```
/*****************************************************************************
 * Package: GSM                                                              *
 * Class  : QCDCorrection                                                    *
 *                                                                           *
 * Description:                                                              *
 *      2 loopQCD Corrections to the effective weak couplings                *
 *      in perturbative QCD                                                  *
 * Sources:                                                                  *
 *      B. Kniehl, Nucl. Phys. B347 (1990) 86                                *
 *                                                                           *
 * The Code of this class has been ported to C++ from Fortran code authored  *
 *    by B. Kniehl that is also used in the Fortran package ZFITTER          *
 *                                                                           *
 *****************************************************************************/

#include "TMath.h"

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

#include "GSM/QCDCorrections.h"
#include "GSM/ZMath.h"

using namespace Gfitter;
using std::complex;

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

  BookParameter( "MZ"              , & p_MZ );
  BookParameter( "mt"              , & p_mt );
  BookTheory   ( "GSM::WMass"      , & t_MW );
}

void GSM::QCDCorrections::Initialise()
{
  m_cone  = 1.0;
```

Hinweis:

Kommentare mit Hinweisen auf ZFitter sind grün markiert

Übereinstimmungen sind gelb markiert.

Auffällige Stellen bzw. Anmerkungen sind violett markiert

Alle Matches in dieser Klasse stammen aus dem Fortran Paket bkqcdl5_14.f.

Anhang 10

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      m_zeta3 = GMath::Zeta3();
46    }
47
48    void GSM::QCDCorrections::UpdateLocalFlags( GReference& /* ref */ )
49    {
50      m_isUpToDate_Update = kFALSE;
51    }
52
53    void GSM::QCDCorrections::Update()
54    {
55      if (m_isUpToDate_Update) return;
56
57      // now, it is uptodate (I mean... it will be)
58      m_isUpToDate_Update = kTRUE;
59
60      m_kmQCD = KMQCD( p_MZ*p_MZ, GMath::IPow( GetMW(),2 ), p_mt*p_mt, p_MZ*p_MZ );
61      m_rmQCD = RMQCD( p_MZ*p_MZ, GMath::IPow( GetMW(),2 ), p_mt*p_mt, p_MZ*p_MZ );
62
63      SetUpToDate();
64    }
65
66    // ------------------------ additional part from ZFITTER Authors -----------------------------
67
68    // Correction to kappa_f
69    complex<Double_t> GSM::QCDCorrections::KMQCD( const Double_t& MZ2, const Double_t& MW2,
70                                                  const Double_t& mt2, const Double_t& S ) const
71    {
72      Double_t CW2 = MW2/MZ2;
73      Double_t SW2 = 1.0 - CW2;
74      Double_t QTM = 2/3.0;
75      Double_t QBM = 1/3.0;
76      complex<Double_t> XCR = MZ2/(4.0*mt2);
77      complex<Double_t> XCS = S/(4.0*mt2);
78      complex<Double_t> XCX = MW2/mt2;
79      Double_t VT2 = GMath::IPow(1.0 - 4.0*QTM*SW2,2);
80
81      complex<Double_t> kmqcd = ( CW2/(SW2*SW2)/4.0*mt2/MW2*(VT2*CV1(XCR) + CA1(XCR))
82                                  - CW2/(SW2*SW2)*mt2/MW2*CF1(XCX)
83                                  + 1.0/SW2*(1.0-4.0*QTM*SW2)*QTM*mt2/S*CV1(XCS)
84                                  - 1.0/(16.0*SW2*SW2)*(2.0*(1.0-2.0*QBM*SW2)*TMath::Log(MZ2/mt2)
85                                                        + 4.0*(1.0-4.0*QBM*SW2)*QBM*SW2*TMath::Log(S/MZ2)) );
86      return kmqcd;
87    }
```

Match 1

```
52        FUNCTION XKMQCD(AMZ2,AMW2,AMT2,S)
53    *
54        IMPLICIT REAL*8(A-W,Y-Z)
55        IMPLICIT COMPLEX*16(X)
56        COMPLEX*16 CV1,CA1,CF1
57    *
58        CW2 =AMW2/AMZ2
59        SW2 =1D0-CW2
60    *
61        QTM=2D0/3D0
62        QBM=1D0/3D0
63        XCR=DCMPLX(AMZ2/(4D0*AMT2))
64        XCS=DCMPLX( S  /(4D0*AMT2))
65        XCX=DCMPLX(AMW2/AMT2)
66    *
67        XKMQCD=CW2/SW2**2/4D0*AMT2/AMW2
68    &       *((1D0-4D0*QTM*SW2)**2*CV1(XCR)+CA1(XCR))
69    &       -CW2/SW2**2*AMT2/AMW2*CF1(XCX)
70    &       +1D0/SW2*(1D0-4D0*QTM*SW2)*QTM*AMT2/S*CV1(XCS)
71    &       -1D0/16D0/SW2**2*(2D0*(1D0-2D0*QBM*SW2)*LOG(AMZ2/AMT2)
72    &       +4D0*(1D0-4D0*QBM*SW2)*QBM*SW2*LOG(S/AMZ2))
73    *
74        END
```

Match 2

```
88
89  // Correction to rho_f
90  complex<Double_t> GSM::QCDCorrections::RMQCD( const Double_t& MZ2, const Double_t& MW2,
91                                                const Double_t& mt2, const Double_t& S ) const
92  {
93    Double_t CW2  = MW2/MZ2;
94    Double_t SW2  = 1.0 - CW2;
95    Double_t SMZ2 = S/MZ2;
96    Double_t DMZ2 = 1.0 - SMZ2;
97    Double_t QTM  = 2/3.0;
98    Double_t QBM  = 1/3.0;
99    Double_t D2 = 1.6449340668482;
100   Double_t D3 = 1.2020569031596;
101
102   Double_t VT2 = GMath::IPow(1.0 - 4.0*QTM*SW2,2);
103   Double_t VB2 = GMath::IPow(1.0 - 4.0*QBM*SW2,2);
104
105   complex<Double_t> rmqcd(0.0,0.0);
106
107   if (TMath::Abs(DMZ2) < 1e-3) {
108
109     Double_t ALTZ = -mt2/MZ2;
110
111     complex<Double_t> PVFTZ = ZMath::PolVfi(ALTZ);
112     complex<Double_t> PAFTZ = ZMath::PolAfi(ALTZ);
113
114     complex<Double_t> DVFTZ = ZMath::DPolVfi(ALTZ);
115     complex<Double_t> DAFTZ = ZMath::DPolAfi(ALTZ);
116     rmqcd = 1.0/(12.0*SW2)*( 3.0/(4.0*CW2)*(1.0 + VB2)
117                            + mt2/(4.0*MW2)*(VT2*(DVFTZ/ALTZ - PVFTZ)
118                                                + DAFTZ/ALTZ - PAFTZ)
119                            - mt2/MW2*(3.0*D2 +105/8.0) );
120   }
121   else {
122     complex<Double_t> XCR   = MZ2/(4.0*mt2);
123     complex<Double_t> XCS   = S/(4.0*mt2);
124     complex<Double_t> XV1r  = CV1(XCR);
125     complex<Double_t> XA1r  = CA1(XCR);
126     complex<Double_t> XV1rs = CV1(XCS);
127     complex<Double_t> XA1rs = CA1(XCS);
128
129     rmqcd = 1.0/(4.0*SW2*CW2)*( mt2/MZ2*(VT2*XV1r + XA1r)
```

```
76        FUNCTION XRMQCD(AMZ2,AMW2,AMT2,S)
77  *
78        IMPLICIT REAL*8(A-W,Y-Z)
79        IMPLICIT COMPLEX*16(X)
80        COMPLEX*16 CV1,CA1
81  *
82        DATA D2/1.6449340668482D0/
83        DATA D3/1.2020569031596D0/
84        DATA EPS/1D-3/
85  *
86        CW2 =AMW2/AMZ2
87        SW2 =1D0-CW2
88  *
89        QTM=2D0/3D0
90        QBM=1D0/3D0
91        XCR=DCMPLX(AMZ2/(4D0*AMT2))
92        XCS=DCMPLX( S  /(4D0*AMT2))
93  *
94        XV1r =CV1(XCR)
95        XA1r =CA1(XCR)
96        XV1rs=CV1(XCS)
97        XA1rs=CA1(XCS)
98  *
99        SMZ2=S/AMZ2
100       DMZ2=1.D0-SMZ2
101 *
102       IF(ABS(DMZ2).LT.EPS) THEN
103         VT =1D0-4D0*QTM*SW2
104         VB =1D0-4D0*QBM*SW2
105         VT2=VT**2
106         VB2=VB**2
107 *
108         ALTZ=-AMT2/AMZ2
109         ALTS=-AMT2/S
110         XPVFTS=XPVFI(ALTS)
111         XPVFTZ=XPVFI(ALTZ)
112         XPAFTS=XPAFI(ALTS)
113         XPAFTZ=XPAFI(ALTZ)
114 *
115         XDVFTZ=XDPVFI(ALTZ)
116         XDAFTZ=XDPAFI(ALTZ)
117         XRMQCD=1D0/(12D0*SW2)
118 &          *(3D0/4D0/CW2*(1D0+VB2)
119 &          +AMT2/4D0/AMW2*(VT2*(XDVFTZ/ALTZ-XPVFTZ)
120 &          +XDAFTZ/ALTZ-XPAFTZ)
121 &          -AMT2/AMW2*(3D0*D2+105D0/8D0))
122       ELSE
123         XRMQCD=1D0/4D0/SW2/CW2
124 &          *(AMT2/AMZ2*((1D0-4D0*QTM*SW2)**2*XV1r+XA1r)
125 &          +AMT2/(AMZ2-S)*((1-4*QTM*SW2)**2*(XV1rs-XV1r)+XA1rs-XA1r)
126 &          +2D0*AMT2/AMZ2*(-23D0/8D0+D2+3D0*D3)
127 &      -1D0/4D0*(1D0+(1D0-4D0*QBM*SW2)**2)*S/(AMZ2-S)*LOG(S/AMZ2))
128       ENDIF
129 *
130       END
```

```
                        + mt2/(MZ2-S)*(VT2*(XV1rs - XV1r) + XA1rs - XA1r)
                        + 2.0*mt2/MZ2*(-23.0/8.0*D2+3.0*D3)
                        - 1.0/4.0*(1.0 + QBM)*S/(MZ2-S)*TMath::Log(S/MZ2) );
  }

  return rmqcd;
}

// ----------------------- Kniehl's part starts here -----------------------------

// V_1(r) defined by Eq. (13) in Nucl. Phys. B347 (1990)
complex<Double_t> GSM::QCDCorrections::CV1( const complex<Double_t>& r ) const
{
  complex<Double_t> cv1(0.0,0.0);
  complex<Double_t> I  (0.0,1.0);

  if      (r == cv1) cv1 = 0.0;
  else if (r == m_cone) {
    m_logger << kWARNING <<  "Warning: Coulomb singularity of V_1(r) at r = 1 !" << GEndl ;
    cv1 = I*TMath::Pi()*TMath::Pi()*TMath::Pi();
  }
  else {
    complex<Double_t> C1  = ZMath::CSqrt(m_cone-r,-m_cone);
    complex<Double_t> C2  = ZMath::CSqrt(-r,-m_cone);
    complex<Double_t> CS  = C1/C2;
    complex<Double_t> CP  = C1 + C2;
    complex<Double_t> CM  = C1 - C2;
    complex<Double_t> CMS = CM*CM;
    complex<Double_t> CMQ = CMS*CMS;
    complex<Double_t> CF  = ZMath::Log(CP);
    complex<Double_t> CFS = CF*CF;
    complex<Double_t> CG  = ZMath::Log(CP-CM);
    complex<Double_t> CH  = ZMath::Log(CP+CM);
    complex<Double_t> CD  = ZMath::Li2(CMS,m_cone) - ZMath::Li2(CMQ,m_cone);

    cv1 = ( 4.0*(r - 1.0/(4.0*r))*( 2.0*ZMath::Li3(CMS,m_cone)
                                  - ZMath::Li3(CMQ,m_cone) + 8.0/3.0*CF*CD
                                  + 4.0*CFS*(-CF+CG/3.0 + 2.0/3.0*CH) )
          + CS*(8.0/3.0*(r + 1.0/2.0)*(CD + CF*(-3.0*CF+2.0*CG+4.0*CH))-2.0*(r+3.0/2.0)*CF)
          - 8.0*(r-1.0/6.0-7.0/(48.0*r))*CFS+13.0/6.0 + m_zeta3/r );
  }

  return cv1;
```

4,3,3,3

Match 3

```
      FUNCTION CV1(CR)
C     ****************
C V_1(r) defined by Eq. (10) in
C B.A. Kniehl, Nucl. Phys. B347 (1990) 86.
C r = (s + i*epsilon)/(4*m^2), where s may be complex.
C
      IMPLICIT LOGICAL (A-Z)
      COMPLEX*16 CR,CV1
      REAL*8 PI
      COMPLEX*16 C1,C2,CD,CF,CFS,CG,CH,CLI2,CLI3,CM,CMQ,CMS,CONE,CP,CRT,
     .           CS,CZERO,CZETA2,CZETA3
      DATA CZERO,CONE/(0.D0,0.D0),(1.D0,0.D0)/
      PI=4.D0*DATAN(1.D0)
      CZETA2=DCMPLX(PI**2/6.D0)
      CZETA3=CLI3(CONE,CONE)
      IF (CR.EQ.CZERO) THEN
          CV1=CZERO
      ELSE IF (CR.EQ.CONE) THEN
          CV1=DCMPLX(0.D0,PI**3)
          WRITE (6,*)
     .         'Warning: Coulomb singularity of V_1(r) at r = 1 !'
      ELSE
          C1=CRT(CONE-CR,-CONE)
          C2=CRT(-CR,-CONE)
          CS=C1/C2
          CP=C1+C2
          CM=C1-C2
          CMS=CM**2
          CMQ=CMS**2
          CF=CDLOG(CP)
          CFS=CF**2
          CG=CDLOG(CP-CM)
          CH=CDLOG(CP+CM)
          CD=CLI2(CMS,CONE)-CLI2(CMQ,CONE)
          CV1=4.D0*(CR-1.D0/(4.D0*CR))*(2.D0*CLI3(CMS,CONE)
     .          -CLI3(CMQ,CONE)+8.D0/3.D0*CF*CD
     .          +4.D0*CFS*(-CF+CG/3.D0+2.D0/3.D0*CH))
     .          +CS*(8.D0/3.D0*(CR+1.D0/2.D0)*(CD
     .          +CF*(-3.D0*CF+2.D0*CG+4.D0*CH))-2.D0*(CR+3.D0/2.D0)*CF)
     .          -8.D0*(CR-1.D0/6.D0-7.D0/(48.D0*CR))*CFS+13.D0/6.D0
     .          +CZETA3/CR
      END IF
      RETURN
C***********************************************************************
      END
```

Match 4

```
}

// A_1(r) defined by Eq. (14) in Nucl. Phys. B347 (1990)
complex<Double_t> GSM::QCDCorrections::CA1( const complex<Double_t>& r ) const
{

  complex<Double_t> ca1(0.0,0.0);

  if      (r == ca1)    ca1 = 3.0*(-2.0*m_zeta3 - GMath::Zeta2() + 7.0/4.0);
  else if (r == m_cone) ca1 = - 2.0*m_zeta3 -3.0/8.0*GMath::Zeta2() + 29.0/12.0;
  else {
    complex<Double_t> C1  = ZMath::CSqrt(m_cone-r,-m_cone);
    complex<Double_t> C2  = ZMath::CSqrt(-r,-m_cone);
    complex<Double_t> CS  = C1/C2;
    complex<Double_t> CP  = C1 + C2;
    complex<Double_t> CM  = C1 - C2;
    complex<Double_t> CMS = CM*CM;
    complex<Double_t> CMQ = CMS*CMS;
    complex<Double_t> CF  = ZMath::Log(CP);
    complex<Double_t> CFS = CF*CF;
    complex<Double_t> CG  = ZMath::Log(CP-CM);
    complex<Double_t> CH  = ZMath::Log(CP+CM);
    complex<Double_t> CD  = ZMath::Li2(CMS,m_cone) - ZMath::Li2(CMQ,m_cone);
    ca1 = ( 4.0*(r - 3.0/2.0 + 1.0/(2.0*r))*( 2.0*ZMath::Li3(CMS,m_cone)
                                    - ZMath::Li3(CMQ,m_cone) + 8.0/3.0*CF*CD
                                    + 4.0*CFS*(-CF+CG/3.0+2.0/3.0*CH) )
          + CS*(8.0/3.0*(r - 1.0)*(CD + CF*(-3.0*CF+2.0*CG+4.0*CH)) - 2.0*(r - 3.0 + 1.0/(4.0*r))*CF)
           - 8.0*(r - 11.0/12.0 + 5.0/(48.0*r) + 1.0/(32.0*r*r))*CFS
           - 3.0*GMath::Zeta2()+13.0/6.0+(-2.0*m_zeta3+1.0/4.0)/r );
  }

  return ca1;
}
```

```
      FUNCTION CA1(CR)
C     ****************
C A_1(r) defined by Eq. (11) in
C B.A. Kniehl, Nucl. Phys. B347 (1990) 86.
C r = (s + i*epsilon)/(4*m^2), where s may be complex.
C
      IMPLICIT LOGICAL (A-Z)
      COMPLEX*16 CR,CA1
      REAL*8 PI
      COMPLEX*16 C1,C2,CD,CF,CFS,CG,CH,CLI2,CLI3,CM,CMQ,CMS,CONE,CP,CRT,
     .           CS,CZERO,CZETA2,CZETA3
      DATA CZERO,CONE/(0.D0,0.D0),(1.D0,0.D0)/
      PI=4.D0*DATAN(1.D0)
      CZETA2=DCMPLX(PI**2/6.D0)
      CZETA3=CLI3(CONE,CONE)
      IF (CR.EQ.CZERO) THEN
          CA1=3.D0*(-2.D0*CZETA3-CZETA2+7.D0/4.D0)
      ELSE IF (CR.EQ.CONE) THEN
          CA1=-2.D0*CZETA3-3.D0/8.D0*CZETA2+29.D0/12.D0
      ELSE
          C1=CRT(CONE-CR,-CONE)
          C2=CRT(-CR,-CONE)
          CS=C1/C2
          CP=C1+C2
          CM=C1-C2
          CMS=CM**2
          CMQ=CMS**2
          CF=CDLOG(CP)
          CFS=CF**2
          CG=CDLOG(CP-CM)
          CH=CDLOG(CP+CM)
          CD=CLI2(CMS,CONE)-CLI2(CMQ,CONE)
          CA1=4.D0*(CR-3.D0/2.D0+1.D0/(2.D0*CR))*(2.D0*CLI3(CMS,CONE)
     .         -CLI3(CMQ,CONE)+8.D0/3.D0*CF*CD
     .         +4.D0*CFS*(-CF+CG/3.D0+2.D0/3.D0*CH))
     .         +CS*(8.D0/3.D0*(CR-1.D0)*(CD
     .         +CF*(-3.D0*CF+2.D0*CG+4.D0*CH))
     .         -2.D0*(CR-3.D0+1.D0/(4.D0*CR))*CF)
     .         -8.D0*(CR-11.D0/12.D0+5.D0/(48.D0*CR)
     .         +1.D0/(32.D0*CR**2))*CFS
     .         -3.D0*CZETA2+13.D0/6.D0+(-2.D0*CZETA3+1.D0/4.D0)/CR
      END IF
      RETURN
C***********************************************************************
      END
```

Match 5

```
208  // F_1(r) defined by Eq. (15) in Nucl. Phys. B347 (1990)
209  complex<Double_t> GSM::QCDCorrections::CF1( const complex<Double_t>& x ) const
210  {
211    complex<Double_t> cf1 (0.0,0.0);
212
213    if      (x == cf1)    cf1 = -3.0/2.0*m_zeta3 - GMath::Zeta2()/2.0 + 23.0/16.0;
214    else if (x == m_cone) cf1 = -m_zeta3/2.0-GMath::Zeta2()/12.0 + 7.0/8.0;
215    else {
216      complex<Double_t> xS  = x*x;
217      complex<Double_t> CA  = ZMath::CLog(-x,-m_cone);
218      complex<Double_t> CB  = ZMath::CLog(m_cone-x,-m_cone);
219      complex<Double_t> CBS = CB*CB;
220      complex<Double_t> CD  = ZMath::Li2(m_cone/(m_cone-x),m_cone);
221
222      cf1 = ( (x-3.0/2.0+1.0/(2.0*xS))*( ZMath::Li3(m_cone/(m_cone-x),m_cone)
223                                       + 2.0/3.0*CB*CD-CBS/6.0*(CA-CB) )
224            + (x+1.0/2.0-1.0/(2.0*x))/3.0*(CD-CA*CB)
225            + (x-1.0/8.0-1.0/x+5.0/(8.0*xS))/3.0*CBS
226            - (x-5.0/2.0+2.0/(3.0*x)+5.0/(6.0*xS))/4.0*CB
227            - 3.0/4.0*GMath::Zeta2()+13.0/12.0-5.0/(24.0*x)
228            - m_zeta3/(2.0*xS) );
229    }
230
231    return cf1;
232  }
233
```

```
226        FUNCTION CF1(CX)
227  C     ****************
228  C F_1(x) defined by Eq. (12) in
229  C B.A. Kniehl, Nucl. Phys. B347 (1990) 86.
230  C x = (s + i*epsilon)/m^2, where s may be complex.
231  C
232        IMPLICIT LOGICAL (A-Z)
233        COMPLEX*16 CX,CF1
234        REAL*8 PI
235        COMPLEX*16 CA,CB,CBS,CD,CLI2,CLI3,CLN,CXS,CONE,CZERO,CZETA2,CZETA3
236        DATA CZERO,CONE/(0.D0,0.D0),(1.D0,0.D0)/
237        PI=4.D0*DATAN(1.D0)
238        CZETA2=DCMPLX(PI**2/6.D0)
239        CZETA3=CLI3(CONE,CONE)
240        IF (CX.EQ.CZERO) THEN
241            CF1=-3.D0/2.D0*CZETA3-CZETA2/2.D0+23.D0/16.D0
242        ELSE IF (CX.EQ.CONE) THEN
243            CF1=-CZETA3/2.D0-CZETA2/12.D0+7.D0/8.D0
244        ELSE
245            CXS=CX**2
246            CA=CLN(-CX,-CONE)
247            CB=CLN(CONE-CX,-CONE)
248            CBS=CB**2
249            CD=CLI2(CONE/(CONE-CX),CONE)
250            CF1=(CX-3.D0/2.D0+1.D0/(2.D0*CXS))*(CLI3(CONE/(CONE-CX),CONE)
251           .         +2.D0/3.D0*CB*CD-CBS/6.D0*(CA-CB))
252           .         +(CX+1.D0/2.D0-1.D0/(2.D0*CX))/3.D0*(CD-CA*CB)
253           .         +(CX-1.D0/8.D0-1.D0/CX+5.D0/(8.D0*CXS))/3.D0*CBS
254           .         -(CX-5.D0/2.D0+2.D0/(3.D0*CX)+5.D0/(6.D0*CXS))/4.D0*CB
255           .         -3.D0/4.D0*CZETA2+13.D0/12.D0-5.D0/(24.D0*CX)
256           .         -CZETA3/(2.D0*CXS)
257            END IF
258        RETURN
259  C*******************************************************************
260        END
```