Calculating the Compton scattering cross section using Schoonschip

The Feynman diagrams for $e + \gamma \to e + \gamma$ are

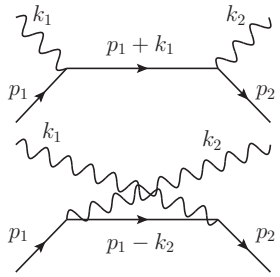

Figure 1: The solid line represents an electron and the wiggly line represents a photon. k_1 is the incoming photon and k_2 is the outgoing photon.

These diagrams can be expressed in terms of a complex amplitude that takes the form

$$\mathcal{M} = e^2 \left[\bar{u}(p_2) \not{\varepsilon}_2 \left(\frac{m - i(\not{p}_1 + \not{k}_1)}{(p_1 + k_1)^2 + m^2} \right) \not{\varepsilon}_1 u(p_1) + \bar{u}(p_2) \not{\varepsilon}_1 \left(\frac{m - i(\not{p}_1 - \not{k}_2)}{(p_1 - k_2)^2 + m^2} \right) \not{\varepsilon}_2 u(p_1) \right] , \qquad (1)$$

where e is the electron charge, $u(p_1)$ and $\bar{u}(p_2)$ are the initial and final electron wave functions and ε_1 and ε_2 are the initial and final photon polarization vectors. The ratios in parenthesis are electron propagators that describe how the electrons move from the initial to the final state. $\not{p}$ is shorthand for $p_\mu \gamma_\mu$, where γ_μ, $\mu = 1, 2, 3, 4$, is a set 4×4 matrices[1] that represent the spin content of the electron . A repeated suffix implies a summation over $1, 2, 3, 4$. The four vector is $p_\mu = (\vec{p}, ip_0)$. For a free particle $p_0 = E(\vec{p}) = \sqrt{\vec{p}^2 + m^2}$, so $p_\mu p_\mu = p \cdot p = -m^2$, where m is the mass of the particle.

The amplitude in Eq. (1) is a complex number formed from a structure of the form $\tilde{v}_2 \, V \, v_1$, where $\tilde{v}_2$ is the "row vector" $\bar{u}(p_2)$ and v_1 is the "column vector" $u(p_1)$. V is a product of 4×4 matrices consisting of everything in between.

To calculate the cross section it is necessary to obtain an expression for the transition rate $\mathcal{M}\mathcal{M}^*$ in terms of the energy and scattering angle by using explicit forms of p_1, p_2, k_1, k_2 in a convenient reference frame. When reducing $\mathcal{M}\mathcal{M}^*$ to a real number, one can usually sums over the final electron and photon spins and averages over the initial electron and photon spins. There are well known techniques available to do this but the algebra is, needless, to say tedious. In particular when electrons (or any spin 1/2 fermions) are involved, the reduction will require taking the trace of a product of γ matrices (see the Appendix for an example of the traces of some products of gamma matrices). Schoonschip, a symbolic manipulation program created by M. Veltman, performs these reduction techniques. The program will need a choice of spatial coordinates, a choice of a frame and the corresponding momenta. We choose the rest frame of the initial electron with the initial photon propagating in the z direction and the scattered photon in the $x - z$ plane. Then, the momenta are

[1] see Appendix

$$
\begin{aligned}
k_1 &= (0,0,\omega,i\omega) \\
p_1 &= (0,0,0,im_e) \\
k_2 &= (\omega'\sin(\theta),0,\omega'\cos(\theta),i\omega') \\
p_2 &= p_1+k_1-k_2
\end{aligned} \tag{2}
$$

Since $p_2^2=-m_e^2$, the last line in Eq. (2) can be used to express ω' in terms of ω and $\cos(\theta)$ as

$$
\omega' = \frac{\omega}{1+\omega(1-\cos(\theta))/m_e}\,. \tag{3}
$$

With these choices, a Schoonschip program that does the job is

```
A me,cos
C A lists the algebraic variables.
V p1,p2,q1,q2,k,kp,e,ep,q
C V lists the vectors
I i1,i2,mu,mup,nu,nup
C I lists the indices.
F Ms=u,Mu=u
C lists the functions.
Z Sig=Ms*Conjg(Ms)-Mu*Conjg(Ms)-Ms*Conjg(Mu)+Mu*Conjg(Mu)
C Z defines the quantity to be evaluated.
Id,Ms=Ubg(i1,me,p2)*G(i1,mu)*(me-i*G(i1,q1))*G(i1,nu)*Ug(i1,me,p1)*p1Dk^-1
Id,Mu=Ubg(i1,me,p2)*G(i1,nu)*(me-i*G(i1,q2))*G(i1,mu)*Ug(i1,me,p1)*p1Dkp^-1
C Id defines a substitution.
Id,Spin,i1
C Spin sums over the electron spins.
P out
*yep
C The last two terms tell Schoonschip to print out the result of the spin
C sum.
Id,Trick,Trace,i1
C Trick, Trace simplifies the result of Spin and takes the trace (see Appendix).
Id,q1(mu~)=p1(mu)+k(mu)
Id,Dotpr,q1(mu~)=p1(mu)+k(mu)
Id,q2(mu~)=p1(mu)-kp(mu)
Id,Dotpr,q2(mu~)=p1(mu)-kp(mu)
Id,Dotpr,p2(mu~)=p1(mu)+k(mu)-kp(mu)
Id,kDkp=p1Dk-p1Dkp
Id,p1Dp1=-me^2
Al,p2Dp2=-me^2
Al,kDk=0
Al,kpDkp=0
C Rest frame of the target
Id,p1Dk=-me*w
```

```
Al,p1Dkp=-me*wp
Id,p1Dk^-1=-me^-1*w^-1
Al,p1Dkp^-1=-me^-1*wp^-1
Id,p1Dk^-2=me^-2*w^-2
Al,p1Dkp^-2=me^-2*wp^-2
Id,Multi,me*wp^-1=me*w^-1 + (1-cos)
Id,me^2*wp^-1=me^2*w^-1 + me*(1-cos)
Id,me^2*wp^-2=me^2*w^-2+2*me*w^-1*(1-cos)+(1-cos)^2
Id,cos^2=1-sin^2
C Average over initial electron and photon spins and include the factor 4
C from the electron propagators.
Id,Addfac,1/16
B me
C B brackets the result in powers of me.
P out
*end
```

When this file is processed by Schoonschip, the result is (comments left out):

```
Schoonschip, 68000 version of June 27, 1991. Public version.
Date: Tue May 15 2001 18:56:05.  Memory: start 00020008, length 476860.

Command line: compton.e compton.t

A me,cos
V p1,p2,q1,q2,k,kp,e,ep,q
I i1,i2,mu,mup,nu,nup
F Ms=u,Mu=u
Z Sig=Ms*Conjg(Ms)-Mu*Conjg(Ms)-Ms*Conjg(Mu)+Mu*Conjg(Mu)
L 2 Id,Ms=Ubg(i1,me,p2)*G(i1,mu)*(me-i*G(i1,q1))*G(i1,nu)*Ug(i1,me,p1)*p1Dk^-1
L 4 Id,Mu=Ubg(i1,me,p2)*G(i1,nu)*(me-i*G(i1,q2))*G(i1,mu)*Ug(i1,me,p1)*p1Dkp^-1
L 6 Id,Spin,i1
> P out
*yep

Sig =
 + G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
  * ( me^2*p1Dk^-1*p1Dkp^-1 )

 + G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*Gi(i1)
  * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

 + G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
  * ( - i*me*p1Dk^-1*p1Dkp^-1 )

 + G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
  * ( me^2*p1Dk^-1*p1Dkp^-1 )
```

```
+ G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
 * ( - me^2*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*Gi(i1)
 * ( - i*me^3*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
 * ( i*me*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
 * ( - me^2*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
 * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,nu)*Gi(i1)
 * ( - me^4*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
 * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
 * ( - i*me^3*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,mu)*Gi(i1)
 * ( me^4*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
 * ( - me^2*p1Dk^-2 )

+ G(i1,mu)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
 * ( - i*me^3*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
 * ( - i*me*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*Gi(i1)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
 * ( - p1Dk^-1*p1Dkp^-1 )
```

```
+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
 * ( - i*me*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
 * ( i*me*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*Gi(i1)
 * ( - me^2*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
 * ( p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
 * ( i*me*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,nu)*Gi(i1)
 * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
 * ( - i*me*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
 * ( - me^2*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,mu)*Gi(i1)
 * ( - i*me^3*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
 * ( i*me*p1Dk^-2 )

+ G(i1,mu)*G(i1,q1)*G(i1,nu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
 * ( - me^2*p1Dk^-2 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
 * ( - me^2*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*Gi(i1)
 * ( - i*me^3*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
```

```
 * ( i*me*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
 * ( - me^2*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*Gi(i1)
 * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
 * ( - i*me*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
 * ( - i*me^3*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,nu)*Gi(i1)
 * ( me^4*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
 * ( - me^2*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
 * ( - i*me^3*p1Dkp^-2 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
 * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,mu)*Gi(i1)
 * ( - me^4*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
 * ( me^2*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
 * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

+ G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
 * ( i*me*p1Dkp^-2 )

+ G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,nu)*Gi(i1)
 * ( - me^2*p1Dkp^-2 )
```

```
 + G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
  * ( p1Dkp^-2 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
  * ( i*me*p1Dkp^-2 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
  * ( - i*me*p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,mu)*Gi(i1)
  * ( me^2*p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
  * ( - p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*G(i1,p1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
  * ( - i*me*p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,nu)*G(i1,p2)
  * ( - me^2*p1Dkp^-2 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,nu)*Gi(i1)
  * ( - i*me^3*p1Dkp^-2 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*G(i1,p2)
  * ( i*me*p1Dkp^-2 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,mu)*G(i1,q2)*G(i1,nu)*Gi(i1)
  * ( - me^2*p1Dkp^-2 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,mu)*G(i1,p2)
  * ( me^2*p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,mu)*Gi(i1)
  * ( i*me^3*p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*G(i1,p2)
  * ( - i*me*p1Dk^-1*p1Dkp^-1 )

 + G(i1,nu)*G(i1,q2)*G(i1,mu)*Gi(i1)*G(i1,nu)*G(i1,q1)*G(i1,mu)*Gi(i1)
  * ( me^2*p1Dk^-1*p1Dkp^-1 ) + 0.

L 1 Id,Trick,Trace,i1
L 9 Id,q1(mu~)=p1(mu)+k(mu)
L10 Id,Dotpr,q1(mu~)=p1(mu)+k(mu)
```

```
L11 Id,q2(mu~)=p1(mu)-kp(mu)
L12 Id,Dotpr,q2(mu~)=p1(mu)-kp(mu)
L13 Id,Dotpr,p2(mu~)=p1(mu)+k(mu)-kp(mu)
L14 Id,kDkp=p1Dk-p1Dkp
L15 Id,p1Dp1=-me^2
L15 Al,p2Dp2=-me^2
L15 Al,kDk=0
L15 Al,kpDkp=0
C Rest frame of the target
L16 Id,p1Dk=-me*w
L16 Al,p1Dkp=-me*wp
L17 Id,p1Dk^-1=-me^-1*w^-1
L17 Al,p1Dkp^-1=-me^-1*wp^-1
L18 Id,p1Dk^-2=me^-2*w^-2
L18 Al,p1Dkp^-2=me^-2*wp^-2
L19 Id,Multi,me*wp^-1=me*w^-1 + (1-cos)
L21 Id,me^2*wp^-1=me^2*w^-1 + me*(1-cos)
L23 Id,me^2*wp^-2=me^2*w^-2+2*me*w^-1*(1-cos)+(1-cos)^2
L25 Id,cos^2=1-sin^2
C Average over initial electron and photon spins and include the factor 1/4
C from the electron propagators.
L26 Id,Addfac,1/16
B me
> P out
*end

Sig =  + 2*w*wp^-1 + 2*w^-1*wp - 2*sin^2 + 0.


End run. Time 1 sec.
```

Here, I printed out the step where the electron spin is summed to show how many products of gamma matrices are generated. Removing

```
P out
*yep
```

will suppress the pages of γ products.

Thus, the desired quantity $\mathcal{M}\mathcal{M}*$ is

$$\mathcal{M}\mathcal{M}* = 2\left(\frac{\omega}{\omega'} + \frac{\omega'}{\omega} - \sin^2(\theta)\right) . \tag{4}$$

To get the differential cross section $d\sigma/d\Omega$ we need to multiply this result by the coupling $e^4 = 16\pi^2\alpha^2$ and a kinematic factor that includes the incoming flux and the final state phase space, which is

$$\frac{\omega'^2}{(8\pi)^2 m_e^2\,\omega^2} . \tag{5}$$

The final result is the Klein-Nishina formula

$$\frac{d\sigma}{d\Omega} = \frac{1}{2}\frac{\alpha^2\,\omega'^2}{m_e^2\,\omega^2}\left(\frac{\omega}{\omega'} + \frac{\omega'}{\omega} - \sin^2(\theta)\right) . \tag{6}$$

Appendix

The γ matrices are 4×4 matrices that satisfy the anti commutation relation

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = 2\delta_{\mu\nu} I\,, \tag{7}$$

where I is the unit matrix. A common representation is

$$\begin{aligned}
\gamma_1 &= \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix} \quad \gamma_2 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix} \\
\gamma_3 &= \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & i \\ i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix} \quad \gamma_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}
\end{aligned} \tag{8}$$

Taking the trace of Eq. (7) (the trace of a 4×4 matrix $M_{\alpha\,\beta}$ is $\mathrm{Tr}(M)$=$M_{11} + \cdots + M_{44}$) gives

$$\mathrm{Tr}(\gamma_\mu\gamma_\nu) + \mathrm{Tr}(\gamma_\nu\gamma_\mu) = 8\delta_{\mu\nu}\,. \tag{9}$$

From the definition of the trace, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ for $n \times n$ matrices, so

$$\mathrm{Tr}(\gamma_\mu\gamma_\nu) = 4\delta_{\mu\nu}. \tag{10}$$

It turns out that the trace of the product of an odd number of γ's vanishes so consider $\gamma_\mu\gamma_\nu\gamma_\lambda\gamma_\rho$. Using Eq. (9), one can show that

$$\gamma_\mu\gamma_\nu\gamma_\lambda\gamma_\rho + \gamma_\nu\gamma_\lambda\gamma_\rho\gamma_\mu = 2(\delta_{\mu\nu}\gamma_\lambda\gamma\rho - \delta_{\mu\lambda}\gamma_\nu\gamma_\rho + \delta_{\mu\rho}\gamma_\nu\gamma_\lambda)\,. \tag{11}$$

Then, taking the trace gives

$$\mathrm{Tr}(\gamma_\mu\gamma_\nu\gamma_\lambda\gamma_\rho) = 4(\delta_{\mu\nu}\delta_{\lambda\rho} - \delta_{\mu\lambda}\delta_{\nu\rho} + \delta_{\mu\rho}\delta_{\nu\lambda})\,. \tag{12}$$

It is easy to see how quickly this escalates as the number of γ's increases. The Trick,Trace commands do all this automatically.