```
/*****************************************************************************
 * Package: GSM                                                              *
 * Class  : Z0ZFitter                                                        *
 *                                                                           *
 * Description:                                                              *
 *      Auxiliary Theory of the ZFitter option using the OMS approach        *
 *      Computes effective weak mixing angle and Partial Z widths            *
 *                                                                           *
 * Sources:                                                                  *
 *      - Bardin et al, hep-ph/9709229v1                                     *
 *      - Bardin et al, hep-ph/9412201v3                                     *
 *      - Bardin et al,,Comput.Phys.Commun. 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 "Gfitter/GVariable.h"
#include "Gfitter/GStore.h"

#include "GSM/Z0ZFitter.h"
#include "GSM/ZMath.h"

#include "GSM/MW.h"

using namespace Gfitter;
using std::complex;
using namespace Gfitter::GTypes;

ClassImp(GSM::Z0ZFitter)

GSM::Z0ZFitter::Z0ZFitter()
  : Z0Base(),
```

Hinweis:

Kommentare mit Hinweisen auf ZFitter sind grün markiert

Übereinstimmungen sind gelb markiert.

Auffällige Stellen bzw. Anmerkungen sind violett markiert

Anhang 7

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        m_isUpToDate_Update( kFALSE )
44    {
45      SetTheoryName( GetName() );
46      SetExistDerivative( kFALSE );
47
48      const TString& omstype = gStore()->GetVariable( "GSMFlags::OMSType" )->GetStringValue();
49      m_logger << kINFO << "Using OMS type: \"" << omstype << "\"" << GEndl;
50
51      if      (omstype == "OMS1" )  m_OMSType  = OMS1;
52      else if (omstype == "OMS1_2") m_OMSType  = OMS1_2;
53      else if (omstype == "OMS2")   m_OMSType  = OMS2;
54      else {
55        m_logger << kFATAL << "unknown value for \"GSMFlags::OMStype\": \"" << omstype << "\""
56                 << ". Possible are: \"OMS1\", \"OMS1-2\" \"OMS2\""
57                 << GEndl;
58      }
59
60      const TString& logMH = gStore()->GetVariable( "GSMFlags::logMH" )->GetStringValue();
61      m_logger << kINFO << "Using logMH: \"" << logMH << "\"" << GEndl;
62
63      if      (logMH == "Yes" ) m_logMH  = kTRUE;
64      else if (logMH == "No" )  m_logMH  = kFALSE;
65      else {
66        m_logger << kFATAL << "unknown value for \"GSMFlags::logMH\": \"" << logMH << "\""
67                 << ". Possible are: \"Yes\" and \"No\"\""
68                 << GEndl;
69      }
70
71      BookParameter( "DeltaOMS"                 , &p_DeltaOMS );
72
73      BookTheory   ( "GSM::Vertex"              , &t_Vertex );
74      BookTheory   ( "GSM::QCDCorrections"      , &t_QCDCorr );
75      BookTheory   ( "GSM::FermionPart"         , &t_FermionPart );
76      BookTheory   ( "GSM::BosonPart"           , &t_BosonPart );
77      BookTheory   ( "GSM::EW2Loop"             , &t_EW2Loop );
78      BookTheory   ( "GSM::RadiatorFunctions"   , &t_radFun );
79
80      BookTheory   ( "GSM::RunningAlphaQCD"     , &t_AlphasRun );
81      BookTheory   ( "GSM::WMass"               , &t_MW );
82    }
83
84    void GSM::Z0ZFitter::Initialise()
```

```
85  {
86    //non-factorizable EW*QCD corrections
87    m_EWQCD[0] = -0.113e-3;
88    m_EWQCD[1] = -0.160e-3;
89    m_EWQCD[2] =  m_EWQCD[0];
90    m_EWQCD[3] =  m_EWQCD[1];
91    m_EWQCD[4] = -0.040e-3;
92  }
93
94  void GSM::Z0ZFitter::UpdateLocalFlags( GReference& /* ref */ )
95  {
96    m_isUpToDate_Update = kFALSE;
97  }
98
99  void GSM::Z0ZFitter::Update()
100 {
101   if (m_isUpToDate_Update) return;
102
103   // now, it is uptodate (I mean... it will be)
104   m_isUpToDate_Update = kTRUE;
105
106   Double_t pi        = TMath::Pi();
107
108   Double_t MW        = GetMW();
109   Double_t MZ        = p_MZ;
110   Double_t mt        = p_mt;
111   Double_t MH        = GetMH().GetValue();
112   if (m_logMH) MH = TMath::Exp( GetMH().GetValue() );
113
114   // alphas at MZ and mtop
115   Double_t alphasMZ  = GetAlphasRun().EvolveAlphas(p_MZ);;
116
117   if (TMath::IsNaN(p_mt)) m_logger << kFATAL << "<Z0ZFitter::Update> p_mt is NaN !" << GEndl;
118   Double_t alphasTop = GetAlphasRun().EvolveAlphas(p_mt);;
119
120   m_R = MW*MW/(MZ*MZ);
121
122   // leading order contribution
123   // W and Z self-energies at different scales
124   // MPhoZAtMZ = photon Z mixing function
125   // bosonic + ferminoic part
```

```
126     // see dizet6_42.f line 3133-3137
127     m_WAtMW      =   GetBosonPart().GetWbAtMW()     + GetFermionPart().GetWfAtMW();
128     m_ZFAtMZ     =   GetBosonPart().GetZbFAtMZ()    + GetFermionPart().GetZfFAtMZ();
129     m_WAt0       =   GetBosonPart().GetWbAt0()      + GetFermionPart().GetWfAt0();
130     m_MPhoZAtMZ  =   GetBosonPart().GetMbPhoZAtMZ() + GetFermionPart().GetMfPhoZAtMZ();
131     m_ZAtMZ      = ( GetBosonPart().GetZbAtMZ()     + GetFermionPart().GetZfAtMZ() - 1/(4.0*pi)
132                    * GConstants::alphaQED()*pow( GetFermionPart().GetMfPhoZAtMZ(), 2 ) );
133
134     // QCD corrections
135     // see dizet6_42 line 1858 & 1859
136     Double_t v2u = GMath::IPow(1.0-4.0*2/3.0*(1.0-m_R),2);
137     Double_t v2d = GMath::IPow(1.0-4.0*1/3.0*(1.0-m_R),2);
138
139     // light and heavy quarks
140     // (3.415) of hep-ph/9908433
141     // The Standard Model in the Making page 413-414
142     // see dizet6_42.f line 3181
143     m_rhoQCD     = ( GConstants::alphaQED()/(1.0*pi)
144                    * alphasTop/pi*real( GetQCDCorr().GetRMQCD() )
145                    + GConstants::alphaQED()/(1.0*pi)
146                    * alphasMZ/(8.0*pi*m_R*(1.0-m_R))
147                    *(v2u+v2d+2.0) );
148
149     // (3.416) of hep-ph/9908433
150     // The Standard Model in the Making page 413-414
151     // see dizet6_42.f line 3183
152     m_kapQCD     = ( GConstants::alphaQED()/(1.0*pi)
153                    * alphasTop/pi*real( GetQCDCorr().GetKMQCD() )
154                    + GConstants::alphaQED()/(1.0*pi)
155                    * alphasMZ/pi
156                    * m_R/(2.0*(1.0-m_R)*(1.0-m_R))*TMath::Log(m_R) );
157
158
159     // additional corrections (reminders)
160     // see dizet6_42.f line 3050
161     Double_t WZCorr  = m_R/(1.0-m_R)*real( m_WAtMW - m_ZAtMZ )/(1.0 -m_R);
162     // see dizet6_42.f line 3052
163     Double_t Scale   = GConstants::alphaQED()/(4*pi*(1.0-m_R))*(41/6.0 - 11/3.0*m_R)*TMath::Log(m_R);
164     // see dizet6_42.f line 3055
165     Double_t TopCon  = p_GF*mt*mt/( 8.0*GMath::Sqrt2()*pi*pi );
166     // see dizet6_42.f line 3055 ||~ bzw. line 3056???
167     Double_t RhoTop4 = 3.0*TopCon*(1.0 + TopCon*ZMath::BarbMc( MH/mt ));
168     // see dizet6_42.f line 3064
169     Double_t TBCor0  = 3.0*TopCon*ZMath::Afmt3( alphasTop, mt*mt, MZ*MZ, (1.0-m_R) );
```

Match 1

```
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
```

Match 2

```
1858  * VECTOR COUPLINGS FOR QCDCOR
1859        VB=1D0-4D0*CQM(6)*R1
1860        VT=1D0-4D0*CQM(5)*R1

3174        IF     (IQCD.EQ.1) THEN
3175          ROQCD=AL4PI*DREAL(XRQCDS(ALSZ,ALST,AMZ2,AMW2,AMT2,SSZ))
3176          AKQCD=AL4PI*DREAL(XKQCDS(ALST,AMZ2,AMW2,AMT2,SSZ))
3177        ELSEIF(IQCD.EQ.2) THEN
3178          ROQCD=AL4PI*DREAL(XROQCD(ALSZ,ALST,AMZ2,AMW2,AMT2,SSZ))
3179          AKQCD=AL4PI*DREAL(XKAQCD(ALST,AMZ2,AMW2,AMT2,SSZ))
3180        ELSEIF(IQCD.EQ.3) THEN
3181          ROQCD=AL1PI*ALST/PI*DREAL(XRMQCD(AMZ2,AMW2,AMT2,SSZ))
3182       &        +AL1PI*ALSZ/PI/8D0/SW2/CW2*(VT2+VB2+2D0)
3183  * light quarks are added (25/02/1998), TWO doublets --> 2
3184          AKQCD=AL1PI*ALST/PI*DREAL(XKMQCD(AMZ2,AMW2,AMT2,SSZ))
3185       &        +AL1PI*ALSZ/PI*CW2/2D0/SW2**2*LOG(CW2)
3186        ENDIF
```

Anfang Match 3

```
3048        ELSEIF(IAMT4 .GE. 3) THEN
3049  *
3050          DWZ1AL=R/R1*DREAL(XWZ1R1+XDWZ1F)
3051          RENORM=SQRT(2D0)*GMU*AMZ2*R1*R/PI*ALFAI
3052          SCALE = AL4PI/R1*(41D0/6D0-11D0/3D0*R)*ALR
3053          CORKAP=(AL4PI*DWZ1AL+SCALE)+.75D0*AL4PI/SW2**2*AMT2/AMZ2
3054          DRHOT =.75D0*AL4PI/SW2/R*AMT2/AMZ2
3055          TOPX2 =GMU*AMT2/DSQRT(2.D0)/8.D0/PI2
3056          DRHOT4=3D0*TOPX2*(1D0+TOPX2*AMT4C)
3057          IF(IQCD.EQ.0) THEN
3058           TBQCD0=0D0
3059           TBQCDL=0D0
3060            ELSE
3061             IF(IAFMT.EQ.0) THEN
3062              TBQCD0=-TOPX2*CALXI/PI*2D0*(1D0+PI2/3D0)
3063               ELSE
3064              TBQCD0=3*TOPX2*AFMT3(CALST,AMT2,AMZ2,SW2)
```

```
170     // see dizet6_42.f line 3226
171     Double_t DkTBCor = 3.0*TopCon*ZMath::TbQCD( alphasTop, mt*mt, MZ*MZ, (1.0-m_R) );
172
173     // see dizet6_42.f line 3051
174     m_Renorm        = GMath::Sqrt2()*p_GF*MZ*MZ*(1.0-m_R)*m_R/(pi*GConstants::alphaQED());
175     // see dizet6_42.f line 3053
176     m_KapCorr       = ( GConstants::alphaQED()/(4*pi)*WZCorr + Scale
177                     + 0.75*GConstants::alphaQED()/(4*pi*(1.0-m_R)*(1.0-m_R))*mt*mt/(MZ*MZ) );
178
179     // see dizet6_42.f line 3054
180     m_RhoTop        = 0.75*GConstants::alphaQED()/(4*pi*m_R*(1.0-m_R))*mt*mt/(MZ*MZ);
181     // see dizet6_42.f line 3067
182     m_TBCorL        = -GConstants::alphaQED()/(4*pi)*alphasTop/pi*mt*mt/(MW*MW*(1.0-m_R))*(0.5+pi*pi/6.0);
183
184     // see dizet6_42.f line 3071-3072
185     m_RhoRemAdd     = -1.0 - m_TBCorL +  (1.0-m_R)/m_R*m_KapCorr - m_RhoTop;
186     m_KapRemAdd     = -1.0 - m_R/(1.0-m_R)*(m_TBCorL + m_RhoTop) + m_KapCorr + DkTBCor;
187
188     // see hep-ph/9908433 page 77, DROBAR
189     // see dizet6_42.f line 2300
190     m_DRhoBar       = ( TBCor0 - (GConstants::alphaQED()/(4*pi)*WZCorr + Scale)
191                     *GMath::Sqrt2()*p_GF*MZ*MZ*GMath::IPow( 1-m_R , 2 )/pi*1/GConstants::alphaQED()
192                     + GetEW2Loop().GetDrho2L() );
193     // see hep-ph/9908433 page 77, DROBLO
194     // see dizet6_42.f line 2302
195     m_DRhoBarLO     = (  - (GConstants::alphaQED()/(4*pi)*WZCorr + Scale)
196                      *GMath::Sqrt2()*p_GF*MZ*MZ*GMath::IPow( 1-m_R , 2 )/pi*1/GConstants::alphaQED() );
197
198
199     // eq. (2.4.23) of hep-ph/9908433
200     // bosonic reminder term
201     // see dizet6_42.f line 2176-2178 ||~ bzw. line 2021-2025 / 2252???
202     m_DrRem         =  ( GConstants::alphaQED()/(4*pi)
203                     *(-2.0/3.0 + 1.0/(1.0-m_R)
204                       *(real( GetBosonPart().GetWbAt0() ) - real( GetBosonPart().GetWbAtMW() )
205                         - 5.0/8.0*m_R*(1.0+m_R) + 11.0/2.0 + 9.0/4.0*m_R/(1.0-m_R)*TMath::Log(m_R))
206                     - GConstants::alphaQED()/(4*pi*(1.0-m_R))*(1.0/6.0 + 7.0*m_R)*TMath::Log(m_R) );
207     // fermionic reminder term
208     m_DrRem         += ( GConstants::alphaQED()/(4*pi*(1.0-m_R))
209                     *( real( GetFermionPart().GetWfAt0() ) - real( GetFermionPart().GetWfAtMW() ) )
210                     + GConstants::alphaQED()/(pi*(1.0-m_R))*TMath::Log(m_R) );
```

Ende Match 3

```
211
```

```
3225    * Here TBQCDR is called for FOKAPP
3226              DKTB3R=3*TOPX2*TBQCDR(CALST,AMT2,AMZ2,SW2)
3227            ENDIF
```

```
3062              TBQCD0=-TOPX2*CALXI/PI*2D0*(1D0+PI2/3D0)
3063               ELSE
3064              TBQCD0=3*TOPX2*AFMT3(CALST,AMT2,AMZ2,SW2)
3065    * Here TBQCDR has to be called!!! ROKAPP - 3
3066              ENDIF
3067             TBQCDL=-AL4PI*ALST/PI*AMT2/AMW2/R1*(.5D0+PI2/6D0)
3068           ENDIF
3069           ROFACL=1/(1-DRHOT4-TBQCD0)
3070           AKFACL=(1+R/SW2*(DRHOT4+TBQCD0)-CORKAP*RENORM)
3071           ROFACR=ROFACI        -DRHOT-TBQCDL         -1
3072           AKFACR=AKFACI-R/SW2*(DRHOT+TBQCDL)+CORKAP-1
3073         ENDIF
```

```
2300          DROBAR=3D0*AXF*TBQCD0-(AL4PI*DWZ1AL+SCALE)
2301         &        *SQRT(2D0)*GMU*AMZ2*R1**2/PI*ALFAI+DRHOD
2302          DROBLO=                -(AL4PI*DWZ1AL+SCALE)
2303         &        *SQRT(2D0)*GMU*AMZ2*R1**2/PI*ALFAI
```

```
2175    * See p.18 of PCWG-notebook
2176           DRREMF= DR1FER+R/SW2*DRHO1-DALFA1+TBQCD3+CORRDR
2177           DRREMB= DR1BOS-DRHIG1
2178           DRREM = DRREMF+DRREMB
2179           DRBIG=1-(1+R/SW2*DRIRR-DRHIGS)*(1D0-DALFA)
```

```
2020    DWZ1AL=R/R1*DREAL(XWZ1R1+XDWZ1F)
2021    RXX=-2D0/3D0+4D0/3D0*(SL2+SQ2)+DWZ1AL
2022   *   +(W0AL-WM1AL-5D0/8D0*R2-5D0/8D0*R+11D0/2D0+9D0/4D0*R/R1*ALR)/R1
2023    RXXFER=4D0/3D0*(SL2+SQ2)+R/R1*DREAL(XDWZ1F)+(W0F-DREAL(XWM1F))/R1
2024    RXXBOS=-2D0/3D0+R/R1*DREAL(XWZ1R1)+(W0
2025   &   -DREAL(XWM1)-5D0/8D0*R2-5D0/8D0*R+11D0/2D0+9D0/4D0*R/R1*ALR)/R1
2026    AMH=SQRT(AMH2)
2027    AMW=SQRT(AMW2)
```

```
2252          SCALEB = AL4PI/R1*(1D0/6D0+7D0*R)*ALR
```

```
212     // corrections for Z->bb
213     // there isn't a full two loop calculation
214     // see dizet6_42.f line 3789-3804||~ bzw. line 3798-3804
215     Double_t xt      = GConstants::GF()*mt*mt/(8*GMath::Sqrt2()*pi*pi);||~line 3798
216     m_taub1          = -2.0*xt*( 1.0 - pi/3.0*alphasTop ); ||~line 3800
217     m_taub2          = -2.0*xt*( xt*ZMath::BarbMb( MH/mt ) );
218
219     // see dizet6_42.f line 3069-3070
220     m_RhoCL          = 1/( 1.0 - RhoTop4 - TBCor0 );
221     m_KapCL          = ( 1.0 + m_R/(1.0-m_R)*(RhoTop4 + TBCor0) - m_KapCorr*m_Renorm );
222
223     // see dizet6_42.f line 2972
224     m_CorrBB         = GConstants::alphaQED()/(8.0*pi*(1.0-m_R))*mt*mt/(MW*MW);
225
226     // for Z decay
227     // see dizet6_42.f line 3167
228     m_kapmix         = ( 1/pi*GConstants::alphaQED()
229                        *alphasMZ/(24.0*(1-m_R)*(1-m_R))*(2.0*(1-m_R)-1.0) );
230
231
232     SetUpToDate();
233   }
234
235   // effective weak mixing angle
236   // see also class Sin2ThetaF.h
237   Double_t GSM::Z0ZFitter::GetSin2Eff( GTypes::Particle ParticleType )
238   {
239     Double_t sineff = 0;
240
241     sineff = GetSinEffF().GetSin2ThetaF( ParticleType );
242
243     return sineff;
244   }
245
246   // eq. (2.4.21) of hep-ph/9908433
247   // formfactor rho_Z_f
248   Double_t GSM::Z0ZFitter::rhof( GTypes::Particle ParticleType, Double_t Charge )
249   {
250     Update();
251     Double_t rho    = 0;
252     Double_t RhoRem = 0;
253
```

Match 4

Match 5

Match 6

```
3786    IF(IBARB.EQ.0.OR.IBARB.EQ.-1) THEN
3787     AMT4B=(27-PI2)/3
3788      ELSEIF(IBARB.EQ.1) THEN
3789     RBTH=AMT2/AMH2
3790     ALRB=LOG(RBTH)
3791     AMT4B=1D0/144*(311D0+24*PI2+282*ALRB+90*ALRB**2
3792    &       -4D0*RBTH*(40D0+ 6*PI2+ 15*ALRB+18*ALRB**2)
3793    &       +3D0*RBTH**2*(242.09D0-60*PI2-454.2D0*ALRB-180*ALRB**2))
3794      ELSEIF(IBARB.EQ.2) THEN
3795     RBARB=SQRT(AMH2/AMT2)
3796     AMT4B=FBARBB(RBARB)
3797    ENDIF
3798    TOPX2 = GMU*AMT2/DSQRT(2.D0)/8.D0/PI2
3799    IF(IFTJR.EQ.1) THEN
3800      TAUBB1=-2*TOPX2*(1-PI/3*CALST)
3801    ELSE
3802      TAUBB1=-2*TOPX2
3803    ENDIF
3804    TAUBB2=-2*TOPX2*TOPX2*AMT4B*V_TB**2
```

```
3069          ROFACL=1/(1-DRHOT4-TBQCD0)
3070          AKFACL=(1+R/SW2*(DRHOT4+TBQCD0)-CORKAP*RENORM)
3071          ROFACR=ROFACI        -DRHOT-TBQCDL         -1
3072          AKFACR=AKFACI-R/SW2*(DRHOT+TBQCDL)+CORKAP-1
```

```
2971        TAUBB2=-2*TOPX2*(TOPX2*AMT4B)*V_TB**2
2972        CORBB =+AL1PI/8/SW2*AMT2/AMW2*V_TB**2
```

```
3167        AKMIX=AL1PI*ALSZ/8D0/3D0/SW2**2*(2D0*SW2-1D0)
3168        AR1IM=DIMAG(XRO1)
3169        AK1IM=DIMAG(XAK1)+AKMIX
```

```
switch(ParticleType){
case kElectron:
case kMuon:
case kTau:
case kNeutrino:
case kUp:
case kDown:
case kStrange:
case kCharm:
  // see dizet6_42.f line 3153 & 3190 & 3231
  RhoRem = ( sqrt(GMath::IPow(1.0 + real(rhofLO(ParticleType,Charge)),2)
           + GMath::IPow(imag(rhofLO(ParticleType,Charge)),2)) + m_rhoQCD + m_RhoRemAdd );

  // see dizet6_42.f line 3248-3258
  switch(m_OMSType){
  case OMS1:
    rho = ( (1.0 + m_Renorm*RhoRem + GetEW2Loop().GetRhoRem( TMath::Abs(Charge)))
          /( 1.0 - m_DRhoBar*(1.0 - m_Renorm*m_DrRem) ) );
    break;
  case OMS1_2:
    rho = ( (1.0 + m_Renorm*RhoRem)/(1.0 - m_DRhoBar*(1.0 - m_Renorm*m_DrRem))
          + GetEW2Loop().GetRhoRem( TMath::Abs(Charge) ) );
    break;
  case OMS2:
    rho = ( 1.0 + m_DRhoBar - m_DRhoBarLO*m_Renorm*m_DrRem + m_DRhoBarLO*m_DRhoBarLO
          +m_Renorm*RhoRem*(1.0 + m_DRhoBarLO) + GetEW2Loop().GetRhoRem( TMath::Abs(Charge) ) );
    break;
  default:
    m_logger << kFATAL << "unknown value for \"m_QMSType\": \"" << GetOMSType() << "\""
         << GEndl;
  }
  break;
case kBottom:
    // bottoms don't have full two loop corrections
    // and taub is handling in
    // see dizet6_42.f line 2989 & 3071
  RhoRem = real(rhofLO( ParticleType, Charge )) + m_rhoQCD + 2.0*m_CorrBB - m_RhoTop - m_TBCorL;

```

Match 7

```
3153      RO1=SQRT((1D0+DREAL(XRO1))**2+(DIMAG(XRO1))**2)
3154      AK1=1D0+DREAL(XAK1)
3190      ROFACI=RO1+ROQCD
3191      AKFACI=AK1+AKQCD
3231       ROFACR=ROFACI       -TBQCDL+CORRHO-1
3232       AKFACR=AKFACI-R/SW2*TBQCDL+CORKAP-1+DKTB3R
```

Match 8

```
3248
3249       IF(IFACT.EQ.0)      THEN
3250        ROFAC=(1D0+RENORM*ROFACR+DROREM*SCALER2)
3251     &                          /(1D0-DROBAR*(1D0-DF1BAR))
3252       ELSEIF(IFACT.EQ.1) THEN
3253        ROFAC=(1D0+RENORM*ROFACR)/(1D0-DROBAR*(1D0-DF1BAR))
3254     &                          +DROREM*SCALER2
3255       ELSEIF(IFACT.EQ.2) THEN
3256        ROFAC=1D0+DROBAR-DROBLO*DF1BAR+DROBLO**2
3257     &            +RENORM*ROFACR*(1D0+DROBLO)
3258     &                          +DROREM*SCALER2
3259       ENDIF
```

Match 9

```
2989      ROFACI=1D0+RO1+ROQCD+2*CORBB*IBFLA
2990      AKFACI=1D0+AK1+AKQCD-  CORBB*IBFLA
3069       ROFACL=1/(1-DRHOT4-TBQCD0)
3070       AKFACL=(1+R/SW2*(DRHOT4+TBQCD0)-CORKAP*RENORM)
3071       ROFACR=ROFACI       -DRHOT-TBQCDL        -1
3072       AKFACR=AKFACI-R/SW2*(DRHOT+TBQCDL)+CORKAP-1
```

```
292         // see dizet6_42.f line 3082-3101
293         switch(m_OMSType){
294         case OMS1:
295           rho = 1.0/(1.0/(m_RhoCL*GMath::IPow( (1 + m_taub1 + m_taub2),2 )) - 1.0*RhoRem);
296           break;
297         case OMS1_2:
298           rho = ( m_RhoCL*GMath::IPow( (1 + m_taub1 + m_taub2),2 )
299                 *(1.0 + RhoRem*m_RhoCL*GMath::IPow((1 + m_taub1 + m_taub2),2)) ) ;
300           break;
301         case OMS2:
302           rho = m_RhoCL*GMath::IPow( (1 + m_taub1 + m_taub2),2 )*(1.0 + RhoRem);
303           break;
304         default:
305           m_logger << kFATAL << "unknown value for \"m_QMSType\": \"" << GetOMSType() << "\""
306                    << GEndl;
307         }
308         break;
309       default:
310         m_logger << kFATAL << "unknown value for \"ParticleType\": \"" << GTypes::GetName(ParticleType) << "\""
311                  << GEndl;
312       }
313
314       return rho + rho*p_DeltaOMS;
315     }
316
317     // eq. (2.4.24) of hep-ph/9908433
318     // formfactor kappa_Z_f
319     Double_t GSM::Z0ZFitter::kappaf( GTypes::Particle ParticleType, Double_t Charge )
320     {
321       Update();
322       Double_t kappa = 0;
323       Double_t KapRem = 0;
324       Double_t ka = 0,Da = 0;
325
326       switch(ParticleType){
327       case kElectron:
328       case kMuon:
329       case kTau:
330       case kNeutrino:
331       case kUp:
332       case kDown:
333       case kStrange:
334       case kCharm:
335         // see dizet6_42.f line 3154 & 3191 & 3232
336         KapRem   = ( 1.0 + real(kappafLO(ParticleType,Charge)) + m_kapQCD + m_KapRemAdd );
```

Match 10

Match 11

```
3082        IF(IFACT.EQ.0)THEN
3083         ROFACI=1/(1/ROFACL-SCALER*ROFACR)
3084         AKFACI=AKFACL*(1+SCALER*AKFACR)
3085        ELSEIF(IFACT.EQ.1) THEN
3086         ROFACI=ROFACL*(1+SCALER*ROFACR*ROFACL)
3087         AKFACI=AKFACL+SCALER*AKFACR
3088        ELSEIF(IFACT.EQ.2) THEN
3089         ROFACI=ROFACL*(1+SCALER*ROFACR)
3090         AKFACI=AKFACL+SCALER*AKFACR
3091        ELSEIF(IFACT.EQ.3) THEN
3092         ROFACI=ROFACL+SCALER*ROFACR
3093         AKFACI=AKFACL+SCALER*AKFACR
3094          ELSE
3095         ROFACI=ROFACL
3096         AKFACI=AKFACL
3097        ENDIF
3098       ELSEIF(IEWLC.EQ.0) THEN
3099        ROFACI=ROFACL
3100        AKFACI=AKFACL
3101       ELSE
```

```
3153        RO1=SQRT((1D0+DREAL(XRO1))**2+(DIMAG(XRO1))**2)
3154        AK1=1D0+DREAL(XAK1)
```

```
3190        ROFACI=RO1+ROQCD
3191        AKFACI=AK1+AKQCD
3192        ROFACM=ROFACI
3193        AKFACM=AKFACI
```

```
3231         ROFACR=ROFACI       -TBQCDL+CORRHO-1
3232         AKFACR=AKFACI-R/SW2*TBQCDL+CORKAP-1+DKTB3R
```

```
337       // see dizet6_42.f line 3261-3271
338     switch(m_OMSType){
339     case OMS1:
340       kappa = ( (1.0 + m_Renorm*KapRem + GetEW2Loop().GetDkdRem( TMath::Abs( Charge ) ))
341             *(1.0 + m_R/(1.0-m_R)*m_DRhoBar*(1.0 - m_Renorm*m_DrRem)) );
342       break;
343     case OMS1_2:
344       kappa = ( (1.0 + m_Renorm*KapRem)*(1.0 + m_R/(1.0-m_R)*m_DRhoBar*(1.0 - m_Renorm*m_DrRem))
345             + GetEW2Loop().GetDkdRem( TMath::Abs( Charge ) ) );
346       break;
347     case OMS2:
348       kappa = ( 1.0 + m_R/(1.0-m_R)*m_DRhoBar - m_R/(1.0-m_R)*m_DRhoBarLO*m_Renorm*m_DrRem
349             + m_Renorm*KapRem*(1.0 + m_R/(1.0-m_R)*m_DRhoBarLO)
350             + GetEW2Loop().GetDkdRem( TMath::Abs( Charge ) ) );
351       break;
352     default:
353       m_logger << kFATAL << "unknown value for \"m_QMSType\": \"" << GetOMSType() << "\""
354             << GEndl;
355     }
356     Da      = GetDAlphaQED().DAlphaQEDMZ()*GConstants::alphaQED()/(4.0*TMath::Pi());
357     ka      = GMath::IPow( GConstants::alphaQED()/(1.0 - Da), 2 )*35/18.0*(1.0 - 8/3.0*kappa*(1.0-m_R));
358     kappa  += ka/(1.0-m_R);          ||~line 3273-3274
359     break;
360   case kBottom:
361     // bottoms don't have full two loop corrections
362     // and taub is handling in
363     // see dizet6_42.f line 2990 & 3072
364     KapRem    = ( real(kappafLO( ParticleType, Charge )) + m_kapQCD - m_CorrBB
365               - m_R/(1.0-m_R)*(m_RhoTop + m_TBCorL) + m_KapCorr );
366
367      // see dizet6_42.f line 3082-3101
368     switch(m_OMSType){
369     case OMS1:
370       kappa = m_KapCL/(1 + m_taub1 + m_taub2)*( 1.0 + 1.0*KapRem );
371       break;
372     case OMS1_2:
373       kappa = m_KapCL/(1 + m_taub1 + m_taub2) + KapRem;
374       break;
375     case OMS2:
376       kappa = m_KapCL/(1 + m_taub1 + m_taub2) + KapRem;
377       break;
378     default:
379       m_logger << kFATAL << "unknown value for \"m_QMSType\": \"" << GetOMSType() << "\""
380             << GEndl;
```

Match 12

```
3260   ROFACI=ROFAC
3261   IF(IFACT.EQ.0) THEN
3262    AKFAC=(1D0+RENORM*AKFACR+DKDREM*SCALER2)
3263  &                          *(1D0+R/SW2*DROBAR*(1D0-DF1BAR))
3264   ELSEIF(IFACT.EQ.1) THEN
3265    AKFAC=(1D0+RENORM*AKFACR)*(1D0+R/SW2*DROBAR*(1D0-DF1BAR))
3266  &                          +DKDREM*SCALER2
3267   ELSEIF(IFACT.EQ.2) THEN
3268    AKFAC=1D0+R/SW2*DROBAR-R/SW2*DROBLO*DF1BAR
3269  &           +RENORM*AKFACR *(1D0+R/SW2*DROBLO)
3270  &                          +DKDREM*SCALER2
3271   ENDIF
3272
3273   ADDIM=1D0/ALFAI**2/(1D0-DALFA)**2*35D0/18*(1D0-8D0/3*AKFAC*SW2)
3274   AKFACI=AKFAC+ADDIM/SW2
3275   SINEFF=AKFACI*SW2
```

```
2989        ROFACI=1D0+RO1+ROQCD+2*CORBB*IBFLA
2990        AKFACI=1D0+AK1+AKQCD-  CORBB*IBFLA
```

```
3069        ROFACL=1/(1-DRHOT4-TBQCD0)
3070        AKFACL=(1+R/SW2*(DRHOT4+TBQCD0)-CORKAP*RENORM)
3071        ROFACR=ROFACI        -DRHOT-TBQCDL         -1
3072        AKFACR=AKFACI-R/SW2*(DRHOT+TBQCDL)+CORKAP-1
```

Match 13

Match 14

```
3082        IF(IFACT.EQ.0)THEN
3083         ROFACI=1/(1/ROFACL-SCALER*ROFACR)
3084         AKFACI=AKFACL*(1+SCALER*AKFACR)
3085        ELSEIF(IFACT.EQ.1) THEN
3086         ROFACI=ROFACL*(1+SCALER*ROFACR*ROFACL)
3087         AKFACI=AKFACL+SCALER*AKFACR
3088        ELSEIF(IFACT.EQ.2) THEN
3089         ROFACI=ROFACL*(1+SCALER*ROFACR)
3090         AKFACI=AKFACL+SCALER*AKFACR
3091        ELSEIF(IFACT.EQ.3) THEN
3092         ROFACI=ROFACL+SCALER*ROFACR
3093         AKFACI=AKFACL+SCALER*AKFACR
3094          ELSE
3095         ROFACI=ROFACL
3096         AKFACI=AKFACL
3097         ENDIF
3098        ELSEIF(IEWLC.EQ.0) THEN
3099         ROFACI=ROFACL
3100         AKFACI=AKFACL
3101        ELSE
```

```
381         }
382         break;
383       default:
384         m_logger << kFATAL << "unknown value for \"ParticleType\": \"" << GTypes::GetName(ParticleType) << "\""
385                  << GEndl;
386       }
387
388       return kappa + p_DeltaOMS*kappa;
389     }
390
391     // leading order of rhof defined in (253) of hep-ph/9709229v1
392     complex<Double_t> GSM::Z0ZFitter::rhofLO( GTypes::Particle ParticleType, Double_t Charge )
393     {
394       complex<Double_t> rho = 0;
395
396       switch(ParticleType){
397       case kElectron:
398       case kMuon:
399       case kTau:
400       case kNeutrino:
401       case kUp:
402       case kDown:
403       case kStrange:
404       case kCharm:
405         // see dizet6_42.f line 3147
406         rho = ( GConstants::alphaQED()/(4*TMath::Pi()*(1.0-m_R))*
407                 ( real(m_ZAtMZ + m_ZFAtMZ) - m_WAt0 + 5.0/8.0*m_R*(1.0+m_R)
408                   - 11.0/2.0 - 9.0*m_R/(4.0*(1.0-m_R))*TMath::Log(m_R) + 2.0*Uff(ParticleType, Charge)) );
409         break;
410       case kBottom:
411         // see dizet6_42.f line 2907
412         rho = ( GConstants::alphaQED()/(4*TMath::Pi()*(1.0-m_R))*
413                 ( real(m_ZAtMZ) + real(m_ZFAtMZ) - real(m_WAt0) + 5.0/8.0*m_R*(m_R + 1.0)
414                   - 11.0/2.0 - 9.0*m_R/(4.0*(1.0-m_R))*TMath::Log(m_R) + Uff(ParticleType, Charge)) );
415         break;
416       default:
417         m_logger << kFATAL << "false input for ParticleType" << GTypes::GetName(ParticleType) << GEndl;
418         break;
419       }
420
421       return rho;
422     }
```

Match 15

```
3147        XRO1=AL4PI/R1*(DREAL(XZM1A+XZFM1A)-W0A+5.D0/8.D0*R*(1D0+R)
3148       &    -11.D0/2.D0-9.D0/4.D0*R/R1*ALR+2D0*XUFF)
```

```
2907        RO1=AL4PI/R1*(ZM1A+ZFM1A-W0A+5.D0/8.D0*R2+5.D0/8.D0*R
2908       &    -11.D0/2.D0-9.D0/4.D0*R/R1*ALR+UFF)
```

Ende Match 15

```
423
424  // leading order to kappaf defined in (254) of hep-ph/9709229v1
425  complex<Double_t> GSM::Z0ZFitter::kappafLO( GTypes::Particle ParticleType, Double_t Charge )
426  {
427    complex<Double_t> kappa = 0;
428
429    switch(ParticleType){
430    case kElectron:
431    case kMuon:
432    case kTau:
433    case kNeutrino:
434    case kUp:
435    case kDown:
436    case kStrange:
437    case kCharm:
438      // see dizet6_42.f line 3150
439      kappa = ( GConstants::alphaQED()/(4*TMath::Pi()*(1-m_R))*
440                ( m_R/(1-m_R)*real(m_ZAtMZ - m_WAtMW)
441                  + m_MPhoZAtMZ + GMath::IPow((1-m_R),2)/m_R*Charge*Charge*GetVertex().GetV1ZZ()
442                  - Uff(ParticleType, Charge) ) );
443      break;
444    case kBottom:
445      // see dizet6_42.f line 2910
446      kappa = ( GConstants::alphaQED()/(4*TMath::Pi()*(1-m_R))*
447                ( m_R/(1-m_R)*( real( m_ZAtMZ ) - real( m_WAtMW ) )
448                  + real( m_MPhoZAtMZ ) + GMath::IPow((1-m_R),2)/m_R*Charge*Charge*real( GetVertex().GetV1ZZ() )
449                  - 0.5*Uff(ParticleType, Charge) ) );
450      break;
451    default:
452      m_logger << kFATAL << "false input for ParticleType : " << GTypes::GetName(ParticleType) << GEndl;
453      break;
454    }
455
456    return kappa;
457
458  }
459
460  // eq. (255) of hep-ph/9709229v1
461  complex<Double_t> GSM::Z0ZFitter::Uff( GTypes::Particle ParticleType, Double_t Charge )
462  {
463    complex<Double_t> uff = 0;
464
```

Match 16

```
3150        XAK1=AL4PI/R1*(R/R1*DREAL(XZM1A-XWM1A)
3151       &     +XAMM1A+R12/R*CH2*XV1ZZ-XUFF)
```

```
2910        AK1=AL4PI/R1*(R/R1*(ZM1A-WM1A)+AMM1A+R12/R*CH2*V1ZZ-.5D0*UFF)
```

Ende Match 16

```
465     switch(ParticleType){
466     case kElectron:
467     case kMuon:
468     case kTau:
469     case kNeutrino:
470     case kUp:
471     case kDown:
472     case kStrange:
473     case kCharm:
474       // see dizet6_42.f line 3144
475       uff = ( 0.25/m_R*(1.0 - 6.0*TMath::Abs(Charge)*(1.0-m_R)
476                         + 12.0*Charge*Charge*(1.0-m_R)*(1.0-m_R))*GetVertex().GetV1ZZ()
477               + (0.5 - m_R - TMath::Abs(Charge)*(1.0-m_R))*GetVertex().GetV1ZW()
478               + m_R*GetVertex().GetV2ZWW() );
479       break;
480     case kBottom:
481       // see dizet6_42.f line 2903
482       uff = ( 0.5/m_R*(1.0 - 6.0*TMath::Abs(Charge)*(1.0-m_R)
483                        + 12.0*Charge*Charge*(1.0-m_R)*(1.0-m_R))*real(GetVertex().GetV1ZZ())
484               + (1.0 - 2.0*m_R - 2.0*TMath::Abs(Charge)*(1.0-m_R))*real(GetVertex().GetV1ZW())
485               + 2.0* m_R*real(GetVertex().GetV2ZWW()) + 2.0*GetVertex().GetVtb() );
486       break;
487     default:
488       m_logger << kFATAL << "false input for ParticleType : " << GTypes::GetName(ParticleType) << GEndl;
489       break;
490
491     }
492
493     return uff;
494   }
495
496   // partial decay width of the Z Boson
497   // eq. (2.49) and (2.4.10) of hep-ph/9908433
498   Double_t GSM::Z0ZFitter::GammaZff( GTypes::Particle ParticleType, Double_t Charge, Double_t mf )
499   {
500     Update();
501
502     Double_t Gamma0     = ( p_GF*GMath::IPow(p_MZ,3)/
503                             (12.0*TMath::Pi()*GMath::Sqrt2()) );
504     Double_t aQEDMZ     = GConstants::alphaQED()/( 1 - GetDAlphaQED().DAlphaQEDMZt() );
505     Double_t Rhof       = rhof( ParticleType, Charge );
506     Double_t T3         = 0.5*TMath::Sign(1.0,Charge+0.1);
507     Double_t SqrRatio   = 1;
```

Match 17

Ende Match 17

```
3144        XUFF=0.25D0/R*(1D0-6D0*CH*R1+12D0*R12*CH2)*XV1ZZ
3145       &     +(0.5D0-R-CH*R1)*XV1ZW+R*V2ZWW+VTB
```

```
2903        UFF=(0.5D0/R-3.D0/R*CH*R1+6.D0*R12/R*CH2)*V1ZZ
2904       &    +(1.D0-2.D0*R-2.D0*CH*R1)*V1ZW
2905       &    +2.D0*R*V2ZWW+2.D0*VTB
```

```
508     Double_t RatioMfMZ  = 0;
509     Double_t RadV       = 0;
510     Double_t RadA       = 0;
511     Double_t AddEWQCD   = 0;
512     Int_t ncf           = 0;
513     Int_t nf            = 0;
514
515     switch(ParticleType){
516     case kElectron :
517     case kMuon     :
518     case kTau      :
519     case kNeutrino :
520       RadV      = 1.0 + 0.75*aQEDMZ/TMath::Pi()*Charge*Charge;
521       RadA      = RadV;
522       RatioMfMZ  = GMath::IPow(mf/p_MZ,2);
523       SqrRatio   = sqrt(1.0-4.0*RatioMfMZ);
524       ncf       = 1;
525       break;
526     case kBottom  :
527       ++nf;
528     case kStrange :
529       ++nf;
530     case kCharm   :
531       ++nf;
532     case kDown    :
533       ++nf;
534     case kUp      :
535       ++nf;
536       RadV      = GetRadFun().GetRVq( ParticleType, Charge);
537       RadA      = GetRadFun().GetRAq( ParticleType, Charge, T3);
538       RatioMfMZ  = 0;
539       SqrRatio   = 1;
540       ncf       = 3;
541       AddEWQCD   = m_EWQCD[nf-1];
542       break;
543     default:
544       m_logger << kFATAL << "false input for ParticleType : " << GTypes::GetName(ParticleType) << GEndl;
545       break;
546     }
547
548     // see dizet6_42.f line 3169
549     Double_t KappaImag  = imag(kappafLO(ParticleType,Charge)) + m_kapmix;
550     if( ParticleType == kBottom ) KappaImag = 0;
551     Double_t sineff     = GetSin2Eff(ParticleType);
552     Double_t RatioCplgs = 1.0 - 4.0*TMath::Abs(Charge)*sineff;
553
```

Match 18

```
2711      IF(INF.LE.4.OR.ABS(QCDCOR((MAX(0,2*INDQ(INF)-1)))-1).LT.1D-8) THEN
2712        RQCDV=1D0+0.75D0*CALQED/PI*AQFI(INF)**2
2713        RQCDA=RQCDV
2714      ELSE
2715        RQCDV=QCDCOR(MAX(0,2*INDQ(INF)-1))
2716        RQCDA=QCDCOR(MAX(0,2*INDQ(INF)   ))
2717      ENDIF
```

Match 19

```
3167      AKMIX=AL1PI*ALSZ/8D0/3D0/SW2**2*(2D0*SW2-1D0)
3168      AR1IM=DIMAG(XRO1)
3169      AK1IM=DIMAG(XAK1)+AKMIX
```

```
554     // see dizet6_42.f line 2734-2793||~2738 - 2739???
555     Double_t Gamma      = ( Gamma0*Rhof*SqrRatio*
556                         ( (1.0+2.0*RatioMfMZ)
557                          *( (RatioCplgs*RatioCplgs + 16.0*(1-m_R)*(1-m_R)*
558                             Charge*Charge*KappaImag*KappaImag)*RadV
559                           + RadA)/2.0 - 3.0*RatioMfMZ*RadA) );
560
561     return Gamma*ncf + AddEWQCD;
562
563   }
564
565
```

Match 20

```
2734  *
2735        IF(IFACT.LE.3) THEN
2736         VF1L=1-4*SW2M*AQFI(INF)*AKFACI
2737         GAMWI=CONSTZ*ROFACI*SQR*((1+2*RAT)*(VF1L**2+1)/2-3*RAT)
2738         GAM1I=CONSTZ*ROFACI*SQR*((1+2*RAT)
2739        &     *((VF1L**2+16*SW2M**2*(AQFI(INF))**2*AK1IM**2)*RQCDV+RQCDA)/2
2740        &                              -3*RAT*RQCDA)
2741        &      +ICZAK*ARCZAK(INDQ(INF))
2742         GAM1IA=CONSTZ*ROFACI*SQR*((1+2*RAT)
2743        &      *(VF1L**2*RQCDV+RQCDA)/2-3*RAT*RQCDA)
2744        &      +ICZAK*ARCZAK(INDQ(INF))
2745         GAM1II=CONSTZ*ROFACI*SQR*(1+2*RAT)
2746        &      *(16*SW2M**2*(AQFI(INF))**2*AK1IM**2*RQCDV)/2
2747  * FOR Z->BB APPLIED ONLY FOR IFACT.LE.3
2748         BF1L=1-4*SW2M*AQFI(INF)*AKFABI
2749         BAMWI=CONSTZ*ROFABI*SQR*((1+2*RAT)*(BF1L**2+1)/2-3*RAT)
2750         BAM1I=CONSTZ*ROFABI*SQR*((1+2*RAT)
2751        &     *((BF1L**2+16*SW2M**2*(AQFI(INF))**2*AK1IM**2)*RQCDV+RQCDA)/2
2752        &                              -3*RAT*RQCDA)
2753        &      +ICZAK*ARCZAK(INDQ(INF))
2754         BAM1IA=CONSTZ*ROFABI*SQR*((1+2*RAT)
2755        &      *(BF1L**2*RQCDV+RQCDA)/2-3*RAT*RQCDA)
2756        &      +ICZAK*ARCZAK(INDQ(INF))
2757         BAM1II=CONSTZ*ROFABI*SQR*(1+2*RAT)
2758        &      *(16*SW2M**2*(AQFI(INF))**2*AK1IM**2*RQCDV)/2
2759        ELSEIF(IFACT.EQ.4) THEN
```

```
2759        ELSEIF(IFACT.EQ.4) THEN
2760         ROFACL=ROFACI
2761         AKFACL=AKFACI
2762         ROFACR=SCALER*ROFACR
2763         AKFACR=SCALER*AKFACR
2764         AKFACI=AKFACL+AKFACR
2765         VF1LL=1-4*SW2M*AQFI(INF)*AKFACL
2766         VF1LR= -4*SW2M*AQFI(INF)*AKFACR
2767         GAMWI=CONSTZ*SQR*(ROFACL*((1+2*RAT)*( VF1LL**2+2*VF1LL*VF1LR+1)/2
2768        &                                                    -3*RAT)
2769        &                  +ROFACR*((1+2*RAT)*( VF1LL**2+1 )/2 -3*RAT)     )
2770         GAM1I=CONSTZ*SQR*(ROFACL*((1+2*RAT)
2771        & *(((VF1LL**2+2*VF1LL*VF1LR)+16*SW2M**2*(AQFI(INF))**2*AK1IM**2)/2
2772        &                                    *RQCDV+RQCDA/2)-3*RAT*RQCDA)
2773        &                  +ROFACR*((1+2*RAT)*( VF1LL**2/2
2774        &                                    *RQCDV+RQCDA/2)-3*RAT*RQCDA) )
2775        &      +ICZAK*ARCZAK(INDQ(INF))
2776        ELSEIF(IFACT.EQ.5) THEN
2777         ROFACL=ROFACI
2778         AKFACL=AKFACI
2779         ROFACR=SCALER*ROFACR
2780         AKFACR=SCALER*AKFACR
2781         AKFACI=AKFACL+AKFACR
2782         VF1LL=1-4*SW2M*AQFI(INF)*AKFACL
2783         VF1LR= -4*SW2M*AQFI(INF)*AKFACR
2784         GAMWI=CONSTZ*SQR*(ROFACL*((1+2*RAT)*( VF1LL**2+2*VF1LL*VF1LR+1)/2
2785        &                                                    -3*RAT)
2786        &                  +ROFACR*((1+2*RAT)*( VF1LL**2+1 )/2 -3*RAT)     )
2787         GAM1I=CONSTZ*SQR*(ROFACL*((1+2*RAT)
2788        &      *((VF1LL**2+16*SW2M**2*(AQFI(INF))**2*AK1IM**2)/2
2789        &                                      *RQCDV+RQCDA/2)-3*RAT*RQCDA)
2790        &                  +ROFACL*((1+2*RAT)*VF1LL*VF1LR)
2791        &                  +ROFACR*((1+2*RAT)*( VF1LL**2+1)/2 -3*RAT)      )
2792        &      +ICZAK*ARCZAK(INDQ(INF))
2793        ENDIF
```