```
/*****************************************************************************
 * Package: GSM                                                              *
 * Class  : RadiatorFunctions                                                *
 *                                                                           *
 * Description:                                                              *
 *      Auxiliary Theory computes QCD radiator functions                     *
 *      for Z and W decay                                                    *
 *                                                                           *
 * Papers:                                                                   *
 *      - Degrassi, Bardin, The Standard Model in the Making, Oxford 1999    *
 *      - Bardin et al, CPC 133 (2001) 229-395, hep-ph/9908433               *
 *      - Baikov et al., PR Lett. 101(2008) 012002, arXiv:0709.1983          *
 *      - Bardin et al, ZFitter package dizet6_42.f                          *
 *                                                                           *
 * This class also contains code lines ported to C++ from the Fortran package *
 * ZFITTER                                                                   *
 *                                                                           *
 *****************************************************************************/
#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 )
```

Hinweis:

Kommentare mit Hinweisen auf ZFitter sind grün markiert

Übereinstimmungen sind gelb markiert.

Auffällige Stellen bzw. Anmerkungen sind violett markiert

Anhang 4

zum Gutachten DESY ZFitter_GFitter vom 17.03.2014

Lübeck, den 17. März 2014

Dr. Ulrich Obermöller
-Sachverständiger-

Von der Handelskammer Hamburg
Dr. Ulrich Obermöller
Sachverständiger für Systeme und Anwendungen der Informationsverarbeitung, insbesondere Individualsoftware und Software-Qualitätssicherung
öffentlich bestellt und vereidigt

```
45  {
46    SetTheoryName( GetName() );
47    SetExistDerivative( kFALSE );
48
49    BookParameter( "MZ",                      &p_MZ );
50    BookParameter( "mt",                     &p_mt );
51    BookParameter( "DeltaAlphasTheoC05_Scale",  &p_DeltaAlphasTheoC05_Scale );
52    BookParameter( "DeltaAlphasTheoCMt3_Scale", &p_DeltaAlphasTheoCMt3_Scale );
53
54    BookTheory   ( "GSM::RunningAlphaQCD",      &t_AlphasRun );
55    BookTheory   ( "GSM::DAlphaQED",           &t_DAlphaQED );
56    BookTheory   ( "GSM::QMassRunning",         &t_qMassRun );
57  }
58
59  void GSM::RadiatorFunctions::Initialise()
60  {
61    if (gStore()->ExistVariable( "GSMFlags::UseNLOAlphasOnlyInRadFun" )) {
62      m_useNLOAlphasOnly = gStore()->GetVariable( "GSMFlags::UseNLOAlphasOnlyInRadFun" )->GetBoolValue();
63    }
64
65    Double_t zeta2 = GMath::Zeta2();
66    Double_t zeta3 = GMath::Zeta3();
67    Double_t zeta4 = GMath::Zeta4();
68    Double_t zeta5 = GMath::Zeta5();
69
70    // five active falvors
71    m_nf1   = 5;
72
73    // see for coefficients The Standard Model in the Making page 504
74    // and hep-ph/9908433 page 100-102
75
76    // massless non-singlet corrections
77    m_C01   = 1.0;
78    m_C02   = 365/24.0 - 11*zeta3 + (-11/12.0 + 2/3.0*zeta3)*m_nf1; ||~ line 5049 - 5053
79    m_C03   = ( 87029/288.0 - 121/8.0*zeta2 - 1103/4.0*zeta3 + 275/6.0*zeta5
80              + (-7847/216.0 + 11/6.0*zeta2 + 262/9.0*zeta3 - 25/9.0*zeta5)*m_nf1
81              + (151/162.0 - 1/18.0*zeta2 - 19/27.0*zeta3)*m_nf1*m_nf1 );
82    // NNNLO calculation from arXiv:0709.1983
83    m_C04   = -156.61 + 18.77*m_nf1 -0.7974*m_nf1*m_nf1 + 0.0215*m_nf1*m_nf1*m_nf1
84
85    // rough estimate from geometric series: C05 = C04*(C04/C03)
86    m_C05   = -68.078*(1 - p_DeltaAlphasTheoC05_Scale);
87
88    // Quadratic massive corrections
```

Match 1

```
5046  * Massless corrections
5047  *
5048        ANF=5D0
5049        COEF01=1D0
5050        COEF02=365D0/24-11D0*D3+(-11D0/12+2D0/3*D3)*ANF
5051        COEF03=87029D0/288 -121D0/8*D2-1103D0/4*D3+275D0/6*D5
5052       &      +(-7847D0/216+ 11D0/6*D2+ 262D0/9*D3- 25D0/9*D5)*ANF
5053       &      +(151D0/162  - 1D0/18*D2- 19D0/27*D3           )*ANF**2
5054  *
```

```
89      // only running mc and mb => sqrt(s) > 15 GeV
90      // light quarks
91      m_CL1   = 0; ||~ line 5062 - 5064
92      m_CL2   = 0;
93      m_CL3   = -80.0 + 60*zeta3 + (32/9.0 - 8/3.0*zeta3)*m_nf1;
94
95      // only heavy quarks
96      // vector part
97      m_CV1   = 12.0; ||~ line 5068 - 5072
98      m_CV2   = 253/2.0 - 13/3.0*m_nf1;
99      m_CV3   = ( 2522.0 - 855/2.0*zeta2 + 310/3.0*zeta3 - 5225/6.0*zeta5
100             + (-4942/27.0 + 34*zeta2 - 394/27.0*zeta3 + 1045/27.0*zeta5)*m_nf1
101             + (125/54.0 - 2/3.0*zeta2)*m_nf1*m_nf1 );
102
103     // axial part
104     m_CA0   = -6.0; ||~ line 5074 - 5080
105     m_CA1   = -22.0;
106     m_CA2   = -8221/24.0 + 57*zeta2 + 117*zeta3 + (151/12.0 - 2*zeta2 - 4*zeta3)*m_nf1
107     m_CA3   = ( - 4544045/864.0 + 1340*zeta2 + 118915/36.0*zeta3 - 1270*zeta5
108             + (71621/162.0 - 209/2.0*zeta2 - 216*zeta3 + 5*zeta4 + 55*zeta5)*m_nf1
109             + (-13171/1944.0 + 16/9.0*zeta2 + 26/9.0*zeta3)*m_nf1*m_nf1 );
110
111     // Quartic massive corrections
112      m_C42   = 13/3.0 - 4.0*GMath::Zeta3();||~ line 5114 / 5116
113
114     m_CV40  = -6.; ||~ line 5020 - 5021
115     m_CV41  = -22.;
116     // first term -3029/12. replaced!
117     // (taken from dizet6_42.f line 5120)
118     m_CV42  = -3173/12.0 + 162.*GMath::Zeta2() + 112.0*GMath::Zeta3() +
119        (143/18.0 - 4.*GMath::Zeta2() - 8/3.0*GMath::Zeta3())*m_nf1;
120
121     m_CVL42 = -11/2. + 1/3.*m_nf1; ||~ line 5128
122
123     m_CA40  = 6.; ||~ line 5026 - 5027
124     m_CA41  = 10.;
125     // first term -3389/12. replaced!
126     // (taken from dizet6_42.f line 5126)
127     m_CA42  = 3533/12.0 - 162.*GMath::Zeta2() - 220.0*GMath::Zeta3() +
128        (-41/6. + 4.*GMath::Zeta2() + 16/3.0*GMath::Zeta3())*m_nf1;
129
130     m_CAL42 = 77/2.0 - 7/3.0*m_nf1; ||~ line 5129
131
132 }
```

Match 1

```
5060  * Light quarks
5061  *
5062        COEFL1=0D0
5063        COEFL2=0D0
5064        COEFL3=-80D0+60D0*D3+(32D0/9-8D0/3*D3)*ANF
5065  *

5066  * Heavy quarks
5067  *
5068        COEFV1=12D0
5069        COEFV2=253D0/2-13D0/3*ANF
5070        COEFV3=  2522D0   - 855D0/2*D2+ 310D0/3*D3- 5225D0/6*D5
5071       &      +(-4942D0/27+   34D0*D2-394D0/27*D3+1045D0/27*D5)*ANF
5072       &      +(125D0/54  -   2D0/3*D2                        )*ANF**2
5073  *

5074        COEFA0=-6D0
5075        COEFA1=-22D0
5076        COEFA2=-8221D0/24+57D0*D2+117D0*D3
5077       &      +(151D0/12 - 2D0*D2-  4D0*D3)*ANF
5078        COEFA3=-4544045D0/864+   1340*D2+118915D0/36*D3  -1270D0*D5
5079       &      +(71621D0/162  -209D0/2*D2   -216D0*D3+5D0*D4+55D0*D5)*ANF
5080       &      +(-13171D0/1944+ 16D0/9*D2  +26D0/9*D3             )*ANF**2
5081  *

5113        ALMC=LOG(RLMC)
5114        R4LC=RLMC**2*(13D0/3-ALMC-4D0*D3)*ALFSPI**2
5115        ALMB=LOG(RLMB)
5116        R4LB=RLMB**2*(13D0/3-ALMB-4D0*D3)*ALFSPI**2

5120        RV40=-6D0-22D0*ALFSPI+(+12D0-3173D0/12+27D0*PI2+112D0*D3
5121       &     +(143D0/18-2D0/3*PI2- 8D0/3*D3)*ANF)*ALFSPI**2
5122  *

5126        RA40=+6D0+10D0*ALFSPI+(-12D0+3533D0/12-27D0*PI2-220D0*D3
5127       &     +(-41D0/6 +2D0/3*PI2+16D0/3*D3)*ANF)*ALFSPI**2
5128        RV4L=(-11D0/2+1D0/3*ANF)*ALFSPI**2
5129        RA4L=(+77D0/2-7D0/3*ANF)*ALFSPI**2
```

```
133
134  void GSM::RadiatorFunctions::UpdateLocalFlags( GReference& /* ref */ )
135  {
136    m_isUpToDate_Update = kFALSE;
137  }
138
139  void GSM::RadiatorFunctions::Update()
140  {
141    if (m_isUpToDate_Update) return;
142
143    // now, it is uptodate (I mean... it will be)
144    m_isUpToDate_Update = kTRUE;
145
146    Double_t pi    = TMath::Pi();
147
148    Double_t MSMc  = GetQMassRun().GetRunningQuarkMass( GTypes::kCharm );
149    Double_t MSMb  = GetQMassRun().GetRunningQuarkMass( GTypes::kBottom );
150
151    m_asMZpi       = GetAlphasMZ()/pi;
152    m_aQEDMZpi     = (GConstants::alphaQED()/( 1.0 - GetDAlphaQED().DAlphaQEDMZt() ))/pi;
153
154    m_RatioMcMZ    = GMath::IPow( MSMc/p_MZ,2);
155    m_RatioMbMZ    = GMath::IPow( MSMb/p_MZ,2);
156    m_RatioMtMZ    = GMath::IPow( p_MZ/p_mt,2);
157
158    m_RatioMcMt    = GMath::IPow( MSMc/p_mt,2);
159    m_RatioMbMt    = GMath::IPow( MSMb/p_mt,2);
160
161    // light quark corrections
162    //   Quadratic corrections (mc and mb term)
163    m_LightQu2     = ( m_RatioMcMZ + m_RatioMbMZ )*m_CL3*GMath::IPow(m_asMZpi,3);
164
165    //   Quartic corrections (mc and mb term)
166    m_LightQu4Mc   = m_RatioMcMZ*m_RatioMcMZ*(m_C42 - TMath::Log(m_RatioMcMZ))*m_asMZpi*m_asMZpi;
167    m_LightQu4Mb   = m_RatioMbMZ*m_RatioMbMZ*(m_C42 - TMath::Log(m_RatioMbMZ))*m_asMZpi*m_asMZpi;
168
169    // Power suppressed top mass correction
170    m_Ct2   = m_RatioMtMZ*(44/675.0 - 2/135.0*TMath::Log(m_RatioMtMZ));
171
172    // Singlet axial corrections
173    m_CMt2 = -37/12.0 + TMath::Log(m_RatioMtMZ) + 7/81.0*m_RatioMtMZ + 0.0132*m_RatioMtMZ*m_RatioMtMZ;
174    // + 8/3.0 term not in publications => taken from dizet6_42.f line 5149
175    m_CMt3 = -5075/216.0 + 8/3.0 + 23/6.0*GMath::Zeta2() + GMath::Zeta3()
176       + 67/18.0*TMath::Log(m_RatioMtMZ) + 23/12.0*GMath::IPow(TMath::Log(m_RatioMtMZ),2);
```

Match 2

```
5156       QCDCON(2*IQ-1)=1+ALFSPI
5157      &                +1D0/4*ALQED/PI*CHARQU(IQ)**2*(3*ZQED-ALFSPI*ZMIX)
5158      &                +(1.40923D0+(44D0/675-2D0/135*ALMT)*RLMT)*ALFSPI**2
5159      &                +(-12.76706D0)*ALFSPI**3
5160      &+12*AMQRUN(IQ)**2/SQS**2*ALFSPI*(1+8.7D0*ALFSPI+45.15D0*ALFSPI**2)
```

```
5148       CAI2=-37D0/12+ALMT+7D0/81*RLMT+.0132D0*RLMT**2
5149       CAI3=-5651D0/216+8D0/3+23D0/6*D2+D3+67D0/18*ALMT+23D0/12*ALMT**2
```

Match 3

```
177
178     m_CMt4 = m_CMt3/m_CMt2*m_CMt3*(1.0 - p_DeltaAlphasTheoCMt3_Scale);
179
180     SetUpToDate();
181   }
182
183
184   // ---------------------- Radiator-functions for computing Z->qq ----------------------------
185
186   // vector part; Ve = Vector
187   // eq. (3.6.3) from  hep-ph/9908433
188   Double_t GSM::RadiatorFunctions::GetRVq( GTypes::Particle ParticleType, Double_t Charge )
189   {
190     Update();
191
192     // heavy quark corrections
193     //   Quadratic corrections (mq^2 term only for charm and bottom case)
194     Double_t VeHeavyQu2 = m_CV1*m_asMZpi + m_CV2*GMath::IPow(m_asMZpi,2);
195     if (!m_useNLOAlphasOnly) VeHeavyQu2 += m_CV3*GMath::IPow(m_asMZpi,3);
196
197
198     // Quartic corrections
199     //   particulary bottom corretions (mq^4 term only for charm and bottom case)
200
201     // m_CV40 term is missing in publications, taken from dizet6_42.f line 5128
202     Double_t VeHeavyQu4 = m_CV40 + m_CV41*m_asMZpi + m_CV42*GMath::IPow(m_asMZpi,2);
203     // this term in dizet6_42.f, but it is double counting see AxialQCD4
204     // !!! term makes no difference in result !!!
205     // VeHeavyQu4 += 12.*GMath::IPow(m_asMZpi,2);
206     Double_t Vector4    = m_CVL42*GMath::IPow(m_asMZpi,2);
207
208     // mc and mb terms
209     Double_t VectorQCD2 = m_LightQu2;
210     Double_t VectorQCD4 = m_LightQu4Mc + m_LightQu4Mb;
211
212     if( ParticleType == GTypes::kCharm ){
213       VectorQCD2 += m_RatioMcMZ * VeHeavyQu2;
214       VectorQCD4 += ( GMath::IPow(m_RatioMcMZ,2)*(VeHeavyQu4 + Vector4*TMath::Log(m_RatioMcMZ))
215                   + 12.0*GMath::IPow(m_RatioMbMZ*m_asMZpi,2) );
216     }
217     else if( ParticleType == GTypes::kBottom){
218       VectorQCD2 += m_RatioMbMZ * VeHeavyQu2;
219       VectorQCD4 += ( GMath::IPow(m_RatioMbMZ,2)*(VeHeavyQu4 + Vector4*TMath::Log(m_RatioMbMZ)) ||~ line 5136-5137
220                   + 12.0*GMath::IPow(m_RatioMcMZ*m_asMZpi,2)
```

```
5131        QCDC4V(1)=R4LC+R4LB
5132        QCDC4V(2)=R4LC+R4LB
5133        QCDC4V(3)=R4LC+R4LB+RLMC**2*(RV40+RV4L*ALMC)+12*RLMB**2*ALFSPI**2
5134        QCDC4V(4)=R4LC+R4LB
5135        QCDC4V(5)=0D0
5136        QCDC4V(6)=R4LC+R4LB+RLMB**2*(RV40+RV4L*ALMB)+12*RLMC**2*ALFSPI**2
5137       &                    -RLMB**3*(8D0+16D0/27*(155D0+6D0*ALMB)*ALFSPI)
```

Match 4

```
                - 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 + 3/4.*Charge*Charge*m_aQEDMZpi + m_asMZpi - 1/4.0*Charge*Charge*m_aQEDMZpi*m_asMZpi
          + (m_C02 + m_Ct2)*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
// eq. (3.6.4) from  hep-ph/9908433
Double_t GSM::RadiatorFunctions::GetRAq( GTypes::Particle ParticleType, Double_t Charge, Double_t T3 )
{
  Update();

  // heavy quark corrections
  //   Quadratic corrections (mq^2 term only for charm and bottom case)
  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 (mq^4 term only for charm and bottom case)
  Double_t AxHeavyQu4 = m_CA40 + m_CA41*m_asMZpi + m_CA42*GMath::IPow(m_asMZpi,2);
  // this term in dizet6_42.f, but it is double counting see AxialQCD4
  // AxHeavyQu4 += - 12.*GMath::IPow(m_asMZpi,2);
  // !!! term makes no difference in result !!!
  Double_t Axial4     = m_CAL42*GMath::IPow(m_asMZpi,2);

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

  // in publication: AxSingle = 6*m_RatioMbMZ*(3.0 + TMath::Log(m_RatioMtMZ)) ...
  // in publication: AxSingle = ... (8/81. + 1/54.) ...
  // differences corrected with dizet6_42.f
  if( ParticleType == GTypes::kCharm ){
    AxialQCD2 += m_RatioMcMZ * AxHeavyQu2;
    AxialQCD4 += ( GMath::IPow(m_RatioMcMZ,2)*(AxHeavyQu4 + Axial4*TMath::Log(m_RatioMcMZ))
```

```
265                 - 12.0*GMath::IPow(m_RatioMbMZ*m_asMZpi,2) );
266       AxSingle   = ( -6.0*m_RatioMcMZ*(-3.0 + TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi
267                 -10.0*m_RatioMcMt*(8/81.0 - 1/54.0*TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi );
268     }
269     else if( ParticleType == GTypes::kBottom ){
270       AxialQCD2 += m_RatioMbMZ * AxHeavyQu2;
271       AxialQCD4 += ( GMath::IPow(m_RatioMbMZ,2)*(AxHeavyQu4 + Axial4*TMath::Log(m_RatioMbMZ))
272                 - 12.0*GMath::IPow(m_RatioMcMZ*m_asMZpi,2) );
273       AxSingle   = ( -6.0*m_RatioMbMZ*(-3.0 + TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi
274                 -10.0*m_RatioMbMt*(8/81.0 - 1/54.0*TMath::Log(m_RatioMtMZ))*m_asMZpi*m_asMZpi );
275     }
276
277     Double_t RadA = ( 1.0 + 3/4.*Charge*Charge*m_aQEDMZpi + m_asMZpi - 1/4.0*Charge*Charge*m_aQEDMZpi*m_asMZpi
278                 + (m_C02 + m_Ct2 - 2.0*T3*m_CMt2)*GMath::IPow(m_asMZpi,2)
279                 + (m_useNLOAlphasOnly ? 0 : ( (m_C03 - 2.0*T3*m_CMt3)*GMath::IPow(m_asMZpi,3) +
280                                   (m_C04 - 2.0*T3*m_CMt4)*GMath::IPow(m_asMZpi,4) +
281                                   (m_C05)*GMath::IPow(m_asMZpi,5) ))
282                 + AxialQCD2 + AxialQCD4 + AxSingle);
283
284     return RadA;
285   }
```

Match 5

```
5178  * Axial singlet contributions
5179  *
5180        IF     (IQ.EQ.3) THEN
5181         RASING=-6D0*AMCRUN**2/SQS**2*(-3D0+ALMT)*ALFSPI**2
5182       &        -10D0*AMCRUN**2/AMT**2*(8D0/81-1D0/54*ALMT)*ALFSPI**2
5183        ELSEIF(IQ.EQ.6) THEN
5184         RASING=-6D0*AMBRUN**2/SQS**2*(-3D0+ALMT)*ALFSPI**2
5185       &        -10D0*AMBRUN**2/AMT**2*(8D0/81-1D0/54*ALMT)*ALFSPI**2
5186        ELSE
5187         RASING=0D0
5188        ENDIF
```