```
/*****************************************************************************
 * Package: GSM                                                              *
 * Class  : Vertex                                                           *
 *                                                                           *
 * Description:                                                              *
 *      Auxiliary Theory for Vertex corrections                              *
 *                                                                           *
 * Sources                                                                   *
 *      - Bardin et al, hep-ph/9709229v1,                                    *
 *      - Bardin et al, CPC. 133 (2001) 229, hep-ph/9908433                  *
 *      - 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 "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/ZMath.h"
#include "GSM/Vertex.h"

using namespace Gfitter;
using std::complex;

ClassImp(GSM::Vertex)

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

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

Hinweis:

Kommentare mit Hinweisen auf ZFitter sind grün markiert

Übereinstimmungen sind gelb markiert.

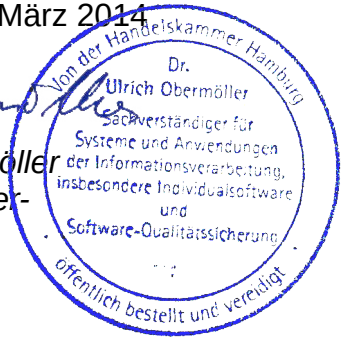
Auffällige Stellen bzw. Anmerkungen sind violett markiert

Anhang 6

zum Gutachten DESY ZFitter_GFitter vom 17.03.2014

Lübeck, den 17. März 2014

Dr. Ulrich Obermöller
-Sachverständiger-



```
43      BookTheory   ( "GSM::WMass" , & t_MW );
44    }
45
46    void GSM::Vertex::UpdateLocalFlags( GReference& /* ref */ )
47    {
48      m_isUpToDate_Update = kFALSE;
49    }
50
51    void GSM::Vertex::Update()
52    {
53      if (m_isUpToDate_Update) return;
54
55      // now, it is uptodate (I mean... it will be)
56      m_isUpToDate_Update = kTRUE;
57
58      Double_t MW      = GetMW();
59      Double_t MZ      = p_MZ;
60      Double_t mt      = p_mt;
61
62      Double_t mt2     = mt*mt;
63      Double_t MW2     = MW*MW;
64      Double_t MZ2     = MZ*MZ;
65      Double_t R       = MW2/MZ2;
66
67      Double_t Sqr     = TMath::Sqrt(4.0*R - 1.0);
68      Double_t Atan    = TMath::ATan(Sqr/(2.0*R - 1.0));
69      complex<Double_t>  I (0,1.0);
70
71      // Vertex functions in the limit of vanishing fermion masses
72      // for Z decay
73      // eq. hep-ph/9709229v1 (265)
74      // Spence( 1 ) = Li2( 1 ) = Zeta2
75      // see dizet6_42.f line 3311 & 3141
76      m_V1ZZ          = - 5.5 - 8.0*( GMath::Zeta2() - TMath::DiLog(2.0) ) + I*imag(V1b(-MZ2, MZ2));
77
78      // eq. hep-ph/9709229v1 (265)
79      // see also in hep-ph/9908433 (A.4.33)
80       // see dizet6_42.f line 3313 & 3142
81      m_V1ZW          = ( -3.5 - 2.0*R - (3.0 + 2.0*R)*TMath::Log(R) - 2.0*(1.0 + R)*(1.0 + R)
82                      *(GMath::Zeta2() - TMath::DiLog(1.0 + 1.0/R)) + I*imag(V1b(-MZ2,MW2)) );
83
84      // eq. hep-ph/9709229v1 (266)
```

Match 1

Match 2

```
3130  * Basic EW RHO and KAPPA
3131  *
3132        CH2=CH*CH
3133        W0A  =W0+W0F
3134        XZM1A =XZM1 +XZM1F
3135        XZFM1A=XZFM1+XZFM1F
3136        XWM1A =XWM1 +XWM1F
3137        XAMM1A=XAMM1+XAMM1F
3138  *
3139        V1ZIM=DIMAG(XV1B(-AMZ2,AMZ2))
3140        V1WIM=DIMAG(XV1B(-AMZ2,AMW2))
3141        XV1ZZ=DCMPLX(V1ZZ,V1ZIM)
3142        XV1ZW=DCMPLX(V1ZW,V1WIM)
```

```
3306  * Z-BOSON CHAIN ********************
3307  * FILLS CDZVZW (Z PART)
3308  *
3309  5     SR=SQRT(4.D0*R-1.D0)
3310        AT=ATAN(SR/(2.D0*R-1.D0))
3311        V1ZZ=-5.5D0-8.D0*(F1-SPENCE(2.D0))
3312        SPERR=SPENCE(1.D0+1.D0/R)
3313        V1ZW=-3.5D0-2.D0*R-(3.D0+2.D0*R)*ALR-2.D0*(1.D0+R)**2*(F1-SPERR)
3314        V2ZWW=2.D0/9.D0/R2+43.D0/18.D0/R-1.D0/6.D0-2.D0*R
3315       *     +(-1.D0/12.D0/R2-1.5D0/R+7.D0/3.D0+2.D0*R)*SR*AT
3316       *     -2.D0*R*(2.D0+R)*AT**2
```

```
85      // see also in hep-ph/9908433 (A.4.34)
86       // see dizet6_42.f line 3314
87      m_V2ZWW         = ( 2.0/(9.0*R*R) + 43.0/(18.0*R) - 1.0/6.0 - 2.0*R
88                    + (-1.0/(12.0*R*R) - 1.5/R + 7.0/3.0 + 2.0*R)*Sqr*Atan - 2.0*R*(2.0 + R)*Atan*Atan );
89
90      Double_t lam = GMath::IPow(MZ2,2) - 4.0*MW2*MZ2;
91
92      // -------- for W decay ----------
93      // see dizet6_42.f line 3333
94      m_V1WZ  = ( -5.0 - 2.0/R + (3.0 + 2.0/R)*TMath::Log(R)
95              -2.0*GMath::IPow((1.0+R)/R,2)*(GMath::Zeta2() - TMath::DiLog(1.0 + R)) );
96
97      // see dizet6_42.f line 3335
98      m_V2WWZ = ( -9/(4.0*R) -1/(12.0*R*R) + 23/18.0
99              + (0.5/R - 0.75/(R*R) - 1/(24.0*GMath::IPow(R,3)) + 1.0)*TMath::Log(R)
100             - real(ZMath::L(-MW2,MW2,MZ2))*(5/(6.0*R) + 1/(24.0*R*R) + 0.5)/MW2
101             + (0.5 + 1.0/R)*lam*GMath::IPow( real(ZMath::J(-MW2,MW2,MZ2)), 2 )
102             - (0.5 + 1.0/R)*GMath::IPow( TMath::Log(R), 2 ) );
103
104     // top quark additions
105     Double_t RtW      = mt2/MW2;
106     Double_t RtW1     = RtW - 1.0;
107
108     Double_t J0W = 0, S3W = 0, S30W = 0;
109     Double_t J0t = 0, S3t = 0, S30t = 0;
110
111     // s = -MZ2
112     ZMath::S3Wana( mt2, MW2, -MZ2, J0W, S3W, S30W);
113     ZMath::S3Wana( MW2, mt2, -MZ2, J0t, S3t, S30t);
114
115     // Information to this point in hep-ph/9709229 page 88-89,
116     // hep-ph/9908433 page 175-177
117     // and dizet6_42.f (ZFitter) line 3374-3393
118     Double_t WWv2  = ( - 2.0*R*(2.00 + R)*MZ2*(S3W - S30W) + RtW
119                   *((3.0*R*R + 2.5*R - 2.0 - (2.0*R - 0.5)*RtW +R*(0.5 - R)*RtW*RtW)*MZ2*S3W
120                    -(R + 1.00 - (0.5 - R)*RtW)*(J0W - 2.0) +(2.0*R + 3/(2.0*RtW1*RtW1)
121                                                - 2.0/RtW1 + 0.5 - (0.5 - R)*RtW)*TMath::Log(RtW)
122                   -(R + 3/(2.0*RtW1) + 0.75 - (0.5 - R)*RtW) + 0.25/R*(J0W - 3.0)*1.0) );
123     Double_t WWv11 = ( + 2.0*(1.0 + R)*(1.0 + R)*MZ2*(S3t-S30t) + (2.0*R + 3.0)*(J0t + TMath::Log(RtW) + TMath::Log(R))
124                   - RtW*( R*(3.0*R + 2.0 - RtW - R*RtW*RtW)*MZ2*S3t + (R + 0.5 + R*RtW)*(J0t + TMath::Log(RtW) - 2.0)
125                         - (2.00*R + 3/(2.0*RtW1*RtW1) - 2.0/RtW1 + 0.5 + R*RtW)*TMath::Log(RtW)
126                         + R + 3/(2.0*RtW1) + 5/4.0 + R*RtW ) );
```

Match 3

Match 4

Match 5

Match 6

```
3329  * W-BOSON CHAIN ********************
3330  * FILLS CDZVZW (W PART)
3331  *
3332  8     ALAM=AMZ2*AMZ2-4.D0*AMW2*AMZ2
3333        V1WZ=-5.D0-2.D0/R+(3.D0+2.D0/R)*ALR
3334  *     -2.D0*R1W2/R2*(SPENCE(1.D0)-SPENCE(R1W))
3335        V2WWZ=-9.D0/4.D0/R-1.D0/12.D0/R2+23.D0/18.D0
3336  *      +(1.D0/2.D0/R-3.D0/4.D0/R2
3337  *      -1.D0/24.D0/R/R2+1.D0)*ALR
3338  *      -DREAL(XL(-AMW2,AMW2,AMZ2))
3339  *      *(5.D0/6.D0/R+1.D0/24.D0/R2+1.D0/2.D0)/AMW2
3340  *      +(1.D0/2.D0+1.D0/R)*ALAM*DREAL(XJ(-AMW2,AMW2,AMZ2))
3341  *      *DREAL(XJ(-AMW2,AMW2,AMZ2))-(1.D0/2.D0+1.D0/R)*ALR*ALR
3342  9     CONTINUE
3343  *
3344        END
```

```
3374        WWv2 =-2D0*RWS*(2D0+RWS)*S*(S3w-S3w0)
3375  &       +RTW*((3D0*RWS**2+2.5D0*RWS-2D0-(2D0*RWS-.5D0)*RTW
3376  &            +RWS*(.5D0-RWS)*RTW**2)*S*S3w
3377  &            -(RWS+1D0-(.5D0-RWS)*RTW)*(AJ0w-2D0)
3378  &            +(2D0*RWS+3D0/2/RTW1**2-2D0/RTW1+1D0/2
3379  &            -(.5D0-RWS)*RTW)*ALRT
3380  &            -(RWS+3D0/2/RTW1+3D0/4-(.5D0-RWS)*RTW)
3381  &            +.25D0/RWS*(AJ0w-3D0)*NUNI
3382  &            )
3383        WWv11=+2D0*(1D0+RWS)**2*S*(S3t-S3t0)
3384  &       +(2D0*RWS+3D0)*(AJ0t+ALRT+LOG(RWS))
3385  &       -RTW*(RWS*(3D0*RWS+2D0-RTW-RWS*RTW**2)*S*S3t
3386  &            +(RWS+.5D0+RWS*RTW)*(AJ0t+ALRT-2D0)
3387  &            -(2D0*RWS+3D0/2/RTW1**2-2D0/RTW1+1D0/2+RWS*RTW)*ALRT
3388  &            +RWS+3D0/2/RTW1+5D0/4+RWS*RTW
3389  &            )
3390        WWv12=-RTW*(RWS*(2D0+RWS-2D0*RWS*RTW+RWS*RTW**2)*S*S3t
3391  &            -(.5D0-RWS+RWS*RTW)*(AJ0t+ALRT-1D0)+RWS*RTW*ALRT
3392  &            )
3393  *
```

```
127     Double_t WWv12 = ( -RtW*(R*(2.0 + R - 2.0*R*RtW + R*RtW*RtW)*MZ2*S3t
128                        -(0.5 - R + R*RtW)*(J0t + TMath::Log(RtW) - 1.0) + R*RtW*TMath::Log(RtW) ) );
129
130     // see dizet6_42.f line 3322                                    Match 7
131     Double_t v_tb  = 1;
132     m_Vtb          = v_tb*v_tb*( R*WWv2 - 0.5*(1.0 - 2.0*(1.0 - R)*(2/3.0))*WWv11 - 0.5*WWv12 );
133
134     // now parameters are up-to-date
135     SetUpToDate();
136   }
137
138
139   // hep-ph/9709229v1 (265) - (267)
140   // deal with the imaginary part
141     // see dizet6_42.f line 918-932
142   complex<Double_t> GSM::Vertex::V1b( Double_t Q2, Double_t MV2 )
143   {
144     Double_t r      = Q2/MV2;
145     Double_t Logr   = TMath::Log(TMath::Abs(r));
146     Double_t Rev1b  = ( -3.5 + 2.0/r + (3.0-2.0/r)*Logr + 2.0 * GMath::IPow(1.0-1.0/r,2) *
147                       (TMath::DiLog(1.0-r) - GMath::Zeta2()) );
148     Double_t Aiv1b = 0;                                            Match 8
149
150     if (Q2 < 0) Aiv1b = TMath::Pi()*(-3.0+2.0/r+2.0*GMath::IPow((1.0-1.0/r),2)*TMath::Log(1.0-r));
151     complex<Double_t> v1b(Rev1b,Aiv1b);
152
153     return v1b;
154   }
155
156
157
158
```

```
3318        IF(INDF.EQ.5) THEN
3319   *
3320         CALL VTBANA(1,AMZ2,WWv2,WWv11,WWv12)
3321         QBM=1D0/3D0
3322        VTB=V_TB**2*(R*WWv2-.5D0*(1D0-2D0*R1*(1D0-QBM))*WWv11-.5D0*WWv12)
3323   *
3324        ELSE
3325         VTB=0.D0
3326        ENDIF
3327        GO TO 9
```

```
918         FUNCTION XV1B(Q2,AMV2)
919    *
920         IMPLICIT REAL*8(A-H,O-W,Y-Z)
921         IMPLICIT COMPLEX*16(X)
922         COMMON/CDZCON/PI,PI2,F1,D3,ALFAI,AL4PI,AL2PI,AL1PI
923    *
924         AL=Q2/AMV2
925         ALA=LOG(ABS(AL))
926         REV1B=-3.5D0+2.D0/AL+(3.D0-2.D0/AL)*ALA
927    *       +2.D0*(1.D0-1.D0/AL)**2*(SPENCE(1.D0-AL)-F1)
928         AIV1B=0.D0
929         IF(Q2.LT.0.D0)
930    &AIV1B=PI*(-3.D0+2.D0/AL+2.D0*(1.D0-1.D0/AL)**2*LOG(1.D0-AL))
931         XV1B=DCMPLX(REV1B,AIV1B)
932         END
```