq_th:D_sub(Mssq_dot,[D(k1,p1),D(k2,p1)])
(%o57) -16*m^4/((E^2-cos(th)*v*E^2)*(cos(th)*v*E^2+E^2))
        +8*(E^2-cos(th)*v*E^2)/(cos(th)*v*E^2+E^2)+16*m^2/(cos(th)*v*E^2+E^2)
        -8*m^4/(cos(th)*v*E^2+E^2)^2+16*m^2/(E^2-cos(th)*v*E^2)
        -8*m^4/(E^2-cos(th)*v*E^2)^2+8*(cos(th)*v*E^2+E^2)/(E^2-cos(th)*v*E^2)
(%i58) Mssq_th:trigsimp(subst(E^2 = m^2/(1-v^2),Mssq_th))
(%o58) -((16*sin(th)^4+16)*v^4-32*sin(th)^2*v^2-16)
        /(cos(th)^4*v^4-2*cos(th)^2*v^2+1)
(%i59) " simplify the denominator - override default factor "
(%i60) mt1d:denom(Mssq_th)
(%o60) cos(th)^4*v^4-2*cos(th)^2*v^2+1
(%i61) mt1d:ratsubst(x,v^2*cos(th)^2,mt1d)
(%o61) x^2-2*x+1
(%i62) mt1d:factor(mt1d)
(%o62) (x-1)^2
(%i63) mt1d:subst(x = v^2*cos(th)^2,mt1d)
(%o63) (cos(th)^2*v^2-1)^2
(%i64) " simplify the numerator "
(%i65) mt1n:num(Mssq_th)
(%o65) -(16*sin(th)^4+16)*v^4+32*sin(th)^2*v^2+16
(%i66) " pull out factor of 16 from numerator "
(%i67) mt1n16:factor(mt1n)/16
(%o67) -sin(th)^4*v^4-v^4+2*sin(th)^2*v^2+1
(%i68) " change sin(th) to cos(th) in numerator "
(%i69) mt1n16:expand(ts(mt1n16,th))
(%o69) -cos(th)^4*v^4+2*cos(th)^2*v^4-2*v^4-2*cos(th)^2*v^2+2*v^2+1
(%i70) " simplified num/denom expression: "
(%i71) Mssq_unpol:16*mt1n16/mt1d
(%o71) 16*(-cos(th)^4*v^4+2*cos(th)^2*v^4-2*v^4-2*cos(th)^2*v^2+2*v^2+1)
        /(cos(th)^2*v^2-1)^2
(%i72) " divide by 4 from averaging over initial spin "
(%i73) " and polarization "
(%i74) dsigdo_CM:A*Mssq_unpol/4
(%i75) (display2d:true,display(dsigdo_CM),display2d:false)
dsigdo_CM =
            2       4      4        2      4      4        2      2      2
       alpha  (- cos (th) v  + 2 cos (th) v  - 2 v  - 2 cos (th) v  + 2 v  + 1)
       ------------------------------------------------------------------------
                                  2      2     2  2
                           4 v (cos (th) v  - 1)  E
```

```
(%i76) " which agrees with Greiner and Reinhardt, p.201, eqn (17)"
(%i77) "------------------------------------------------------------"
(%o77) "pair1.mac"
```

12.15 pair2.mac: Polarized Two Photon Pair Annihilation Amplitudes

The pair annihilation to two photon event is

$$e^-(p_1,\sigma_1) + e^+(p_2,\sigma_2) \to \gamma(k_1,\lambda_1) + \gamma(k_2,\lambda_2). \tag{12.79}$$

With the definitions

$$s = (p_1+p_2)^2, \quad t = (p_1-k_1)^2, \quad u = (p_1-k_2)^2. \tag{12.80}$$

If we use explicit Dirac spinors and matrices and photon polarization 3-vectors to calculate polarized amplitudes in which both the leptons and photons have definite helicities in the CM frame, we start with the polarized amplitude already written down in the previous section dealing with **`pair1.mac`**.

With the abbreviations

$$u_1 = u(p_1,\sigma_1), \quad \bar v_2 = \bar v(p_2,\sigma_2), \quad e_1^* = e^*_{k_1,\lambda_1}, \quad e_2^* = e^*_{k_2,\lambda_2}, \tag{12.81}$$

$M = M_1 + M_2$, with

$$M_1 = \frac{e^2\,\bar v_2 \not e_2^*\,(\not p_1 - \not k_1 + m)\,\not e_1^*\,u_1}{t-m^2} \tag{12.82}$$

and

$$M_2 = \frac{e^2\,\bar v_2 \not e_1^*\,(\not p_1 - \not k_2 + m)\,\not e_2^*\,u_1}{u-m^2}. \tag{12.83}$$

Here we use explicit Dirac spinor methods in the CM frame, letting each lepton have the energy **`E = M/2`**, with **`M`** the total CM frame energy, and then each lepton has 3-momentum magnitude **`p = E*v = M*v/2`**. We also have the dimensionless lepton CM frame speed **`v = sqrt( 1 - 4*m^2/M^2 )`**. Each outgoing photon has energy **`k = M/2`** and each photon has 3-momentum magnitude **`k`**.

We use the photon polarization conventions of F. Gross. (See Franz Gross, Rel. Q. Mech. and Fld. Theory, pages 54, 218, and Schwinger, PSF I, Sec. 3-13-98.)

The outgoing photons are chosen to be travelling along the $+z$ and $-z$ axes. We let $\mathbf{e}(\hat z,\lambda)$ be the photon polarization 3-vector describing a photon travelling in the $+z$ direction, with $\lambda = 1$ being a positive helicity state and $\lambda = -1$ being a negative helicity state. Then

$$\mathbf{e}(\hat z, 1) = -\frac{1}{\sqrt2}\,(\hat x + i\,\hat y), \tag{12.84}$$

and

$$\mathbf{e}(\hat z, -1) = \frac{1}{\sqrt2}\,(\hat x - i\,\hat y). \tag{12.85}$$

We let $\mathbf{e}(-\hat z,\lambda)$ be the photon polarization 3-vector describing a photon travelling in the $-z$ direction, with $\lambda = 1$ being a positive helicity state and $\lambda = -1$ being a negative helicity state. Then

$$\mathbf{e}(-\hat z, 1) = \frac{1}{\sqrt2}\,(\hat x - i\,\hat y) = R_y(\pi)\,\mathbf{e}(\hat z, 1), \tag{12.86}$$

and

$$\mathbf{e}(-\hat{z},-1) = -\frac{1}{\sqrt{2}}\,(\hat{x} + i\,\hat{y}) = R_y(\pi)\,\mathbf{e}(\hat{z},-1). \tag{12.87}$$

For our calculation, since the photons are in the final state, we need complex conjugates of these. The first example discussed is **RR --> RR**, which means two right-handed electrons (actually positive helicity) turn into two positive helicity photons. For this first example, we define CM frame components with the line:

```
comp_def(p1(M/2,M*v*sin(th)/2,0,M*v*cos(th)/2),k1(M/2,0,0,M/2),
              k2(M/2,0,0,(-M)/2),e1_cc(0,(-1)/sqrt(2),%i/sqrt(2),0),
              e2_cc(0,1/sqrt(2),%i/sqrt(2),0))
```

which shows that the photon with 4-momentum **k1** is travelling along the positive **z** axis, and the polarization state of this photon is described by the 4-vector **e1_cc** with the components **(0,(-1)/sqrt(2),%i/sqrt(2),0)** in which the three spatial components agree with $\mathbf{e}(\hat{z},1)^*$.

Likewise, the photon with 4-momentum **k2** is travelling along the negative **z** axis, and the symbol **e2_cc** is the 4-vector describing its positive helicity state, with components **(0,1/sqrt(2),%i/sqrt(2),0)** whose spatial components agree with $\mathbf{e}(-\hat{z},1)^*$.

The cases considered are **RR --> RR**, **RR --> LL**, **RR --> RL**, **RR --> LR**, **RL --> RL**, **RL --> LR**, and **RL --> RR** (this last case has amplitude equal to zero).

Since we want to compare our results with Schwinger's results, some involved manipulations are necessary in each case to arrive at a comparable form.

```
(%i78) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("pair2.mac");
read and interpret file: #pc:/work5/pair2.mac
(%i3) " ======================================================"
(%i4) "  file pair2.mac "
(%i5) "  Maxima by Example, Ch. 12 "
(%i6) "  Dirac Algebra and Quantum Electrodynamics "
(%i7) "  Edwin L Woollett, woollett@charter.net "
(%i8) "  http://www.csulb.edu/~woollett  "
(%i9) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-04

(%i10) " ---------------------------------------------"
(%i11) "   POLARIZED PAIR ANNIHILATION in CM Frame    "
(%i12) " ---------------------------------------------"
(%i13) "  e(-,p1) + e(+,p2) --> gamma(k1) + gamma(k2) "
(%i14) " ---------------------------------------------"
(%i15) "   AMPLITUDES USING EXPLICIT DIRAC SPINORS    "
(%i16) " ---------------------------------------------"
(%i17) " Define the needed Dirac spinors and barred spinors. "
(%i18) " For example, RR --> RR amplitude without the factor e^2, and"
(%i19) " either t-m^2 or u-m^2 in the denominator . The photon 3-momenta"
(%i20) " magnitudes are equal to M/2, where M = 2*E = CM frame energy."
(%i21) " M >= 2*m, and 'low energy' corresponds to M at its lower limit,"
(%i22) " for which v is close to value 0, and 'high energy' corresponds"
(%i23) " to M >> m, for which v approaches its maximum value 1 "
(%i24) " The lepton 3-momentum magnitudes are each equal to E*v = M*v/2 "
(%i25) " where v is the dimensionless velocity with "
(%i26) " v^2 = p^2/E^2 = (E^2-m^2)/E^2 = 1- 4*m^2/M^2 "
(%i27) "  All particle energies are equal to M/2."
(%i28) " ------------------------------------------------"
```

```
(%i29) " CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,+1)  gamma(k2,+1) "
(%i30) " ------------------------------------------------------"
(%i31) " Both the electron and positron have helicity +1. "
(%i32) " Each of the final gammas have helicity +1. "
(%i33) "For a given set of helicity quantum numbers,  the amplitude is the"
(%i34) " sum of two terms:  Mtn/(t-m^2) + Mun/(u-m^2) , where  "
(%i35) " Mtd = t-m^2 = -2*D(k1,p1), Mud = u-m^2 = -2*D(k2,p1). "
(%i36) " ------------------------------------------------------"
(%i37) " To simplify the photon helicity 3-vectors, let k1_3vec be "
(%i38) " along the positive z axis, and k2_3vec be along the "
(%i39) " minus z axis, p1_3vec in the z-x plane at an angle of "
(%i40) " th radians to k1_3vec."
(%i41) (up1:UU(M/2,M*v/2,th,0,1),vp2b:sbar(VV(M/2,M*v/2,%pi-th,%pi,1)))
(%i42) " ------------------------------------------------------"
(%i43) " We need to define the components of the four vectors "
(%i44) " p1, k1, k2, e1_cc, and e2_cc relative to the CM frame"
(%i45) " For the photon polarization 4-vectors, we use the phase"
(%i46) " conventions of Franz Gross, Rel. Q. Mech. and Fld. Theory"
(%i47) " pages 54, 218, and 230. e1_cc means complex conjugate of e1. "
(%i48) " Without the 'cc', e1 means here e (z_hat,+1) "
(%i49) " and e2 means here e (-z_hat,+1) "
(%i50) comp_def(p1(M/2,M*v*sin(th)/2,0,M*v*cos(th)/2),k1(M/2,0,0,M/2),
                k2(M/2,0,0,(-M)/2),e1_cc(0,(-1)/sqrt(2),%i/sqrt(2),0),
                e2_cc(0,1/sqrt(2),%i/sqrt(2),0))
(%i51) " denominators "
(%i52) Mtd:-2*pullfac(noncov(D(k1,p1)),M^2/4)
(%o52) -(1-cos(th)*v)*M^2/2
(%i53) Mud:-2*pullfac(noncov(D(k2,p1)),M^2/4)
(%o53) -(cos(th)*v+1)*M^2/2
(%i54) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o54) matrix([2*m,0,-2*(M/2-cos(th)*v*M/2),0],[0,0,0,0],
              [-2*(cos(th)*v*M/2-M/2),0,2*m,0],[0,0,0,0])
(%i55) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o55) matrix([0,0,0,0],[0,2*m,0,-2*(cos(th)*v*M/2+M/2)],[0,0,0,0],
              [0,-2*(-cos(th)*v*M/2-M/2),0,2*m])
(%i56) " M1 = Mtn/Mtd "
(%i57) Mtn:vp2b . g1 . up1
(%o57) sin((%pi-th)/2)*sqrt((v+1)*M/2)
                      *(2*m*cos(th/2)*sqrt((v+1)*M/2)
                       -2*cos(th/2)*sqrt(-(v-1)*M/2)*(cos(th)*v*M/2-M/2))
       -sin((%pi-th)/2)*sqrt(-(v-1)*M/2)
                        *(2*m*cos(th/2)*sqrt(-(v-1)*M/2)
                         -2*cos(th/2)*sqrt((v+1)*M/2)*(M/2-cos(th)*v*M/2))
(%i58) expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o58) -cos(th/2)^2*cos(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
       +cos(th/2)^2*sqrt(1-v)*sqrt(v+1)*M^2+2*m*cos(th/2)^2*v*M
(%i59) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o59) -2*m*cos(th/2)^2*cos(th)*v*M+2*m*cos(th/2)^2*v*M+2*m*cos(th/2)^2*M
(%i60) pullfac(%,2*m*M*cos(th/2)^2)
(%o60) 2*m*cos(th/2)^2*(-cos(th)*v+v+1)*M
(%i61) expand(ratsubst((cos(th)+1)/2,cos(th/2)^2,%))
(%o61) -m*cos(th)^2*v*M+m*v*M+m*cos(th)*M+m*M
(%i62) Mtn:pullfac(%,m*M)
(%o62) m*(-cos(th)^2*v+v+cos(th)+1)*M
(%i63) M1:Mtn/Mtd
(%o63) -2*m*(-cos(th)^2*v+v+cos(th)+1)/((1-cos(th)*v)*M)
(%i64) " M2 = Mun/Mud "
```

```
(%i65) Mun:vp2b . g2 . up1
(%o65) cos((%pi-th)/2)*sqrt((v+1)*M/2)
                      *(2*m*sin(th/2)*sqrt((v+1)*M/2)
                       -2*sin(th/2)*sqrt(-(v-1)*M/2)*(-cos(th)*v*M/2-M/2))
       -cos((%pi-th)/2)*sqrt(-(v-1)*M/2)
                        *(2*m*sin(th/2)*sqrt(-(v-1)*M/2)
                         -2*sin(th/2)*sqrt((v+1)*M/2)*(cos(th)*v*M/2+M/2))
(%i66) expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o66) sin(th/2)^2*cos(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
       +sin(th/2)^2*sqrt(1-v)*sqrt(v+1)*M^2+2*m*sin(th/2)^2*v*M
(%i67) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o67) 2*m*sin(th/2)^2*cos(th)*v*M+2*m*sin(th/2)^2*v*M+2*m*sin(th/2)^2*M
(%i68) pullfac(%,2*m*M*sin(th/2)^2)
(%o68) 2*m*sin(th/2)^2*(cos(th)*v+v+1)*M
(%i69) Mun:expand(ratsubst((1-cos(th))/2,sin(th/2)^2,%))
(%o69) -m*cos(th)^2*v*M+m*v*M-m*cos(th)*M+m*M
(%i70) Mun:pullfac(%,m*M)
(%o70) m*(-cos(th)^2*v+v-cos(th)+1)*M
(%i71) M2:Mun/Mud
(%o71) -2*m*(-cos(th)^2*v+v-cos(th)+1)/((cos(th)*v+1)*M)
(%i72) " Add amplitudes and simplify trig functions "
(%i73) Mfi:trigsimp(M2+M1)
(%o73) (4*m*v+4*m)/((cos(th)^2*v^2-1)*M)
(%i74) " factor numerator and pull out minus sign in denom "
(%i75) Mfi_n:factor(num(Mfi))
(%o75) 4*m*(v+1)
(%i76) Mfi_d:pullfac(expand(denom(Mfi)),-M)
(%o76) -(1-cos(th)^2*v^2)*M
(%i77) Mfi:Mfi_n/Mfi_d
(%o77) -4*m*(v+1)/((1-cos(th)^2*v^2)*M)
(%i78) (display2d:true,
        disp("CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,+1)  gamma(k2,+1) "),
        display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,+1)  gamma(k2,+1)

                                    4 m (v + 1)
                          Mfi = - --------------------
                                          2      2
                                  (1 - cos (th) v ) M

(%i79) " which agrees with Schwinger, PSF I, (3-13-98)"
(%i80) "-----------------------------------------------"
(%i81) "CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,-1)  gamma(k2,-1) "
(%i82) " -----------------------------------------------"
(%i83) comp_def(e1_cc(0,1/sqrt(2),%i/sqrt(2),0),
                e2_cc(0,(-1)/sqrt(2),%i/sqrt(2),0))
(%i84) listarray(e1_cc)
(%o84) [0,1/sqrt(2),%i/sqrt(2),0]
(%i85) listarray(e2_cc)
(%o85) [0,-1/sqrt(2),%i/sqrt(2),0]
(%i86) " the spinors and denominators are the same "
(%i87) " recompute the g1 and g2 matrices "
(%i88) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o88) matrix([0,0,0,0],[0,2*m,0,-2*(cos(th)*v*M/2-M/2)],[0,0,0,0],
              [0,-2*(M/2-cos(th)*v*M/2),0,2*m])
(%i89) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o89) matrix([2*m,0,-2*(-cos(th)*v*M/2-M/2),0],[0,0,0,0],
              [-2*(cos(th)*v*M/2+M/2),0,2*m,0],[0,0,0,0])
(%i90) " ------------------------------------"
(%i91) " M1 = Mtn/Mtd "
```

```
(%i92) Mtn:vp2b . g1 . up1
(%o92) cos((%pi-th)/2)*sqrt((v+1)*M/2)
                      *(2*m*sin(th/2)*sqrt((v+1)*M/2)
                       -2*sin(th/2)*sqrt(-(v-1)*M/2)*(M/2-cos(th)*v*M/2))
       -cos((%pi-th)/2)*sqrt(-(v-1)*M/2)
                        *(2*m*sin(th/2)*sqrt(-(v-1)*M/2)
                         -2*sin(th/2)*sqrt((v+1)*M/2)*(cos(th)*v*M/2-M/2))
(%i93) expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o93) sin(th/2)^2*cos(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
       -sin(th/2)^2*sqrt(1-v)*sqrt(v+1)*M^2+2*m*sin(th/2)^2*v*M
(%i94) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o94) 2*m*sin(th/2)^2*cos(th)*v*M+2*m*sin(th/2)^2*v*M-2*m*sin(th/2)^2*M
(%i95) pullfac(%,2*m*M*sin(th/2)^2)
(%o95) 2*m*sin(th/2)^2*(cos(th)*v+v-1)*M
(%i96) expand(ratsubst((1-cos(th))/2,sin(th/2)^2,%))
(%o96) -m*cos(th)^2*v*M+m*v*M+m*cos(th)*M-m*M
(%i97) Mtn:pullfac(%,m*M)
(%o97) m*(-cos(th)^2*v+v+cos(th)-1)*M
(%i98) M1:Mtn/Mtd
(%o98) -2*m*(-cos(th)^2*v+v+cos(th)-1)/((1-cos(th)*v)*M)
(%i99) "------------------------------"
(%i100) " M2 = Mun/Mud "
(%i101) Mun:vp2b . g2 . up1
(%o101) sin((%pi-th)/2)*sqrt((v+1)*M/2)
                       *(2*m*cos(th/2)*sqrt((v+1)*M/2)
                        -2*cos(th/2)*sqrt(-(v-1)*M/2)*(cos(th)*v*M/2+M/2))
        -sin((%pi-th)/2)*sqrt(-(v-1)*M/2)
                         *(2*m*cos(th/2)*sqrt(-(v-1)*M/2)
                          -2*cos(th/2)*sqrt((v+1)*M/2)*(-cos(th)*v*M/2-M/2))
(%i102) expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o102) -cos(th/2)^2*cos(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
        -cos(th/2)^2*sqrt(1-v)*sqrt(v+1)*M^2+2*m*cos(th/2)^2*v*M
(%i103) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o103) -2*m*cos(th/2)^2*cos(th)*v*M+2*m*cos(th/2)^2*v*M-2*m*cos(th/2)^2*M
(%i104) pullfac(%,2*m*M*cos(th/2)^2)
(%o104) 2*m*cos(th/2)^2*(-cos(th)*v+v-1)*M
(%i105) expand(ratsubst((cos(th)+1)/2,cos(th/2)^2,%))
(%o105) -m*cos(th)^2*v*M+m*v*M-m*cos(th)*M-m*M
(%i106) Mun:pullfac(%,m*M)
(%o106) m*(-cos(th)^2*v+v-cos(th)-1)*M
(%i107) M2:Mun/Mud
(%o107) -2*m*(-cos(th)^2*v+v-cos(th)-1)/((cos(th)*v+1)*M)
(%i108) " Add amplitudes and simplify trig functions "
(%i109) Mfi:trigsimp(M2+M1)
(%o109) (4*m*v-4*m)/((cos(th)^2*v^2-1)*M)
(%i110) " factor numerator and pull out minus sign in denom "
(%i111) Mfi_n:pullfac(num(Mfi),-4*m)
(%o111) -4*m*(1-v)
(%i112) Mfi_d:pullfac(expand(denom(Mfi)),-M)
(%o112) -(1-cos(th)^2*v^2)*M
(%i113) Mfi:Mfi_n/Mfi_d
(%o113) 4*m*(1-v)/((1-cos(th)^2*v^2)*M)
(%i114) (display2d:true,
         disp("CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,-1)  gamma(k2,-1) "),
         display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,-1)  gamma(k2,-1)

                                    4 m (1 - v)
                           Mfi = -------------------
                                        2      2
                                 (1 - cos (th) v ) M
```

```
(%i115) " which agrees with Schwinger, PSF I, (3-13-98)"
(%i116) "-----------------------------------------------"
(%i117) "CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,1)  gamma(k2,-1) "
(%i118) " -----------------------------------------------"
(%i119) comp_def(e1_cc(0,(-1)/sqrt(2),%i/sqrt(2),0))
(%i120) listarray(e1_cc)
(%o120) [0,-1/sqrt(2),%i/sqrt(2),0]
(%i121) " the spinors and denominators are the same "
(%i122) " recompute the g1 and g2 matrices "
(%i123) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o123) matrix([0,0,0,0],[0,0,sin(th)*v*M,0],[0,0,0,0],[-sin(th)*v*M,0,0,0])
(%i124) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o124) matrix([0,0,0,0],[0,0,sin(th)*v*M,0],[0,0,0,0],[-sin(th)*v*M,0,0,0])
(%i125) " ------------------------------------"
(%i126) " M1 = Mtn/Mtd "
(%i127) Mtn:vp2b . g1 . up1
(%o127) -2*cos((%pi-th)/2)*cos(th/2)*sin(th)*v*M*sqrt(-(v-1)*M/2)
          *sqrt((v+1)*M/2)
(%i128) expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o128) -cos(th/2)*sin(th/2)*sin(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
(%i129) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o129) -2*m*cos(th/2)*sin(th/2)*sin(th)*v*M
(%i130) Mtn:ratsubst(sin(th)/2,cos(th/2)*sin(th/2),%)
(%o130) -m*sin(th)^2*v*M
(%i131) M1:Mtn/Mtd
(%o131) 2*m*sin(th)^2*v/((1-cos(th)*v)*M)
(%i132) "------------------------------"
(%i133) " M2 = Mun/Mud "
(%i134) Mun:vp2b . g2 . up1
(%o134) -2*cos((%pi-th)/2)*cos(th/2)*sin(th)*v*M*sqrt(-(v-1)*M/2)
          *sqrt((v+1)*M/2)
(%i135) expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o135) -cos(th/2)*sin(th/2)*sin(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
(%i136) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o136) -2*m*cos(th/2)*sin(th/2)*sin(th)*v*M
(%i137) Mun:ratsubst(sin(th)/2,cos(th/2)*sin(th/2),%)
(%o137) -m*sin(th)^2*v*M
(%i138) M2:Mun/Mud
(%o138) 2*m*sin(th)^2*v/((cos(th)*v+1)*M)
(%i139) " ---------------------------------------"
(%i140) " Add amplitudes and simplify trig functions "
(%i141) Mfi:trigsimp(M2+M1)
(%o141) -4*m*sin(th)^2*v/((cos(th)^2*v^2-1)*M)
(%i142) Mfi:num(Mfi)/pullfac(expand(denom(Mfi)),-M)
(%o142) 4*m*sin(th)^2*v/((1-cos(th)^2*v^2)*M)
(%i143) (display2d:true,
         disp("CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,+1)  gamma(k2,-1) "),
         display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,+1)  gamma(k2,-1)

                                      2
                               4 m sin (th) v
                        Mfi = -------------------
                                     2      2
                              (1 - cos (th) v ) M

(%i144) " -----------------------------------------------"
(%i145) "-----------------------------------------------"
(%i146) "CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,-1)  gamma(k2,+1) "
(%i147) " -----------------------------------------------"
(%i148) comp_def(e1_cc(0,1/sqrt(2),%i/sqrt(2),0),
                 e2_cc(0,1/sqrt(2),%i/sqrt(2),0))
```

```
(%i149) listarray(e1_cc)
(%o149) [0,1/sqrt(2),%i/sqrt(2),0]
(%i150) listarray(e2_cc)
(%o150) [0,1/sqrt(2),%i/sqrt(2),0]
(%i151) " the spinors and denominators are the same "
(%i152) " recompute the g1 and g2 matrices "
(%i153) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o153) matrix([0,0,0,sin(th)*v*M],[0,0,0,0],[0,-sin(th)*v*M,0,0],[0,0,0,0])
(%i154) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o154) matrix([0,0,0,sin(th)*v*M],[0,0,0,0],[0,-sin(th)*v*M,0,0],[0,0,0,0])
(%i155) " ------------------------------------"
(%i156) " ------------------------------------"
(%i157) " M1 = Mtn/Mtd "
(%i158) Mtn:vp2b . g1 . up1
(%o158) -2*sin((%pi-th)/2)*sin(th/2)*sin(th)*v*M*sqrt(-(v-1)*M/2)
          *sqrt((v+1)*M/2)
(%i159) expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o159) -cos(th/2)*sin(th/2)*sin(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
(%i160) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o160) -2*m*cos(th/2)*sin(th/2)*sin(th)*v*M
(%i161) Mtn:ratsubst(sin(th)/2,cos(th/2)*sin(th/2),%)
(%o161) -m*sin(th)^2*v*M
(%i162) M1:Mtn/Mtd
(%o162) 2*m*sin(th)^2*v/((1-cos(th)*v)*M)
(%i163) "------------------------------"
(%i164) " M2 = Mun/Mud "
(%i165) Mun:vp2b . g2 . up1
(%o165) -2*sin((%pi-th)/2)*sin(th/2)*sin(th)*v*M*sqrt(-(v-1)*M/2)
          *sqrt((v+1)*M/2)
(%i166) expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o166) -cos(th/2)*sin(th/2)*sin(th)*sqrt(1-v)*v*sqrt(v+1)*M^2
(%i167) expand(ratsubst(2*m/M,sqrt(1-v^2),rootscontract(%)))
(%o167) -2*m*cos(th/2)*sin(th/2)*sin(th)*v*M
(%i168) Mun:ratsubst(sin(th)/2,cos(th/2)*sin(th/2),%)
(%o168) -m*sin(th)^2*v*M
(%i169) M2:Mun/Mud
(%o169) 2*m*sin(th)^2*v/((cos(th)*v+1)*M)
(%i170) " --------------------------------"
(%i171) Mfi:trigsimp(M2+M1)
(%o171) -4*m*sin(th)^2*v/((cos(th)^2*v^2-1)*M)
(%i172) Mfi:num(Mfi)/pullfac(expand(denom(Mfi)),-M)
(%o172) 4*m*sin(th)^2*v/((1-cos(th)^2*v^2)*M)
(%i173) (display2d:true,
         disp("CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,-1)  gamma(k2,+1) "),
         display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,+1) --> gamma(k1,-1)  gamma(k2,+1)

                                        2
                                 4 m sin (th) v
                          Mfi = -------------------
                                        2      2
                                (1 - cos (th) v ) M

(%i174) " ------------------------------------------------"
(%i175) " Schwinger's amplitude for these last two cases differs by a minus sign "
(%i176) " ------------------------------------------------ "
(%i177) " CASES LEPTONS HAVE OPPOSITE HELICITIES: see Schwinger (3-13-97). "
(%i178) "------------------------------------------"
(%i179) "CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,+1)  gamma(k2,-1) "
(%i180) " ------------------------------------------------"
```

```
(%i181) " recompute vp2b "
(%i182) vp2b:sbar(VV(M/2,M*v/2,%pi-th,%pi,-1))
(%i183) " denominators Mtd, Mud remain the same "
(%i184) " recompute g1 and g2 "
(%i185) comp_def(e1_cc(0,(-1)/sqrt(2),%i/sqrt(2),0),
                 e2_cc(0,(-1)/sqrt(2),%i/sqrt(2),0))
(%i186) listarray(e1_cc)
(%o186) [0,-1/sqrt(2),%i/sqrt(2),0]
(%i187) listarray(e2_cc)
(%o187) [0,-1/sqrt(2),%i/sqrt(2),0]
(%i188) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o188) matrix([0,0,0,0],[0,0,sin(th)*v*M,0],[0,0,0,0],[-sin(th)*v*M,0,0,0])
(%i189) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o189) matrix([0,0,0,0],[0,0,sin(th)*v*M,0],[0,0,0,0],[-sin(th)*v*M,0,0,0])
(%i190) " ------------------------------------"
(%i191) " ------------------------------------"
(%i192) " M1 = Mtn/Mtd "
(%i193) Mtn:vp2b . g1 . up1
(%o193) sin((%pi-th)/2)*cos(th/2)*sin(th)*v*(v+1)*M^2/2
         -sin((%pi-th)/2)*cos(th/2)*sin(th)*(v-1)*v*M^2/2
(%i194) Mtn:expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o194) cos(th/2)^2*sin(th)*v*M^2
(%i195) M1:Mtn/Mtd
(%o195) -2*cos(th/2)^2*sin(th)*v/(1-cos(th)*v)
(%i196) "-----------------------------"
(%i197) " M2 = Mun/Mud "
(%i198) Mun:vp2b . g2 . up1
(%o198) sin((%pi-th)/2)*cos(th/2)*sin(th)*v*(v+1)*M^2/2
         -sin((%pi-th)/2)*cos(th/2)*sin(th)*(v-1)*v*M^2/2
(%i199) Mun:expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o199) cos(th/2)^2*sin(th)*v*M^2
(%i200) M2:Mun/Mud
(%o200) -2*cos(th/2)^2*sin(th)*v/(cos(th)*v+1)
(%i201) " ----------------------------"
(%i202) trigsimp(M2+M1)
(%o202) 4*cos(th/2)^2*sin(th)*v/(cos(th)^2*v^2-1)
(%i203) Mfi:(-num(%))/(-denom(%))
(%o203) -4*cos(th/2)^2*sin(th)*v/(1-cos(th)^2*v^2)
(%i204) (display2d:true,
         disp("CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,+1)  gamma(k2,-1) "),
         display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,+1)  gamma(k2,-1)

                                    2 th
                               4 cos (--) sin(th) v
                                    2
                        Mfi = - --------------------
                                        2      2
                                  1 - cos (th) v

(%i205) " ------------------------------------------------"
(%i206) " This differs by a sign compared to Schwinger. "
(%i207) " ------------------------------------------------"
(%i208) "CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,-1)  gamma(k2,+1) "
(%i209) " ------------------------------------------------"
(%i210) " spinors and denominators are the same , recompute g1 and g2"
(%i211) comp_def(e1_cc(0,1/sqrt(2),%i/sqrt(2),0),
                 e2_cc(0,1/sqrt(2),%i/sqrt(2),0))
(%i212) listarray(e1_cc)
(%o212) [0,1/sqrt(2),%i/sqrt(2),0]
(%i213) listarray(e2_cc)
(%o213) [0,1/sqrt(2),%i/sqrt(2),0]
```

```
(%i214) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o214) matrix([0,0,0,sin(th)*v*M],[0,0,0,0],[0,-sin(th)*v*M,0,0],[0,0,0,0])
(%i215) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o215) matrix([0,0,0,sin(th)*v*M],[0,0,0,0],[0,-sin(th)*v*M,0,0],[0,0,0,0])
(%i216) " M1 = Mtn/Mtd "
(%i217) Mtn:vp2b . g1 . up1
(%o217) cos((%pi-th)/2)*sin(th/2)*sin(th)*(v-1)*v*M^2/2
         -cos((%pi-th)/2)*sin(th/2)*sin(th)*v*(v+1)*M^2/2
(%i218) Mtn:expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o218) -sin(th/2)^2*sin(th)*v*M^2
(%i219) M1:Mtn/Mtd
(%o219) 2*sin(th/2)^2*sin(th)*v/(1-cos(th)*v)
(%i220) Mun:vp2b . g2 . up1
(%o220) cos((%pi-th)/2)*sin(th/2)*sin(th)*(v-1)*v*M^2/2
         -cos((%pi-th)/2)*sin(th/2)*sin(th)*v*(v+1)*M^2/2
(%i221) Mun:expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o221) -sin(th/2)^2*sin(th)*v*M^2
(%i222) M2:Mun/Mud
(%o222) 2*sin(th/2)^2*sin(th)*v/(cos(th)*v+1)
(%i223) " ----------------------------"
(%i224) trigsimp(M2+M1)
(%o224) -4*sin(th/2)^2*sin(th)*v/(cos(th)^2*v^2-1)
(%i225) Mfi:(-num(%))/(-denom(%))
(%o225) 4*sin(th/2)^2*sin(th)*v/(1-cos(th)^2*v^2)
(%i226) (display2d:true,
         disp("CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,-1)  gamma(k2,+1) "),
         display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,-1)  gamma(k2,+1)

                                    2 th
                                4 sin (--) sin(th) v
                                     2
                           Mfi = --------------------
                                          2      2
                                  1 - cos (th) v

(%i227) " ------------------------------------------------"
(%i228) " This differs by a sign compared to Schwinger. "
(%i229) " ------------------------------------------------"
(%i230) "CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,+1)  gamma(k2,+1) "
(%i231) " ------------------------------------------------"
(%i232) comp_def(e1_cc(0,(-1)/sqrt(2),%i/sqrt(2),0),
                 e2_cc(0,1/sqrt(2),%i/sqrt(2),0))
(%i233) listarray(e1_cc)
(%o233) [0,-1/sqrt(2),%i/sqrt(2),0]
(%i234) listarray(e2_cc)
(%o234) [0,1/sqrt(2),%i/sqrt(2),0]
(%i235) g1:sL(e2_cc) . (m*I4-sL(k1)+sL(p1)) . sL(e1_cc)
(%o235) matrix([2*m,0,-2*(M/2-cos(th)*v*M/2),0],[0,0,0,0],
                [-2*(cos(th)*v*M/2-M/2),0,2*m,0],[0,0,0,0])
(%i236) g2:sL(e1_cc) . (m*I4-sL(k2)+sL(p1)) . sL(e2_cc)
(%o236) matrix([0,0,0,0],[0,2*m,0,-2*(cos(th)*v*M/2+M/2)],[0,0,0,0],
                [0,-2*(-cos(th)*v*M/2-M/2),0,2*m])
(%i237) " M1 = Mtn/Mtd "
(%i238) Mtn:vp2b . g1 . up1
(%o238) cos((%pi-th)/2)*sqrt(-(v-1)*M/2)
                       *(2*m*cos(th/2)*sqrt((v+1)*M/2)
                        -2*cos(th/2)*sqrt(-(v-1)*M/2)*(cos(th)*v*M/2-M/2))
         -cos((%pi-th)/2)*sqrt((v+1)*M/2)
                          *(2*m*cos(th/2)*sqrt(-(v-1)*M/2)
                           -2*cos(th/2)*sqrt((v+1)*M/2)*(M/2-cos(th)*v*M/2))
```

```
(%i239) expand(ratsubst(sin(th/2),cos((%pi-th)/2),%))
(%o239) cos(th/2)*sin(th/2)*M^2-cos(th/2)*sin(th/2)*cos(th)*v*M^2
(%i240) expand(ratsubst(sin(th)/2,cos(th/2)*sin(th/2),%))
(%o240) sin(th)*M^2/2-cos(th)*sin(th)*v*M^2/2
(%i241) Mtn:pullfac(%,sin(th)*M^2/2)
(%o241) sin(th)*(1-cos(th)*v)*M^2/2
(%i242) M1:Mtn/Mtd
(%o242) -sin(th)
(%i243) Mun:vp2b . g2 . up1
(%o243) sin((%pi-th)/2)*sqrt((v+1)*M/2)
                       *(2*m*sin(th/2)*sqrt(-(v-1)*M/2)
                        -2*sin(th/2)*sqrt((v+1)*M/2)*(cos(th)*v*M/2+M/2))
         -sin((%pi-th)/2)*sqrt(-(v-1)*M/2)
                          *(2*m*sin(th/2)*sqrt((v+1)*M/2)
                           -2*sin(th/2)*sqrt(-(v-1)*M/2)*(-cos(th)*v*M/2-M/2))
(%i244) expand(ratsubst(cos(th/2),sin((%pi-th)/2),%))
(%o244) -cos(th/2)*sin(th/2)*cos(th)*v*M^2-cos(th/2)*sin(th/2)*M^2
(%i245) expand(ratsubst(sin(th)/2,cos(th/2)*sin(th/2),%))
(%o245) -cos(th)*sin(th)*v*M^2/2-sin(th)*M^2/2
(%i246) Mun:factor(%)
(%o246) -sin(th)*(cos(th)*v+1)*M^2/2
(%i247) M2:Mun/Mud
(%o247) sin(th)
(%i248) Mfi:M2+M1
(%o248) 0
(%i249) (display2d:true,
         disp("CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,+1)  gamma(k2,+1) "),
         display(Mfi),display2d:false)
         CASE: e(-,p1,+1)  e(+,p2,-1) --> gamma(k1,+1)  gamma(k2,+1)

                                    Mfi = 0

(%i250) " The amplitude is zero for this case. "
(%o250) "pair2.mac"
```

12.16 moller3.mac: Squared Polarized Amplitudes Using Symbolic or Explicit Matrix Trace Methods

We provide here examples of using trace methods for the square of a polarized amplitude in the case of arbitrary energy (finite mass) Moller scattering. This detailed verification using both explicit matrix traces and the symbolic traces takes several minutes to carry out for each case, with the symbolic **nc_tr** method (equivalent to **noncov (tr(..))**) requiring about twice the computation time as the explicit matrix method **m_tr**.

The new feature is the **tr** or **m_tr** argument **S(sv, Sp)**, in which either **sv = 1** or **sv = -1**, and **Sp** is a symbol for the particle spin 4-vector associated with the particle's 4-momentum.

In the explicit matrix method, such an argument inserts a matrix factor **P(sv,Sp)**, which turns into the matrix **(I4 + sv*Gam[5].sL(Sp))/2**. This is the matrix version of the finite mass spin projection operator, and requires a frame dependent definition (using **comp_def**) of the components of the spin 4-vector corresponding to a particle's 4-momentum. (See our section on high energy physics notation.)

When used with the symbolic **tr** method, the result is a factor $\frac{1}{2}(1 + \sigma\,\gamma^5\,\not{s})$, with $\sigma = \pm 1$ being the value of **sv** and $\not{s}$ being $(Sp)_\mu\,\gamma^\mu$.

Here is the batch file **moller3.mac**:

```
/* file moller3.mac
   m_tr and nc_tr comparisons of
   the square of polarized amplitudes
   compared with dirac spinor results
   found in moller2.mac
   */

  " moller3.mac "$
    "*****************************************"$
   print ("      ver: ",_binfo%,"  date: ",mydate )$
  " Maxima by Example, Ch. 12 "$
  " Dirac Algebra and Quantum Electrodynamics "$
  " Edwin L. Woollett "$
  " http://www.csulb.edu/~woollett  "$
  " woollett@charter.net "$
  "SQUARED POLARIZED AMPLITUDES VIA SYMBOLIC AND MATRIX TRACE METHODS"$
  "  FOR ARBITRARY ENERGY MOLLER SCATTERING  "$
  "  e(-,p1,sv1) + e(-,p2,sv2) --> e(-,p3,sv3) + e(-,p4,sv4) "$
  " ------------------------------------------------------"$

  invar (D(p1,p1) = m^2,
         D(p2,p2) = m^2,
         D(p3,p3) = m^2,
         D(p4,p4) = m^2,
         D(p1,Sp1) = 0,
         D(Sp1,Sp1) = -1,
         D(p2,Sp2) = 0,
         D(Sp2,Sp2) = -1,
         D(p3,Sp3) = 0,
         D(Sp3,Sp3) = -1,
         D(p4,Sp4) = 0,
         D(Sp4,Sp4) = -1 )$


   comp_def ( p1( E,0,0,p),
              Sp1 (p/m,0,0,E/m),
              p2( E,0,0,-p),
              Sp2 (p/m,0,0,-E/m),
              p3 (E,p*sin(th),0,p*cos(th)),
              Sp3 (p/m,E*sin(th)/m,0,E*cos(th)/m),
              p4 (E,-p*sin(th),0,-p*cos(th)),
              Sp4 (p/m,-E*sin(th)/m,0,-E*cos(th)/m))$



  p_Em (expr) := expand (ratsubst(E^2-m^2,p^2,expr))$

  E_pm (expr) := expand (ratsubst (p^2 + m^2,E^2,expr))$
```

```
  s_th : VP (p1+p2,p1+p2);

  t_th : VP (p1-p3,p1-p3);

  u_th : VP (p1-p4,p1-p4);


  t_th2 : to_ao2 (t_th,th);

  u_th2 : to_ao2 (u_th,th);

  t_thE : p_Em(t_th);

  u_thE : p_Em(u_th);

   load("MSQcomp.mac")$
```

and here is the automated comparison code which Moller3.mac loads:

```
/* file MSQcomp.mac */

/* comparison  code for moller3.mac */
  disp (" MSQcomp() to compare matrix trace, symbolic trace
  methods for square of polarized amplitudes
  with square of spinor amplitudes. This is an
  automated comparison, assuming A_spinor is
  globally defined and also global sv1,sv2,sv3,sv4.
  To see progress details, set details to true.")$
     details:false$
 /*********************************************************/
MSQcomp () :=

   block(

   if not details then (disp (" this comparison could take 3 plus minutes ")),

       A_spinor_th : fr_ao2 (A_spinor,th),

        A_spSQ : E_pm (A_spinor_th^2),

       if details then display (A_spSQ),


   M1n_m : trigsimp (E_pm (mcon ( m_tr (S(sv3,Sp3),p3+m,mu,S(sv1,Sp1),p1+m,nu)*
                m_tr (S(sv4,Sp4),p4+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu))),

     if details then (disp ("M1n_m"), display (M1n_m)),
```

```
M1n_s : trigsimp (E_pm (mcon ( nc_tr (S(sv3,Sp3),p3+m,mu,S(sv1,Sp1),p1+m,nu)*
               nc_tr (S(sv4,Sp4),p4+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu))),

   if details then (disp ("M1n_s"),display (M1n_s)),


  M1n_diff : M1n_s - M1n_m,

  if details then print ("  M1n_s - M1n_m = ",M1n_diff),

  if M1n_diff # 0 then (
          disp ("M1n symbolic  method # matrix method disagree "),
          return (M1n_diff)),


 M2n_m : trigsimp (E_pm (mcon ( m_tr (S(sv4,Sp4),p4+m,mu,S(sv1,Sp1),p1+m,nu)*
              m_tr (S(sv3,Sp3),p3+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu))),

  if details then disp ("M2n_m"),


M2n_s : trigsimp (E_pm (mcon ( nc_tr (S(sv4,Sp4),p4+m,mu,S(sv1,Sp1),p1+m,nu)*
              nc_tr (S(sv3,Sp3),p3+m,mu,S(sv2,Sp2),p2+m,nu), mu,nu))),

  if details then disp ("M2n_s"),

  M2n_diff : M2n_s - M2n_m,

  if details then print ("  M2n_s - M2n_m = ",M2n_diff),

  if M2n_diff # 0 then (
          disp ("M2n symbolic  method # matrix method disagree "),
          return (M2n_diff)),

  M12n_m : trigsimp (E_pm (mcon ( m_tr (S(sv3,Sp3),p3+m,mu,S(sv1,Sp1),p1+m,
                       nu,S(sv4,Sp4),p4+m,mu,S(sv2,Sp2),p2+m,nu),mu,nu))),

  if details then disp ("M12n_m"),

  M12n_s : trigsimp (E_pm (nc_tr (S(sv3,Sp3),p3+m,mu,S(sv1,Sp1),p1+m,
                       nu,S(sv4,Sp4),p4+m,mu,S(sv2,Sp2),p2+m,nu))),

  if details then disp ("M12n_s"),


  M12n_diff : M12n_s - M12n_m,

  if details then print ("  M12n_s - M12n_m = ",M12n_diff),
```

```
    if M12n_diff # 0 then (
            disp ("M21n symbolic  method # matrix method disagree "),
            return (M21n_diff)),



    M21n_m : trigsimp (E_pm (mcon ( m_tr (S(sv4,Sp4),p4+m,mu,S(sv1,Sp1),p1+m,
                      nu,S(sv3,Sp3),p3+m,mu,S(sv2,Sp2),p2+m,nu),mu,nu))),

    if details then disp ("M21n_m"),


    M21n_s : trigsimp (E_pm (nc_tr (S(sv4,Sp4),p4+m,mu,S(sv1,Sp1),p1+m,
                         nu,S(sv3,Sp3),p3+m,mu,S(sv2,Sp2),p2+m,nu))),

    if details then disp ("M21n_s"),


    if details then print ("  M21n_s - M21n_m = ",M21n_s - M21n_m),

    M21n_diff : M21n_s - M21n_m,

    if details then print ("  M21n_s - M21n_m = ",M21n_diff),

    if M21n_diff # 0 then (
            disp ("M21n symbolic  method # matrix method disagree "),
            return (M21n_diff)),

    MfiSQ : expand (M1n_m/t_th^2 + M2n_m/u_th^2 - M12n_m/(t_th*u_th)
                 - M21n_m/(t_th*u_th)),

    MfiSQ : trigsimp (MfiSQ),

    MSQ_diff : trigsimp (MfiSQ - A_spSQ),

    if MSQ_diff = 0 then disp (" agreement ")

     else  disp (" no agreement "),

     display ([sv1,sv2,sv3,sv4]))$
```

and here are some cases worked out. After loading **dirac2.mac** and batching **moller3.mac** (which loads also **MSQcomp.mac**), interactive work begins with input **(%i28)** below:

```
(%i35) load(dirac2);
(%o1) "c:/work5/dirac2.mac"
(%i2) batch ("moller3.mac");
read and interpret file: #pc:/work5/moller3.mac
(%i3) " moller3.mac "
(%i4) "****************************************"
```

```
(%i5) print("      ver: ",_binfo%,"  date: ",mydate)
      ver:  Maxima 5.23.2   date:  2011-04-13

(%i6) " Maxima by Example, Ch. 12 "
(%i7) " Dirac Algebra and Quantum Electrodynamics "
(%i8) " Edwin L. Woollett "
(%i9) " http://www.csulb.edu/~woollett  "
(%i10) " woollett@charter.net "
(%i11) " SQUARED POLARIZED AMPLITUDES VIA SYMBOLIC AND MATRIX TRACE METHODS"
(%i12) "  FOR ARBITRARY ENERGY MOLLER SCATTERING  "
(%i13) "  e(-,p1,sv1) + e(-,p2,sv2) --> e(-,p3,sv3) + e(-,p4,sv4) "
(%i14) " ------------------------------------------------------------"
(%i15) invar(D(p1,p1) = m^2,D(p2,p2) = m^2,D(p3,p3) = m^2,D(p4,p4) = m^2,
             D(p1,Sp1) = 0,D(Sp1,Sp1) = -1,D(p2,Sp2) = 0,D(Sp2,Sp2) = -1,
             D(p3,Sp3) = 0,D(Sp3,Sp3) = -1,D(p4,Sp4) = 0,D(Sp4,Sp4) = -1)
(%i16) comp_def(p1(E,0,0,p),Sp1(p/m,0,0,E/m),p2(E,0,0,-p),Sp2(p/m,0,0,(-E)/m),
                p3(E,p*sin(th),0,p*cos(th)),
                Sp3(p/m,E*sin(th)/m,0,E*cos(th)/m),
                p4(E,-p*sin(th),0,-p*cos(th)),
                Sp4(p/m,(-E*sin(th))/m,0,(-E*cos(th))/m))
(%i17) p_Em(expr):=expand(ratsubst(E^2-m^2,p^2,expr))
(%i18) E_pm(expr):=expand(ratsubst(m^2+p^2,E^2,expr))
(%i19) s_th:VP(p2+p1,p2+p1)
(%o19) 4*E^2
(%i20) t_th:VP(p1-p3,p1-p3)
(%o20) 2*p^2*cos(th)-2*p^2
(%i21) u_th:VP(p1-p4,p1-p4)
(%o21) -2*p^2*cos(th)-2*p^2
(%i22) t_th2:to_ao2(t_th,th)
(%o22) -4*p^2*sin(th/2)^2
(%i23) u_th2:to_ao2(u_th,th)
(%o23) 4*p^2*sin(th/2)^2-4*p^2
(%i24) t_thE:p_Em(t_th)
(%o24) 2*cos(th)*E^2-2*E^2-2*m^2*cos(th)+2*m^2
(%i25) u_thE:p_Em(u_th)
(%o25) -2*cos(th)*E^2-2*E^2+2*m^2*cos(th)+2*m^2
(%i26) load("MSQcomp.mac")
" MSQcomp() to compare matrix trace, symbolic trace
  methods for square of polarized amplitudes
  with square of spinor amplitudes. This is an
  automated comparison, assuming A_spinor is
  globally defined and also global sv1,sv2,sv3,sv4.
  To see progress details, set details to true."
(%o27) "moller3.mac"
(%i28) /* case RR --> LL */
         [sv1,sv2,sv3,sv4]:[1,1,-1,-1]$
(%i29) A_spinor : 2*m^2/p^2$
(%i30) MSQcomp();
" each case comparison could take 2 plus minutes "
" agreement "
[sv1,sv2,sv3,sv4] = [1,1,-1,-1]
(%o30) done
(%i31) time(%);
(%o31) [115.91]
(%i32) /* case RR --> RL */
         [sv1,sv2,sv3,sv4]:[1,1,1,-1]$
(%i33) A_spinor : m*sin(th/2)*E/(p^2*cos(th/2))-m*cos(th/2)*E/(p^2*sin(th/2))$
(%i34) MSQcomp();
" each case comparison could take 2 plus minutes "
" agreement "
[sv1,sv2,sv3,sv4] = [1,1,1,-1]
(%o34) done
```

```
(%i35) time(%);
(%o35) [120.71]
(%i36) /* case RR --> RR  */
        [sv1,sv2,sv3,sv4]:[1,1,1,1]$
(%i37) A_spinor : m^2*sin(th/2)^2/(p^2*cos(th/2)^2)+2*sin(th/2)^2/cos(th/2)^2
                                  +m^2*cos(th/2)^2/(p^2*sin(th/2)^2)
                                  +2*cos(th/2)^2/sin(th/2)^2+4$
(%i38) MSQcomp();
" each case comparison could take 2 plus minutes "
" agreement "
[sv1,sv2,sv3,sv4] = [1,1,1,1]
(%o38) done
(%i39) time(%);
(%o39) [122.34]
(%i40) /* case RL --> RL */
        [sv1,sv2,sv3,sv4]:[1,-1,1,-1]$
(%i41) A_spinor : m^2*cos(th/2)^2/(p^2*sin(th/2)^2)+
            2*cos(th/2)^2/sin(th/2)^2-m^2/p^2$
(%i42) MSQcomp();
" each case comparison could take 2 plus minutes "
" agreement "
[sv1,sv2,sv3,sv4] = [1,-1,1,-1]
(%o42) done
(%i43) time(%);
(%o43) [122.5]
(%i44) /* case RL --> LR */
         [sv1,sv2,sv3,sv4]:[1,-1,-1,1]$
(%i45) A_spinor : m^2*sin(th/2)^2/(p^2*cos(th/2)^2)+
             2*sin(th/2)^2/cos(th/2)^2-m^2/p^2$
(%i46) MSQcomp();
" each case comparison could take 2 plus minutes "
" agreement "
[sv1,sv2,sv3,sv4] = [1,-1,-1,1]
(%o46) done
(%i47) time(%);
(%o47) [123.52]
```

12.17 List of Dirac Package Files and Example Batch Files

The **Dirac package** consists of

- **dirac2.mac**: This file loads in the other files and includes some utility functions and program startup settings.
- **simplifying-new.lisp**: This lisp code file has been modified slightly (by Maxima developer Barton Willis) from the file **simplifying.lisp** which appears in **.../share/contrib**. This code defines a recursive method to progressively simplify expressions. We list the functions which make use of this recursive simplification method in the last section of this chapter.
- **dgcon2.mac**: This file has code for symbolic Lorentz index contraction.
- **dgtrace2.mac**: This file has code for symbolic traces of products of gamma matrices.
- **dgeval2**: This file has code for frame dependent evaluation, such as **noncov**.
- **dgmatrix2.mac**: This file has most of the code used for explicit Dirac spinor and matrix calculations.

The file **dgfunctions2.txt** has an semi-alphabetical listing of most of the Dirac package functions, together with a reference to the file in which the code can be found. There are eleven **example batch files** discussed in this chapter. In alphabetical order these are:

- **bhabha1.mac**: High energy electron-positron scattering.

- **bhabha2.mac**: Arbitrary energy electron-positron scattering.
- **compton0.mac**: Photon plus scalar charged particle scattering.
- **compton1.mac**: Photon plus Lepton scattering.
- **moller0.mac**: Scalar plus scalar charged particle scattering.
- **moller1.mac**: High energy limit of electron plus electron scattering.
- **moller2.mac**: Arbitrary energy electron plus electron scattering.
- **moller3.mac**: Use of explicit matrix trace methods and symbolic trace methods for the squared polarized amplitudes for arbitrary energy electron plus electron scattering.
- **pair1.mac**: Unpolarized two photon annihilation of an electron-positron pair.
- **pair2.mac**: Polarized two photon annihilation of an electron-positron pair.
- **photon1.mac**: Sum identities for photon polarization 3-vectors.

In addition to the above files, there is the file **MSQcomp.mac** which defines a function used by (and loaded by) **moller3.mac**.

12.18 Test Suite Files

There are sixteen test suite files, and once **dirac2.mac** has been loaded, there are convenient abbreviations available to run each of these sixteen test files separately.

- **dgintro2-test.mac**: run using **tin()**,
- **dgtrace2-test.mac**: run using **tt1()**,
- **dgtrace2_test.mac**: run using **tt2()**,
- **dgcon2-test.mac**: run using **tcon()**,
- **dgmcon-test.mac**: run using **tmcon()**,
- **dgeval2-test.mac**: run using **teval()**,
- **bhabha1-test.mac**: run using **tb1()**,
- **bhabha2-test.mac**: run using **tb2()**,
- **compton0-test.mac**: run using **tc0()**,
- **compton1-test.mac**: run using **tc1()**,
- **moller0-test.mac**: run using **tm0()**,
- **moller1-test.mac**: run using **tm1()**,
- **moller2-test.mac**: run using **tm2()**,
- **moller3-test.mac**: run using **tm3()**,
- **pair1-test.mac**: run using **tp1()**,
- **pair2-test.mac**: run using **tp2()**

When you make your own modifications to the Dirac package (perhaps to fit your own research goals), you can assure yourself that your new code has not broken anything significant by running these sixteen test files. Once you have loaded **dirac2.mac**, you can continue to use the above abbreviations to run one test file at a time, and you do not need to manually reload **dirac2.mac**.

To run a test file manually (without using the abbreviations defined in **dirac2.mac**), you would need two lines for each test file. Here is an example:

```
load ("dirac2.mac");
batch ("dgintro2-test.mac", test );
```

12.19 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.