Prosodic manipulation of speech using knowledge of instants of significant excitation

Institute for Perception Research
PO Box 513, 5600 MB Eindhoven

23.12.1994

Rapport no. 1029

Prosodic manipulation of speech using knowledge of instants of significant excitation

B. Yegnanarayana and
Remco Teunen

Prosodic Manipulation of Speech using Knowledge of Instants of Significant Excitation

B. Yegnanarayana and Remco Teunen

Abstract

In this report we describe a method for manipulation of prosodic (duration and intonation) information in a speech signal. The method is based on the use of the knowledge of instants of significant excitation, which can be derived from speech signals. The prosodic information can be mainly attributed to the excitation source of the vocal tract system. Therefore the speech signal is decomposed into an approximate source and system components, and the source component is modified according to the specified prosodic manipulation. The method is implemented using the IPO/OTS Software Library, with flexibility to manipulate the source and system components in a desired manner.

1 Introduction

In many applications and for studies in speech perception it is often desirable to generate speech with specified characteristics or to modify a given speech signal by incorporating some specified features. The features may include changes in the vocal tract system and source characteristics. These characteristics at a segmental level may correspond to, for example, the average pitch, vocal tract length and the source-tract interaction within each pitch period. At the suprasegmental level, the characteristics of interest are the durations of units at syllable or higher levels, intonation, and the speaking rate. In this report we address the issue of modifying a given speech signal to incorporate specified features mainly at suprasegmental level. The emphasis is on the manipulation of prosodic features such as speaking rate and intonation.

In order to generate natural sounding speech with some desired prosodic features incorporated, several time and frequency domain methods were proposed in the literature[1,2]. Methods were also proposed based on modelling the speech signal, like Linear Prediction (LP) analysis, and sinusoidal modelling[3,4]. Waveform based techniques like PSOLA and WSOLA [5,6] produce natural sounding speech, provided the modifications in the scale factors for duration and pitch are small. Also waveform based techniques rely on the identification of excitation moments in a pitch period. Methods that combine features of waveform and transform based methods have been proposed to overcome some of the limitations of the individual methods[7].

The critical step in most of the methods is the identification of the moments of excitation, especially in the voiced speech. In fact, the instants of significant excitation of the vocal tract system define the primary source information. Recently a new method has been developed for extraction of these instants from continuous speech signal [8]. The method identifies all the significant instants including the instants of glottal closure, onset of burst, etc. The method also identifies major secondary excitations, if any, within each period, besides random excitation instants in the unvoiced, aspirated and silence regions.

Availability of these instants of excitation makes prosodic manipulation easier, in principle, as it is these instants that need to be modified to realize any desired prosodic characteristics. We focus mainly on the issue of manipulation of speaking rate and pitch period, although it is also possible to affect changes in the segmental characteristics as well. We discuss the procedure to incorporate the desired prosodic modifications, but the procedure to derive the modification rules themselves is not within the scope of this work.

In the next section we give the basic principle for obtaining the significant instants, which is the key input for the prosodic manipulation procedure described in Sec.3. The implementation of the procedure is described briefly in Sec.4 and the details of the implementation are given in the Appendices.

2 Extraction of instants of significant excitation

Recently a method has been proposed for determining the instants of significant excitations in speech signals[8]. The method is based on the global phase characteristics of minimum phase signals. The average slope of the unwrapped phase of the short-time Fourier transform of the linear prediction residual is calculated as a function of time. This is called phase slope function. Instants where the phase slope function makes a positive zerocrossing are identified as significant excitation instants. Here significant excitation refers mainly to the instant of glottal closure within a pitch period in voiced speech, although the method also gives the instants at the onset of other significant events like burst and secondary excitations in a pitch period such as glottal opening in voiced speech.

Fig.1 shows a speech signal, its linear prediction (10th order) residual, phase slope function and a residual gain plot showing the strengths of impulses at the instants of excitation. The strengths of the impluses in the gain plot correspond to the average energy of the LP residual per sample in the interval between two successive instants. From the figure it is clear that in voiced speech the significant excitation mainly takes place at the instant of glottal closure within each period, although in some case a strong secondary excitation is also identified at the glottal opening. The method identifies instants at the onset of other significant events also such as burst or release of stop sounds. In the unvoiced, silence and aspirated regions the instants are randomly positioned. Typically the instants in the voiced regions can be distinguished from those in the nonvoiced regions by the quasiperiodic nature of the glottal excitation, which is reflected in the quasiperiodicity of the extracted instants. The instants in the unvoiced and silence regions can be distinguished, if necessary, using the gain information and the average spacing between the instants. Some postprocessing of the gain plot is required to delete the instants due to minor excitations within a pitch period in the voiced speech, and also to label a given instant as belonging to voiced or nonvoiced category. In the present study this postprocessing is realized by manual editing of the gain plot file using additionally the information in the speech signal and the LP residual. Fig.1d shows the edited gain plot with voiced (V) and unvoiced (U) labels marked on it. All nonvoiced segments are marked as unvoiced.

The availability of the instants of significant excitations with voiced and nonvoiced labels eliminates the need for extraction of pitch for performing prosodic manipulations. Moreover, these instants will enable us to select the significant portion of the residual signal for generating the excitation signal for synthesis. These instants also preserve the microprosodic information, especially after the vowel onset in a voiceless consonant-vowel syllable. In the next section we show how to perform prosodic manipulation using the information of the instants of excitation.

3 Prosodic manipulation

The main objective is to modify a given speech signal to incorporate the desired pitch and durational changes, while preserving the natural segmental characteristics. The segemental characteristics include features of the excitation and vocal tract system within each pitch period in the case of voiced speech. For unvoiced speech it is not critical to preserve the segmental characteristics. It is possible to incorporate the natural variations in the prosodic features of the speaker for different pitch and rates of speaking, provided that prosodic information is made available. The prosodic information can be acquired by analysing large amount of data, but it is not within the scope of the present study.

The modifications in the pitch and speaking rate are presented in the form of multiplication factors. Since the LP residual signal is available, it is possible to keep as much of the signal as needed, to preserve the naturalness at the segmental level. This is similar to the philosophy of PSOLA method where the naturalness is sought to be preserved by selecting a windowed speech waveform [5].At higher pitch frequencies removing a portion of the residual signal will produce distortion, even though it is from the less significant part of the residual. This is because it is not the way natural speech is produced at higher pitch frequencies. This is a matter for detailed investigation, once the basic software for prosodic manipulation is available. If the excitation signal is to be generated using a model for glottal pulse in voiced regions and random noise in unvoiced regions, then obviously it gives more flexibility for manipulation. But the choice between selecting a portion of the residual or a model for excitation depends on the desired degree of naturalness and the level tolerance for distortion due to truncation of the residual. Through the proposed algorithm we provide both the options for generating the excitation signal to enable one to experiment with various alternatives. Using the excitation signal together with the linear predictor coeffficients (LPCs) representing the system at each of the significant excitation instants, it is possible to synthesize speech incorporating the desired prosodic modifications. Since the transfer function of the vocal tract system is represented by the LPCs, it is also possible to modify the system characteristics, if necessary, before synthesizing the speech signal.

The data available for prosodic manipulation is the speech signal, the significant excitation instants in the form of a gain function, and the LPC data file with Voiced (V)/ Unvoiced (U) labels. Centered around each of these instants a windowed speech signal is taken, and a residual signal is obtained by passing the speech signal through the inverse filter defined by the predetermined LPCs. From the residual signal around the instant, the required number of samples are taken to associate with the current instant. The gain per sample is computed at the instant by computing the square root of the mean squared energy of the residual signal associated with the instant.

The basic approach in prosodic manipulation is to derive an excitation signal incorporating the desired modification in the speaking rate and the pitch period. This is done by first taking the instants information in the gain function, and creating a new instants file incorporating the speaking rate and pitch period modifications specified in the form of scale factors for these parameters. We associate with each instant, the time, pitch period (interval between successive instants), LP residual and LPCs. For speaking rate/duration manipulation, obtain the new scaled time instants using the time scale multiplication factor. Likewise, for pitch manipulation, the pitch period associated with each instant is scaled appropriately. Now a new set of instants and the parameters at those instants are determined as follows (see Fig.2):

Proceeding from left to right, the first instant is copied as a new instant. The next new instant should be at the pitch period away from the first one, the period information being available in the parameter list associated with the first instant. Determine which of the old instants are closer to the new instant. Associate the parameter information of the old instant to the new instant. It is also possible to obtain an interpolated value of the pitch period for the current new instant from the pitch periods at the old instants which are on either side of the current new instant. Use the pitch period value in the parameter list at the current new instant to obtain the next new instant. This process is repeated until the end of all the instants derived from the original speech data.

Problems will arise while copying the residual samples at the new instants from the parameter list associated with the old instants, if the new pitch period is smaller than the old value at that instant. The required number of residual samples around the instant are copied. But to avoid discontinuity due to this partial selection of the residual samples, the residual signal sample are multiplied with a Hanning window. The signal is high pass filtered (cut-off frequency of about 50 Hz) to remove the very low frequency components including the zero frequency component. This will ensure that the resulting residual signal has zero mean. This process may produce some distortion, especially when the pitch period is scaled down significantly, say by a multiplication factor of 0.5 or lower. If the scaled pitch period is larger than the old one, the additional excitation samples needed in each pitch period are set to zero. The resulting excitation samples are appropriately scaled to obtain the gain value specified in the parameter list for that instant.

For instants labelled as unvoiced, the required number of residual samples are copied from the residual signal associated with the instants. For these instants, the entry in the pitch period field associated with the instants is not modified. Therefore if the interval between instants increases due to expansion of the time scale (slow speaking rate), some segments of the residual samples belonging to the unvoiced portion may be repeated. Sometimes this will produce some audible distortion. One way to overcome this is to use random samples with appropriate gain, instead of repeating the residual samples as is being done in the current implementation.

Speech signal is generated by exciting the all-pole model defined by the LPCs and the gain parameter with the new excitation signal. It is also possible to vary the all-pole model characteristics within a pitch period to reflect the differences in the vocal tract system in the closed and open phases of glottal vibration. This is realized approximately by using in the open phase a set of LPCs which correspond to the poles moved towards the origin in the z-plane. This creates an effect of damping of formants in the all-pole model representing the vocal tract system. This damping effect can be controlled by using a parameter to modify the LPCs. The parameter is simply the radius (r) of a circle in the z-plane concentric with the unit circle.

As mentioned earlier, it is possible to generate the excitation signal completely using a model for glottal pulse (see Fig.3) for voiced segments, and random noise for unvoiced segments, and appropriately synchronizing them with the information associated with the instants. The glottal pulse model shown in Fig.3 is a model similar to the LF model described in the literature[9].

4 Implementation of prosodic manipulation

The above procedure is implemented using the routines, data structures and other features given in IPO/OTS software library[10]. The two key data structures are: (a) InPulseData, which contains at each input instants the residual samples, gain, pitch period, and LPCs with voiced unvoiced labels. (b) OutPulseData, which contains the corresponding data for the output instants. The implementation details are given in Appendix-I and Appendix-II for the cases using the actual residual signal (PROG-1) and synthetic source signal (PROG-2), respectively.

The output results of the programs can be examined through the GIPOS (Graphical Interactive Processing Of Speech ans audio signals) software available at the Intitute for Perception Research. It is also possible to specify any desired pitch contour within the GIPOS framework, and the prosodic manipulation program generates speech with that pitch information together with the V/U frame decision already available in the parameter file.

Typical sessions of running the programs PROG-1 and PROG-2 are illustrated in the Appendix-III and Appendix-IV, respectively. They show the options available with the package for generating speech with any desired prosodic characteristics.

References

[1] S.Seneff, System to independently modify excitation and/or spectrum of speech waveform without explicit pitch extraction, *IEEE Trans on ASSP*, vol.ASSP-30, pp.566-578, 1982.

[2] M.Portnoff, Time-scale modification of speech based on short-time Fourier analysis, *IEEE Trans on ASSP*, vol.ASSP-29, pp.374-390, 1981.

[3] B.S.Atal and S.L.Hanauer, Speech analysis and synthesis by linear prediction of the speech wave, *Journal of the Acoustical Society of America*, vol.50, pp.637-655, 1971.

[4] Thomas F.Quatieri and Robert J. McAulay, Shape invariant time-scale and pitch modification of speech, *IEEE Trans on Signal Processing*, vol.40, pp.497-510, March 1992.

[5] C.Hamon, E.Moulines, and F.Charpentier, A diphone systhesis system based on time-domain prosodic modifications of speech, *Proceedings of ICASSP-89*, pp.238-241, Glasgow, 1989.

[6] W.Verhelst and M.Roelands, An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech, *Proceedings of ICASSP-93*, pp. 554-558, Minneapolis, 1993.

[7] R.N.J. Veldhuis and H. He, Time-scale and pitch modifications of speech signals and resynthesis from the discrete short-time Fourier transform, *Intitute for Perception Research Manuscript* No.1034, IPO, Eindhoven, The Netherlands, July 1994.

[8] R.L.H.M. Smits and B.Yegnanarayana, Determination of instants of significant excitation in speech using group-delay function, *Institute for Perception Research Manuscript* No.886/II, IPO, Eindhoven, The Netherlands, August 1994.

[9] D.G.Childers and C.F. Wong, Measuring and modelling vocal source-tract interaction, *IEEE Trans on Biomedical Engineering*, vol.41, N0.7, pp.663-671, July 1994.

[10] T.J.G.Veenker, Programmer's guide - IPO/OTS Software Library, *Institute for Perception Research Manual* No.129, IPO, Eindhoven, The Netherlands, August 1994.

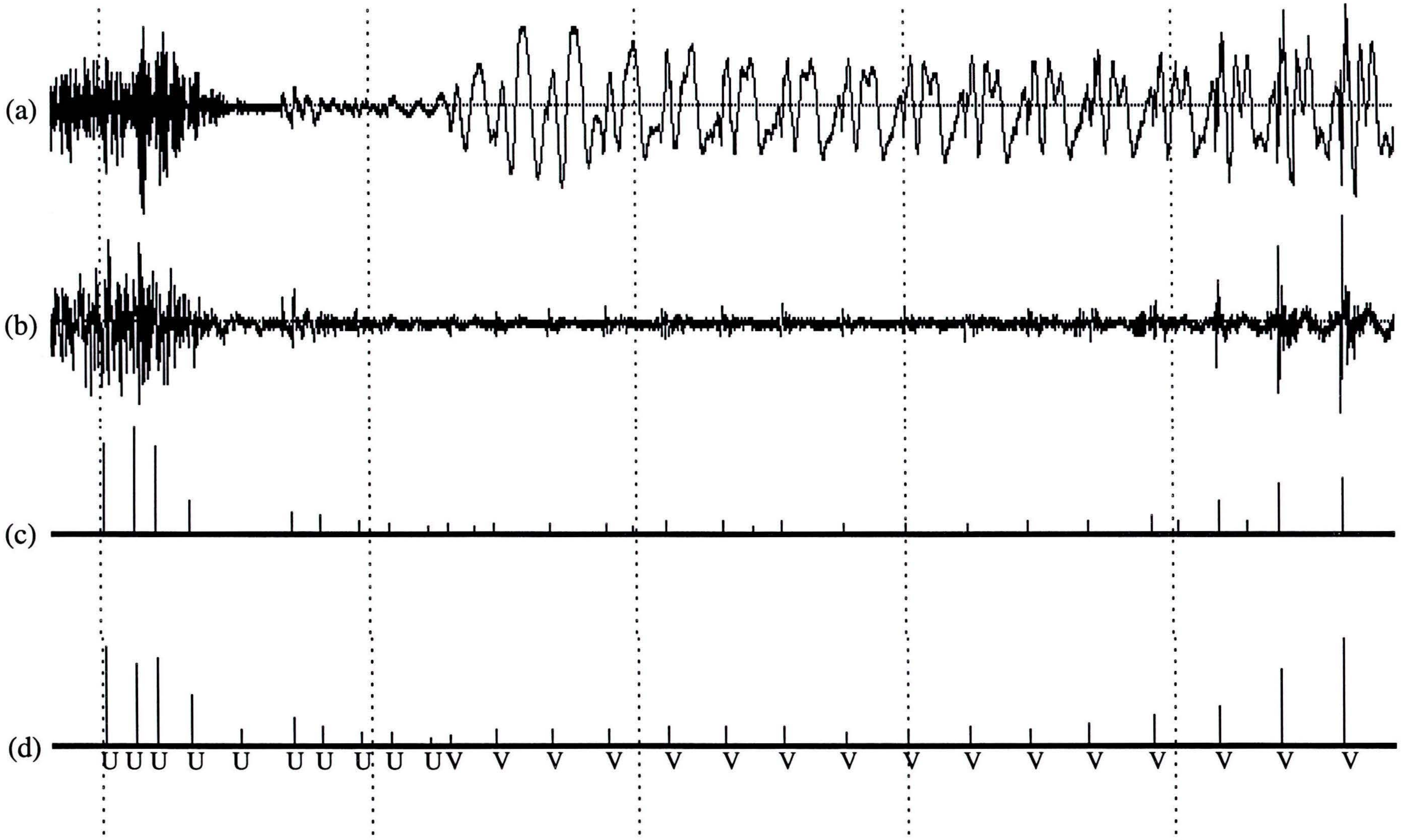

Fig. 1. Illustration of the extraction of instants of significant excitation (a) speech signal, (b) linear prediction residual, (c) gain plot showing the strength of the inpulses of the significant instants, and (d) edited gainplot with V/U labels.

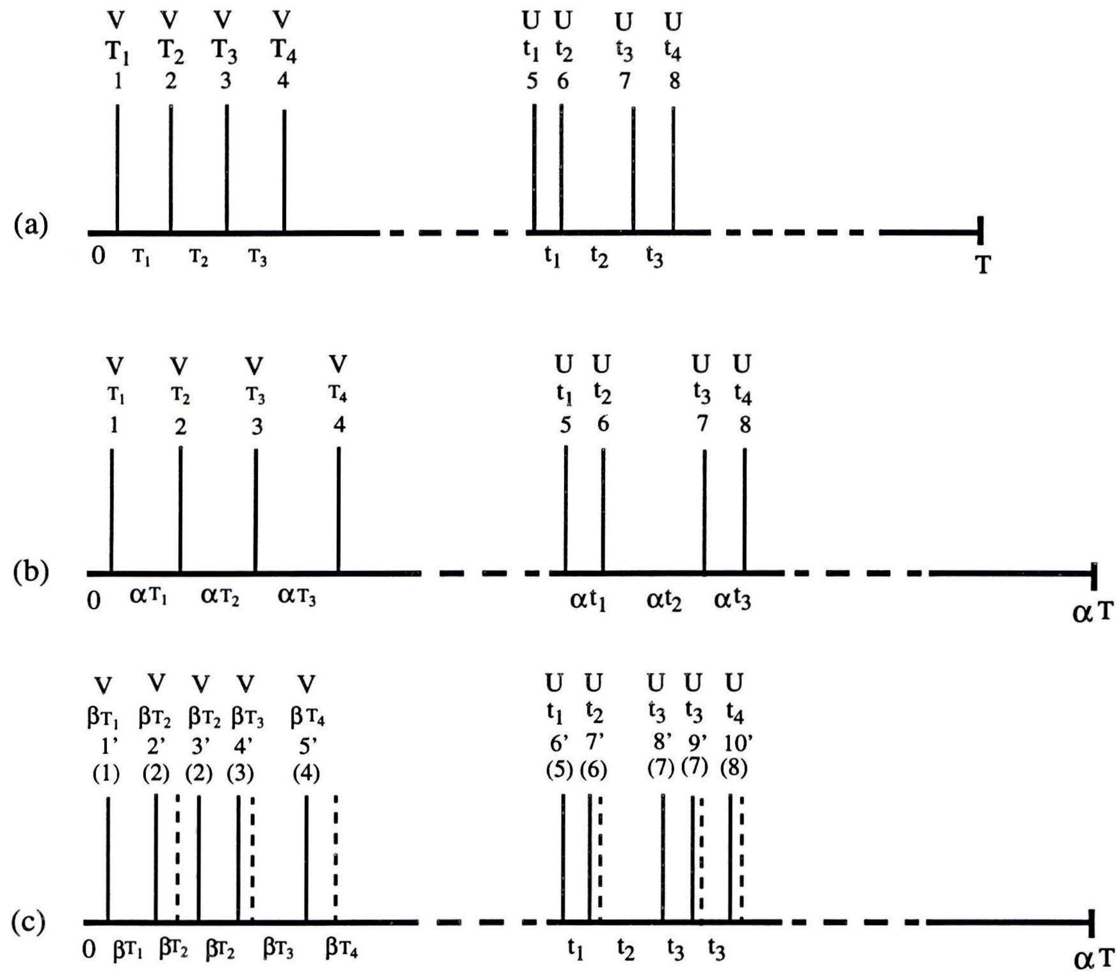

Fig. 2. Illustration of prosodic manipulation. V and U are the voiced and unvoiced labels for the instants. T_i's are intervals between instants in voiced regions, and t_i's are intervals between instants in unvoiced regions.

(a) Instants in the input data.

(b) Instants shifted due to time scale multiplication factor α.

(c) The new instants and the entries in the pitch period field at each instant in voiced and unvoiced regions, where the pitch period is modified by a factor β. Note that the spacing between inpulses is βT_i in voiced regions and t_i in unvoiced regions.

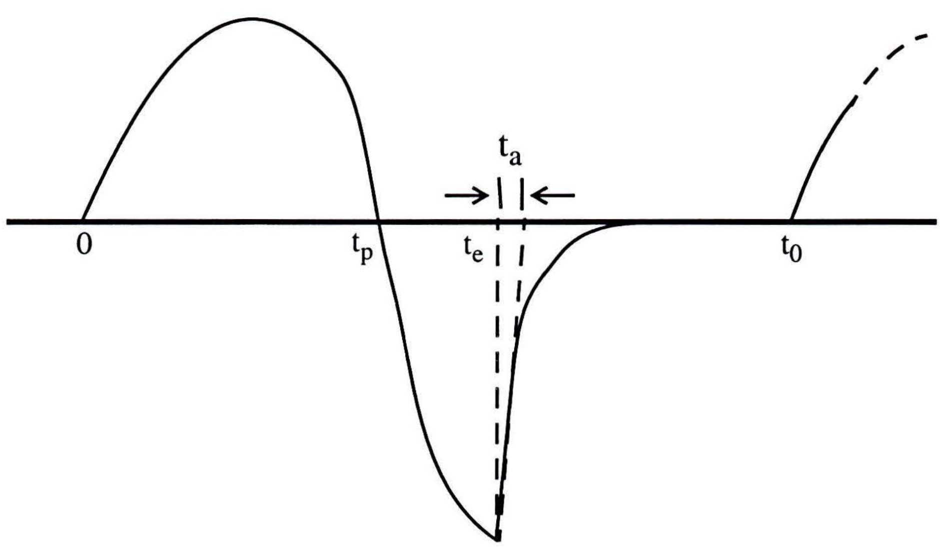

Fig. 3. Shape of the differentiated glottal waveform used for synthetic case, showing the different parameters as in LF model (Ref. 9), where t_0 is one pitch period.

```
/*===========================================================================*
 *   Name:            ReleasePulseData                                       *
 *                                                                           *
 *   Description:     Releases memory occupied by the given pulse data structure *
 *                    'PulseData'.                                           *
 *                                                                           *
 *   Arguments:       PulseData = Pulse data structure to be released.       *
 *===========================================================================*/
ReleasePulseData(PulseData)
{
    If PulseData is not NULL then do :
    {
        For all nodes of PulseData do :
        {
            Free memory used by residual signal of current node.
            Free memory used by LPC frame of current node.
        }
        Free memory used by the structure PulseData itself.
    }
}
EOF : ReleasePulseData()


/*===========================================================================*
 *   Name:            ReadResGnFile                                          *
 *                                                                           *
 *   Description:     Read excitation instant data from file (gain is ignored). *
 *                    This routine also creates the nodes of InPulseData. Each *
 *                    non-zero sample of the inputfile is an excitation instant. *
 *                                                                           *
 *   Arguments:       FileName    = Name of the inputfile which contains the *
 *                                  residual gain data (instant data).       *
 *                    InPulseData = The information gathered will be stored in *
 *                                  this structure.                          *
 *===========================================================================*/
ReadResGnFile(FileName, InPulseData)
{
    Open an AIFF file named FileName with residual gain data and create a
    buffer to speed up reading of the file.

    For all samples in the file do :
    {
        If the buffer is empty, fill it with new data from file.

        Get a new sample value from the buffer.

        If the current sample value is greater than zero, an instant
        is found. Create a new node for InPulseData, fill the node with
        data & do some housekeeping.
    }
    Finish the job; close the file and exit.
}
EOF : ReadResGnFile()


/*===========================================================================*
 *   Name:            ReadLPCsFile                                           *
 *                                                                           *
 *   Description:     Read LPCs from file & store them in InPulseData.       *
 *                                                                           *
 *   Arguments:       FileName    = Name of the inputfile which contains the *
 *                                  LPC data.                                *
 *                    InPulseData = The information gathered will be stored in *
 *                                  this structure.                          *
 *===========================================================================*/
ReadLPCsFile(FileName, InPulseData)
{
    Open a LVSA file named FileName with LPC data and copy some header
    information to InPulseData. This information is needed later to create a new
    LVSA-file with LPC data.

    For all nodes of InPulseData do :
    {
        Allocate memory for LPC frame of current node.

        While the correct LPC frame is not found do :
        {
            Read the current LPC frame.

            Check if this LPC frame is better than the next frame. If not
            goto next frame & try again.
        }
    }
    Close the file and exit the routine.
}
EOF : ReadLPCsFile()


/*===========================================================================*
 *   Name:            ReadFile                                               *
 *                                                                           *
 *   Description:     Read speech signal from file and calculate the residual *
```

```
 *                    signal. Store the residual signal in the nodes. Note that   *
 *                    the speech signal is not stored in the nodes, it is used    *
 *                    only to calculate the residual.                             *
 *                                                                                *
 *   Arguments:       FileName    = Name of the inputfile which contains the      *
 *                                  speech signal.                                *
 *                    InPulseData = The information gathered will be stored in    *
 *                                  this structure.                               *
 *===============================================================================*/
ReadFile(FileName, InPulseData)
{
    Open an AIFF file named FileName with speech data and copy some header data
    to InPulseData. This information is needed later to create a new AIFF-files
    (for the residual gain data, the residual signal and the speech signal).

    Create a buffer for the file to speed up reading.

    For all nodes of InPulseData do :
    {
        Adjust the LPC frame of the current node so the frame duration is
        correct (= half pitchperiod of the previous node + half pitchperiod
        of current node).

        Calculate what piece of the speech signal to pick for generation of
        the residual signal. Only the residual signal centred around the
        instant time (node time) is used (so you might get gaps between
        residual data of adjacent nodes).

        Allocate memory for the residual signal of the current node.

        If data in speech buffer is wrong, fill it with new data from file.

        Setup inverse LPC filter; obtain the coefficients.
        Setup pre-emphasis filter; obtain the coefficient.

        Filter the speech signal; first pre-emphasis, then inverse LPC filtering.
        Store the resulting residual signal in the current node. Also calculate
        the gain of the residual signal.
    }
    Close the file and exit the routine.
}
EOF : ReadFile()


/*===============================================================================*
 *   Name:            InputFiles2InPulseData                                     *
 *                                                                               *
 *   Description:     Read all inputdata from files & put it in InputPulseData.  *
 *                                                                               *
 *   Arguments:       InResGnFileName = Name of the inputfile which contains the *
 *                                      residual gain data.                      *
 *                    InLPCsFileName  = Name of the inputfile which contains the *
 *                                      LPC data.                                *
 *                    InFileName      = Name of the inputfile which contains the *
 *                                      speech signal.                           *
 *                    InPulseData     = The information gathered will be stored  *
 *                                      in this structure.                       *
 *===============================================================================*/
InputFiles2InPulseData(InResGnFileName, InLPCsFileName, InFileName, &InPulseData)
{
    Allocate memory for InPulseData & fill it with zero's.

    Call ReadResGnFile(InResGnFileName, InPulseData).
    Call ReadLPCsFile(InLPCsFileName, InPulseData).
    Call ReadFile(InFileName, InPulseData).
}
EOF : InputFiles2InPulseData()


/*===============================================================================*
 *   Name:            ProsodicManipulation                                       *
 *                                                                               *
 *   Description:     Generate OutPulseData by manipulating InPulseData given    *
 *                    the parameters TimeMulVal and PitchPeriodMulVal (and the   *
 *                    pitch contour specified in the LPC frames).                *
 *                                                                               *
 *   Arguments:       InPulseData            = This structure contains all the   *
 *                                             inputdata about the speech signal.*
 *                    TimeMulVal             = Time multiplication factor.       *
 *                    PitchPeriodMulVal      = Pitchperiod multiplication factor.*
 *                    WindowWidthMulVal      = Window width multiplication factor*
 *                                             for reducing the number of residual*
 *                                             samples around each instant.      *
 *                    UsePitchContour        = TRUE if the pitchcontour specified*
 *                                             in the LPC frames should be used. *
 *                    UsePitchInterpolation  = TRUE if pitch(period)interpolation*
 *                                             should be used.                   *
 *                    UseGainInterpolation   = TRUE if gain interpolation should *
 *                                             be used.                          *
 *                    OutPulseData           = This structure will be filled with*
 *                                             data about the output speech      *
 *                                             signal. The structure can then be *
```

```
*                                            used to generate outputfiles.      *
*==============================================================================*/
ProsodicManipulation(InPulseData, TimeMulVal, PitchPeriodMulVal,
    WindowWidthMulVal, UsePitchContour, UsePitchInterpolation,
    UseGainInterpolation, &OutPulseData)
{
    Allocate memory for OutPulseData and initialize the structure.

    For all samples do :
    {
        Look for nearest input node & other 'bounding' input node.

        Allocate memory for the new output node & copy some data from the
        nearest input node.

        Calculate the pitchperiod of the output node. Use pitch interpolation
        if the segment is voiced and no pitchcontour is specified. If a
        pitchcontour is specified in the LPC file and the segment is voiced
        then take the pitchperiod from the LPC frame of the nearest input node.
        In unvoiced segments, always just copy the pitchperiod from the
        nearest input node.

        Calculate what the gain scale factor of the residual should be.
        This is the quotient of the interpolated gain and the original gain.

        Calculate the new window width.

        Allocate memory for LPC frame & residual signal. Copy LPC frame from
        input node to output node & adjust some fields (frame duration, pitch
        and gain).

        Copy the residual signal from the nearest input node to the current
        output node. If the pitchperiod is smaller, the center of the
        original residual signal should be copied, so samples should be deleted
        from both sides of the original residual signal. If the pitchperiod is
        larger, just copy the original residual signal.

        Apply gain correction, windowing and highpass filtering to the
        residual signal (in that order). Highpass filtering is used to remove
        the low frequency bias due to manipulating the residual signal.

        Calculate the time of the next output node.
    }
    Ready! Exit routine.
}
EOF : ProsodicManipulation()


/*==============================================================================*
*   Name:            WriteResGnFile                                             *
*                                                                               *
*   Description:     Write excitation instant & gain data to file.              *
*                                                                               *
*   Arguments:       OutPulseData = This structure contains inputdata for the   *
*                                   generation of the file.                     *
*                    FileName     = Name of outputfile for the residual gain    *
*                                   data.                                       *
*==============================================================================*/
WriteResGnFile(OutPulseData, FileName)
{
    Create an AIFF file named FileName for residual gain data and create a
    buffer to speed up writing to the file.

    Until all samples (and all nodes of OutPulseData) have been processed do :
    {
        If the current sample number is equal to the time of the current node,
        append the (scaled up) gain sample of the current node to the buffer and
        goto the next node. Otherwise append a zero to the buffer.

        If the buffer is full, write it to file.
    }
    Close file and exit routine.
}
EOF : WriteResGnFile()


/*==============================================================================*
*   Name:            WriteResFile                                               *
*                                                                               *
*   Description:     Write residual signal to file.                             *
*                                                                               *
*   Arguments:       OutPulseData = This structure contains inputdata for the   *
*                                   generation of the file.                     *
*                    FileName     = Name of outputfile for the residual signal. *
*==============================================================================*/
WriteResFile(OutPulseData, FileName)
{
    Create an AIFF file named FileName for the residual data and create a
    buffer to speed up writing to the file.

    Until all samples have been processed do :
    {
```

```
        If not in gap between nodes, get a sample from the residual signal of
        the current node. Otherwise make the sample value zero.

        Write the residual sample to the buffer and goto the next node.
        If the buffer is full, write it to file.
    }
    Close file and exit routine.
}
EOF : WriteResFile()


/*=========================================================================*
 *   Name:            WriteLPCsFile                                         *
 *                                                                          *
 *   Description:     Write LPC data to file.                               *
 *                                                                          *
 *   Arguments:       OutPulseData = This structure contains inputdata for the  *
 *                                   generation of the file.               *
 *                    FileName     = Name of outputfile for the LPC data.   *
 *=========================================================================*/
WriteLPCsFile(OutPulseData, FileName)
{
    Create a LVSA file named FileName for the LPC data.

    Check to see if the file should start with a dummy frame. The start time of
    a LPC frame should be equal to the time of the node to which the frame
    belongs. If the time of the first node is greater than zero, a dummy frame
    is needed. If so do :
    {
        Allocate memory for dummy LPC frame.
        Fill dummy frame with dummy data.
        Write dummy frame to file.
    }
    For nodes of OutPulseData do :
    {
        Write LPC frame of current node to file.
    }
    Close file and exit routine.
}
EOF : WriteLPCsFile()


/*=========================================================================*
 *   Name:            WriteFile                                             *
 *                                                                          *
 *   Description:     Write speech signal to file.                          *
 *                                                                          *
 *   Arguments:       OutPulseData = This structure contains inputdata for the  *
 *                                   generation of the file.               *
 *                    LPCAdjFac    = Multiplication factor for the LPC bandwidth *
 *                                   change.                                *
 *                    FileName     = Name of outputfile for the speech signal.  *
 *=========================================================================*/
WriteFile(OutPulseData, LPCAdjFac, FileName)
{
    Create an AIFF file named FileName for speech data and create a buffer to
    speed up writing to the file.

    For all samples of the residual signal do :
    {
        If not in gap between nodes, get a sample from the residual signal of
        the current node. Otherwise make the sample value zero.

        Calculate what the current LPC adjustment factor should be (to
        change the bandwidth of the all-pole filter). Use LPCAdjFac.

        Setup LPC Filter; calculate the coefficients.
        Setup de-emphasis Filter; obtain the coefficient.

        Apply the filters to the residual sample and store the resulting
        speech sample in the buffer. First apply the LPC filter, then
        the de-emphasis filter.

        If the buffer is full, write it to file.
    }
    Close file and exit routine.
}
EOF : WriteFile()


/*=========================================================================*
 *   Name:            OutPulseData2OutputFiles                              *
 *                                                                          *
 *   Description:     Write data from OutPulseData to files.                *
 *                                                                          *
 *   Arguments:       OutPulseData      = This structure contains inputdata for  *
 *                                        the generation of the files.     *
 *                    LPCAdjFac         = Multiplication factor for LPC bandwidth *
 *                                        change.                           *
 *                    OutResGnFileName = Name of outputfile for residual gain  *
 *                                        data.                             *
 *                    OutResFileName    = Name of outputfile for residual data.  *
```

```
*                       OutLPCsFileName  = Name of outputfile for LPC data.        *
*                       OutFileName      = Name of outputfile for speech data.     *
*===========================================================================*/
OutPulseData2OutputFiles(OutPulseData, LPCAdjFac, OutResGnFileName,
    OutResFileName, OutLPCsFileName, OutFileName)
{
    Call WriteResGnFile(OutPulseData, OutResGnFileName).
    Call WriteResFile(OutPulseData, OutResFileName).
    Call WriteLPCsFile(OutPulseData, OutLPCsFileName).
    Call WriteFile(OutPulseData, LPCAdjFac, OutFileName).
}
EOF : OutPulseData2OutputFiles()


/*===========================================================================*
*   Name:           main                                                    *
*                                                                           *
*   Description:    A basic (& not very user-friendly) interface for the    *
*                   routines above. Use only for test purposes.             *
*===========================================================================*/
main()
{
    Put some program information on the screen (name, version and copyrights).

    Ask if male voice should be used or female voice.
    Ask if gain interpolation should be used.
    Ask if pitch interpolation should be used.
    Ask if pitch contour in LPC file should be used. If not, then
    the original pitch contour will be maintained.
    Ask for time multiplication factor (float).
    Ask for pitchperiod multiplication factor (float).
    Ask for windowwidth multiplication factor (float).
    Ask for LPC adjustment factor (to change the bandwidth) (float).

    Setup filenames & print what parameter values will be used.

    Call InputFiles2InPulseData; read inputfiles and put data in the
    structure InPulseData.

    If call to InputFiles2InPulseData was ok, call ProsodicManipulation;
    create structure OutPulseData from InPulseData, using the given parameters.

    If call to ProsodicManipulation was ok, call OutPulseData2OutputFile;
    write data from the structure OutPulseData to the outputfiles.

    Release structures.
    If there was an error, say so.
}
EOF : main()
```

```
/*=========================================================================*
 *   Name:              ResWave_Impulse                                    *
 *                                                                         *
 *   Description:       Impulse excitation : The routines gives back the requested  *
 *                      sample of residual signal of the given node. This routine   *
 *                      also signals the calling routine when to update the LPCs    *
 *                      and when to change the bandwidth. The routine creates a     *
 *                      zero-mean impulse-residual signal.                 *
 *                                                                         *
 *   Arguments:         Node         = Pointer to the node.                *
 *                      ResSampleNr = Number of the requested sample. Should be    *
 *                                     equal to or greater than zero and smaller   *
 *                                     than the pitchperiod of Node.       *
 *                      UpdateLPCs   = Target variable. This flag signals the calling*
 *                                     routine that the LPCs should be updated or  *
 *                                     LPC bandwidth change should be applied.     *
 *                      ErrorCode    = Return errorcode. MSG_SUCCESS if ok,  else  *
 *                                     MSG_?.                              *
 *   Return:            Sample of residual wave.                           *
 *=========================================================================*/
ResWave_Impulse(Node, ResSampleNr, &UpdateLPCs, &ErrorCode)
{
    Set UpdateLPCs to the correct value; if ResSampleNr is zero, then the
    LPCs should be updated. If the segment is voiced and ResSampleNr is
    equal or greater than half the pitchperiod, then LPC bandwidth change
    should be applied.

    if this is a new node, check if the frame is unvoiced and if so do :
    {
        Allocate memory for buffer to store noise in. Fill the buffer with
        noise (with maximum gain).

        Calculate the mean value of the noise and adjust the signal so
        its zero-mean.

        Calculate the gain of the noise and adjust it if that gain is
        not equal to the gain specified in Node.
    }
    if the segment is unvoiced return a sample from the noise buffer,
    else return an impulse (scaled gain of node) if ResSampleNr is zero
    else some small negative value to make the signal zero mean.
}
EOF : ResWave_Pulse()


/*=========================================================================*
 *   Name:              ResWave_GlottalPulse                               *
 *                                                                         *
 *   Description:       Glottal pusle excitation : The routines gives back the      *
 *                      requested sample of residual signal of the given node. This *
 *                      routine also signals the calling routine when to update the *
 *                      LPCs and when to change the bandwidth. The routine uses a   *
 *                      model derived from the LF model to create residual signal.  *
 *                                                                         *
 *                      Residual(t)  = A*exp(Alfa*t)*sin(t*pi/Tp), for 0<=t<=Tp,    *
 *                                   = B*exp(Alfa*t)*sin(t*pi/Tp), for Tp<=t<=Te,   *
 *                                   = B*exp(Alfa*Te)*sin(Te*pi/Tp)        *
 *                                     (exp(-(t-Te)/Ta)-                   *
 *                                     exp(-(T0-Te)/Ta)) /                 *
 *                                     (1-exp(-(T0-Te)/Ta)),       for Te<=t<=T0.   *
 *                                                                         *
 *                      T0           = Pitchperiod.                        *
 *                                                                         *
 *   Arguments:         Node         = Pointer to the node.                *
 *                      ResSampleNr = Number of the requested sample. Should be    *
 *                                     equal to or greater than zero and smaller   *
 *                                     than the pitchperiod of Node.       *
 *                      UpdateLPCs   = Target variable. This flag signals the calling*
 *                                     routine that the LPCs should be updated or  *
 *                                     LPC bandwidth change should be applied.     *
 *                      ErrorCode    = Return errorcode. MSG_SUCCESS if ok,  else  *
 *                                     MSG_?.                              *
 *                                                                         *
 *   Return:            Sample of residual wave.                           *
 *=========================================================================*/
ResWave_GlottalPulse(Node, ResSampleNr, &UpdateLPCs, &ErrorCode)
{
    if the pitchperiod is less than 5, use a zero-mean impulse waveform.
    Else if this is a new node, check if the frame is unvoiced and if so do :
    {
        Allocate memory for buffer to store noise in. Fill the buffer with
        noise (with maximum gain).

        Calculate the mean value of the noise and adjust the signal so
        its zero-mean.

        Calculate the gain of the noise and adjust it if that gain is
        not equal to the gain specified in Node.
    }
    Else if this is a new node and the frame is voiced do :
    {
        Calculate Tp, Te and Ta.
```

```

        Allocate memory for buffers to store the signal in. The positive
        part of the signal is stored in a different buffer than the negative
        part, so the signal can be made zero-mean by just scaling the signal
        of one of the two buffers.

        Fill the buffers with the glottal pulse.

        Calculate the mean value of the signal and adjust the signal so
        its zero-mean.

        Calculate the gain of the signal and adjust it if that gain is
        not equal to the gain specified in Node.
    }
    If the segment is unvoiced do :
    {
        Set UpdateLPCs so that the LPCs are updated at the beginning of
        the signal. Return a sample from the noise buffer.
    }
    Else do :
    {
        Set UpdateLPCs so that the LPCs are updated at time Tp and the
        bandwidth is changed for times less than Tp.

        Return a sample from the signal buffers.
    }
}
EOF : ResWave_GlottalPulse()


/*============================================================================*
 *   Name:           ReleasePulseData                                         *
 *                                                                            *
 *   Description:    Releases memory occupied by the given pulse data structure *
 *                   'PulseData'.                                             *
 *                                                                            *
 *   Arguments:      PulseData = Pulse data structure to be released.         *
 *============================================================================*/
ReleasePulseData(PulseData)
{
    If PulseData is not NULL then do :
    {
        For all nodes of PulseData do :
        {
            Free memory used by LPC frame of current node.
        }
        Free memory used by the structure PulseData itself.
    }
}
EOF : ReleasePulseData()


/*============================================================================*
 *   Name:           ReadResGnFile                                            *
 *                                                                            *
 *   Description:    Read excitation instant & gain data from file. This      *
 *                   routine also creates the nodes of InPulseData. Each      *
 *                   non-zero sample of the inputfile is an excitation instant. *
 *                                                                            *
 *   Arguments:      FileName    = Name of the inputfile which contains the   *
 *                                 residual gain data.                        *
 *                   InPulseData = The information gathered will be stored in *
 *                                 this structure.                            *
 *============================================================================*/
ReadResGnFile(FileName, InPulseData)
{
    Open an AIFF file named FileName with residual gain data and create a
    buffer to speed up reading of the file.

    For all samples in the file do :
    {
        If the buffer is empty, fill it with new data from file.

        Get a new sample value from the buffer. If greater then zero this value
        is the gain of the residual signal scaled up by a factor.

        If the current sample value is greater than zero, an instant
        is found. Create a new node for InPulseData, fill the node with
        data & do some housekeeping.
    }
    Finish the job; close the file and exit.
}
EOF : ReadResGnFile()


/*============================================================================*
 *   Name:           ReadLPCsFile                                             *
 *                                                                            *
 *   Description:    Read LPCs from file & store them in InPulseData.         *
 *                                                                            *
 *   Arguments:      FileName    = Name of the inputfile which contains the   *
 *                                 LPC data.                                  *
```

```
*                    InPulseData = The information gathered will be stored in    *
*                                  this structure.                               *
*============================================================================*/
ReadLPCsFile(FileName, InPulseData)
{
    Open a LVSA file named FileName with LPC data and copy some header
    information to InPulseData. This information is needed later to create a new
    LVSA-file with LPC data.

    For all nodes of InPulseData do :
    {
        Allocate memory for LPC frame of current node.

        While the correct LPC frame is not found do :
        {
            Read the current LPC frame.

            Check if this LPC frame is better (nearer) than the next frame.
            If not goto next frame & try again.
        }
    }
    Close the file and exit the routine.
}
EOF : ReadLPCsFile()


/*============================================================================*
*   Name:           InputFiles2InPulseData                                   *
*                                                                            *
*   Description:    Read all inputdata from files & put it in InputPulseData.  *
*                                                                            *
*   Arguments:      InResGnFileName = Name of the inputfile which contains the  *
*                                     residual gain data.                    *
*                   InLPCsFileName  = Name of the inputfile which contains the  *
*                                     LPC data.                              *
*                   InPulseData     = The information gathered will be stored  *
*                                     in this structure.                     *
*============================================================================*/
InputFiles2InPulseData(InResGnFileName, InLPCsFileName, &InPulseData)
{
    Allocate memory for InPulseData & fill it with zero's.

    Call ReadResGnFile(InResGnFileName, InPulseData).
    Call ReadLPCsFile(InLPCsFileName, InPulseData).
}
EOF : InputFiles2InPulseData()


/*============================================================================*
*   Name:           ProsodicManipulation                                     *
*                                                                            *
*   Description:    Generate OutPulseData by manipulating InPulseData given  *
*                   the parameters TimeMulVal and PitchPeriodMulVal (and the  *
*                   pitch contour specified in the LPC frames).              *
*                                                                            *
*   Arguments:      InPulseData            = This structure contains all the  *
*                                            inputdata about the speech signal. *
*                   TimeMulVal             = Time multiplication factor.     *
*                   PitchPeriodMulVal      = Pitchperiod multiplication factor. *
*                   UsePitchContour        = TRUE if the pitchcontour specified *
*                                            in the LPC frames should be used. *
*                   UsePitchInterpolation  = TRUE if pitch(period)interpolation *
*                                            should be used.                 *
*                   UseGainInterpolation   = TRUE if gain interpolation should  *
*                                            be used.                        *
*                   OutPulseData           = This structure will be filled with *
*                                            data about the output speech    *
*                                            signal. The structure can then be  *
*                                            used to generate outputfiles.   *
*============================================================================*/
ProsodicManipulation(InPulseData, TimeMulVal, PitchPeriodMulVal,
    UsePitchContour, UsePitchInterpolation, UseGainInterpolation, &OutPulseData)
{
    Allocate memory for OutPulseData and initialize the structure.

    For all samples do :
    {
        Look for nearest input node & other 'bounding' input node.

        Allocate memory for the new output node & copy some data from the
        nearest input node.

        Calculate the pitchperiod of the output node. Use pitch interpolation
        if the segment is voiced and no pitchcontour is specified. If a
        pitchcontour is specified in the LPC file and the segment is voiced
        then take the pitchperiod from the LPC frame of the nearest input node.
        In unvoiced segments, always just copy the pitchperiod from the
        nearest input node.

        Calculate what the gain of the residual should be. This value is the
        interpolated gain (using the nearest input node & other bounding
        input node.
```

```

        Allocate memory for the LPC frame. Copy LPC frame from
        input node to output node & adjust some fields (frame duration, pitch
        and gain).

        Calculate the time of the next output node.
    }
    Ready! Exit routine.
}
EOF : ProsodicManipulation()


/*=========================================================================*
*   Name:            WriteResGnFile                                        *
*                                                                          *
*   Description:     Write excitation instant & gain data to file.         *
*                                                                          *
*   Arguments:       OutPulseData = This structure contains inputdata for the  *
*                                   generation of the file.               *
*                    FileName     = Name of outputfile for the residual gain  *
*                                   data.                                  *
*=========================================================================*/
WriteResGnFile(OutPulseData, FileName)
{
    Create an AIFF file named FileName for residual gain data and create a
    buffer to speed up writing to the file.

    Until all samples (and all nodes of OutPulseData) have been processed do :
    {
        If the current sample number is equal to the time of the current node,
        append the (scaled up) gain sample of the current node to the buffer
        and goto the next node. Otherwise append a zero to the buffer.

        If the buffer is full, write it to file.
    }
    Close file and exit routine.
}
EOF : WriteResGnFile()


/*=========================================================================*
*   Name:            WriteResFile                                          *
*                                                                          *
*   Description:     Write residual signal to file.                        *
*                                                                          *
*   Arguments:       OutPulseData = This structure contains inputdata for the  *
*                                   generation of the file.               *
*                    FileName     = Name of outputfile for the residual signal. *
*                    ResWaveProc  = Pointer to the procedure which generates  *
*                                   the waveform of the residual signal.   *
*=========================================================================*/
WriteResFile(OutPulseData, FileName, ResWaveProc)
{
    Create an AIFF file named FileName for the residual data and create a
    buffer to speed up writing to the file.

    Until all samples have been processed do :
    {
        If not in gap between nodes, get a sample from the residual signal of
        the current node by calling ResWaveProc. Otherwise make the sample
        value zero.

        Write the residual sample to the buffer and goto the next node.
        If the buffer is full, write it to file.
    }
    Close file and exit routine.
}
EOF : WriteResFile()


/*=========================================================================*
*   Name:            WriteLPCsFile                                         *
*                                                                          *
*   Description:     Write LPC data to file.                               *
*                                                                          *
*   Arguments:       OutPulseData = This structure contains inputdata for the  *
*                                   generation of the file.               *
*                    FileName     = Name of outputfile for the LPC data.   *
*=========================================================================*/
WriteLPCsFile(OutPulseData, FileName)
{
    Create a LVSA file named FileName for the LPC data.

    Check to see if the file should start with a dummy frame. The start time of
    a LPC frame should be equal to the time of the node to which the frame
    belongs. If the time of the first node is greater than zero, a dummy frame
    is needed. If so do :
    {
        Allocate memory for dummy LPC frame.
        Fill dummy frame with dummy data.
        Write dummy frame to file.
    }
```

```
    For nodes of OutPulseData do :
    {
        Write LPC frame of current node to file.
    }
    Close file and exit routine.
}
EOF : WriteLPCsFile()


/*==========================================================================*
*   Name:           WriteFile                                               *
*                                                                           *
*   Description:    Write speech signal to file.                            *
*                                                                           *
*   Arguments:      OutPulseData = This structure contains inputdata for the *
*                                  generation of the file.                  *
*                   LPCAdjFac    = Multiplication factor for the LPC bandwidth *
*                                  change.                                  *
*                   FileName     = Name of outputfile for the speech signal. *
*                   ResWaveProc  = Pointer to the procedure which generates  *
*                                  the waveform of the residual signal.     *
*==========================================================================*/
WriteFile(OutPulseData, LPCAdjFac, FileName, ResWaveProc)
{
    Create an AIFF file named FileName for speech data and create a buffer to
    speed up writing to the file.

    For all samples of the residual signal do :
    {
        If not in gap between nodes, get a sample from the residual signal of
        the current node by calling ResWaveProc. Otherwise make the sample
        value zero.

        Calculate what the current LPC adjustment factor should be (to
        change the bandwidth of the all-pole filter). Use LPCAdjFac.

        Setup LPC Filter; calculate the coefficients.
        Setup de-emphasis Filter; calculate the coefficient.

        Apply the filters to the residual sample and store the resulting
        speech sample in the buffer. First apply the LPC filter, then
        the de-emphasis filter.

        If the buffer is full, write it to file.
    }
    Close file and exit routine.
}
EOF : WriteFile()


/*==========================================================================*
*   Name:           OutPulseData2OutputFiles                                *
*                                                                           *
*   Description:    Write data from OutPulseData to files.                  *
*                                                                           *
*   Arguments:      OutPulseData      = This structure contains inputdata for *
*                                       the generation of the files.        *
*                   LPCAdjFac         = Multiplication factor for LPC bandwidth *
*                                       change.                             *
*                   OutResGnFileName = Name of outputfile for residual gain *
*                                       data.                               *
*                   OutResFileName    = Name of outputfile for residual data. *
*                   OutLPCsFileName   = Name of outputfile for LPC data.    *
*                   OutFileName       = Name of outputfile for speech data. *
*                   ResWaveProc       = Pointer to the procedure which generates *
*                                       the waveform of the residual signal. *
*==========================================================================*/
OutPulseData2OutputFiles(OutPulseData, LPCAdjFac, OutResGnFileName,
    OutResFileName, OutLPCsFileName, OutFileName, ResWaveProc)
{
    Call WriteResGnFile(OutPulseData, OutResGnFileName).
    Call WriteResFile(OutPulseData, OutResFileName, ResWaveProc).
    Call WriteLPCsFile(OutPulseData, OutLPCsFileName).
    Call WriteFile(OutPulseData, LPCAdjFac, OutFileName, ResWaveProc).
}
EOF : OutPulseData2OutputFiles()


/*==========================================================================*
*   Name:           main                                                    *
*                                                                           *
*   Description:    A basic (& not very user-friendly) interface for the    *
*                   routines above. Use only for test purposes.             *
*==========================================================================*/
main()
{
    Put some program information on the screen (name, version and copyrights).

    Ask if male voice should be used or female voice.
    Ask if gain interpolation should be used.
    Ask if pitch interpolation should be used.
    Ask if pitch contour in LPC file should be used. Is not, then
```

```
        the original pitch contour will be maintained.
        Ask for time multiplication factor (float).
        Ask for pitchperiod multiplication factor (float).
        Ask for LPC adjustment factor (to change the bandwidth).
        Ask if glottal pulse excitation or impulse excitation should be used.

        If glottal pulse excitation should be used do :
        {
            Ask for Tp-factor (Tp = Tp-factor * pitchperiod). See ResWave_GlottalPulse.
            Ask for Te-factor (Te = Te-factor * pitchperiod). See ResWave_GlottalPulse.
            Ask for Ta-factor (Ta = Ta-factor * pitchperiod). See ResWave_GlottalPulse.
            Ask for alfa. See ResWave_GlottalPulse.
        }
        Setup filenames & print what parameter values will be used.

        Call InputFiles2InPulseData; read inputfiles and put data in the
        structure InPulseData.

        If call to InputFiles2InPulseData was ok, call ProsodicManipulation;
        create structure OutPulseData from InPulseData, using the given parameters.

        If call to ProsodicManipulation was ok, call OutPulseData2OutputFile (with
        the right parameters) to write data from the structure OutPulseData to
        the outputfiles.

        Release structures.
        If there was an error, say so.
    }
    EOF : main()
```

```

ProsMan.Nat Version Oct 28 1994.
Prosodic manipulations using a natural residual signal.
Copyright (c) IPO 1994. All Rights Reserved.

Male or female [M/F]?                                   : m
Use gain interpolation [Y/N]?                           : y
Use pitch interpolation? [Y/N]?                         : y
Use pitch contour in LPC file [Y/N]?                    : n
Time multiplication factor X (0.2 <= X <= 5).           : 1
Pitchperiod multiplication factor X (0.2 <= X <= 5).    : 1
Windowwidth multiplication factor X (0 <= X <= 1).      : 1
LPC adj factor X (bandwidth change) (0.8 <= X <= 1).    : 0.96

Using the following parameter values :
-------------------------------------
Male / Female           = Male
UseGainInterpolation    = Yes
UsePitchInterpolation   = Yes
UsePitchcontour         = No
TimeMulVal              = 1.000000
PitchPeriodMulval       = 1.000000
WindowWidthMulVal       = 1.000000
LPCAdjFac               = 0.960000
-------------------------------------


Entering routine 'InputFiles2InPulseData'.
{
   Entering routine 'ReadResGnFile'.
   {
      Opening ResGnFile 'dicmaleresgn.aiff'.
      NrOfSamples = 36660.
      Created buffer of length 20000.
      Current sample at time 0. Found 1 excitation instants.
      Closing ResGnFile.
   }  ['ReadResGnFile']

   Entering routine 'ReadLPCsFile'.
   {
      Opening LPCsFile 'dicmaleap.lvsa'.
      Number of parameter frames = 731.
      Current node = 0 (time = 31). Duration of frame 1 from 50 to 100.
      Closing LPCsFile.
   }  ['ReadLPCsFile']

   Entering routine 'ReadFile'.
   {
      Opening File 'dicmale.aiff'.
      NrOfSamples = 36608.
      Created buffer of length 20000.
      Current node = 0 (time = 31). ResTimeOffset = -12, WindowWidth = 25.
      Closing File.
      Buffer misses = 2, buffer hits = 560.
   }  ['ReadFile']

}  ['InputFiles2InPulseData']

Entering routine 'ProsodicManipulation'.
{
   HP filter coefficient Alfa = 0.945093.
   InNode = 0, OutNode = 0.
   MaxResGainScaling = 1.000000, MinResGainScaling = 1.000000.
}  ['ProsodicManipulation']

Entering routine 'OutPulseData2OutputFiles'.
{
   Entering routine 'WriteResGnFile'.
   {
      Creating ResGnFile 'out.dicmaleresgn.aiff'.
      NrOfSamples = 36629.
      Created buffer of size 20000.
      Current node = 0.
      Closing ResGnFile.
   }  ['WriteResGnFile']

   Entering routine 'WriteResFile'.
   {
      Creating ResFile 'out.dicmaleres.aiff'.
      Current node = 0.
      Total size of gaps = 5485.
      Closing ResFile.
   }  ['WriteResFile']

   Entering routine 'WriteLPCsFile'.
   {
      Creating LPCsFile 'out.dicmaleap.lvsa'.
      Current node = 0.
      Closing LPCsFile.
   }  ['WriteLPCsFile']

   Entering routine 'WriteFile'.
   {
```

```
      Creating File 'out.dicmale.aiff'.
      Current node = 1.
      Closing File.
   }  ['WriteFile']

}  ['OutPulseData2OutputFiles']

Entering routine 'ReleasePulseData'.
{
   Current node = 0.
}  ['ReleasePulseData']

Entering routine 'ReleasePulseData'.
{
   Current node = 0.
}  ['ReleasePulseData']

Ok. Ready.
```

Appendix-IV : Output of PROG-2


```

ProsMan.Syn Version Oct 28 1994.
Prosodic manipulations using a synthetic residual signal.
Copyright (c) IPO 1994. All Rights Reserved.

Male or Female [M/F]?                                    : m
Impulse or Glottal pulse excitation [I/G]?               : g
Use gain interpolation [Y/N]?                            : y
Use pitch interpolation? [Y/N]?                          : y
Use pitch contour in LPC file [Y/N]?                     : n
Time multiplication factor X (0.2 <= X <= 5).            : 1
Pitchperiod multiplication factor X (0.2 <= X <= 5).  : 1
LPC adj factor X (bandwidth change) (0.8 <= X <= 1). : 0.96
Tp (as factor of pitchperiod) (0 < Tp < 1).             : 0.3
Te (as factor of pitchperiod) (0.400 < Tp < 1).         : 0.4
Ta (as factor of pitchperiod) (0 < Ta < 1).             : 0.01
Alfa (as in LF model) (0 <= Tp <= 3).                   : 0.2

Using the following parameter values :
--------------------------------------
Male / Female           = Male
UseGainInterpolation    = Yes
UsePitchInterpolation   = Yes
UsePitchcontour         = No
TimeMulVal              = 1.000000
PitchPeriodMulval       = 1.000000
LPCAdjFac               = 0.960000
Impulse / Glottal pulse = Glottal pulse
Tp (as factor of T0)    = 0.300000
Te (as factor of T0)    = 0.400000
Ta (as factor of T0)    = 0.010000
Alfa                    = 0.200000
--------------------------------------

Entering routine 'InputFiles2InPulseData'.
{
   Entering routine 'ReadResGnFile'.
   {
      Opening ResGnFile 'dicmaleresgn.aiff'.
      NrOfSamples = 36660.
      Created buffer of length 20000.
      Current sample at time 31. Found 1 excitation instants.
      Closing ResGnFile.
   }  ['ReadResGnFile']

   Entering routine 'ReadLPCsFile'.
   {
      Opening LPCsFile 'dicmaleap.lvsa'.
      Number of parameter frames = 731.
      Current node = 0 (time = 31). Duration of frame 1 from 50 to 100.
      Closing LPCsFile.
   }  ['ReadLPCsFile']

}  ['InputFiles2InPulseData']

Entering routine 'ProsodicManipulation'.
{
   InNode = 0, OutNode = 0.
}  ['ProsodicManipulation']

Entering routine 'OutPulseData2OutputFiles'.
{
   Entering routine 'WriteResGnFile'.
   {
      Creating ResGnFile 'out.dicmaleresgn.aiff'.
      NrOfSamples = 36629.
      Created buffer of size 20000.
      Current node = 0.
      Closing ResGnFile.
   }  ['WriteResGnFile']

   Entering routine 'WriteResFile'.
   {
      Creating ResFile 'out.dicmaleres.aiff'.
      Current node = 0.
      Total size of gaps = 31.
      Closing ResFile.
   }  ['WriteResFile']

   Entering routine 'WriteLPCsFile'.
   {
      Creating LPCsFile 'out.dicmaleap.lvsa'.
      Current node = 0.
      Closing LPCsFile.
   }  ['WriteLPCsFile']

   Entering routine 'WriteFile'.
   {
      Creating File 'out.dicmale.aiff'.
      Current node = 1.
      Closing File.
   }  ['WriteFile']
```

```
} ['OutPulseData2OutputFiles']

Entering routine 'ReleasePulseData'.
{
   Current node = 0.
} ['ReleasePulseData']

Entering routine 'ReleasePulseData'.
{
   Current node = 0.
} ['ReleasePulseData']

Ok. Ready.
```