```
filename: RadiatorFunctions-detailed.cxx
commented printout of gfitter/GSM/RadiatorFunctions.cxx, J. Haller, 2011-03-03
"Look here only at one function in  more detail: m_CA3 "
This function agrees with zfitter/dizet and disagreees with the reference given, from the SMM book

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

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

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

/*****************************************************************************
 * Project: GSM - Electroweak fitting package                                *
 * Package: GSM                                                              *
 * Class  : RadiatorFunctions                                                *
 *                                                                           *
 * Description:                                                              *
 *      Auxiliary Theory computes QCD radiator functions                     *
 *      for Z and W decay                                                    *
 *                                                                           *
 * Papers:                                                                   *
 *      The Standard Model in the Making, Oxford 1999                        *
 *                                                                           *
 * Authors (alphabetical):                                                   *
 *      Martin Goebel   <martin.goebel@desy.de>  - DESY, Germany             *
 *                                                                           *
 * 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: RadiatorFunctions.cxx,v 1.18 2008/07/15 04:21:01 hoecker Exp $
 *****************************************************************************/
#include "TMath.h"
#include "Riostream.h"

#include "Gfitter/GMath.h"
#include "Gfitter/GStore.h"
#include "Gfitter/GConstants.h"
#include "Gfitter/GParameterRef.h"
#include "Gfitter/GTheoryRef.h"
#include "Gfitter/GVariable.h"

#include "GSM/RadiatorFunctions.h"
#include "GSM/QMassRunning.h"
#include "GSM/DAlphaQED.h"
#include "GSM/AlphaQCDAtQ.h"

#include <iomanip>

using namespace Gfitter;

ClassImp(GSM::RadiatorFunctions)

GSM::RadiatorFunctions::RadiatorFunctions()
   : Gfitter::GAuxTheory(),
   m_useNLOAlphasOnly ( kFALSE ),
   m_isUpToDate_Update( kFALSE )
{
   SetTheoryName( GetName() );
   SetExistDerivative( kFALSE );

   BookParameter( "MZ",                        &p_MZ );
   BookParameter( "mt",                        &p_mt );
   BookParameter( "DeltaAlphasTheoC05_Scale",  &p_DeltaAlphasTheoC05_Scale );
```

```
   BookParameter( "DeltaAlphasTheoCMt3_Scale", &p_DeltaAlphasTheoCMt3_Scale );

   BookTheory   ( "GSM::AlphaQCDAtQ/MZ",       &t_AlphasMZ );
   BookTheory   ( "GSM::DAlphaQED",            &t_DAlphaQED );
   BookTheory   ( "GSM::QMassRunning",         &t_qMassRun );
}

void GSM::RadiatorFunctions::Initialise()
{
   if (gStore()->ExistVariable( "GSMFlags::UseNLOAlphasOnlyInRadFun" )) {
      m_useNLOAlphasOnly = gStore()->GetVariable( "GSMFlags::UseNLOAlphasOnlyInRadFun" )-
      >GetBoolValue();
   }

   Double_t zeta2 = GMath::Zeta2();
   Double_t zeta3 = GMath::Zeta3();
   Double_t zeta4 = GMath::Zeta4();
   Double_t zeta5 = GMath::Zeta5();

   m_nf1   = 5;

   // see for coefficients The Standard Model in the Making page 504

   // massless non-singlet corrections
   m_C01   = 1.0;
   m_C02   = 365/24.0 - 11*zeta3 + (-11/12.0 + 2/3.0*zeta3)*m_nf1;
   m_C03   = ( 87029/288.0 - 121/8.0*zeta2 - 1103/4.0*zeta3 + 275/6.0*zeta5
             + (-7847/216.0 + 11/6.0*zeta2 + 262/9.0*zeta3 - 25/9.0*zeta5)*m_nf1
             + (151/162.0 - 1/18.0*zeta2 - 19/27.0*zeta3)*m_nf1*m_nf1 );
   m_C04   = -156.61 + 18.77*m_nf1 -0.7974*m_nf1*m_nf1 + 0.0215*m_nf1*m_nf1*m_nf1;
   // not used, since cancellation
   m_C05   = -68.078*(1 - p_DeltaAlphasTheoC05_Scale); // rough estimate from geometric series: C05 =
   C04*(C04/C03)
   //m_C05 = 0;


   // quadratic massive corrections
   // only running mc and mb => sqrt(s) > 15 GeV
   // light quarks
   m_CL1   = 0;
   m_CL2   = 0;
   m_CL3   = -80.0 + 60*zeta3 + (32/9.0 - 8/3.0*zeta3)*m_nf1;

   // heavy quarks
   // vector part
   m_CV1   = 12.0;
   m_CV2   = 253/2.0 - 13/3.0*m_nf1;
   m_CV3   = ( 2522.0 - 855/2.0*zeta2 + 310/3.0*zeta3 - 5225/6.0*zeta5
             + (-4942/27.0 + 34*zeta2 - 394/27.0*zeta3 + 1045/27.0*zeta5)*m_nf1
             + (125/54.0 - 2/3.0*zeta2)*m_nf1*m_nf1 );

   // axial part
   m_CA0   = -6.0;
   m_CA1   = -22.0;
   m_CA2   = -8221/24.0 + 57*zeta2 + 117*zeta3 + (151/12.0 - 2*zeta2 - 4*zeta3)*m_nf1;
   m_CA3   = ( - 4544045/864.0 + 1340*zeta2 + 118915/36.0*zeta3 - 1270*zeta5
             + (71621/162.0 - 209/2.0*zeta2 - 216*zeta3 + 5*zeta4 + 55*zeta5)*m_nf1
             + (-13171/1944.0 + 16/9.0*zeta2 + 26/9.0*zeta3)*m_nf1*m_nf1 );
}
"-------------------------------------------------------------------------------------------------"
"compare to zfitter/dizet6_42.f  lines 5060-5080 : [even the linebreaks agree]"

* Light quarks
*
      COEFL1=0D0
      COEFL2=0D0
      COEFL3=-80D0+60D0*D3+(32D0/9-8D0/3*D3)*ANF
*
* Heavy quarks
*
      COEFV1=12D0
      COEFV2=253D0/2-13D0/3*ANF
      COEFV3=  2522D0   - 855D0/2*D2+ 310D0/3*D3- 5225D0/6*D5
```

```
     &      +(-4942D0/27+    34D0*D2-394D0/27*D3+1045D0/27*D5)*ANF
     &      +(125D0/54  -   2D0/3*D2                      )*ANF**2
*
      COEFA0=-6D0
      COEFA1=-22D0
      COEFA2=-8221D0/24+57D0*D2+117D0*D3
     &      +(151D0/12 - 2D0*D2-  4D0*D3)*ANF
      COEFA3=-4544045D0/864+   1340*D2+118915D0/36*D3  -1270D0*D5
     &      +(71621D0/162  -209D0/2*D2   -216D0*D3+5D0*D4+55D0*D5)*ANF
     &      +(-13171D0/1944+ 16D0/9*D2  +26D0/9*D3            )*ANF**2
*

"comment:"
Look at "-1270D0*D5" in "m_CA3 = COEFA3". They agree.
Files:
RadiatorFunctions-sm-making-p505.pdf
RadiatorFunctions-Goebel_dipl-p85.pdf
Looking into the SMM book, page 505, eq. (12.38), C_{23}^A, see there instead:
"-127\zeta(5)".
The same "-127\zeta(5)" is found in the diploma thesis of M. Goebel.
Conclusion:
The thesis follows the book SMM, and the program follows dizet. Certainly without much reason, just due
to copying.

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

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

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

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

   Double_t pi    = TMath::Pi();

   Double_t MSMc  = GetQMassRun().GetRunningQuarkMass( GTypes::kCharm );
   Double_t MSMb  = GetQMassRun().GetRunningQuarkMass( GTypes::kBottom );

   m_asMZpi       = GetAlphasMZ()/pi;
   m_aQEDMZpi     = (GConstants::alphaQED()/( 1.0 - GetDAlphaQED().DAlphaQEDMZt() ))/pi;

   m_RatioMcMZ    = GMath::IPow( MSMc/p_MZ,2);
   m_RatioMbMZ    = GMath::IPow( MSMb/p_MZ,2);
   m_RatioMtMZ    = GMath::IPow( p_MZ /p_mt,2);

   m_RatioMcMt    = GMath::IPow( MSMc/p_mt,2);
   m_RatioMbMt    = GMath::IPow( MSMb/p_mt,2);

   // light quark corrections
   //   Quadratic corrections
   m_LightQu2     = ( m_RatioMcMZ + m_RatioMbMZ )*m_CL3*GMath::IPow(m_asMZpi,3);

   //   Quartic corrections
   m_LightQu4Mc   = ( m_RatioMcMZ*m_RatioMcMZ
                      *(13/3.0 - TMath::Log(m_RatioMcMZ) - 4.0*GMath::Zeta3())
                      *m_asMZpi* m_asMZpi);
   m_LightQu4Mb   = ( m_RatioMbMZ*m_RatioMbMZ
                      *(13/3.0 - TMath::Log(m_RatioMbMZ) - 4.0*GMath::Zeta3())
                      *m_asMZpi*m_asMZpi );
   SetUpToDate();
}


// Radiator-functions for computing Z->qq
// Equation are in The Standard Model in the Making
// page 503 - 505

// vector part; Ve = Vector
```

```
Double_t GSM::RadiatorFunctions::GetRVq( GTypes::Particle ParticleType, Double_t Charge )
{
   Update();

   // heavy quark corrections
   //   Quadratic corrections
   Double_t VeHeavyQu2 = m_CV1*m_asMZpi + m_CV2*GMath::IPow(m_asMZpi,2);
   if (!m_useNLOAlphasOnly) VeHeavyQu2 += m_CV3*GMath::IPow(m_asMZpi,3);


   //   Quartic corrections
   //     particulary bottom corretions
   Double_t VeHeavyQu4 = ( -6.0 - 22.0*m_asMZpi
                           + (12.0 - 3173/12.0 + 27.0*GMath::IPow(TMath::Pi(),2) + 112.0*GMath::Zeta3()
                              + (143/18.0 -2/3.0* GMath::IPow(TMath::Pi(),2) -
                              8/3.0*GMath::Zeta3())*m_nf1)
                           * GMath::IPow(m_asMZpi,2) );
   Double_t Vector4  = (-11/2.0 + 1/3.0*m_nf1)*GMath::IPow(m_asMZpi,2);

   Double_t VectorQCD2 = m_LightQu2;
   Double_t VectorQCD4 = m_LightQu4Mc + m_LightQu4Mb;

   if( ParticleType == GTypes::kCharm ){
      VectorQCD2 += m_RatioMcMZ * VeHeavyQu2;
      VectorQCD4 += ( GMath::IPow(m_RatioMcMZ,2)*(VeHeavyQu4 + Vector4*TMath::Log(m_RatioMcMZ))
                      + 12.0*GMath::IPow(m_RatioMbMZ*m_asMZpi,2) );
   }
   else if( ParticleType == GTypes::kBottom){
      VectorQCD2 += m_RatioMbMZ * VeHeavyQu2;
      VectorQCD4 += ( GMath::IPow(m_RatioMbMZ,2)*(VeHeavyQu4 + Vector4*TMath::Log(m_RatioMbMZ))
                      + 12.0*GMath::IPow(m_RatioMcMZ*m_asMZpi,2)
                      - GMath::IPow(m_RatioMbMZ,3)
                      *(8.0 + 16/27.0*(155.0 + 6.0*TMath::Log(m_RatioMbMZ))*m_asMZpi) );
   }

   Double_t RadV = ( 1.0 + m_asMZpi + 1/4.0*m_aQEDMZpi*Charge*Charge*(3.0 - m_asMZpi)
                     + (m_C02 + (44/675.0 - 2/135.0*TMath::Log(m_RatioMtMZ))*m_RatioMtMZ)
                     *GMath::IPow(m_asMZpi,2)
                     + (m_useNLOAlphasOnly ? 0 : ( m_C03*GMath::IPow(m_asMZpi,3) +
                                                   m_C04*GMath::IPow(m_asMZpi,4) +
                                                   m_C05*GMath::IPow(m_asMZpi,5) ))
                     + VectorQCD2 + VectorQCD4 );
   return RadV;
}

// axial part
Double_t GSM::RadiatorFunctions::GetRAq( GTypes::Particle ParticleType, Double_t Charge, Double_t T3 )
{
   Update();

   // heavy quark corrections
   //   Quadratic corrections
   Double_t AxHeavyQu2 = ( m_CA0 + m_CA1*m_asMZpi + m_CA2*GMath::IPow(m_asMZpi,2)
                           + (m_useNLOAlphasOnly ? 0 : m_CA3*GMath::IPow(m_asMZpi,3)) );

   //   Quartic corrections
   //     particulary bottom corretions
   Double_t AxHeavyQu4 = ( 6.0 + 10.0*m_asMZpi
                           + (-12.0 + 3533/12.0 - 27.0*GMath::IPow(TMath::Pi(),2) -
                           220.0*GMath::Zeta3()
                              + (-41/6.0 + 2/3.0*GMath::IPow(TMath::Pi(),2)
                                 + 16/3.0*GMath::Zeta3())*m_nf1)* GMath::IPow(m_asMZpi,2) );
   Double_t Axial4     = (77/2.0 - 7/3.0*m_nf1)*GMath::IPow(m_asMZpi,2);

   Double_t AxialQCD2  = m_LightQu2;
   Double_t AxialQCD4  = m_LightQu4Mc + m_LightQu4Mb;
   // axial single contribution
   Double_t AxSingle   = 0;


   if( ParticleType == GTypes::kCharm ){
      AxialQCD2 += m_RatioMcMZ * AxHeavyQu2;
      AxialQCD4 += ( GMath::IPow(m_RatioMcMZ,2)*(AxHeavyQu4 + Axial4*TMath::Log(m_RatioMcMZ))
```

```
                    - 12.0*GMath::IPow(m_RatioMbMZ*m_asMZpi,2) );
      AxSingle   = ( -6.0*m_RatioMcMZ*(-3.0 + TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi
                    -10.0*m_RatioMcMt*(8/81.0 - 1/54.0*TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi );
   }
   else if( ParticleType == GTypes::kBottom ){
      AxialQCD2 += m_RatioMbMZ * AxHeavyQu2;
      AxialQCD4 += ( GMath::IPow(m_RatioMbMZ,2)*(AxHeavyQu4 + Axial4*TMath::Log(m_RatioMbMZ))
                    - 12.0*GMath::IPow(m_RatioMcMZ*m_asMZpi,2) );
      AxSingle   = ( -6.0*m_RatioMbMZ*(-3.0 + TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi
                    -10.0*m_RatioMbMt*(8/81.0 - 1/54.0*TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi );
   }

   Double_t CMt2 = -37/12.0 + TMath::Log(m_RatioMtMZ) + 7/81.0*m_RatioMtMZ +
   0.0132*m_RatioMtMZ*m_RatioMtMZ;
   Double_t CMt3 = ( - 5075/216.0 + 8/3.0 + 23/6.0*GMath::Zeta2() + GMath::Zeta3()
                    + 67/18.0*TMath::Log(m_RatioMtMZ) + 23/12.0*GMath::IPow(TMath::Log(m_RatioMtMZ),2)
                    );

   Double_t CMt4 = CMt3/CMt2*CMt3*(1.0 - p_DeltaAlphasTheoCMt3_Scale);

   Double_t RadA = ( 1.0 + m_asMZpi + 1./4.0*m_aQEDMZpi*Charge*Charge*(3.0 - m_asMZpi)
                    + (m_C02 + (44/675.0 - 2/135.0*TMath::Log(m_RatioMtMZ))*m_RatioMtMZ -
                    2.0*T3*CMt2)*GMath::IPow(m_asMZpi,2)
                    + (m_useNLOAlphasOnly ? 0 : ( (m_C03 - 2.0*T3*CMt3)*GMath::IPow(m_asMZpi,3) +
                                                  (m_C04 - 2.0*T3*CMt4)*GMath::IPow(m_asMZpi,4) +
                                                  (m_C05)*GMath::IPow(m_asMZpi,5) ))
                    + AxialQCD2 + AxialQCD4 + AxSingle);
   return RadA;
}
```