```
1   /*****************************************************************************
2    * Package: GSM                                                              *
3    * Class  : FermionPart                                                      *
4    *                                                                           *
5    * Description:                                                              *
6    *      Auxiliary Theory for ferminic part of self energies                  *
7    *      one loop core of ZFitter option                                      *
8    *                                                                           *
9    * Sources:                                                                  *
10   *      Bardin et al., hep-ph/9709229                                        *
11   *      Bardin et al, CPC. 133 (2001) 229, hep-ph/9908433                    *
12   *      Bardin et al., ZFitter package dizet6_42.f                           *
13   *                                                                           *
14   * Additional Information                                                    *
15   *      The Standard Model in the Making, Oxford 1999                        *
16   *      Nucl. Phys. B197 (1982) 1-44 / first summary of one loop core        *
17   *                                                                           *
18   * This class also contains code lines ported to C++ from the Fortran package *
19   * ZFITTER                                                                   *
20   *                                                                           *
21   *****************************************************************************/
22  #include <math.h>
23
24  #include "Riostream.h"
25  #include "TMath.h"
26
27  #include "Gfitter/GMath.h"
28  #include "Gfitter/GConstants.h"
29  #include "Gfitter/GTheory.h"
30  #include "Gfitter/GTheoryRef.h"
31  #include "Gfitter/GParameterRef.h"
32  #include "Gfitter/GReference.h"
33
34  #include "GSM/FermionPart.h"
35  #include "GSM/ZMath.h"
36
37  using std::complex;
38
39  using namespace Gfitter;
40  GSM::FermionPart::FermionPart()
41    : Gfitter::GAuxTheory(),
42      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 5

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

```
43  {
44    SetTheoryName( GetName() );
45    SetExistDerivative( kFALSE );
46
47    BookParameter( "MZ", &p_MZ );
48    BookParameter( "mt", &p_mt );
49
50    BookTheory   ( "GSM::WMass",        &t_MW );
51    BookTheory   ( "GSM::QPoleMass", &t_QPoleMass );
52  }
53
54  void GSM::FermionPart::UpdateLocalFlags( GReference& /* ref */ )
55  {
56    m_isUpToDate_Update = kFALSE;
57  }
58
59  void GSM::FermionPart::Update()
60  {
61    if (m_isUpToDate_Update) return;
62
63    // now, it is uptodate (I mean... it will be)
64    m_isUpToDate_Update = kTRUE;
65
66    m_MW2 = GMath::IPow( GetMW(), 2 );
67    m_MZ2 = p_MZ*p_MZ;
68
69    m_R   = m_MW2/m_MZ2;
70    m_R1  = 1.0 - m_R;
71    m_R12 = m_R1*m_R1;
72
73    // lepton masses
74    m_ml[0][0] = 0;
75    m_ml[0][1] = 0;
76    m_ml[0][2] = 0;
77    m_ml[1][0] = GConstants::me();
78    m_ml[1][1] = GConstants::mmu();
79    m_ml[1][2] = GConstants::mtau();
80
81    // quark masses
82    m_mq[0][0] = GetQPoleMass().GetPoleQMass( GTypes::kUp );
83    m_mq[0][1] = GetQPoleMass().GetPoleQMass( GTypes::kCharm );
84    m_mq[0][2] = p_mt;
```

```
85      m_mq[1][0] = GetQPoleMass().GetPoleQMass( GTypes::kDown );
86      m_mq[1][1] = GetQPoleMass().GetPoleQMass( GTypes::kStrange );
87      m_mq[1][2] = GetQPoleMass().GetPoleQMass( GTypes::kBottom );
88
89      // quark charges
90      m_chq[0] = GMath::TwoThird();
91      m_chq[1] = GMath::OneThird();
92
93      // member variables
94      // W boson self energies ( fermionic part )
95      m_WfAt0  = 0.0;
96      m_WfAtMW = 0.0;
97      // Z boson self energies ( fermionic part )
98      m_ZfAtMZ = 0.0;
99      m_ZfAt0  = 0.0;
100
101     // computation of fermionic part to self-energies
102     // for additional information see also ZFitter
103     // package dizet6_42.f line 1910-1960
104     for (Int_t i=0; i<3 ; i++) {
105
106       // massive leptons and quarks
107       Double_t ml2  = m_ml[1][i]*m_ml[1][i];
108       Double_t mu2  = m_mq[0][i]*m_mq[0][i];
109       Double_t md2  = m_mq[1][i]*m_mq[1][i];
110       Double_t RLepW  = ml2/m_MW2;
111       Double_t RUpW   = mu2/m_MW2;
112       Double_t RDownW = md2/m_MW2;
113       Double_t LLepW  = TMath::Log(RLepW);
114       Double_t LUpW   = TMath::Log(RUpW);
115       Double_t LDownW = TMath::Log(RDownW);
116
117       // eq. (262) of hep-ph/9709229
118       // W boson self-energy at 0 GeV
119       // neutrino part = 0
120       // see dizet6_42.f line 1919
121       m_WfAt0 = m_WfAt0 + 0.5*(RLepW*LLepW - RLepW/2.0);
122
123        // see dizet6_42.f line 1921-1926
124       if (RUpW != RDownW){
125         m_WfAt0 = ( m_WfAt0 + 3/2.0*((RUpW*RUpW*LUpW - RDownW*RDownW*LDownW)/(RUpW-RDownW)
126                                      -(RUpW+RDownW)/2.0) );
```

Match 1

```
1919        W0F=W0F+1D0/2D0*(RMLW*ALLW-RMLW/2D0)
1920        Z0F=Z0F+1D0/2D0*RMLW*ALLW
1921        IF(RMTW.NE.RMBW) THEN
1922         W0F=W0F+3D0/2D0*((RMTW**2*ALTW-RMBW**2*ALBW)/(RMTW-RMBW)
1923     *     -(RMTW+RMBW)/2D0)
1924        ELSE
1925         W0F=W0F+3D0*(RMTW*ALTW-RMTW/2D0)
1926        ENDIF
```

Match 1

```
127     }
128     else m_WfAt0 = m_WfAt0 + 3.0*(RUpW*LUpW - RUpW/2.0);
129
130     // eq. (261) of hep-ph/9709229
131     // Z boson self-energy at 0 GeV
132     // neutrino part = 0
133     // see dizet6_42.f line 1920
134     m_ZfAt0 = m_ZfAt0 + 0.5*RLepW*LLepW;
135     // see dizet6_42.f line 1927
136     m_ZfAt0 = m_ZfAt0 + 3.0/2.0*(RUpW*LUpW + RDownW*LDownW);
137
138     // eq. (262) of hep-ph/9709229
139     // W boson self-energy at MW
140     // neutrino part = 0
141     // see dizet6_42.f line 1929
142     m_WfAtMW = ( m_WfAtMW - 2.0*ZMath::I3(m_MW2, -m_MW2, ml2, 0.0)
143             + RLepW*ZMath::I1(m_MW2, -m_MW2, ml2, 0.0) );
144
145
146     // see dizet6_42.f line 1931
147     m_WfAtMW = ( m_WfAtMW - 6.0*ZMath::I3(m_MW2, -m_MW2, mu2, md2)
148             + 3.0*(RUpW*ZMath::I1(m_MW2, -m_MW2, mu2, md2)
149                 + RDownW*ZMath::I1(m_MW2, -m_MW2, md2, mu2)) );
150
151     // eq. (261) of hep-ph/9709229
152     // Z boson self-energy at MZ
153
154     // neutrino part
155     // eq. (261) of hep-ph/9709229 has changed, because masses are zero
156     complex<Double_t> clogWZ (TMath::Log(m_MW2/m_MZ2),TMath::Pi());
157
158     m_ZfAtMZ = m_ZfAtMZ + 1.0/(6.0*m_R)*(clogWZ + 5.0/3.0); ||~ Line 1899
159
160     // massive fermion part
161     // Born weak couplings
162     // see dizet6_42.f line 1935-1944
163     Double_t v2l = 1.0 + GMath::IPow(1.0-4.0*m_R1,2); ||~ Line 1936, 1939, 1940
164     Double_t v2t = 1.0 + GMath::IPow(1.0-4.0*m_R1*m_chq[0],2);
165     Double_t v2b = 1.0 + GMath::IPow(1.0-4.0*m_R1*m_chq[1],2);
166
167     m_ZfAtMZ = ( m_ZfAtMZ - 0.5*v2l/m_R*ZMath::I3(m_MW2, -m_MZ2, ml2, ml2)
168             + 0.5*RLepW*ZMath::I0(m_MW2, -m_MZ2, ml2, ml2) ); ||~ Line 1937/38
```

```
1899        XZM1F=XZM1F+1D0/R/6D0*(XALR+5D0/3D0)

1920        Z0F=Z0F+1D0/2D0*RMLW*ALLW
1921        IF(RMTW.NE.RMBW) THEN
1922         W0F=W0F+3D0/2D0*((RMTW**2*ALTW-RMBW**2*ALBW)/(RMTW-RMBW)
1923     *     -(RMTW+RMBW)/2D0)
1924        ELSE
1925         W0F=W0F+3D0*(RMTW*ALTW-RMTW/2D0)
1926        ENDIF
1927        Z0F=Z0F+3D0/2D0*(RMTW*ALTW+RMBW*ALBW)
1928     *
1929        XWM1F=XWM1F-2D0*XI3(AMW2MU,-AMW2,AML2,0D0 )
1930     *      +     RMLW*XI1(AMW2MU,-AMW2,AML2,0D0 )
1931        XWM1F=XWM1F-6D0*XI3(AMW2MU,-AMW2,AMT2,AMB2)
1932     *      +3D0*(RMTW*XI1(AMW2MU,-AMW2,AMT2,AMB2)
1933     *      +     RMBW*XI1(AMW2MU,-AMW2,AMB2,AMT2))
1934
1935     *
1936        V2PA2L=1D0+(1D0-4D0*R1*CLM(2*I  ))**2
1937        XZM1F=XZM1F-1D0/2D0*V2PA2L/R*XI3(AMW2MU,-AMZ2,AML2,AML2)
1938     *      +          1D0/2D0*RMLW*XI0(AMW2MU,-AMZ2,AML2,AML2)
1939        V2PA2T=1D0+(1D0-4D0*R1*CQM(2*I-1))**2
1940        V2PA2B=1D0+(1D0-4D0*R1*CQM(2*I  ))**2
1941        XZM1F=XZM1F-3D0/2D0*V2PA2T/R*XI3(AMW2MU,-AMZ2,AMT2,AMT2)
1942     *      +          3D0/2D0*RMTW*XI0(AMW2MU,-AMZ2,AMT2,AMT2)
1943        XZM1F=XZM1F-3D0/2D0*V2PA2B/R*XI3(AMW2MU,-AMZ2,AMB2,AMB2)
1944     *      +          3D0/2D0*RMBW*XI0(AMW2MU,-AMZ2,AMB2,AMB2)
1945     1   CONTINUE
1946     *
1947     *   DERIVATIVES, USED ONLY IN FORMFACTORS AND PARTIAL WIDTHS
1948     *
```

```
169       m_ZfAtMZ = ( m_ZfAtMZ - 3.0/2.0*v2t/m_R*ZMath::I3(m_MW2, -m_MZ2, mu2, mu2)
170                    + 3.0/2.0*RUpW*ZMath::I0(m_MW2, -m_MZ2, mu2, mu2) ); ||~ Line 1941/42
171       m_ZfAtMZ = ( m_ZfAtMZ - 3.0/2.0*v2b/m_R*ZMath::I3(m_MW2, -m_MZ2, md2, md2)
172                    + 3.0/2.0*RDownW*ZMath::I0(m_MW2, -m_MZ2, md2, md2) ); ||~ Line 1943/44
173     }
174
175     // derivatives of self energies / F = finite
176     m_ZfFAtMZ     = DerivativeZF( -m_MZ2 ); ||~ Line 1950
177     m_WfFAtMW     = DerivativeWF( -m_MW2 ); ||~ Line 1949
178
179     // photon Z mixing function
180     m_MfPhoZAtMZ  = MfphoZ( -m_MZ2 ); ||~ Line 1951
181
182     SetUpToDate();
183   }
184
185   // eq. (264) of hep-ph/9709229v1
186   // see ZFitter package dizet6_42.f line 1329-1352 ( function XAMF )
187   complex<Double_t> GSM::FermionPart::MfphoZ( const Double_t& Q2 ) const
188   {
189     complex<Double_t> chmQ1 (0,0);
190
191     // leptons
192     for (Int_t i=0; i<3 ; i++) {
193       Double_t ml2 = m_ml[1][i]*m_ml[1][i];
194       chmQ1 = chmQ1 + ZMath::I3(m_MW2, Q2, ml2, ml2);
195     }
196
197     complex<Double_t> chmQ2 = chmQ1;
198
199     // quarks
200     for (Int_t i=0; i<3; i++) {
201       for (Int_t j=0; j<2; j++) {
202         Double_t mq2 = m_mq[j][i]*m_mq[j][i];
203         chmQ1  = chmQ1 + 3.0*m_chq[j]*ZMath::I3(m_MW2, Q2, mq2, mq2);
204         chmQ2  = chmQ2 + 3.0*m_chq[j]*m_chq[j]*ZMath::I3(m_MW2, Q2, mq2, mq2);
205       }
206     }
207     return 8.0*m_R1*chmQ2 - 2.0*chmQ1;
208   }
209
210   // see ZFitter package dizet6_42.f line 1249-1279 ( function XDWF)
```

Match 1

```
1941        XZM1F=XZM1F-3D0/2D0*V2PA2T/R*XI3(AMW2MU,-AMZ2,AMT2,AMT2)
1942       *     +           3D0/2D0*RMTW*XI0(AMW2MU,-AMZ2,AMT2,AMT2)
1943        XZM1F=XZM1F-3D0/2D0*V2PA2B/R*XI3(AMW2MU,-AMZ2,AMB2,AMB2)
1944       *     +           3D0/2D0*RMBW*XI0(AMW2MU,-AMZ2,AMB2,AMB2)
1945     1  CONTINUE
1946       *
1947       *     DERIVATIVES, USED ONLY IN FORMFACTORS AND PARTIAL WIDTHS
1948       *
1949        XWFM1F=XDWF(-AMW2)
1950        XZFM1F=XDZF(-AMZ2)
1951        XAMM1F=XAMF(-AMZ2)
```

Match 2

```
1329        FUNCTION XAMF(Q2)
1330   *
1331        IMPLICIT REAL*8(A-H,O-W,Y-Z)
1332        IMPLICIT COMPLEX*16(X)
1333        COMMON/CDZWSM/AMW2,AMZ2,R,R1,R12,R2,AMH2,RW,RW1,RW12,RW2,RZ,RZ1,
1334       *      RZ12,RZ2,ALR,ALRW,ALRZ,SW2M,CW2M,AKSX,R1W,R1W2
1335        COMMON/CDZFER/CLM(8),AML(8),CQM(8),AMQ(8),VB,VT,VB2,VB2T,VT2,VT2T
1336   *
1337        NG=3
1338        XCHMQ1=DCMPLX(0.D0,0.D0)
1339        DO 1 I=1,NG
1340        AML2=AML(2*I)**2
1341        XCHMQ1=XCHMQ1+XI3(AMW2,Q2,AML2,AML2)
1342     1  CONTINUE
1343        XCHQ21=XCHMQ1
1344   * equal for leptons
1345        INQ=2*NG
1346        DO 2 I=1,INQ
1347        AMQ2=AMQ(I)**2
1348        XCHMQ1=XCHMQ1+3.D0*CQM(I)*XI3(AMW2,Q2,AMQ2,AMQ2)
1349        XCHQ21=XCHQ21+3.D0*CQM(I)**2*XI3(AMW2,Q2,AMQ2,AMQ2)
1350     2  CONTINUE
1351        XAMF=8.D0*R1*XCHQ21-2.D0*XCHMQ1
1352   *
```

```
211  complex<Double_t> GSM::FermionPart::DerivativeWF( const Double_t& Q2 ) const
212  {
213     complex<Double_t> SumI3   (0,0); ||~line 1260-1262
214     complex<Double_t> SumDI1u (0,0);
215     complex<Double_t> SumDI1d (0,0);
216
217     // massive leptons
218     for (Int_t i=0; i < 3 ; i++) {
219        Double_t ml2 = m_ml[1][i]*m_ml[1][i];
220        SumI3   = SumI3 + ZMath::I3(m_MW2, Q2, ml2, 0.0) - ZMath::DI3(Q2, m_MW2, ml2, 0, m_MW2);
221        SumDI1d = SumDI1d + ml2/m_MW2*ZMath::DI1(Q2, m_MW2, ml2, 0.0, m_MW2);
222     }
223
224     // quarks
225     for (Int_t i=0; i < 3; i++) {
226        Double_t mu2 = m_mq[0][i]*m_mq[0][i];
227        Double_t md2 = m_mq[1][i]*m_mq[1][i];
228        SumI3   = SumI3 + 3.0*(ZMath::I3(m_MW2, Q2, md2, mu2) - ZMath::DI3(Q2, m_MW2, mu2, md2, m_MW2));
229        SumDI1u = SumDI1u + 3.0*mu2/m_MW2*ZMath::DI1(Q2, m_MW2, mu2, md2, m_MW2);
230        SumDI1d = SumDI1d + 3.0*md2/m_MW2*ZMath::DI1(Q2, m_MW2, md2, mu2, m_MW2);
231     }
232
233     return 2.0*SumI3 + SumDI1u + SumDI1d;
234  }
235
236  // see hep-ph/9709229v1 page 87-90
237  // hep-ph/9908433 page 154-157
238  // and ZFitter package dizet6_42.f line 1281-1327 ( function XDZF)
239  complex<Double_t> GSM::FermionPart::DerivativeZF( const Double_t& Q2 ) const
240  {
241     Double_t LogQZ  = TMath::Log(TMath::Abs(Q2/m_MZ2)); ||~line 1292
242     Double_t QZ     = m_MZ2 + Q2;
243     Double_t il1    = 0;
244     Double_t il2    = 0;
245     if (Q2 < 0) il1 = -TMath::Pi(); ||~line 1295/96
246     if (Q2 > 0) il2 = -TMath::Pi();
247
248     complex<Double_t> l1 ( LogQZ, il1 ); ||~line 1297/98
249     complex<Double_t> l2 ( LogQZ, il2 );
250
251     complex<Double_t> lq (0,0);
252     complex<Double_t> SumNu(0,0);
```

Match 3

Match 4

```
1249        FUNCTION XDWF(Q2)
1250   *
1251        IMPLICIT REAL*8(A-H,O-W,Y-Z)
1252        IMPLICIT COMPLEX*16(X)
1253        COMMON/CDZWSM/AMW2,AMZ2,R,R1,R12,R2,AMH2,RW,RW1,RW12,RW2,RZ,RZ1,
1254       *      RZ12,RZ2,ALR,ALRW,ALRZ,SW2M,CW2M,AKSX,R1W,R1W2
1255        COMMON/CDZFER/CLM(8),AML(8),CQM(8),AMQ(8),VB,VT,VB2,VB2T,VT2,VT2T
1256        COMMON/CDZWSC/SL2,SQ2,W0,W0F,Z0,Z0F,DWZ0R1,DWZ0F,XWM1,XWM1F,XZM1,
1257       &      XZM1F,XWZ1R1,XDWZ1F,XZFM1,XZFM1F,XAMM1,XAMM1F,XWFM1,XWFM1F
1258   *
1259        NG=3
1260        XSI3  =DCMPLX(0D0,0D0)
1261        XSDI1U=DCMPLX(0D0,0D0)
1262        XSDI1D=DCMPLX(0D0,0D0)
1263        DO 1 I=1,NG
1264        AML2=AML(2*I)**2
1265        XI31=XI3 (AMW2,Q2,AML2,0D0)
1266        XI32=XDI3(Q2,AMW2,AML2,0D0)
1267        XSI3=XSI3+XI3(AMW2,Q2,AML2,0D0)-XDI3(Q2,AMW2,AML2,0D0)
1268        XSDI1D=XSDI1D+AML2/AMW2*XDI1(Q2,AMW2,AML2,0D0)
1269      1 CONTINUE
1270        DO 2 I=1,NG
1271        AMU2=AMQ(2*I-1)**2
1272        AMD2=AMQ(2*I  )**2
1273        XSI3=XSI3+3D0*(XI3(AMW2,Q2,AMU2,AMD2)-XDI3(Q2,AMW2,AMU2,AMD2))
1274        XSDI1U=XSDI1U+3D0*AMU2/AMW2*XDI1(Q2,AMW2,AMU2,AMD2)
1275        XSDI1D=XSDI1D+3D0*AMD2/AMW2*XDI1(Q2,AMW2,AMD2,AMU2)
1276      2 CONTINUE
1277        XDWF=2D0*XSI3+XSDI1U+XSDI1D
1278   *
1279        END
```

Match 4

```
253     if (TMath::Abs(QZ) > 1e-3) lq = l1 - m_MZ2/(m_MZ2+Q2)*l2; ||~line 1302
254     else                       lq = l1 + 1.0 + QZ/2.0 + QZ*QZ/3.0;
255
256     // massless neutrinos
257     SumNu = 3.0/(6.0*m_R)*(-5.0/3.0 - TMath::Log(m_R) + lq); ||~line 1303
258
259     complex<Double_t> SumI3 (0,0); ||~line 1304-1307
260     complex<Double_t> SumQMI3 (0,0);
261     complex<Double_t> SumQ2I3 (0,0);
262     complex<Double_t> SumDI0 (0,0);
263
264     // massive leptons
265     for (Int_t i=0; i<3 ; i++) {
266       Double_t ml2 = m_ml[1][i]*m_ml[1][i];
267       SumI3   = SumI3 + ZMath::I3(m_MW2, Q2, ml2, ml2) - ZMath::DI3(Q2, m_MZ2, ml2, ml2, m_MW2);
268       SumDI0  = SumDI0 + ml2/m_MW2*ZMath::DI0(Q2, m_MZ2, ml2, ml2);
269     }
270
271     SumQMI3 = SumI3; ||~line 1313-1314
272     SumQ2I3 = SumI3;
273     // quarks
274     for (Int_t i=0; i<3; i++) {
275       for (Int_t j=0; j<=1;j++) {
276         Double_t mq2 = m_mq[j][i]*m_mq[j][i];
277         SumI3   = SumI3 + 3.0*(ZMath::I3(m_MW2, Q2, mq2, mq2) - ZMath::DI3(Q2, m_MZ2, mq2, mq2, m_MW2));
278         SumQMI3 = SumQMI3 + 3.0*m_chq[j]*(ZMath::I3(m_MW2, Q2, mq2, mq2)
279                                           - ZMath::DI3(Q2, m_MZ2, mq2, mq2, m_MW2));
280         SumQ2I3 = SumQ2I3 + 3.0*m_chq[j]*m_chq[j]*(ZMath::I3(m_MW2, Q2, mq2, mq2)
281                                                    - ZMath::DI3(Q2, m_MZ2, mq2, mq2, m_MW2));
282         SumDI0  = SumDI0 + 3.0*mq2/m_MW2*ZMath::DI0(Q2, m_MZ2, mq2, mq2);
283       }
284     }
285
286     return (8.0*m_R12*SumQ2I3 - 4.0*m_R1*SumQMI3 + SumI3)/m_R + SumDI0/2.0 + SumNu;
287   }
```

```
1281        FUNCTION XDZF(Q2)
1282    *
1283        IMPLICIT REAL*8(A-H,O-W,Y-Z)
1284        IMPLICIT COMPLEX*16(X)
1285        COMMON/CDZWSM/AMW2,AMZ2,R,R1,R12,R2,AMH2,RW,RW1,RW12,RW2,RZ,RZ1,
1286    *       RZ12,RZ2,ALR,ALRW,ALRZ,SW2M,CW2M,AKSX,R1W,R1W2
1287        COMMON/CDZFER/CLM(8),AML(8),CQM(8),AMQ(8),VB,VT,VB2,VB2T,VT2,VT2T
1288        COMMON/CDZCON/PI,PI2,F1,D3,ALFAI,AL4PI,AL2PI,AL1PI
1289        DATA EPS/1.D-3/
1290    *
1291        NG=3
1292        ALQ=LOG(ABS(Q2/AMZ2))
1293        AIL1=0.D0
1294        AIL2=0.D0
1295        IF(Q2.LT.0.D0) AIL1=-PI
1296        IF(Q2.GT.0.D0) AIL2=-PI
1297        XL1=DCMPLX(ALQ,AIL1)
1298        XL2=DCMPLX(ALQ,AIL2)
1299        QD=AMZ2+Q2
1300        RQD=QD/AMZ2
1301        XLQ=XL1+1.D0+RQD/2.D0+RQD*RQD/3.D0
1302        IF(DABS(RQD).GT.EPS) XLQ=XL1-AMZ2/(AMZ2+Q2)*XL2
1303        XSNU=NG/R/6.D0*(-5.D0/3.D0-ALR+XLQ)
1304        XSI3=DCMPLX(0.D0,0.D0)
1305        XSQMI3=DCMPLX(0.D0,0.D0)
1306        XSQ2I3=DCMPLX(0.D0,0.D0)
1307        XSDI0=DCMPLX(0.D0,0.D0)
1308        DO 1 I=1,NG
1309        AML2=AML(2*I)**2
1310        XSI3=XSI3+XI3(AMW2,Q2,AML2,AML2)-XDI3(Q2,AMZ2,AML2,AML2)
1311        XSDI0=XSDI0+AML2/AMW2*XDI0(Q2,AMZ2,AML2,AML2)
1312    1   CONTINUE
1313        XSQMI3=XSI3
1314        XSQ2I3=XSI3
1315        INQ=2*NG
1316        DO 2 I=1,INQ
1317        AMQ2=AMQ(I)**2
1318        XSI3=XSI3+3.D0*(XI3(AMW2,Q2,AMQ2,AMQ2)-XDI3(Q2,AMZ2,AMQ2,AMQ2))
1319        XSQMI3=XSQMI3+3.D0*CQM(I)*(XI3(AMW2,Q2,AMQ2,AMQ2)
1320    *       -XDI3(Q2,AMZ2,AMQ2,AMQ2))
1321        XSQ2I3=XSQ2I3+3.D0*CQM(I)**2*(XI3(AMW2,Q2,AMQ2,AMQ2)
1322    *       -XDI3(Q2,AMZ2,AMQ2,AMQ2))
1323        XSDI0=XSDI0+3.D0*AMQ2/AMW2*XDI0(Q2,AMZ2,AMQ2,AMQ2)
1324    2   CONTINUE
1325        XDZF=(8.D0*R12*XSQ2I3-4.D0*R1*XSQMI3+XSI3)/R+XSDI0/2.D0+XSNU
1326    *
1327        END
```