```
1  /**********************************************************************************
2   * Package: GSM                                                                   *
3   * Class  : Sin2TheatF                                                            *
4   *                                                                                *
5   * Description:                                                                   *
6   *      Auxiliary Theory for the effective weak mixing angle (included 2 loops)   *
7   *                                                                                *
8   * Papers:                                                                        *
9   *      - M. Awramik, M. Czakon, A. Freitas, JHEP0611:048, 2006, hep-ph/0608099   *
10  *      - M. Awramik, M. Czakon, A. Freitas, B.A. Kniehl,                         *
11  *                        Nucl. Phys. B813:174-187, 2009, arXiv:0811.1364         *
12  *                                                                                *
13  **********************************************************************************
14  #include "TMath.h"
15
16  #include "Gfitter/GMath.h"
17  #include "Gfitter/GTheory.h"
18  #include "Gfitter/GTheoryRef.h"
19  #include "Gfitter/GParameterRef.h"
20  #include "Gfitter/GReference.h"
21  #include "Gfitter/GVariable.h"
22  #include "Gfitter/GStore.h"
23  #include "Gfitter/GConstants.h"
24
25  #include "GSM/Sin2ThetaF.h"
26  #include "GSM/DAlphaQED.h"
27  #include "GSM/MH.h"
28
29  using namespace Gfitter;
30
31  ClassImp(GSM::Sin2ThetaF)
32
33  GSM::Sin2ThetaF::Sin2ThetaF()
34    : Gfitter::GAuxTheory()
35  {
```

Hinweis:

Kommentare mit Hinweisen auf ZFitter sind grün markiert

Übereinstimmungen sind gelb markiert.

Auffällige Stellen bzw. Anmerkungen sind violett markiert

Anhang 12

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

```
36      SetTheoryName( GetName() );
37      SetExistDerivative( kFALSE );
38
39      const TString& logMH = gStore()->GetVariable( "GSMFlags::logMH" )->GetStringValue();
40      m_logger << kINFO << "Using logMH: \"" << logMH << "\"" << GEndl;
41
42      if      (logMH == "Yes" ) m_logMH  = kTRUE;
43      else if (logMH == "No" )  m_logMH  = kFALSE;
44      else {
45        m_logger << kFATAL << "unknown value for \"GSMFlags::logMH\": \"" << logMH << "\""
46                 << ". Possible are: \"Yes\" and \"No\"\""
47                 << GEndl;
48      }
49
50      BookParameter( "MZ",                    &p_MZ );
51      BookParameter( "mt",                   &p_mt );
52      BookParameter( "GF",                    &p_GF );
53      BookParameter( "DeltaSin2ThetaF_Scale",   &p_DeltaSin2ThetaF_Scale );
54      BookTheory   ( "GSM::RunningAlphaQCD",    &t_AlphasRun );
55      BookTheory   ( "GSM::DAlphaQED",          &t_DAlphaQED );
56      BookTheory   ( "GSM::MH" , & t_MH );
57    }
58
59    // coefficients of the fitting formulaes / table 5 of hep-ph/0608099
60    void GSM::Sin2ThetaF::Initialise()
61    {
62      // hep-ph/0608099
63      m_s0[0]  = 0.2312527;
64      m_s0[1]  = 0.2308772;
65      m_s0[2]  = 0.2311395;
66      m_s0[3]  = 0.2310286;
67      m_d1[0]  = 4.729e-4;
68      m_d1[1]  = 4.713e-4;
69      m_d1[2]  = 4.726e-4;
70      m_d1[3]  = 4.720e-4;
```

Anmerkung: diese Konstanten
wurden in ZFitter nicht gefunden

```
71      m_d2[0]  = 2.07e-5;
72      m_d2[1]  = 2.05e-5;
73      m_d2[2]  = 2.07e-5;
74      m_d2[3]  = 2.06e-5;
75      m_d3[0]  = 3.85e-6;
76      m_d3[1]  = 3.85e-6;
77      m_d3[2]  = 3.85e-6;
78      m_d3[3]  = 3.85e-6;
79      m_d4[0]  = -1.85e-6;
80      m_d4[1]  = -1.85e-6;
81      m_d4[2]  = -1.85e-6;
82      m_d4[3]  = -1.85e-6;
83      m_d5[0]  = 2.07e-2;
84      m_d5[1]  = 2.06e-2;
85      m_d5[2]  = 2.07e-2;
86      m_d5[3]  = 2.07e-2;
87      m_d6[0]  = -2.851e-3;
88      m_d6[1]  = -2.850e-3;
89      m_d6[2]  = -2.853e-3;
90      m_d6[3]  = -2.848e-3;
91      m_d7[0]  = 1.82e-4;
92      m_d7[1]  = 1.82e-4;
93      m_d7[2]  = 1.83e-4;
94      m_d7[3]  = 1.81e-4;
95      m_d8[0]  = -9.74e-6;
96      m_d8[1]  = -9.71e-6;
97      m_d8[2]  = -9.73e-6;
98      m_d8[3]  = -9.73e-6;
99      m_d9[0]  = 3.98e-4;
100     m_d9[1]  = 3.96e-4;
101     m_d9[2]  = 3.98e-4;
102     m_d9[3]  = 3.97e-4;
103     m_d10[0] = -6.55e-1;
104     m_d10[1] = -6.54e-1;
105     m_d10[2] = -6.55e-1;
```

Anmerkung: diese Konstanten
wurden in ZFitter nicht gefunden

```
106       m_d10[3] = -6.55e-1;
107
108       // arXiv:0811.1364
109       m_s0[4]  =  2.327580e-1;
110       m_d1[4]  =  4.749e-4;
111       m_d2[4]  =  2.03e-5;
112       m_d3[4]  =  3.94e-6;
113       m_d4[4]  = -1.84e-6;
114       m_d5[4]  =  2.08e-2;
115       m_d6[4]  = -9.93e-4;
116       m_d7[4]  =  7.08e-5;
117       m_d8[4]  = -7.61e-6;
118       m_d9[4]  =  4.03e-4;
119       m_d10[4] =  6.61e-1;
120    }
121
122    // eff. ew mixing angle
123    // we just use the leptonic case and computes the all other
124    // angles in relation to this one ( see Z0ZFitter )
125    // formulaes (48), (49) of hep-ph/0608099
126    Double_t GSM::Sin2ThetaF::GetSin2ThetaF( GTypes::Particle ParticleType )
127    {
128       Double_t MH       = GetMH().GetValue(); //p_MH;
129       if( m_logMH ) MH = TMath::Exp( GetMH().GetValue() ); //p_MH );
130
131       Double_t deAlpha    = (GetDAlphaQED().DAlphaQEDMZ()/0.05907) - 1.0;
132       Double_t deAlphaS   = (GetAlphasRun().EvolveAlphas(p_MZ)/0.117) - 1.0;
133
134       Double_t LH         = TMath::Log(MH/100);
135       Double_t dH         = (MH/100.0);
136
137       Double_t dt         = ((p_mt/178.0)*(p_mt/178.0) - 1.0);
138       Double_t dZ         = (p_MZ/91.1876) - 1.0;
139       Double_t sin2ThetaF = 0;
140
```

Anmerkung: diese Konstanten wurden in ZFitter nicht gefunden

Match 1

3.1.2.3

dizet6_42.f

```
2641  *- Sw_eff block:
2642  *--
2643        DALPHA=AL4PI*DREAL(XFOTF3(IALEM,IALE2,IHVP,1,0,DAL5H,-AMZ2))
2644        DELALP=DALPHA/0.05907D0-1D0
2645        DELALS=CALSZ/0.117D0-1D0
2646        DELHIG=LOG(AMH/100D0)
2647     DELHSP=(AMH/100D0)
2648        DELTOP=(AMQ(5)/174.3D0)**2-1D0
2649        DELAMZ=(AMZ/91.1875D0)-1D0
```

```
141     Int_t index = 0;
142
143     if ( ParticleType == GTypes::kElectron || ParticleType == GTypes::kMuon || ParticleType == GTypes::kTau )
144       index = 0; // charged leptons
145     else if ( ParticleType == GTypes::kNeutrino ) {
146       index = 1; // neutrinos
147     }
148     else if ( ParticleType == GTypes::kUp || ParticleType == GTypes::kCharm ) {
149       index = 2; // up, charm
150     }
151     else if ( ParticleType == GTypes::kDown || ParticleType == GTypes::kStrange ) {
152       index = 3; // down, strange
153     }
154     else if ( ParticleType == GTypes::kBottom ) {
155       index = 4; // bottom
156     }
157     else m_logger << kFATAL << "<SIN2ThethaL> Unknown flavour: " << GTypes::GetName(ParticleType) << GEndl;
158
159
160     sin2ThetaF = ( m_s0[index] + m_d1[index]*LH + m_d2[index]*LH*LH
161                 + m_d3[index]*Gfitter::GMath::IPow(LH,4) + m_d4[index]*(dH*dH-1)
162                 + m_d5[index]*deAlpha + m_d6[index]*dt + m_d7[index]*dt*dt
163                 + m_d8[index]*dt*(dH-1) + m_d9[index]*deAlphaS + m_d10[index]*dZ );
164
165     // theretical uncertainty for eff. weak mixing angle
166     sin2ThetaF = sin2ThetaF + p_DeltaSin2ThetaF_Scale*4.7e-5;
167
168     // inform the base class that calculations for this set of parameters the
169     // theory predictions are up-to-date
170     // SetUpToDate();
171
172     // test GF dependency of GF
173     if( false ){
174       Double_t MZ2 = p_MZ*p_MZ;
175
```

```
176         Double_t deltaSin2 = ( 1./2.*( TMath::Sqrt(1-TMath::Sqrt(8.)*TMath::Pi()*GConstants::alphaQED()/(MZ2*GConstants::GF()))
177                                     - TMath::Sqrt(1-TMath::Sqrt(8.)*TMath::Pi()*GConstants::alphaQED()/(MZ2*p_GF)) ) );
178         sin2ThetaF += deltaSin2;
179
180       }
181
182
183       return sin2ThetaF;
184     }
```