```
1
2
3       typedef double XFLOAT
4       typedef double OTA_FLOAT
5
6
7   {
8
9   XFLOAT LpqWeight (CPOLQAData            *POLQAHandle,
10                    const int             pPowerSyllabe,
11                    const int             pPowerTime,
12                    const CDoubleArray    &pDisturbance,
13                    const CDoubleArray    &pTimeWeight)
14  {
15      XFLOAT      resultTime = 0
16      XFLOAT      totalTimeWeightTime = 0
17      int NUMBER_OF_PSQM_FRAMES_PER_SYLLABE = 2 * round(17.5*0.75 - 3.75)
18
19      const int stopFrameIdx = POLQAHandle->statics->stopFrameIdx
20
21      for (int startFrameOfSyllabe = POLQAHandle->statics->startFrameIdx
22           startFrameOfSyllabe <= stopFrameIdx
23           startFrameOfSyllabe += NUMBER_OF_PSQM_FRAMES_PER_SYLLABE/2)
24      {
25          XFLOAT  resultSyllabe = 0
26          int     countSyllabe = 0
27
28          for (int frameIndex = startFrameOfSyllabe
29              frameIndex < startFrameOfSyllabe + NUMBER_OF_PSQM_FRAMES_PER_SYLLABE && frameIndex <=
    stopFrameIdx
30              frameIndex++)
31          {
32              resultSyllabe +=  pow (pDisturbance.m_pData[frameIndex], pPowerSyllabe)
33              countSyllabe++
34          }
35
36          resultSyllabe /= countSyllabe
37          resultSyllabe = pow (resultSyllabe, (XFLOAT) 1.0 / (XFLOAT)pPowerSyllabe)
38
39          resultTime +=  pow (pTimeWeight.m_pData[startFrameOfSyllabe] * resultSyllabe, pPowerTime)
40          totalTimeWeightTime += pow (pTimeWeight.m_pData[startFrameOfSyllabe], pPowerTime)
41      }
42
43      resultTime /= totalTimeWeightTime
44      resultTime = pow (resultTime, (XFLOAT) 1.0 / (XFLOAT)pPowerTime)
45
46      return (XFLOAT) resultTime
47  }
48
49  XFLOAT LppqqWeight (CPOLQAData          *POLQAHandle,
50                      const double            pPowerSyllabe,
51                      const double            pPowerTime,
52                      const CDoubleArray    &pDisturbance,
53                      const CDoubleArray    &pTimeWeight)
54  {
55      XFLOAT      resultTime = 0
56      XFLOAT      totalTimeWeightTime = 0
57      XFLOAT      hulp
58
59      int NUMBER_OF_PSQM_FRAMES_PER_SYLLABE = 2 * round(3.0*0.75)
60      const int stopFrameIdx = POLQAHandle->statics->stopFrameIdx
61
62      for (int startFrameOfSyllabe = POLQAHandle->statics->startFrameIdx
63           startFrameOfSyllabe <= stopFrameIdx
64           startFrameOfSyllabe += NUMBER_OF_PSQM_FRAMES_PER_SYLLABE/2)
65      {
66          XFLOAT  resultSyllabe = 0
67          int     countSyllabe = 0
```

```
68
69          for (int frameIndex = startFrameOfSyllabe
70              frameIndex < startFrameOfSyllabe + NUMBER_OF_PSQM_FRAMES_PER_SYLLABE && frameIndex <=
    stopFrameIdx
71              frameIndex++)
72          {
73              resultSyllabe +=  pow (pDisturbance.m_pData[frameIndex], pPowerSyllabe)
74              countSyllabe++
75          }
76
77          resultSyllabe /= countSyllabe
78          resultSyllabe = pow (resultSyllabe, (XFLOAT) 1.0 / (XFLOAT)pPowerSyllabe)
79          hulp = pow(pTimeWeight.m_pData[startFrameOfSyllabe],20.0)
80          resultTime +=  pow (hulp * resultSyllabe, pPowerTime)
81          totalTimeWeightTime += pow (hulp, pPowerTime)
82      }
83
84      resultTime /= totalTimeWeightTime
85      resultTime = pow (resultTime, (XFLOAT) 1.0 / (XFLOAT)pPowerTime)
86
87      return (XFLOAT) resultTime
88  }
89
90  XFLOAT MappingNoisiness(XFLOAT MovValue)
91      {
92          double y=0
93          {
94
95              const int NumMOVs=1
96              const XFLOAT MinVal[] ={-5.1881600100e+001  }
97              const XFLOAT MaxVal[] ={-1.2932499900e+001  }
98              const XFLOAT a[] ={5.3755432426e-002    }
99              const XFLOAT b[] ={1.8673875713e+000    }
100             const XFLOAT wA[]     ={7.4651073203e+000   }
101             for (int o=0  o<NumMOVs  o++)
102             {
103                 XFLOAT x = b[o] + a[o] * min(max(MovValue, MinVal[o]), MaxVal[o])
104                 y += wA[o] /(1.0 + exp(-x))
105             }
106             y += -9.1276868618e-001
107         }
108         return y
109
110     }
111 XFLOAT MappingContinuity(XFLOAT d0s7t3,XFLOAT TimeClipp, XFLOAT GainVarInd)
112     {
113         double y=0
114         {
115             const int NumMOVs=3
116             XFLOAT MOV[]    ={d0s7t3, TimeClipp, GainVarInd }
117
118             const XFLOAT MinVal[] ={8.7619990000e-001, -3.0000000100e+001, -1.0000000000e-007   }
119             const XFLOAT MaxVal[] ={3.7069000100e+001, 3.2061100100e+001, 5.0000001000e+000 }
120             const XFLOAT a[] ={2.1118267162e-001, 6.1198879625e-002, 3.9752161922e+000  }
121             const XFLOAT b[] ={-2.9017902933e+000, -2.7251827257e+000, -2.8504300267e+000}
122             const XFLOAT wA[]     ={-1.9680354258e+000, -7.9795948206e+000, -3.8537694364e-001  }
123             for (int o=0  o<NumMOVs  o++)
124             {
125                 XFLOAT x = b[o] + a[o] * min(max(MOV[o], MinVal[o]), MaxVal[o])
126                 y += wA[o] /(1.0 + exp(-x))
127             }
128             y+= 4.6632628918e+000
129         }
130         return y
131
132     }
133
134 XFLOAT MappingLoudness( XFLOAT GainVarInd, XFLOAT LoudnessInd)
```

```
135     {
136         XFLOAT y=0
137         {
138             const int NumMOVs=2
139             XFLOAT MOV[]    ={ GainVarInd, LoudnessInd}
140             const XFLOAT MinVal[] ={-1.0000000000e-007, 7.8036999000e+000   }
141             const XFLOAT MaxVal[] ={3.0000001000e+000, 3.0000000100e+001    }
142             const XFLOAT a[] ={3.0956162823e+000, 1.7092435979e-001 }
143             const XFLOAT b[] ={-2.9997559040e+000, -1.7077417161e+000   }
144             const XFLOAT wA[]     ={-9.6883745774e-001, 4.8110793483e+000   }
145             for (int o=0  o<NumMOVs  o++)
146             {
147                 XFLOAT x = b[o] + a[o] * min(max(MOV[o], MinVal[o]), MaxVal[o])
148                 y += wA[o] /(1.0 + exp(-x))
149             }
150             y += -4.7050057860e-001
151         }
152         return y
153 
154     }
155 
156 BOOL CPairParameters::DisturbanceTimeProcess (POLQA_RESULT_DATA*  pOverviewHolder)
157 {
158     XFLOAT SwitchingLevelDeg = mpBGNSwitchingLevel[1]
159     XFLOAT NoiseLevelSpeechDeg = mpNoiseDuringSpeechdB[1]
160     XFLOAT NoiseLevelSilenceDeg = mpNoiseDuringSilencedB[1]
161     XFLOAT NoiseLevelDifference = 1.0
162     if (NoiseLevelSpeechDeg<0)
163     if (NoiseLevelSilenceDeg<0)
164     if (SwitchingLevelDeg>9.0) SwitchingLevelDeg = 9.0
165     if (NoiseLevelDifference<1.0) NoiseLevelDifference = 1.0
166     if (NoiseLevelDifference>4.0) NoiseLevelDifference = 4.0
167     if (NoiseLevelSpeechDeg>0.0 && NoiseLevelSilenceDeg>0.0 &&
    NoiseLevelSilenceDeg>NoiseLevelSpeechDeg) NoiseLevelDifference =
    pow((NoiseLevelSilenceDeg-NoiseLevelSpeechDeg),0.2)
168 
169     XFLOAT disturbanceL [NUMBER_OF_POWERS_OVER_FREQ] [NUMBER_OF_POWERS_OVER_SYL]
    [NUMBER_OF_POWERS_OVER_TIME]
170     XFLOAT addedDisturbanceL [NUMBER_OF_POWERS_OVER_FREQ] [NUMBER_OF_POWERS_OVER_SYL]
    [NUMBER_OF_POWERS_OVER_TIME]
171     XFLOAT  log10aPureFrqLoudnessMean, hulpLevel1=0.0, hulpLevel2
172     XFLOAT crestFactorCompensation000
173     XFLOAT maxFreqBarkLowLimit = 18
174     XFLOAT maxFreqBarkHighLimit = 22
175 
176     int             i, j, k
177 
178     for (i = 0  i < NUMBER_OF_POWERS_OVER_FREQ  i++) {
179         for (j = 0  j < NUMBER_OF_POWERS_OVER_SYL  j++) {
180             for (k = 0  k < NUMBER_OF_POWERS_OVER_TIME  k++) {
181 
182                 disturbanceL [i][j][k] =       LpqWeight (POLQAHandle, MINIMUM_POWER_SYL +
    STEP_POWER_SYL*j,
183                                                           MINIMUM_POWER_TIME + STEP_POWER_TIME*k,
184                                                            aDisturbance [i], aTimeWeight)
185 
186                 addedDisturbanceL [i][j][k] =      LpqWeight (POLQAHandle, MINIMUM_POWER_SYL +
    STEP_POWER_SYL*j,
187                                                               MINIMUM_POWER_TIME + STEP_POWER_TIME*k,
188                                                                aAddedDisturbance [i], aTimeWeight)
189 
190             }
191         }
192     }
193 
194     fractionNoTimeclip = 1.0
195     crestFactorCompensation000 = crestFactor
196     if (crestFactorCompensation000<80.0)    crestFactorCompensation000=80.0
```

```
197     if (crestFactorCompensation000>200.0)   crestFactorCompensation000=200.0
198     crestFactorCompensation000 = pow(crestFactorCompensation000,0.09)/1.6
199
200     XFLOAT n000 = LppqqWeight(POLQAHandle, 1.0, 0.6, aAddedSilentDisturbance[0], aTimeWeight)
201
202     XFLOAT n000mosIntellCorrection = LppqqWeight(POLQAHandle, 0.9 , 1.4, aAddedSilentDisturbance [0],
    aTimeWeight)
203     XFLOAT n000hulp = n000
204     XFLOAT n000hulp2 = n000
205     XFLOAT n011 = fractionNoTimeclip*LpqWeight(POLQAHandle, 3,2, aAddedSilentDisturbance [0],
    aTimeWeight)
206     XFLOAT n120 = fractionNoTimeclip*LppqqWeight(POLQAHandle, 0.1, 2.0, aAddedSilentDisturbance [0],
    aTimeWeight)
207     XFLOAT n161 = fractionNoTimeclip*LpqWeight(POLQAHandle, 13,2, aAddedSilentDisturbance [1],
    aTimeWeight)
208     XFLOAT noiseDisturbanceCorrection = fractionNoTimeclip*LpqWeight(POLQAHandle, 1, 1,
    aAddedSilentDisturbance [1], aTimeWeight)
209
210     XFLOAT predictedMosOverall, predictedMosUnclipped = -1.0, predictedCvcPureIntelligibility
211     XFLOAT predictedMosPureFrq, predictedMosPureFrqClipped, predictedMosPureNoise,
    predictedMosOverallNearTransparent
212     XFLOAT HulpCvc, testMos1, testMos2, testMos3, testMos4, testMos5, newPolqaPlusMos
213     XFLOAT indicatorMultiplicative, LoudnessCompensation
214     XFLOAT PAMD_MOS_noise, PAMD_MOS_continu, PAMD_MOS_color, PAMD_MOS_loudn
215
216     if (n000mosIntellCorrection<2.5) n000mosIntellCorrection = 2.5
217
218 //POLQAMAIN PART 3
219
220     if (maxFreqBarkSource < maxFreqBarkHighLimit) {
221         if (maxFreqBarkSource < maxFreqBarkLowLimit) {
222             if (n000 < 0.2) n000 = 0.2
223             hulpLevel1 = 1.11*pow((n000 + 1.0), 0.12 / noiseContrastParameter)
224         } else {
225             if (n000 < 0.2) n000 = 0.2
226             hulpLevel1 = 1.11*pow((n000 + 1.0), 0.12 / noiseContrastParameter)
227         }
228     } else {
229         if (n000<0.2) n000 = 0.2
230         hulpLevel1 = 1.11*pow((n000 + 1.0), 0.13 / noiseContrastParameter)
231     }
232 //NOISE in silent intervals compensation for addedDisturbanceL loud noises are reduced in impact
233
234     HulpCvc = 1.0
235
236     for (i = 0  i < NUMBER_OF_POWERS_OVER_FREQ  i++) {
237         for (j = 0  j < NUMBER_OF_POWERS_OVER_SYL  j++) {
238             for (k = 0  k < NUMBER_OF_POWERS_OVER_TIME  k++) {
239
240                 disturbanceL[i][j][k] /= HulpCvc
241                 addedDisturbanceL[i][j][k] /= pow(HulpCvc, 2)
242                 if (maxFreqBarkSource < maxFreqBarkHighLimit) {
243                     if (maxFreqBarkSource < maxFreqBarkLowLimit) {
244                         addedDisturbanceL[i][j][k] /= pow((0.9*n000 + 6.0), hulpLevel1)
245                     } else {
246                         addedDisturbanceL[i][j][k] /= pow((0.9*n000 + 6.0), hulpLevel1)
247
248                     }
249                 } else {
250                     addedDisturbanceL[i][j][k] /= pow((n000 + 6.0), hulpLevel1)
251
252                 }
253
254             }
255         }
256     }
257
258 //FREQUENCY SPURT AND TIME INTEGRATION
```

```
259
260     const XFLOAT d0s0t0 = disturbanceL [0][0][0]
261     const XFLOAT d0s1t3 = disturbanceL [0][1][3]
262     const XFLOAT d0s2t0 = disturbanceL [0][2][0]
263     const XFLOAT d0s2t1 = disturbanceL [0][2][1]
264     const XFLOAT d0s3t1 = disturbanceL [0][3][1]
265     const XFLOAT d0s3t3 = disturbanceL [0][3][3]
266     const XFLOAT d0s6t6 = disturbanceL [0][6][6]
267     const XFLOAT d0s7t0 = disturbanceL [0][7][0]
268     const XFLOAT d0s7t3 = disturbanceL [0][7][3]
269     const XFLOAT d1s1t1 = disturbanceL [1][1][1]
270     const XFLOAT d1s1t2 = disturbanceL [1][1][2]
271     const XFLOAT d1s6t1 = disturbanceL [1][6][1]
272     const XFLOAT d1s7t1 = disturbanceL [1][7][1]
273     const XFLOAT d1s7t3 = disturbanceL [1][7][3]
274     const XFLOAT d2s0t1 = disturbanceL [2][0][1]
275     const XFLOAT d2s0t2 = disturbanceL [2][0][2]
276     const XFLOAT d2s0t3 = disturbanceL [2][0][3]
277     const XFLOAT d2s1t1 = disturbanceL [2][1][1]
278     const XFLOAT d2s1t2 = disturbanceL [2][1][2]
279     const XFLOAT d2s2t1 = disturbanceL [2][2][1]
280     const XFLOAT d2s5t1 = disturbanceL [2][5][1]
281     const XFLOAT d2s7t3 = disturbanceL [2][7][3]
282     const XFLOAT d3s0t2 = disturbanceL [3][0][2]
283     const XFLOAT d3s0t3 = disturbanceL [3][0][3]
284     const XFLOAT d3s1t2 = disturbanceL [3][1][2]
285     const XFLOAT d3s1t1 = disturbanceL [3][1][1]
286     const XFLOAT d3s2t2 = disturbanceL [3][2][2]
287     const XFLOAT d3s3t2 = disturbanceL [3][3][2]
288     const XFLOAT d3s4t2 = disturbanceL [3][4][2]
289     const XFLOAT d4s0t0 = disturbanceL [4][0][0]
290     const XFLOAT d4s2t0 = disturbanceL [4][2][0]
291     const XFLOAT d4s5t2 = disturbanceL [4][5][2]
292     const XFLOAT d4s7t3 = disturbanceL [4][7][3]
293     const XFLOAT d5s1t2 = disturbanceL [5][1][2]
294     const XFLOAT d5s0t3 = disturbanceL [5][0][3]
295     const XFLOAT d5s5t3 = disturbanceL [5][5][3]
296     const XFLOAT d5s6t1 = disturbanceL [5][6][1]
297
298     const XFLOAT a0s0t0 = addedDisturbanceL [0][0][0]
299     const XFLOAT a0s0t1 = addedDisturbanceL [0][0][1]
300     const XFLOAT a0s1t0 = addedDisturbanceL [0][1][0]
301     const XFLOAT a0s1t1 = addedDisturbanceL [0][1][1]
302     const XFLOAT a0s1t3 = addedDisturbanceL [0][1][3]
303     const XFLOAT a0s2t0 = addedDisturbanceL [0][2][0]
304     const XFLOAT a0s2t1 = addedDisturbanceL [0][2][1]
305     const XFLOAT a0s2t3 = addedDisturbanceL [0][2][3]
306     const XFLOAT a0s3t0 = addedDisturbanceL [0][3][0]
307     const XFLOAT a0s4t3 = addedDisturbanceL [0][4][3]
308     const XFLOAT a0s5t1 = addedDisturbanceL [0][5][1]
309     const XFLOAT a0s5t2 = addedDisturbanceL [0][5][2]
310     const XFLOAT a0s6t1 = addedDisturbanceL [0][6][1]
311     const XFLOAT a0s7t1 = addedDisturbanceL [0][7][1]
312     const XFLOAT a0s7t2 = addedDisturbanceL [0][7][2]
313     const XFLOAT a1s0t0 = addedDisturbanceL [1][0][0]
314     const XFLOAT a1s3t2 = addedDisturbanceL [1][3][2]
315     const XFLOAT a1s4t1 = addedDisturbanceL [1][4][1]
316     const XFLOAT a1s5t1 = addedDisturbanceL [1][5][1]
317     const XFLOAT a1s7t1 = addedDisturbanceL [1][7][1]
318     const XFLOAT a2s1t2 = addedDisturbanceL [2][1][2]
319     const XFLOAT a2s6t3 = addedDisturbanceL [2][6][3]
320     const XFLOAT a3s0t2 = addedDisturbanceL [3][0][2]
321     const XFLOAT a3s2t2 = addedDisturbanceL [3][2][2]
322     const XFLOAT a3s7t1 = addedDisturbanceL [3][7][1]
323     const XFLOAT a4s4t3 = addedDisturbanceL [4][4][3]
324     const XFLOAT a5s1t1 = addedDisturbanceL [5][1][1]
325
326     XFLOAT rawFrqInd = log10(aPureFrqLoudnessMean+0.0001)
```

```
327
328     XFLOAT aPureFrqLoudnessMean1 = log10(aPureFrqLoudnessMean+0.0001)
329     if (aPureFrqLoudnessMean1<1.0) aPureFrqLoudnessMean1 = 1.0
330
331     XFLOAT aPureFrqLoudnessMean2 = log10(aPureFrqLoudnessMean+0.0001)
332
333     aPureFrqLoudnessMean3 = log10(aPureFrqLoudnessMean+0.0001)
334     if (aPureFrqLoudnessMean3<3.0) aPureFrqLoudnessMean3 = 3.0
335
336     log10aPureFrqLoudnessMean = log10(aPureFrqLoudnessMean + 0.0001)
337     if (log10aPureFrqLoudnessMean<1.5) log10aPureFrqLoudnessMean = 1.5
338
339     predictedMosPureFrq = -0.8453*log10aPureFrqLoudnessMean + 6.0131 + 0.7
340     if (predictedMosPureFrq < 0.8) predictedMosPureFrq = 0.8
341     if (predictedMosPureFrq > 5.0) predictedMosPureFrq = 5.0
342
343     distortedLoudnessTimbreLow /= 450.0
344
345     distortedLoudnessTimbreHigh /= 1.0e5
346     distortedLoudnessTimbreHighSilent /= 1.0e5
347
348     if (distortedLoudnessTimbreLow > 1.0)   distortedLoudnessTimbreLow = 1.0
349
350     XFLOAT hulp1, hulp2, hulp3, hulpA, hulpB, hulpFrq
351     hulpFrq = aPureFrqLoudnessMean2
352     hulpFrq -= 3.9
353     if (hulpFrq<0.0) hulpFrq = 0.0
354     hulpFrq = -0.2*hulpFrq
355
356     newPolqaPlusMos = -0.601*d1s1t1 - a0s1t1
357
358     //MAPPING TO INTERMEDIATE MOS SCORE
359     hulp1 = d0s3t1
360     if (hulp1>45.0) hulp1 = 45.0
361     testMos2 = -0.508*aPureFrqLoudnessMean2 - 0.341*hulp1 - a0s0t1
362     if (testMos2<-30.0) testMos2 = -30.0
363     testMos2 = 0.0062*testMos2*testMos2 + 0.3504*testMos2 + 5.7406
364     testMos2 = 1.1857*testMos2 - 0.8029
365     if (testMos2<0.8) testMos2 = 0.8
366     if (testMos2>4.9) testMos2 = 4.9
367
368     hulp2 = a0s0t1
369     if (hulp2>4.0) hulp2 = 4.0
370     testMos3 = -0.508*aPureFrqLoudnessMean2 - 0.341*d0s3t1 - hulp2
371     if (testMos3<-30.0) testMos3 = -30.0
372     testMos3 = 0.0062*testMos3*testMos3 + 0.3504*testMos3 + 5.7406
373     testMos3 = 1.1857*testMos3 - 0.8029
374     if (testMos3>4.9) testMos3 = 4.9
375
376     hulp2 = a0s0t1
377     if (maxFreqBarkSource < maxFreqBarkHighLimit) {
378         if (maxFreqBarkSource < maxFreqBarkLowLimit) {
379             if (aPureFrqLoudnessMean2<3.3) aPureFrqLoudnessMean2 = 3.3
380             if (hulp2>5.0) hulp2 = 5.0
381             testMos4 = -0.46*aPureFrqLoudnessMean2 - 0.34*d0s3t1 - hulp2
382         } else {
383             if (aPureFrqLoudnessMean2<3.1) aPureFrqLoudnessMean2 = 3.1
384             if (hulp2>5.0) hulp2 = 5.0
385             testMos4 = -0.46*aPureFrqLoudnessMean2 - 0.36*d0s3t1 - 0.95*hulp2
386         }
387     } else {
388         if (aPureFrqLoudnessMean2<3.1) aPureFrqLoudnessMean2 = 3.1
389         if (hulp2>5.5) hulp2 = 5.5
390         testMos4 = -0.44*aPureFrqLoudnessMean2 - 0.38*d0s3t1 - hulp2
391     }
392     if (testMos4<-30.0) testMos4 = -30.0
393     testMos4 = 0.0062*testMos4*testMos4 + 0.3504*testMos4 + 5.7406
394     testMos4 = 1.1857*testMos4 - 0.8029
```

```
    if (testMos4<0.7) testMos4 = 0.7

    hulp1 = d0s3t1
    if (hulp1>50.0) hulp1 = 50.0
    hulp2 = a0s0t1
    if (hulp2>6.0) hulp2 = 6.0
    testMos5 = -0.508*aPureFrqLoudnessMean2 - 0.341*hulp1 - hulp2
    if (testMos5<-30.0) testMos5 = -30.0
    testMos5 = 0.0062*testMos5*testMos5 + 0.3504*testMos5 + 5.7406
    testMos5 = 1.1857*testMos5 - 0.8029
    if (testMos5<0.8) testMos5 = 0.8
    if (testMos5>4.9) testMos5 = 4.9

    predictedMosOverall = testMos4
    if (!(predictedMosOverall == predictedMosOverall)) DebugBreak()

    testMos1 = predictedMosOverall
    testMos1 = testMos1 + 0.024*d2s0t2
    testMos1 = 1.1913*testMos1 - 1.1039
    if (testMos1<0.8) testMos1 = 0.8
    if (testMos1>4.9) testMos1 = 4.9

    testMos1 = predictedMosOverall

    XFLOAT hulpMos
    hulpMos = predictedMosOverall
    hulpLevel1 = predictedMosOverall
    hulpLevel2 = predictedMosOverall
    if (hulpLevel1<2.0) hulpLevel1 = 2.0
    if (hulpLevel2<1.0) hulpLevel2 = 1.0

    predictedMosPureFrqClipped = predictedMosPureFrq
    if (predictedMosPureFrqClipped > 4.0) predictedMosPureFrqClipped = 4.0
    indicatorMultiplicative = abs(MNRU_indicator)
    if (aAvgDistortedPower>1.0e8)
        LoudnessCompensation = 1.0e8
    else LoudnessCompensation = aAvgDistortedPower
    LoudnessCompensation = 3.0e-8*LoudnessCompensation+1.8661
    indicatorMultiplicative*= LoudnessCompensation
    MNRU_indicator = MNRU_indicator*predictedMosPureFrqClipped*pow(aAvgDistortedPower, 0.2) / 5.0e3

    predictedMosOverall = predictedMosOverall -
0.5*correlationOriginalWithDisturbanceCompensation000ForMNRU - MNRU_indicator -
noiseIndicatorHighBandsCompensation000
    predictedMosOverall -= 5.0*extremeTimeclipFramesCompensation000

    if (aListeningCondition == STANDARD_IRS) {
        predictedMosOverall = predictedMosOverall/pow(CVCratioSNRlevelRangecompensation0001, 0.3) -
1.0*frameFlatnessDistortedAvgCompensationSilent000
        predictedMosOverall -= 0.03*n000
        predictedMosOverall /= distortedLoudnessTimbrePerFrameNarrowbandAvg000

    }
    else {
        predictedMosOverall = predictedMosOverall*pow(CVCratioSNRlevelRangecompensation0001, 0.5) -
1.0*frameFlatnessDistortedAvgCompensationSilent000

        if (frameCorrelationTimeDisturbanceAvgCompensation000<0.99)
frameCorrelationTimeDisturbanceAvgCompensation000 = 0.99
        hulp1 = predictedMosOverall
        if (hulp1 < 1.0) hulp1 = 1.0
        hulp1 = pow(hulp1, 0.05)
        hulp1 = pow(frameCorrelationTimeDisturbanceAvgCompensation000, hulp1)
        hulp1 = frameCorrelationTimeDisturbanceAvgCompensation000
        predictedMosOverall /= hulp1

    }
    freqShiftChangesAvg *= predictedMosOverall
```

```
458     predictedMosOverall += freqShiftChangesAvg/100.0
459     predictedMosOverall -= (scaleDistortion2 / 14.0e3 + scaleDistortion4 / 16.0e3)
460
461     if (predictedMosOverall>3.5) predictedMosOverall -=
    (predictedMosOverall/2-0.75)*distortedLoudnessTimbrePerFrameLoudAvg000
462     testMos3 /= pow(aGlobalCompensation3, 0.1)
463
464     hulp1 = aGlobalCompensation1 - 1.0
465     hulp1 *= 2.0
466     if (hulp1 < 1.0) hulp1 = 1.0
467     predictedMosOverall /= pow(aGlobalCompensation1, 0.06 / hulp1)
468     hulp1 = (aGlobalCompensation1loud - 1.2)
469     if (hulp1 < 0.0) hulp1 = 0.0
470     if (hulp1 > 0.15) hulp1 = 0.15
471     hulp2 = 5.0 - predictedMosOverall
472     if (hulp2<1.0) hulp2 = 1.0
473     if (hulp2>4.0) hulp2 = 4.0
474     predictedMosOverall -= hulp1 / (2.0*hulp2)
475
476     predictedMosUnclipped = predictedMosOverall
477     if (predictedMosOverall < 1.00) predictedMosOverall = 1.00
478
479     if (aListeningCondition == STANDARD_IRS) {
480         predictedMosOverall = 0.0423*pow(predictedMosOverall, 3) - 0.1791*pow(predictedMosOverall, 2)
    + 0.7639*predictedMosOverall + 0.377 + 0.02
481
482         if (predictedMosOverall > 5.2) predictedMosOverall = 5.2
483         predictedMosOverall = -0.0060379*pow(predictedMosOverall, 3) -
    0.1273410*pow(predictedMosOverall, 2) + 1.8555029*predictedMosOverall - 0.7199955
484         if (predictedMosOverall < 1.00) predictedMosOverall = 1.00
485
486         if (predictedMosOverall > 4.50) predictedMosOverall = 4.50
487     }
488     else
489
490     {
491         if (maxFreqBarkSource < maxFreqBarkHighLimit)
492         {
493             predictedMosOverall = 0.0274*pow(predictedMosOverall, 3) +
    0.0233*pow(predictedMosOverall, 2) + 0.0823*predictedMosOverall + 0.9162 + 0.02
494             if (predictedMosOverall > 4.75) predictedMosOverall = 4.75
495         }
496         else
497         {
498             predictedMosOverall = 0.0459*pow(predictedMosOverall, 3) -
    0.0991*pow(predictedMosOverall, 2) + 0.2658*predictedMosOverall + 0.7941 + 0.02
499
500             if (predictedMosOverall > 5.15) predictedMosOverall = 5.15
501             predictedMosOverall = -0.06883*pow(predictedMosOverall, 3) +
    0.55725*pow(predictedMosOverall, 2) - 0.24624*predictedMosOverall + 0.686
502             if (predictedMosOverall < 1.00) predictedMosOverall = 1.00
503
504         }
505     }
506
507     XFLOAT predictedMosOverall_WIDE_H = predictedMosOverall
508
509     XFLOAT shiftStabilityIndicator1 = -0.136*constantDelayIndicator + fraction_of_sections_inserted
510     hulp1 = (fraction_of_sections_inserted - 0.00803) / (0.6238 - 0.00803)
511     hulp2 = (a0s4t3 - 0.48483) / (19.4713 - 0.48483)
512     hulp3 = (d0s2t0 - 1.13894) / (21.3116 - 1.13894)
513     hulpA = 1.71017 - 4.95073*hulp1 + 0.90607*hulp2 - 0.605642*hulp3
514     hulpB = 4.20589 - 9.34612*hulp1 - 3.95103*hulp2 + 3.06325*hulp3
515     hulp1 = hulpA / hulpB
516     XFLOAT shiftStabilityIndicator2 = 4.31805 -1.96177*hulpA*hulpB/(hulpA*hulpA+hulpB*hulpB)
517     if (n120>37.0) n120 = 37.0
518     if (n120<19)
519         predictedMosPureNoise = -0.0009*pow(n120, 3) + 0.0352*pow(n120, 2) - 0.494*n120 + 4.75
```

```
    else
        predictedMosPureNoise = 0.0019*pow(n120, 2) - 0.1677*n120 + 4.52
    predictedMosPureNoise *= pow(aGlobalCompensation3, 0.6)
    if (predictedMosPureNoise < 1.0) predictedMosPureNoise = 1.0
    if (predictedMosPureNoise > 4.75) predictedMosPureNoise = 4.75

    predictedMosOverallNearTransparent = predictedMosOverall - 0.574*aPureFrqLoudnessMean2

    pOverviewHolder->m_PredictedMos = predictedMosOverall
    pOverviewHolder->m_PredictedMosUnclipped = predictedMosUnclipped
    pOverviewHolder->m_PredictedMosPureFrq = predictedMosPureFrq
    pOverviewHolder->m_PredictedMosPureNoise = predictedMosPureNoise

    aDistortedLoudnessMeanIndicator2 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator3 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator4 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator5 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator6 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator7 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator8 = aDistortedLoudnessMeanIndicator1
    aDistortedLoudnessMeanIndicator9 = aDistortedLoudnessMeanIndicator1
    if (aDistortedLoudnessMeanIndicator1<6.0) {
        aDistortedLoudnessMeanIndicator1 = pow((7.0 - aDistortedLoudnessMeanIndicator1), 2)
    }
    else {
        aDistortedLoudnessMeanIndicator1 = aDistortedLoudnessMeanIndicator1 - 5.0
    }
    if (aDistortedLoudnessMeanIndicator2<6.0) {
        aDistortedLoudnessMeanIndicator2 = pow((7.0 - aDistortedLoudnessMeanIndicator2), 5)
    }
    else {
        aDistortedLoudnessMeanIndicator2 = pow((aDistortedLoudnessMeanIndicator2 - 5.0), 2)
    }
    if (aDistortedLoudnessMeanIndicator3<6.0) {
        aDistortedLoudnessMeanIndicator3 = pow((7.0 - aDistortedLoudnessMeanIndicator3), 10.0)
    }
    else {
        aDistortedLoudnessMeanIndicator3 = pow((aDistortedLoudnessMeanIndicator3 - 5.0), 5)
    }

    if (aDistortedLoudnessMeanIndicator4<4.0) {
        aDistortedLoudnessMeanIndicator4 = pow((5.0 - aDistortedLoudnessMeanIndicator4), 2)
    }
    else {
        aDistortedLoudnessMeanIndicator4 = aDistortedLoudnessMeanIndicator4 - 3.0
    }
    if (aDistortedLoudnessMeanIndicator5<4.0) {
        aDistortedLoudnessMeanIndicator5 = pow((5.0 - aDistortedLoudnessMeanIndicator5), 5)
    }
    else {
        aDistortedLoudnessMeanIndicator5 = pow((aDistortedLoudnessMeanIndicator5 - 3.0), 2)
    }
    if (aDistortedLoudnessMeanIndicator6<4.0) {
        aDistortedLoudnessMeanIndicator6 = pow((5.0 - aDistortedLoudnessMeanIndicator6), 10.0)
    }
    else {
        aDistortedLoudnessMeanIndicator6 = pow((aDistortedLoudnessMeanIndicator6 - 3.0), 5)
    }

    if (aDistortedLoudnessMeanIndicator7<2.0) {
        aDistortedLoudnessMeanIndicator7 = pow((3.0 - aDistortedLoudnessMeanIndicator7), 2)
    }
    else {
        aDistortedLoudnessMeanIndicator7 = aDistortedLoudnessMeanIndicator7 - 2.0
    }

    if (aDistortedLoudnessMeanIndicator8<2.0) {
        aDistortedLoudnessMeanIndicator8 = pow((3.0 - aDistortedLoudnessMeanIndicator8), 5)
```

```
588     }
589     else {
590         aDistortedLoudnessMeanIndicator8 = pow((aDistortedLoudnessMeanIndicator8 - 2.0), 2)
591     }
592     if (aDistortedLoudnessMeanIndicator9<2.0) {
593         aDistortedLoudnessMeanIndicator9 = pow((3.0 - aDistortedLoudnessMeanIndicator9), 10.0)
594     }
595     else {
596         aDistortedLoudnessMeanIndicator9 = pow((aDistortedLoudnessMeanIndicator9 - 2.0), 5)
597     }
598     aDistortedLoudnessMeanIndicator3 = aGlobalCompensation1
599     aDistortedLoudnessMeanIndicator4 = aGlobalCompensation1
600     aDistortedLoudnessMeanIndicator5 = aGlobalCompensation1
601     aDistortedLoudnessMeanIndicator6 = aGlobalCompensation1
602     aDistortedLoudnessMeanIndicator7 = aGlobalCompensation1 - 1.5
603     aDistortedLoudnessMeanIndicator8 = aGlobalCompensation1 - 1.5
604     aDistortedLoudnessMeanIndicator9 = aGlobalCompensation1 - 1.5
605     if (aDistortedLoudnessMeanIndicator3<1.5) aDistortedLoudnessMeanIndicator3 = 1.5
606     if (aDistortedLoudnessMeanIndicator4<2.0) aDistortedLoudnessMeanIndicator4 = 2.0
607     aDistortedLoudnessMeanIndicator5 =
    aDistortedLoudnessMeanIndicator3*aDistortedLoudnessMeanIndicator3
608     aDistortedLoudnessMeanIndicator6 =
    aDistortedLoudnessMeanIndicator4*aDistortedLoudnessMeanIndicator4
609     if (aDistortedLoudnessMeanIndicator7<0.0) aDistortedLoudnessMeanIndicator7 = 0.0
610     if (aDistortedLoudnessMeanIndicator8<0.5) aDistortedLoudnessMeanIndicator8 = 0.5
611     constantDelayIndicator *= 1.04f
612
613     pOverviewHolder->aIndicator[0] = (XFLOAT)predictedMosOverall
614
615     pOverviewHolder->aIndicator[1] = 0
616     pOverviewHolder->aIndicator[2] = 0
617     pOverviewHolder->aIndicator[3] = 0
618     pOverviewHolder->aIndicator[4] = 0
619
620     pOverviewHolder->aIndicator[5] = (XFLOAT)pOverviewHolder->m_GlobalDelay
621     pOverviewHolder->aIndicator[6] = (XFLOAT)predictedMosOverall_WIDE_H
622     pOverviewHolder->aIndicator[7] = (XFLOAT)aGlobalCompensation1
623     pOverviewHolder->aIndicator[8] = (XFLOAT)aGlobalCompensation1loud
624     pOverviewHolder->aIndicator[9] = (XFLOAT)aPureFrqLoudnessMean2
625     pOverviewHolder->aIndicator[10] = (XFLOAT)envelopeContinuityCompensation000
626     pOverviewHolder->aIndicator[11] = (XFLOAT)predictedMosPureFrq
627     pOverviewHolder->aIndicator[12] = (XFLOAT)reverbIndicator
628
629     pOverviewHolder->aIndicator[13] = (XFLOAT)n000
630     pOverviewHolder->aIndicator[14] = (XFLOAT)n011
631     pOverviewHolder->aIndicator[15] = (XFLOAT)freqShiftChangesAvg
632     pOverviewHolder->aIndicator[16] = (XFLOAT)n000mosIntellCorrection
633     pOverviewHolder->aIndicator[17] = (XFLOAT)predictedMosPureNoise
634     pOverviewHolder->aIndicator[18] = (XFLOAT)shiftStabilityIndicator1
635     pOverviewHolder->aIndicator[19] = (XFLOAT)distortedLoudnessTimbrePerFrameLoudAvg000
636     pOverviewHolder->aIndicator[20] = (XFLOAT)constantDelayIndicator
637     pOverviewHolder->aIndicator[21] = (XFLOAT)frameFlatnessDisturbanceAvgCompensation000silent
638     pOverviewHolder->aIndicator[22] = (XFLOAT)distortedLoudnessTimbreHighPerFrameAvgActive000
639     pOverviewHolder->aIndicator[23] = (XFLOAT)frameFlatnessDistortedAvgCompensationSilent000
640     pOverviewHolder->aIndicator[24] = (XFLOAT)envelopeContinuityCompensation000
641     pOverviewHolder->aIndicator[25] = (XFLOAT)CVCratioSNRlevelRangecompensation0001
642     pOverviewHolder->aIndicator[26] = (XFLOAT)SNRloudnessRatioOKratio
643     pOverviewHolder->aIndicator[27] = (XFLOAT)crestFactorPAMD
644
645     pOverviewHolder->aIndicator[28] = (XFLOAT)avgPitchLoudFramesRise000
646     pOverviewHolder->aIndicator[29] = (XFLOAT)avgPitchLoudFramesDrop000
647     pOverviewHolder->aIndicator[30] = (XFLOAT)avgPitchLoudFrames000
648     pOverviewHolder->aIndicator[31] = (XFLOAT)pitchFreqReference
649
650     pOverviewHolder->aIndicator[32] = (XFLOAT)frameCorrelationTimeDisturbanceAvgCompensation000
651     pOverviewHolder->aIndicator[33] = (XFLOAT)frameCorrelationTimeDisturbanceAvgCompensation000silent
652     pOverviewHolder->aIndicator[34] = (XFLOAT)frameFlatnessDisturbanceAvgCompensation000active
653     pOverviewHolder->aIndicator[35] = (XFLOAT)aAvgDistortedPower
```

```
654     pOverviewHolder->aIndicator[36] = (XFLOAT)aAvgActiveDistortedPower
655     pOverviewHolder->aIndicator[37] = (XFLOAT)MNRU_indicator
656     pOverviewHolder->aIndicator[38] =
    (XFLOAT)correlationOriginalWithDisturbanceCompensation000ForMNRU
657     pOverviewHolder->aIndicator[39] = (XFLOAT)scaleDistortion2
658     pOverviewHolder->aIndicator[40] = (XFLOAT)scaleDistortion4
659
660     pOverviewHolder->aIndicator[41] = (XFLOAT)fraction_of_sections_inserted
661     pOverviewHolder->aIndicator[42] = (XFLOAT)fraction_of_sections_critical
662     pOverviewHolder->aIndicator[43] = (XFLOAT)fraction_of_sections_invalid
663     pOverviewHolder->aIndicator[44] = (XFLOAT)frameCorrelationTimeCompensationDifferenceAvg
664     pOverviewHolder->aIndicator[45] = (XFLOAT)pitchIndicator
665     pOverviewHolder->aIndicator[46] = (XFLOAT)superLowbandCorrection
666     pOverviewHolder->aIndicator[47] = (XFLOAT)superWidebandCorrection
667     pOverviewHolder->aIndicator[48] = (XFLOAT)fullbandCorrection
668
669     ASSERT(NUMBER_OF_POWER_INDICATORS == 4)
670     pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 0] = (XFLOAT)(10 * log10(1e-8 +
    aAvgDistortedPower / 1e7))
671     pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 1] = (XFLOAT)(10 * log10(1e-8 +
    aAvgOriginalPower / 1e7))
672     pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 2] = (XFLOAT)(10 * log10(1e-8 +
    aAvgActiveDistortedPower / 1e7))
673     pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 3] = (XFLOAT)(10 * log10(1e-8 +
    aAvgActiveOriginalPower / 1e7))
674
675     for (i = 0  i < NUMBER_OF_POWERS_OVER_FREQ  i++) {
676         for (j = 0  j < NUMBER_OF_POWERS_OVER_SYL  j++) {
677             for (k = 0  k < NUMBER_OF_POWERS_OVER_TIME  k++) {
678                 pOverviewHolder->aIndicator[NUMBER_OF_POWER_INDICATORS +
    NUMBER_OF_PREDICTION_INDICATORS + (i*NUMBER_OF_POWERS_OVER_SYL +
    j)*NUMBER_OF_POWERS_OVER_TIME + k]
679                     = disturbanceL[i][j][k]
680                 pOverviewHolder->aIndicator[NUMBER_OF_POWER_INDICATORS +
    NUMBER_OF_PREDICTION_INDICATORS + ((NUMBER_OF_POWERS_OVER_FREQ +
    i)*NUMBER_OF_POWERS_OVER_SYL + j)*NUMBER_OF_POWERS_OVER_TIME + k]
681                     = addedDisturbanceL[i][j][k]
682
683             }
684         }
685     }
686
687     return TRUE
688 }
689
690 }
691
692
```