```
    typedef double XFLOAT;
    typedef double OTA_FLOAT;

namespace SQFUNCS_POLQA_INTERNAL
{
int SQcalcASLandNoise(  XFLOAT       *envelope,
                        short       *sVADprofile,
                        long  const numFrames,
                        int   const iStepPerFrame,
                        XFLOAT const minEnvValdB,
                        int   const sampFreq,
                        XFLOAT       *fSpeechAct,
                        XFLOAT       *fActSpeechLevel,
                        XFLOAT       *fNoiseLevel,
                        XFLOAT       *fActSpeechTresh,
                        XFLOAT* const maxSigLenBuff)
{
   XFLOAT   fNoiseEstim     = (XFLOAT)0.0,
            fNoiseEstim_pow = (XFLOAT)1.0,
            fASLEstim       = (XFLOAT)0.0,
            fASLEnergy      = (XFLOAT)0.0,
            fSignalThresh   = (XFLOAT)0.0,
            fPrevThresh     = (XFLOAT)1000.0,
            pfA_EnvDistr[100],
            fTotRMS         = (XFLOAT)0.0;
    int     iCurUttLen      = 0,
            numActFrames    = 0,
            startVADPos     = 0,
            endVADPos       = numFrames-1,
            prevStartVADPos = startVADPos,
            prevEndVADPos   = endVADPos;
    bool    foundTooMuchAct = false;

    XFLOAT const ENV_UNIT    = (XFLOAT)0.032*(XFLOAT)0.5 /
(iStepPerFrame/(XFLOAT)sampFreq);
    int const MIN_UTT_LEN   = ((int)(((3 * ENV_UNIT) > 0) ? (3 * ENV_UNIT)+0.5f : (3 *
ENV_UNIT)-0.5f));
    int const OVERHANG_FWD  = ((int)(((7 * ENV_UNIT) > 0) ? (7 * ENV_UNIT)+0.5f : (7 *
ENV_UNIT)-0.5f));
    int const OVERHANG_BACK = ((int)(((3 * ENV_UNIT) > 0) ? (3 * ENV_UNIT)+0.5f : (3 *
ENV_UNIT)-0.5f));
    int const MAX_NUM_ITER  = 10;

    if (envelope == NULL || sVADprofile == NULL || numFrames < 10 || minEnvValdB >=
(XFLOAT)0.0)
        return -1;

    fCalc_DistrOfVector(envelope, numFrames, minEnvValdB, (XFLOAT)0.0, 100,
pfA_EnvDistr);
    fNoiseEstim = fCalc_PercentilOfDistrVector((XFLOAT)80.0, minEnvValdB, (XFLOAT)0.0,
100, pfA_EnvDistr);
    fASLEstim   = fCalc_PercentilOfDistrVector((XFLOAT)20.0, minEnvValdB, (XFLOAT)0.0,
100, pfA_EnvDistr);
    fSignalThresh = (fASLEstim + 2*fNoiseEstim) / (XFLOAT)3.0;
    fSignalThresh = pow((XFLOAT)10.0, fSignalThresh/(XFLOAT)10.0);

    if (maxSigLenBuff == NULL)
    {
        vsdiv (envelope, (XFLOAT)10.0, envelope, numFrames);
        vexp10(envelope,        envelope, numFrames);
    }
    else
    {
        vsdiv (envelope, (XFLOAT)10.0, envelope, numFrames);
        vexp10(envelope,        envelope, numFrames, maxSigLenBuff);
    }
    sivclr(sVADprofile, numFrames);
    XFLOAT powFloor = (((pow((XFLOAT)10.0, minEnvValdB/(XFLOAT)10.0)) > ((XFLOAT)1e-16))
? (pow((XFLOAT)10.0, minEnvValdB/(XFLOAT)10.0)) : ((XFLOAT)1e-16));

    for (int k = 0; k < MAX_NUM_ITER; k++)
    {
        iCurUttLen      = 0;
        fPrevThresh     = fSignalThresh;
```

```
        fNoiseEstim_pow = pow((XFLOAT)10.0, fNoiseEstim / (XFLOAT)10.0);

        for (int i = startVADPos; i <= endVADPos; i++)
        {
            if (envelope[i] > fSignalThresh)
            {
                sVADprofile[i] = SQ_VAD_ACT_SPEECH;

                if (!iCurUttLen)
                    for (int j = i-1; j >= (((startVADPos) > (i-OVERHANG_BACK)) ?
(startVADPos) : (i-OVERHANG_BACK)); j--)
                        if (envelope[i] < fNoiseEstim_pow)
                            break;
                        else
                            sVADprofile[j] = SQ_VAD_OVERHANG;

                iCurUttLen++;
            }
            else
            {
                if (iCurUttLen >= MIN_UTT_LEN)
                {
                    int uttEndPos = i + OVERHANG_FWD;
                    for (; i < (((uttEndPos) < (endVADPos+1)) ? (uttEndPos) :
(endVADPos+1)); i++)
                        if (envelope[i] < fNoiseEstim_pow)
                            break;
                        else if (envelope[i] > fSignalThresh && envelope[i-1] >
fSignalThresh)
                            break;
                        else
                            sVADprofile[i] = SQ_VAD_OVERHANG;
                    i--;
                }
                else if (iCurUttLen > 0 &&
                         i-iCurUttLen-OVERHANG_BACK-1 >= startVADPos &&
                         sVADprofile[i-iCurUttLen-OVERHANG_BACK-1] == SQ_VAD_NO_SPEECH)
                {
                    bool uttFollowing = true;
                    for (int j = i+OVERHANG_BACK; j < (((i+OVERHANG_BACK +
2*MIN_UTT_LEN) < (endVADPos+1)) ? (i+OVERHANG_BACK + 2*MIN_UTT_LEN)
: (endVADPos+1)); j++)
                        uttFollowing &= envelope[j] > fSignalThresh;

                    for (int j = i; j >= (((startVADPos) > (i-iCurUttLen-OVERHANG_BACK))
? (startVADPos) : (i-iCurUttLen-OVERHANG_BACK)); j--)
                        sVADprofile[j] = uttFollowing ? SQ_VAD_OVERHANG :
SQ_VAD_NO_SPEECH;
                }
                else
                    sVADprofile[i] = SQ_VAD_NO_SPEECH;

                iCurUttLen = 0;
            }
        }

        foundTooMuchAct = AnalyzeVADProfile(sVADprofile, &startVADPos, &endVADPos,
                                            numFrames, sampFreq, iStepPerFrame);

        XFLOAT minBoundary= (0.32*sampFreq)/iStepPerFrame;
        if (abs(startVADPos-prevStartVADPos) > minBoundary ||
            abs(endVADPos  -prevEndVADPos)   > minBoundary)
        {
            if (maxSigLenBuff == NULL)
            {
                matbThresh1(envelope, numFrames, powFloor, MAT_LT);
                vlog10(envelope,               envelope, numFrames);
                vsmul (envelope, (XFLOAT)10.0,        envelope, numFrames);
                vclip (envelope, -minEnvValdB, envelope, numFrames);
            }
            else
            {
                matbThresh1(envelope, numFrames, powFloor, MAT_LT);
                vlog10(envelope,                   maxSigLenBuff, numFrames);
                vsmul (maxSigLenBuff, (XFLOAT)10.0,        envelope,      numFrames);
                vclip (envelope,      -minEnvValdB, envelope,      numFrames);
```

```
            }

            fCalc_DistrOfVector(envelope+startVADPos, endVADPos-startVADPos+1,
minEnvValdB, 0.0f, 100, pfA_EnvDistr);
            fNoiseEstim = fCalc_PercentilOfDistrVector(80.0f, minEnvValdB, 0.0f, 100,
pfA_EnvDistr);
            fASLEstim   = fCalc_PercentilOfDistrVector(20.0f, minEnvValdB, 0.0f, 100,
pfA_EnvDistr);
            fSignalThresh   = (fASLEstim + 2*fNoiseEstim) / 3.0f;
            fSignalThresh   = pow((XFLOAT)10.0, fSignalThresh/(XFLOAT)10.0);
            prevStartVADPos = startVADPos;
            prevEndVADPos   = endVADPos;

            if (maxSigLenBuff == NULL)
            {
                vsdiv (envelope, 10.0f, envelope, numFrames);
                vexp10(envelope,        envelope, numFrames);
            }
            else
            {
                vsdiv (envelope, 10.0f, envelope, numFrames);
                vexp10(envelope,        envelope, numFrames, maxSigLenBuff);
            }
            fPrevThresh = 1000.0f;

            continue;
        }
        else if (foundTooMuchAct && k <= MAX_NUM_ITER - 3)
        {
            fSignalThresh *= 8.0f;
            k = MAX_NUM_ITER - 3;
            continue;
        }

        numActFrames = 0;
        for (int i = startVADPos; i <= endVADPos; i++)
            if (sVADprofile[i] >= SQ_VAD_OVERHANG)
                numActFrames++;

        if (numActFrames > 10 &&
            numActFrames / (XFLOAT)(endVADPos - startVADPos + 1) > (XFLOAT)0.05 &&
            endVADPos - startVADPos + 1 > numActFrames)
        {
            fTotRMS = matSum(envelope+startVADPos, endVADPos - startVADPos + 1);

            fASLEnergy = (XFLOAT)0.0;
            for (int i = startVADPos; i <= endVADPos; i++)
                if (sVADprofile[i] >= SQ_VAD_OVERHANG)
                    fASLEnergy += envelope[i];

            fNoiseEstim = 10.0f * log10((((0) > ((fTotRMS-fASLEnergy) / (endVADPos -
startVADPos + 1 - numActFrames))) ? (0) : ((fTotRMS-fASLEnergy) / (endVADPos
- startVADPos + 1 - numActFrames))) + 1e-12f);
            fASLEstim   = 10.0f * log10(fASLEnergy / numActFrames + 1e-12f);
            fNoiseEstim = (((minEnvValdB) > (fNoiseEstim)) ? (minEnvValdB) :
(fNoiseEstim));
            fASLEstim   = (((minEnvValdB) > (fASLEstim)) ? (minEnvValdB) : (fASLEstim));

            fSignalThresh = (fASLEstim + 2*fNoiseEstim) / (XFLOAT)3.0;
            fSignalThresh = pow((XFLOAT)10.0, fSignalThresh/(XFLOAT)10.0);

            if (fSignalThresh == fPrevThresh)
                break;
        }
        else
            break;
    }

    for (int i = 0; i < startVADPos; i++)
        sVADprofile[i] = SQ_VAD_NO_SPEECH;
    for (int i = endVADPos+1; i < numFrames; i++)
        sVADprofile[i] = SQ_VAD_NO_SPEECH;

    if (maxSigLenBuff == NULL)
    {
        matbThresh1(envelope, numFrames, powFloor, MAT_LT);
```

```
        vlog10(envelope,               envelope, numFrames);
        vsmul (envelope, 10.0f,        envelope, numFrames);
        vclip (envelope, -minEnvValdB, envelope, numFrames);
    }
    else
    {
        matbThresh1(envelope, numFrames, powFloor, MAT_LT);
        vlog10(envelope,                   maxSigLenBuff, numFrames);
        vsmul (maxSigLenBuff, 10.0f,       envelope,      numFrames);
        vclip (envelope,      -minEnvValdB, envelope,     numFrames);
    }

    if (fActSpeechLevel != NULL)
        *fActSpeechLevel = fASLEstim;
    if (fNoiseLevel != NULL)
        *fNoiseLevel     = fNoiseEstim;
    if (fSpeechAct != NULL)
        *fSpeechAct      = numActFrames / (XFLOAT)numFrames;
    if (fActSpeechTresh != NULL)
        *fActSpeechTresh = 10.0f * log10(fSignalThresh + 1e-16f);

    return 0;
}

bool AnalyzeVADProfile (short *VADProfile, int *startVADPos, int *endVADPos,
                        int const numFrames, int const sampFreq, int const
iStepPerFrame)
{
    int numberOfInactStart   = 0;
    int numberOfInactEnd     = 0;
    int maxNumberOfConsecAct = 0;
    int numberOfConsecAct    = 0;

    float const MAX_PERC_OF_CONSECUTIVE_ACTIVE_FRAMES = 0.666f;

    bool countingInactStart = true;
    bool countingInactEnd   = true;

    for(int i = *startVADPos; i <= *endVADPos;  ++i)
    {
        if(VADProfile[i] >= SQ_VAD_OVERHANG)
        {
            ++numberOfConsecAct;
            countingInactStart = false;
        }
        else if(countingInactStart)
        {
            ++numberOfInactStart;
        }
        else
        {
            maxNumberOfConsecAct = (((maxNumberOfConsecAct) > (numberOfConsecAct)) ?
(maxNumberOfConsecAct) : (numberOfConsecAct));
            numberOfConsecAct    = 0;
        }

        if(VADProfile[*endVADPos + *startVADPos - i] <  SQ_VAD_OVERHANG &&
countingInactEnd)
            ++numberOfInactEnd;
        if(VADProfile[*endVADPos + *startVADPos - i] >= SQ_VAD_OVERHANG &&
countingInactEnd)
            countingInactEnd = false;
    }
    maxNumberOfConsecAct = (((maxNumberOfConsecAct) > (numberOfConsecAct)) ?
(maxNumberOfConsecAct) : (numberOfConsecAct));

    int tempStart   = *startVADPos;
    int tempEnd     = *endVADPos;
    int localLenVAD = *endVADPos - *startVADPos + 1;

    if( localLenVAD > 0)
    {
        if(numberOfInactStart*iStepPerFrame > 0.32f*sampFreq)
            tempStart = (((*startVADPos) > (*startVADPos + ((int)(((numberOfInactStart -
0.32f*sampFreq/iStepPerFrame) > 0) ? (numberOfInactStart -
0.32f*sampFreq/iStepPerFrame)+0.5f : (numberOfInactStart -
```

```
0.32f*sampFreq/iStepPerFrame)-0.5f)))) ? (*startVADPos) : (*startVADPos +
((int)(((numberOfInactStart - 0.32f*sampFreq/iStepPerFrame) > 0) ?
(numberOfInactStart - 0.32f*sampFreq/iStepPerFrame)+0.5f :
(numberOfInactStart - 0.32f*sampFreq/iStepPerFrame)-0.5f))));
        if(numberOfInactEnd*iStepPerFrame > 0.32f*sampFreq)
            tempEnd   = (((*endVADPos) < (*endVADPos - ((int)(((numberOfInactEnd -
0.32f*sampFreq/iStepPerFrame) > 0) ? (numberOfInactEnd -
0.32f*sampFreq/iStepPerFrame)+0.5f : (numberOfInactEnd -
0.32f*sampFreq/iStepPerFrame)-0.5f)))) ? (*endVADPos) : (*endVADPos -
((int)(((numberOfInactEnd - 0.32f*sampFreq/iStepPerFrame) > 0) ?
(numberOfInactEnd - 0.32f*sampFreq/iStepPerFrame)+0.5f : (numberOfInactEnd -
0.32f*sampFreq/iStepPerFrame)-0.5f))));

        if(tempStart >= 0 && tempEnd < numFrames && tempEnd > tempStart)
        {
            *endVADPos   = tempEnd;
            *startVADPos = tempStart;
        }
    }

    if(maxNumberOfConsecAct > MAX_PERC_OF_CONSECUTIVE_ACTIVE_FRAMES * numFrames)
    {
        return true;
    }
    else
        return false;
}
}
```