```
filename: ZFitterFermionPart-detailed.cxx
commented printout of gfitter/GSM/ZFitterFermionPart.cxx, J. Haller, 2011-03-03
"Look here only at few functions, namely with 4 or with 3 arguments: m_WfAtMW = XWM1F etc."
The functions agree with zfitter/dizet and disagree with the reference given, from 9709229
"Many other cases are listed in file gfitter-uses-funs-of-zfitter-short/long.txt"

==========================================================================================
===
Correct references:

D. Bardin, G. Passarino, "The Standard Model in the Making", Oxford University Press, 1999

D. Bardin, M. Bilenky, S. Riemann, T. Riemann et al.,hep-ph/9709229, "Electroweak Working Group Report"
section 4.4: ZFITTER basics, p. 86-91

etc.

=========================================================================================
original file begins here, with insertions from zfitter and coments from zfitter group
=========================================================================================

/*****************************************************************************
 * Project: GSM - Electroweak fitting package                                *
 * Package: GSM                                                              *
 * Class  : ZFitterFermionPart                                               *
 *                                                                           *
 * Description:                                                              *
 *      Auxiliary Theory for ferminic part of self energies                  *
 *      one loop core of ZFitter option                                      *
 *                                                                           *
 * Sources:                                                                  *
 *      hep-ph/9709299, hep-ph/9908433                                       *
 *      ZFitter package  dizet6_42.f                                         *
 *                                                                           *
 * Additional Information                                                    *
 *      The Standard Model in the Making, Oxford 1999                        *
 *      Nucl. Phys. B197 (1982) 1-44 / first summary of one loop core        *
 *                                                                           *
 * Authors (alphabetical):                                                   *
 *      Martin Goebel   <martin.goebel@desy.de>  - DESY, Germany             *
 *      Andreas Hoecker <Andreas.Hocker@cern.ch> - CERN, Switzerland         *
 *                                                                           *
 * Copyright (c) 2006:                                                       *
 *      CERN, Switzerland,                                                   *
 *                                                                           *
 * Redistribution and use in source and binary forms, with or without        *
 * modification, are permitted according to the terms listed in LICENSE      *
 * (http://mva.sourceforge.net/license.txt)                                  *
 *                                                                           *
 * File and Version Information:                                             *
 * $Id: ZFitterFermionPart.cxx,v 1.18 2007/10/30 10:39:44 mgoebel Exp $
 *****************************************************************************/
#include <math.h>

#include "Riostream.h"
#include "TMath.h"

#include "Gfitter/GMath.h"
#include "Gfitter/GConstants.h"
#include "Gfitter/GTheory.h"
#include "Gfitter/GTheoryRef.h"
#include "Gfitter/GParameterRef.h"
#include "Gfitter/GReference.h"

#include "GSM/ZFitterFermionPart.h"
#include "GSM/GSMMath.h"
#include "GSM/QPoleMass.h"

using std::complex;

using namespace Gfitter;
GSM::ZFitterFermionPart::ZFitterFermionPart()
```

```
   : Gfitter::GAuxTheory(),
     m_isUpToDate_Update( kFALSE )
{
   SetTheoryName( GetName() );
   SetExistDerivative( kFALSE );

   BookParameter( "MZ", &p_MZ );
   BookParameter( "mt", &p_mt );

   BookTheory   ( "GSM::MW",        &t_MW );
   BookTheory   ( "GSM::QPoleMass", &t_QPoleMass );
}

void GSM::ZFitterFermionPart::UpdateLocalFlags( GReference& /* ref */ )
{
   m_isUpToDate_Update = kFALSE;
}

void GSM::ZFitterFermionPart::Update()
{
   if (m_isUpToDate_Update) return;

   // now, it is uptodate (I mean... it will be)
   m_isUpToDate_Update = kTRUE;

   m_MW2 = GMath::IPow( GetMW(), 2 );
   m_MZ2 = p_MZ*p_MZ;

   m_R   = m_MW2/m_MZ2;
   m_R1  = 1.0 - m_R;
   m_R12 = m_R1*m_R1;

   // lepton masses
   m_ml[0][0] = 0;
   m_ml[0][1] = 0;
   m_ml[0][2] = 0;
   m_ml[1][0] = GConstants::me();
   m_ml[1][1] = GConstants::mmu();
   m_ml[1][2] = GConstants::mtau();

   // quark masses
   m_mq[0][0] = GetQPoleMass().GetPoleQMass( GTypes::kUp );
   m_mq[0][1] = GetQPoleMass().GetPoleQMass( GTypes::kCharm );
   m_mq[0][2] = p_mt;
   m_mq[1][0] = GetQPoleMass().GetPoleQMass( GTypes::kDown );
   m_mq[1][1] = GetQPoleMass().GetPoleQMass( GTypes::kStrange );
   m_mq[1][2] = GetQPoleMass().GetPoleQMass( GTypes::kBottom );

   // quark charges
   m_chq[0] = GMath::TwoThird();
   m_chq[1] = GMath::OneThird();

   // member variables
   // W boson self energies ( fermionic part )
   m_WfAt0  = 0.0;
   m_WfAtMW = 0.0;
   // Z boson self energies ( fermionic part )
   m_ZfAtMZ = 0.0;
   m_ZfAt0  = 0.0;

   // computation of fermionic part to self-energies
   // for additional information see also ZFitter
   // package dizet6_42.f line 1910-1960
   for (Int_t i=0; i<3 ; i++) {

      // massive leptons and quarks
      Double_t ml2  = m_ml[1][i]*m_ml[1][i];
      Double_t mu2  = m_mq[0][i]*m_mq[0][i];
      Double_t md2  = m_mq[1][i]*m_mq[1][i];

      Double_t RLepW  = ml2/m_MW2;
      Double_t RUpW   = mu2/m_MW2;
      Double_t RDownW = md2/m_MW2;
```

```
      Double_t LLepW  = TMath::Log(RLepW);
      Double_t LUpW   = TMath::Log(RUpW);
      Double_t LDownW = TMath::Log(RDownW);

      // eq. (262) of hep-ph/9709229
      // W boson self-energy at 0 GeV
      // neutrino part = 0
      m_WfAt0 = m_WfAt0 + 0.5*(RLepW*LLepW - RLepW/2.0);

      if (RUpW != RDownW){
         m_WfAt0 = ( m_WfAt0 + 3/2.0*((RUpW*RUpW*LUpW - RDownW*RDownW*LDownW)/(RUpW-RDownW)
                                      -(RUpW+RDownW)/2.0) );
      }
      else m_WfAt0 = m_WfAt0 + 3.0*(RUpW*LUpW - RUpW/2.0);

      // eq. (261) of hep-ph/9709229
      // Z boson self-energy at 0 GeV
      // neutrino part = 0
      m_ZfAt0 = m_ZfAt0 + 0.5*RLepW*LLepW;
      m_ZfAt0 = m_ZfAt0 + 3.0/2.0*(RUpW*LUpW + RDownW*LDownW);

      // eq. (262) of hep-ph/9709229
      // W boson self-energy at MW
      // neutrino part = 0
      m_WfAtMW = ( m_WfAtMW - 2.0*GSMMath::I3(m_MW2, -m_MW2, ml2, 0.0)
                   + RLepW*GSMMath::I1(m_MW2, -m_MW2, ml2, 0.0) );

      m_WfAtMW = ( m_WfAtMW - 6.0*GSMMath::I3(m_MW2, -m_MW2, mu2, md2)
                   + 3.0*(RUpW*GSMMath::I1(m_MW2, -m_MW2, mu2, md2)
                          + RDownW*GSMMath::I1(m_MW2, -m_MW2, md2, mu2)) );

      "-----------------------------------------------------------------------------------------------"
"compare to zfitter/dizet6_42.f  lines 1929 ff: [even the linebreaks agree]"

            XWM1F=XWM1F-2D0*XI3(AMW2MU,-AMW2,AML2,0D0 )
     *      +     RMLW*XI1(AMW2MU,-AMW2,AML2,0D0 )
      XWM1F=XWM1F-6D0*XI3(AMW2MU,-AMW2,AMT2,AMB2)
     *      +3D0*(RMTW*XI1(AMW2MU,-AMW2,AMT2,AMB2)
     *      +     RMBW*XI1(AMW2MU,-AMW2,AMB2,AMT2))

     "In contrast, see (262) of hep-ph/9709229, p.88, W_f(s) has I_3(a,b,c), I_1(a,b,c)

     "===============================================================================================

      // eq. (261) of hep-ph/9709229
      // Z boson self-energy at MZ

      // neutrino part
      // eq. (261) has changed, because masses are zero
      complex<Double_t> clogWZ (TMath::Log(m_MW2/m_MZ2),TMath::Pi());
      m_ZfAtMZ = m_ZfAtMZ + 1.0/(6.0*m_R)*(clogWZ + 5.0/3.0);

      // massive fermion part
      // Born weak couplings
      Double_t v2l = 1.0 + GMath::IPow(1.0-4.0*m_R1,2);
      Double_t v2t = 1.0 + GMath::IPow(1.0-4.0*m_R1*m_chq[0],2);
      Double_t v2b = 1.0 + GMath::IPow(1.0-4.0*m_R1*m_chq[1],2);

      m_ZfAtMZ = ( m_ZfAtMZ - 0.5*v2l/m_R*GSMMath::I3(m_MW2, -m_MZ2, ml2, ml2)
                   + 0.5*RLepW*GSMMath::I0(m_MW2, -m_MZ2, ml2, ml2) );
      m_ZfAtMZ = ( m_ZfAtMZ - 3.0/2.0*v2t/m_R*GSMMath::I3(m_MW2, -m_MZ2, mu2, mu2)
                   + 3.0/2.0*RUpW*GSMMath::I0(m_MW2, -m_MZ2, mu2, mu2) );
      m_ZfAtMZ = ( m_ZfAtMZ - 3.0/2.0*v2b/m_R*GSMMath::I3(m_MW2, -m_MZ2, md2, md2)
                   + 3.0/2.0*RDownW*GSMMath::I0(m_MW2, -m_MZ2, md2, md2) );
   }

   // derivatives of self energies / F = finite
   m_ZfFAtMZ     = DerivativeZF( -m_MZ2 );
   m_WfFAtMW     = DerivativeWF( -m_MW2 );

   // photon Z mixing function
   m_MfPhoZAtMZ  = MfphoZ( -m_MZ2 );
```

```
   SetUpToDate();
}

// eq. (264) of hep-ph/9709229v1
// see ZFitter package dizet6_42.f line 1329-1352 ( function XAMF )
complex<Double_t> GSM::ZFitterFermionPart::MfphoZ( const Double_t& Q2 ) const
{
   complex<Double_t> chmQ1 (0,0);

   // leptons
   for (Int_t i=0; i<3 ; i++) {
      Double_t ml2 = m_ml[1][i]*m_ml[1][i];
      chmQ1 = chmQ1 + GSMMath::I3(m_MW2, Q2, ml2, ml2);
   }

   complex<Double_t> chmQ2 = chmQ1;

   // quarks
   for (Int_t i=0; i<3; i++) {
      for (Int_t j=0; j<2; j++) {
         Double_t mq2 = m_mq[j][i]*m_mq[j][i];
         chmQ1  = chmQ1 + 3.0*m_chq[j]*GSMMath::I3(m_MW2, Q2, mq2, mq2);
         chmQ2  = chmQ2 + 3.0*m_chq[j]*m_chq[j]*GSMMath::I3(m_MW2, Q2, mq2, mq2);
      }
   }
   return 8.0*m_R1*chmQ2 - 2.0*chmQ1;
}

// see ZFitter package dizet6_42.f line 1249-1279 ( function XDWF)
complex<Double_t> GSM::ZFitterFermionPart::DerivativeWF( const Double_t& Q2 ) const
{
   complex<Double_t> SumI3   (0,0);
   complex<Double_t> SumDI1u (0,0);
   complex<Double_t> SumDI1d (0,0);

   // massive leptons
   for (Int_t i=0; i < 3 ; i++) {
      Double_t ml2 = m_ml[1][i]*m_ml[1][i];
      SumI3   = SumI3 + GSMMath::I3(m_MW2, Q2, ml2, 0.0) - GSMMath::DI3(Q2, m_MW2, ml2, 0, m_MW2);
      SumDI1d = SumDI1d + ml2/m_MW2*GSMMath::DI1(Q2, m_MW2, ml2, 0.0, m_MW2);
   }

   // quarks
   for (Int_t i=0; i < 3; i++) {
      Double_t mu2 = m_mq[0][i]*m_mq[0][i];
      Double_t md2 = m_mq[1][i]*m_mq[1][i];
      SumI3   = SumI3 + 3.0*(GSMMath::I3(m_MW2, Q2, md2, mu2) - GSMMath::DI3(Q2, m_MW2, mu2, md2,
      m_MW2));
      SumDI1u = SumDI1u + 3.0*mu2/m_MW2*GSMMath::DI1(Q2, m_MW2, mu2, md2, m_MW2);
      SumDI1d = SumDI1d + 3.0*md2/m_MW2*GSMMath::DI1(Q2, m_MW2, md2, mu2, m_MW2);
   }

   return 2.0*SumI3 + SumDI1u + SumDI1d;
}

// see hep-ph/9709229v1 page 87-90
// hep-ph/9908433 page 154-157
// and ZFitter package dizet6_42.f line 1281-1327 ( function XDZF)
complex<Double_t> GSM::ZFitterFermionPart::DerivativeZF( const Double_t& Q2 ) const
{
   Double_t LogQZ  = TMath::Log(TMath::Abs(Q2/m_MZ2));
   Double_t QZ     = m_MZ2 + Q2;
   Double_t il1    = 0;
   Double_t il2    = 0;
   if (Q2 < 0) il1 = -TMath::Pi();
   if (Q2 > 0) il2 = -TMath::Pi();

   complex<Double_t> l1 ( LogQZ, il1 );
   complex<Double_t> l2 ( LogQZ, il2 );

   complex<Double_t> lq (0,0);
   complex<Double_t> SumNu(0,0);
   if (TMath::Abs(QZ) > 1e-3) lq = l1 - m_MZ2/(m_MZ2+Q2)*l2;
```

```
   else                        lq = l1 + 1.0 + QZ/2.0 + QZ*QZ/3.0;

   // massless neutrinos
   SumNu = 3.0/(6.0*m_R)*(-5.0/3.0 - TMath::Log(m_R) + lq);

   complex<Double_t> SumI3 (0,0);
   complex<Double_t> SumQMI3 (0,0);
   complex<Double_t> SumQ2I3 (0,0);
   complex<Double_t> SumDI0 (0,0);

   // massive leptons
   for (Int_t i=0; i<3 ; i++) {
      Double_t ml2 = m_ml[1][i]*m_ml[1][i];
      SumI3   = SumI3 + GSMMath::I3(m_MW2, Q2, ml2, ml2) - GSMMath::DI3(Q2, m_MZ2, ml2, ml2, m_MW2);
      SumDI0  = SumDI0 + ml2/m_MW2*GSMMath::DI0(Q2, m_MZ2, ml2, ml2);
   }

   SumQMI3 = SumI3;
   SumQ2I3 = SumI3;
   // quarks
   for (Int_t i=0; i<3; i++) {
      for (Int_t j=0; j<=1;j++) {
         Double_t mq2 = m_mq[j][i]*m_mq[j][i];
         SumI3   = SumI3 + 3.0*(GSMMath::I3(m_MW2, Q2, mq2, mq2) - GSMMath::DI3(Q2, m_MZ2, mq2, mq2,
         m_MW2));
         SumQMI3 = SumQMI3 + 3.0*m_chq[j]*(GSMMath::I3(m_MW2, Q2, mq2, mq2)
                                           - GSMMath::DI3(Q2, m_MZ2, mq2, mq2, m_MW2));
         SumQ2I3 = SumQ2I3 + 3.0*m_chq[j]*m_chq[j]*(GSMMath::I3(m_MW2, Q2, mq2, mq2)
                                                    - GSMMath::DI3(Q2, m_MZ2, mq2, mq2, m_MW2));
         SumDI0  = SumDI0 + 3.0*mq2/m_MW2*GSMMath::DI0(Q2, m_MZ2, mq2, mq2);
      }
   }

   return (8.0*m_R12*SumQ2I3 - 4.0*m_R1*SumQMI3 + SumI3)/m_R + SumDI0/2.0 + SumNu;
}
```