

# Prosodic manipulation of speech using knowledge of instants of significant excitation

**Citation for published version (APA):**

Yegnanarayana, B., & Teunen, R. (1994). *Prosodic manipulation of speech using knowledge of instants of significant excitation*. (IPO-Rapport; Vol. 1029). Instituut voor Perceptie Onderzoek (IPO).

**Document status and date:**

Published: 23/12/1994

**Document Version:**

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

**Take down policy**

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

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 impulses 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 segmental 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 coefficients (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 and audio signals) software available at the Institute 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 synthesis 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, *Institute 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.

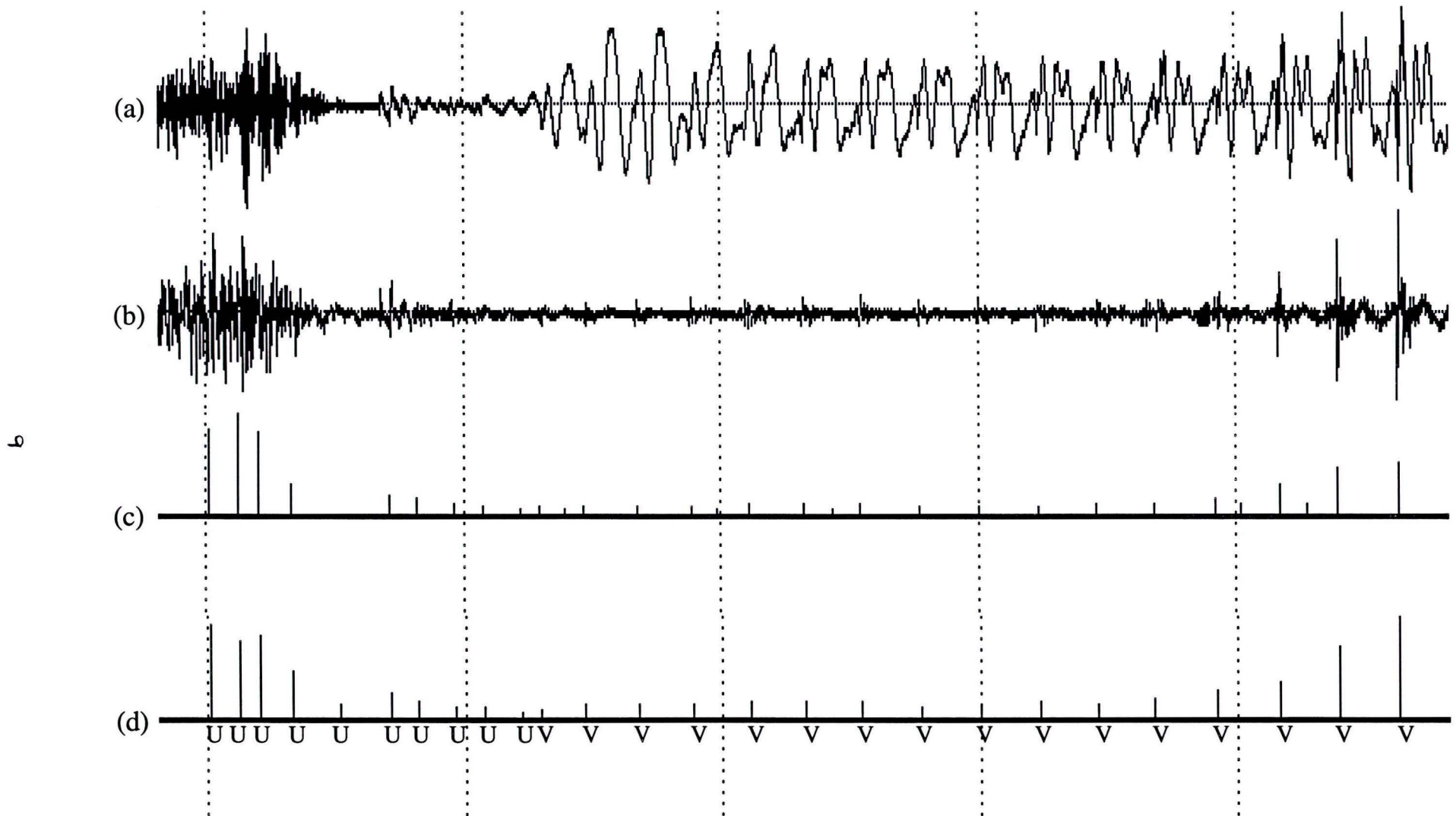


Fig. 1. Illustration of the extraction of instants of significant excitation (a) speech signal, (b) linear prediction residual, (c) gain plot showing the strength of the impulses of the significant instants, and (d) edited gainplot with V/U labels.

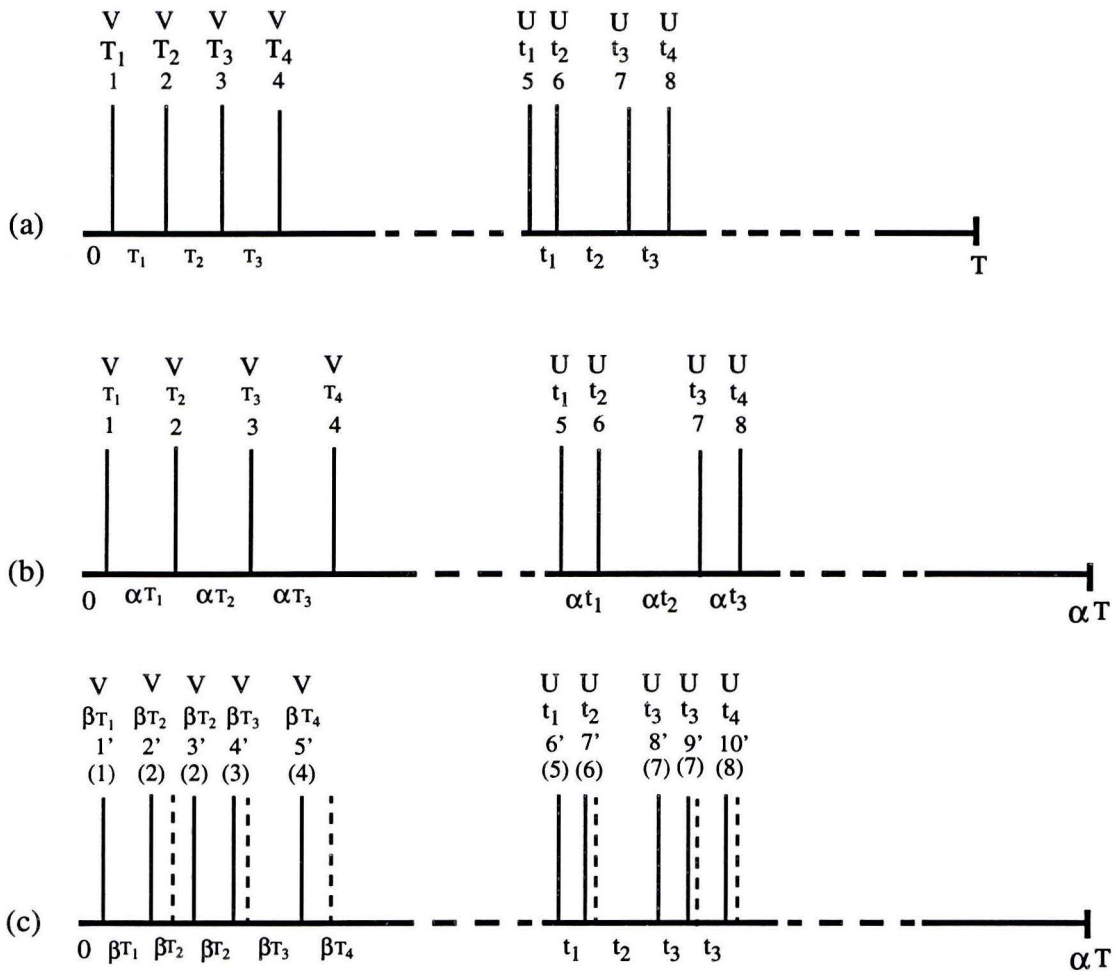


Fig. 2. Illustration of prosodic manipulation. V and U are the voiced and unvoiced labels for the instants.  $T_i$ 's are intervals between instants in voiced regions, and  $t_i$ 's are intervals between instants in unvoiced regions.

(a) Instants in the input data.

(b) Instants shifted due to time scale multiplication factor  $\alpha$ .

(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  $\beta$ . Note that the spacing between impulses is  $\beta T_i$  in voiced regions and  $t_i$  in unvoiced regions.

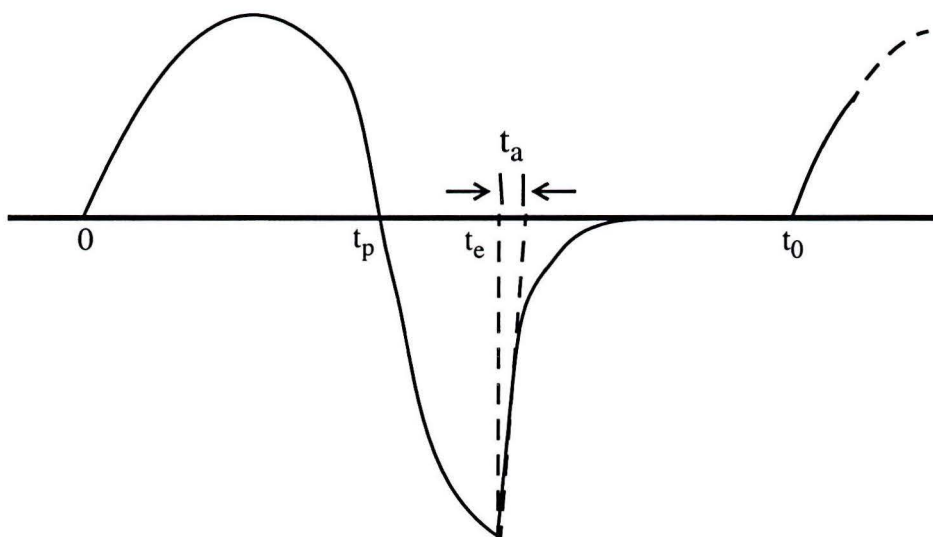


Fig. 3. Shape of the differentiated glottal waveform used for synthetic case, showing the different parameters as in LF model (Ref. 9), where  $t_0$  is one pitch period.

# Appendix-I : PROG-1

```
1 /*=====
2 * Name: ReleasePulseData *
3 * *
4 * Description: Releases memory occupied by the given pulse data structure *
5 * 'PulseData'. *
6 * *
7 * Arguments: PulseData = Pulse data structure to be released. *
8 *=====*/
9 ReleasePulseData(PulseData)
10 {
11     If PulseData is not NULL then do :
12     {
13         For all nodes of PulseData do :
14         {
15             Free memory used by residual signal of current node.
16             Free memory used by LPC frame of current node.
17         }
18         Free memory used by the structure PulseData itself.
19     }
20 }
21 EOF : ReleasePulseData()
22
23
24 /*=====
25 * Name: ReadResGnFile *
26 * *
27 * Description: Read excitation instant data from file (gain is ignored). *
28 * This routine also creates the nodes of InPulseData. Each *
29 * non-zero sample of the inputfile is an excitation instant. *
30 * *
31 * Arguments: FileName = Name of the inputfile which contains the *
32 * residual gain data (instant data). *
33 * InPulseData = The information gathered will be stored in *
34 * this structure. *
35 *=====*/
36 ReadResGnFile(FileName, InPulseData)
37 {
38     Open an AIFF file named FileName with residual gain data and create a
39     buffer to speed up reading of the file.
40
41     For all samples in the file do :
42     {
43         If the buffer is empty, fill it with new data from file.
44
45         Get a new sample value from the buffer.
46
47         If the current sample value is greater than zero, an instant
48         is found. Create a new node for InPulseData, fill the node with
49         data & do some housekeeping.
50     }
51     Finish the job; close the file and exit.
52 }
53 EOF : ReadResGnFile()
54
55
56 /*=====
57 * Name: ReadLPCsFile *
58 * *
59 * Description: Read LPCs from file & store them in InPulseData. *
60 * *
61 * Arguments: FileName = Name of the inputfile which contains the *
62 * LPC data. *
63 * InPulseData = The information gathered will be stored in *
64 * this structure. *
65 *=====*/
66 ReadLPCsFile(FileName, InPulseData)
67 {
68     Open a LVSA file named FileName with LPC data and copy some header
69     information to InPulseData. This information is needed later to create a new
70     LVSA-file with LPC data.
71
72     For all nodes of InPulseData do :
73     {
74         Allocate memory for LPC frame of current node.
75
76         While the correct LPC frame is not found do :
77         {
78             Read the current LPC frame.
79
80             Check if this LPC frame is better than the next frame. If not
81             goto next frame & try again.
82         }
83     }
84     Close the file and exit the routine.
85 }
86 EOF : ReadLPCsFile()
87
88
89 /*=====
90 * Name: ReadFile *
91 * *
92 * Description: Read speech signal from file and calculate the residual *
```

```

93 *          signal. Store the residual signal in the nodes. Note that *
94 *          the speech signal is not stored in the nodes, it is used *
95 *          only to calculate the residual. *
96 * *
97 * Arguments:   FileName      = Name of the inputfile which contains the *
98 *              speech signal. *
99 *              InPulseData = The information gathered will be stored in *
100 *                  this structure. *
101 *=====*/
102 ReadFile(FileName, InPulseData)
103 {
104     Open an AIFF file named FileName with speech data and copy some header data
105     to InPulseData. This information is needed later to create a new AIFF-files
106     (for the residual gain data, the residual signal and the speech signal).
107
108     Create a buffer for the file to speed up reading.
109
110     For all nodes of InPulseData do :
111     {
112         Adjust the LPC frame of the current node so the frame duration is
113         correct (= half pitchperiod of the previous node + half pitchperiod
114         of current node).
115
116         Calculate what piece of the speech signal to pick for generation of
117         the residual signal. Only the residual signal centred around the
118         instant time (node time) is used (so you might get gaps between
119         residual data of adjacent nodes).
120
121         Allocate memory for the residual signal of the current node.
122
123         If data in speech buffer is wrong, fill it with new data from file.
124
125         Setup inverse LPC filter; obtain the coefficients.
126         Setup pre-emphasis filter; obtain the coefficient.
127
128         Filter the speech signal; first pre-emphasis, then inverse LPC filtering.
129         Store the resulting residual signal in the current node. Also calculate
130         the gain of the residual signal.
131     }
132     Close the file and exit the routine.
133 }
134 EOF : ReadFile()
135
136
137 /*=====*/
138 * Name:          InputFiles2InPulseData *
139 * *
140 * Description:   Read all inputdata from files & put it in InputPulseData. *
141 * *
142 * Arguments:    InResGnFileName = Name of the inputfile which contains the *
143 *                residual gain data. *
144 *                InLPCsFileName = Name of the inputfile which contains the *
145 *                LPC data. *
146 *                InFileName     = Name of the inputfile which contains the *
147 *                speech signal. *
148 *                InPulseData    = The information gathered will be stored *
149 *                in this structure. *
150 *=====*/
151 InputFiles2InPulseData(InResGnFileName, InLPCsFileName, InFileName, &InPulseData)
152 {
153     Allocate memory for InPulseData & fill it with zero's.
154
155     Call ReadResGnFile(InResGnFileName, InPulseData).
156     Call ReadLPCsFile(InLPCsFileName, InPulseData).
157     Call ReadFile(InFileName, InPulseData).
158 }
159 EOF : InputFiles2InPulseData()
160
161
162 /*=====*/
163 * Name:          ProsodicManipulation *
164 * *
165 * Description:   Generate OutPulseData by manipulating InPulseData given *
166 *                the parameters TimeMulVal and PitchPeriodMulVal (and the *
167 *                pitch contour specified in the LPC frames). *
168 * *
169 * Arguments:    InPulseData      = This structure contains all the *
170 *                inputdata about the speech signal. *
171 *                TimeMulVal      = Time multiplication factor. *
172 *                PitchPeriodMulVal = Pitchperiod multiplication factor. *
173 *                WindowWidthMulVal = Window width multiplication factor *
174 *                for reducing the number of residual *
175 *                samples around each instant. *
176 *                UsePitchContour = TRUE if the pitchcontour specified *
177 *                in the LPC frames should be used. *
178 *                UsePitchInterpolation = TRUE if pitch(period)interpolation *
179 *                should be used. *
180 *                UseGainInterpolation = TRUE if gain interpolation should *
181 *                be used. *
182 *                OutPulseData    = This structure will be filled with *
183 *                data about the output speech *
184 *                signal. The structure can then be *

```

# Appendix-I : PROG-1

```
185 *                                     used to generate outputfiles. *
186 *=====*/
187 ProsodicManipulation(InPulseData, TimeMulVal, PitchPeriodMulVal,
188   WindowWidthMulVal, UsePitchContour, UsePitchInterpolation,
189   UseGainInterpolation, &OutPulseData)
190 {
191   Allocate memory for OutPulseData and initialize the structure.
192
193   For all samples do :
194   {
195     Look for nearest input node & other 'bounding' input node.
196
197     Allocate memory for the new output node & copy some data from the
198     nearest input node.
199
200     Calculate the pitchperiod of the output node. Use pitch interpolation
201     if the segment is voiced and no pitchcontour is specified. If a
202     pitchcontour is specified in the LPC file and the segment is voiced
203     then take the pitchperiod from the LPC frame of the nearest input node.
204     In unvoiced segments, always just copy the pitchperiod from the
205     nearest input node.
206
207     Calculate what the gain scale factor of the residual should be.
208     This is the quotient of the interpolated gain and the original gain.
209
210     Calculate the new window width.
211
212     Allocate memory for LPC frame & residual signal. Copy LPC frame from
213     input node to output node & adjust some fields (frame duration, pitch
214     and gain).
215
216     Copy the residual signal from the nearest input node to the current
217     output node. If the pitchperiod is smaller, the center of the
218     original residual signal should be copied, so samples should be deleted
219     from both sides of the original residual signal. If the pitchperiod is
220     larger, just copy the original residual signal.
221
222     Apply gain correction, windowing and highpass filtering to the
223     residual signal (in that order). Highpass filtering is used to remove
224     the low frequency bias due to manipulating the residual signal.
225
226     Calculate the time of the next output node.
227   }
228   Ready! Exit routine.
229 }
230 EOF : ProsodicManipulation()
231
232
233 /*=====*/
234 * Name:          WriteResGnFile *
235 * * * * * * * * * * * * * * * * *
236 * Description:   Write excitation instant & gain data to file. *
237 * * * * * * * * * * * * * * * * *
238 * Arguments:     OutPulseData = This structure contains inputdata for the *
239 *                generation of the file. *
240 *                FileName     = Name of outputfile for the residual gain *
241 *                data. *
242 *=====*/
243 WriteResGnFile(OutPulseData, FileName)
244 {
245   Create an AIFF file named FileName for residual gain data and create a
246   buffer to speed up writing to the file.
247
248   Until all samples (and all nodes of OutPulseData) have been processed do :
249   {
250     If the current sample number is equal to the time of the current node,
251     append the (scaled up) gain sample of the current node to the buffer and
252     goto the next node. Otherwise append a zero to the buffer.
253
254     If the buffer is full, write it to file.
255   }
256   Close file and exit routine.
257 }
258 EOF : WriteResGnFile()
259
260
261 /*=====*/
262 * Name:          WriteResFile *
263 * * * * * * * * * * * * * * * * *
264 * Description:   Write residual signal to file. *
265 * * * * * * * * * * * * * * * * *
266 * Arguments:     OutPulseData = This structure contains inputdata for the *
267 *                generation of the file. *
268 *                FileName     = Name of outputfile for the residual signal. *
269 *=====*/
270 WriteResFile(OutPulseData, FileName)
271 {
272   Create an AIFF file named FileName for the residual data and create a
273   buffer to speed up writing to the file.
274
275   Until all samples have been processed do :
276   {
```



```

277         If not in gap between nodes, get a sample from the residual signal of
278         the current node. Otherwise make the sample value zero.
279
280         Write the residual sample to the buffer and goto the next node.
281         If the buffer is full, write it to file.
282     }
283     Close file and exit routine.
284 }
285 EOF : WriteResFile()
286
287
288 /*=====
289 *   Name:          WriteLPCsFile
290 *
291 *   Description:   Write LPC data to file.
292 *
293 *   Arguments:     OutPulseData = This structure contains inputdata for the
294 *                   generation of the file.
295 *                   FileName    = Name of outputfile for the LPC data.
296 *=====*/
297 WriteLPCsFile(OutPulseData, FileName)
298 {
299     Create a LVSA file named FileName for the LPC data.
300
301     Check to see if the file should start with a dummy frame. The start time of
302     a LPC frame should be equal to the time of the node to which the frame
303     belongs. If the time of the first node is greater than zero, a dummy frame
304     is needed. If so do :
305     {
306         Allocate memory for dummy LPC frame.
307         Fill dummy frame with dummy data.
308         Write dummy frame to file.
309     }
310     For nodes of OutPulseData do :
311     {
312         Write LPC frame of current node to file.
313     }
314     Close file and exit routine.
315 }
316 EOF : WriteLPCsFile()
317
318
319 /*=====
320 *   Name:          WriteFile
321 *
322 *   Description:   Write speech signal to file.
323 *
324 *   Arguments:     OutPulseData = This structure contains inputdata for the
325 *                   generation of the file.
326 *                   LPCAdjFac   = Multiplication factor for the LPC bandwidth
327 *                   change.
328 *                   FileName    = Name of outputfile for the speech signal.
329 *=====*/
330 WriteFile(OutPulseData, LPCAdjFac, FileName)
331 {
332     Create an AIFF file named FileName for speech data and create a buffer to
333     speed up writing to the file.
334
335     For all samples of the residual signal do :
336     {
337         If not in gap between nodes, get a sample from the residual signal of
338         the current node. Otherwise make the sample value zero.
339
340         Calculate what the current LPC adjustment factor should be (to
341         change the bandwidth of the all-pole filter). Use LPCAdjFac.
342
343         Setup LPC Filter; calculate the coefficients.
344         Setup de-emphasis Filter; obtain the coefficient.
345
346         Apply the filters to the residual sample and store the resulting
347         speech sample in the buffer. First apply the LPC filter, then
348         the de-emphasis filter.
349
350         If the buffer is full, write it to file.
351     }
352     Close file and exit routine.
353 }
354 EOF : WriteFile()
355
356
357 /*=====
358 *   Name:          OutPulseData2OutputFiles
359 *
360 *   Description:   Write data from OutPulseData to files.
361 *
362 *   Arguments:     OutPulseData = This structure contains inputdata for
363 *                   the generation of the files.
364 *                   LPCAdjFac   = Multiplication factor for LPC bandwidth
365 *                   change.
366 *                   OutResGnFileName = Name of outputfile for residual gain
367 *                   data.
368 *                   OutResFileName = Name of outputfile for residual data.

```

# Appendix-I : PROG-1

```
369 *           OutLPCsFileName = Name of outputfile for LPC data. *
370 *           OutFileName     = Name of outputfile for speech data. *
371 *=====*/
372 OutPulseData2OutputFiles(OutPulseData, LPCAdjFac, OutResGnFileName,
373   OutResFileName, OutLPCsFileName, OutFileName)
374 {
375     Call WriteResGnFile(OutPulseData, OutResGnFileName).
376     Call WriteResFile(OutPulseData, OutResFileName).
377     Call WriteLPCsFile(OutPulseData, OutLPCsFileName).
378     Call WriteFile(OutPulseData, LPCAdjFac, OutFileName).
379 }
380 EOF : OutPulseData2OutputFiles()
381
382
383 /*=====*/
384 *   Name:           main *
385 * *
386 *   Description:    A basic (& not very user-friendly) interface for the *
387 *                 routines above. Use only for test purposes. *
388 *=====*/
389 main()
390 {
391     Put some program information on the screen (name, version and copyrights).
392
393     Ask if male voice should be used or female voice.
394     Ask if gain interpolation should be used.
395     Ask if pitch interpolation should be used.
396     Ask if pitch contour in LPC file should be used. If not, then
397     the original pitch contour will be maintained.
398     Ask for time multiplication factor (float).
399     Ask for pitchperiod multiplication factor (float).
400     Ask for windowwidth multiplication factor (float).
401     Ask for LPC adjustment factor (to change the bandwidth) (float).
402
403     Setup filenames & print what parameter values will be used.
404
405     Call InputFiles2InPulseData; read inputfiles and put data in the
406     structure InPulseData.
407
408     If call to InputFiles2InPulseData was ok, call ProsodicManipulation;
409     create structure OutPulseData from InPulseData, using the given parameters.
410
411     If call to ProsodicManipulation was ok, call OutPulseData2OutputFile;
412     write data from the structure OutPulseData to the outputfiles.
413
414     Release structures.
415     If there was an error, say so.
416 }
417 EOF : main()
```

```

1  /*=====*/
2  *   Name:           ResWave_Impulse                               *
3  *   *
4  *   Description:    Impulse excitation : The routines gives back the requested *
5  *   *               sample of residual signal of the given node. This routine *
6  *   *               also signals the calling routine when to update the LPCs *
7  *   *               and when to change the bandwidth. The routine creates a *
8  *   *               zero-mean impulse-residual signal. *
9  *   *
10 *   Arguments:     Node      = Pointer to the node. *
11 *   *              ResSampleNr = Number of the requested sample. Should be *
12 *   *                       equal to or greater than zero and smaller *
13 *   *                       than the pitchperiod of Node. *
14 *   *              UpdateLPCs = Target variable. This flag signals the calling *
15 *   *                       routine that the LPCs should be updated or *
16 *   *                       LPC bandwidth change should be applied. *
17 *   *              ErrorCode  = Return errorcode. MSG_SUCCESS if ok, else *
18 *   *                       MSG_?. *
19 *   Return:        Sample of residual wave. *
20 *=====*/
21 ResWave_Impulse(Node, ResSampleNr, &UpdateLPCs, &ErrorCode)
22 {
23     Set UpdateLPCs to the correct value; if ResSampleNr is zero, then the
24     LPCs should be updated. If the segment is voiced and ResSampleNr is
25     equal or greater than half the pitchperiod, then LPC bandwidth change
26     should be applied.
27
28     if this is a new node, check if the frame is unvoiced and if so do :
29     {
30         Allocate memory for buffer to store noise in. Fill the buffer with
31         noise (with maximum gain).
32
33         Calculate the mean value of the noise and adjust the signal so
34         its zero-mean.
35
36         Calculate the gain of the noise and adjust it if that gain is
37         not equal to the gain specified in Node.
38     }
39     if the segment is unvoiced return a sample from the noise buffer,
40     else return an impulse (scaled gain of node) if ResSampleNr is zero
41     else some small negative value to make the signal zero mean.
42 }
43 EOF : ResWave_Pulse()
44
45 /*=====*/
46 *   Name:           ResWave_GlottalPulse                           *
47 *   *
48 *   Description:    Glottal pusle excitation : The routines gives back the *
49 *   *               requested sample of residual signal of the given node. This *
50 *   *               routine also signals the calling routine when to update the *
51 *   *               LPCs and when to change the bandwidth. The routine uses a *
52 *   *               model derived from the LF model to create residual signal. *
53 *   *
54 *   *               Residual(t) = A*exp(Alfa*t)*sin(t*pi/Tp), for 0<=t<=Tp, *
55 *   *                       = B*exp(Alfa*t)*sin(t*pi/Tp), for Tp<=t<=Te, *
56 *   *                       = B*exp(Alfa*Te)*sin(Te*pi/Tp) *
57 *   *                       (exp(-(t-Te)/Ta) - *
58 *   *                       exp(-(T0-Te)/Ta)) / *
59 *   *                       (1-exp(-(T0-Te)/Ta)),      for Te<=t<=T0. *
60 *   *
61 *   *               T0      = Pitchperiod. *
62 *   *
63 *   Arguments:     Node      = Pointer to the node. *
64 *   *              ResSampleNr = Number of the requested sample. Should be *
65 *   *                       equal to or greater than zero and smaller *
66 *   *                       than the pitchperiod of Node. *
67 *   *              UpdateLPCs = Target variable. This flag signals the calling *
68 *   *                       routine that the LPCs should be updated or *
69 *   *                       LPC bandwidth change should be applied. *
70 *   *              ErrorCode  = Return errorcode. MSG_SUCCESS if ok, else *
71 *   *                       MSG_?. *
72 *   *
73 *   Return:        Sample of residual wave. *
74 *=====*/
75 ResWave_GlottalPulse(Node, ResSampleNr, &UpdateLPCs, &ErrorCode)
76 {
77     if the pitchperiod is less than 5, use a zero-mean impulse waveform.
78     Else if this is a new node, check if the frame is unvoiced and if so do :
79     {
80         Allocate memory for buffer to store noise in. Fill the buffer with
81         noise (with maximum gain).
82
83         Calculate the mean value of the noise and adjust the signal so
84         its zero-mean.
85
86         Calculate the gain of the noise and adjust it if that gain is
87         not equal to the gain specified in Node.
88     }
89     Else if this is a new node and the frame is voiced do :
90     {
91         Calculate Tp, Te and Ta.
92

```

```

93
94     Allocate memory for buffers to store the signal in. The positive
95     part of the signal is stored in a different buffer than the negative
96     part, so the signal can be made zero-mean by just scaling the signal
97     of one of the two buffers.
98
99     Fill the buffers with the glottal pulse.
100
101     Calculate the mean value of the signal and adjust the signal so
102     its zero-mean.
103
104     Calculate the gain of the signal and adjust it if that gain is
105     not equal to the gain specified in Node.
106 }
107 If the segment is unvoiced do :
108 {
109     Set UpdateLPCs so that the LPCs are updated at the beginning of
110     the signal. Return a sample from the noise buffer.
111 }
112 Else do :
113 {
114     Set UpdateLPCs so that the LPCs are updated at time Tp and the
115     bandwidth is changed for times less than Tp.
116
117     Return a sample from the signal buffers.
118 }
119 }
120 EOF : ResWave_GlottalPulse()
121
122
123 /*=====
124 *   Name:           ReleasePulseData
125 *
126 *   Description:    Releases memory occupied by the given pulse data structure
127 *                   'PulseData'.
128 *
129 *   Arguments:      PulseData = Pulse data structure to be released.
130 *=====*/
131 ReleasePulseData(PulseData)
132 {
133     If PulseData is not NULL then do :
134     {
135         For all nodes of PulseData do :
136         {
137             Free memory used by LPC frame of current node.
138         }
139         Free memory used by the structure PulseData itself.
140     }
141 }
142 EOF : ReleasePulseData()
143
144
145 /*=====
146 *   Name:           ReadResGnFile
147 *
148 *   Description:    Read excitation instant & gain data from file. This
149 *                   routine also creates the nodes of InPulseData. Each
150 *                   non-zero sample of the inputfile is an excitation instant.
151 *
152 *   Arguments:      FileName      = Name of the inputfile which contains the
153 *                   residual gain data.
154 *                   InPulseData = The information gathered will be stored in
155 *                   this structure.
156 *=====*/
157 ReadResGnFile(FileName, InPulseData)
158 {
159     Open an AIFF file named FileName with residual gain data and create a
160     buffer to speed up reading of the file.
161
162     For all samples in the file do :
163     {
164         If the buffer is empty, fill it with new data from file.
165
166         Get a new sample value from the buffer. If greater than zero this value
167         is the gain of the residual signal scaled up by a factor.
168
169         If the current sample value is greater than zero, an instant
170         is found. Create a new node for InPulseData, fill the node with
171         data & do some housekeeping.
172     }
173     Finish the job; close the file and exit.
174 }
175 EOF : ReadResGnFile()
176
177
178 /*=====
179 *   Name:           ReadLPCsFile
180 *
181 *   Description:    Read LPCs from file & store them in InPulseData.
182 *
183 *   Arguments:      FileName      = Name of the inputfile which contains the
184 *                   LPC data.

```

```

185 *           InPulseData = The information gathered will be stored in *
186 *           this structure. *
187 *=====*/
188 ReadLPCsFile(FileName, InPulseData)
189 {
190     Open a LVSA file named FileName with LPC data and copy some header
191     information to InPulseData. This information is needed later to create a new
192     LVSA-file with LPC data.
193
194     For all nodes of InPulseData do :
195     {
196         Allocate memory for LPC frame of current node.
197
198         While the correct LPC frame is not found do :
199         {
200             Read the current LPC frame.
201
202             Check if this LPC frame is better (nearer) than the next frame.
203             If not goto next frame & try again.
204         }
205     }
206     Close the file and exit the routine.
207 }
208 EOF : ReadLPCsFile()
209
210
211 /*=====*/
212 * Name:           InputFiles2InPulseData *
213 * * * * * *
214 * Description:    Read all inputdata from files & put it in InputPulseData. *
215 * * * * * *
216 * Arguments:     InResGnFileName = Name of the inputfile which contains the *
217 *                 residual gain data. *
218 *                 InLPCsFileName = Name of the inputfile which contains the *
219 *                 LPC data. *
220 *                 InPulseData   = The information gathered will be stored *
221 *                 in this structure. *
222 *=====*/
223 InputFiles2InPulseData(InResGnFileName, InLPCsFileName, &InPulseData)
224 {
225     Allocate memory for InPulseData & fill it with zero's.
226
227     Call ReadResGnFile(InResGnFileName, InPulseData).
228     Call ReadLPCsFile(InLPCsFileName, InPulseData).
229 }
230 EOF : InputFiles2InPulseData()
231
232
233 /*=====*/
234 * Name:           ProsodicManipulation *
235 * * * * * *
236 * Description:    Generate OutPulseData by manipulating InPulseData given *
237 *                 the parameters TimeMulVal and PitchPeriodMulVal (and the *
238 *                 pitch contour specified in the LPC frames). *
239 * * * * * *
240 * Arguments:     InPulseData       = This structure contains all the *
241 *                 inputdata about the speech signal. *
242 *                 TimeMulVal       = Time multiplication factor. *
243 *                 PitchPeriodMulVal = Pitchperiod multiplication factor. *
244 *                 UsePitchContour  = TRUE if the pitchcontour specified *
245 *                 in the LPC frames should be used. *
246 *                 UsePitchInterpolation = TRUE if pitch(period)interpolation *
247 *                 should be used. *
248 *                 UseGainInterpolation = TRUE if gain interpolation should *
249 *                 be used. *
250 *                 OutPulseData     = This structure will be filled with *
251 *                 data about the output speech *
252 *                 signal. The structure can then be *
253 *                 used to generate outputfiles. *
254 *=====*/
255 ProsodicManipulation(InPulseData, TimeMulVal, PitchPeriodMulVal,
256 UsePitchContour, UsePitchInterpolation, UseGainInterpolation, &OutPulseData)
257 {
258     Allocate memory for OutPulseData and initialize the structure.
259
260     For all samples do :
261     {
262         Look for nearest input node & other 'bounding' input node.
263
264         Allocate memory for the new output node & copy some data from the
265         nearest input node.
266
267         Calculate the pitchperiod of the output node. Use pitch interpolation
268         if the segment is voiced and no pitchcontour is specified. If a
269         pitchcontour is specified in the LPC file and the segment is voiced
270         then take the pitchperiod from the LPC frame of the nearest input node.
271         In unvoiced segments, always just copy the pitchperiod from the
272         nearest input node.
273
274         Calculate what the gain of the residual should be. This value is the
275         interpolated gain (using the nearest input node & other bounding
276         input node.

```

```

277
278     Allocate memory for the LPC frame. Copy LPC frame from
279     input node to output node & adjust some fields (frame duration, pitch
280     and gain).
281
282     Calculate the time of the next output node.
283 }
284 Ready! Exit routine.
285 }
286 EOF : ProsodicManipulation()
287
288
289 /*=====
290 * Name:          WriteResGnFile
291 *
292 * Description:   Write excitation instant & gain data to file.
293 *
294 * Arguments:    OutPulseData = This structure contains inputdata for the
295 *                generation of the file.
296 *                FileName     = Name of outputfile for the residual gain
297 *                data.
298 *=====
299 WriteResGnFile(OutPulseData, FileName)
300 {
301     Create an AIFF file named FileName for residual gain data and create a
302     buffer to speed up writing to the file.
303
304     Until all samples (and all nodes of OutPulseData) have been processed do :
305     {
306         If the current sample number is equal to the time of the current node,
307         append the (scaled up) gain sample of the current node to the buffer
308         and goto the next node. Otherwise append a zero to the buffer.
309
310         If the buffer is full, write it to file.
311     }
312     Close file and exit routine.
313 }
314 EOF : WriteResGnFile()
315
316
317 /*=====
318 * Name:          WriteResFile
319 *
320 * Description:   Write residual signal to file.
321 *
322 * Arguments:    OutPulseData = This structure contains inputdata for the
323 *                generation of the file.
324 *                FileName     = Name of outputfile for the residual signal.
325 *                ResWaveProc  = Pointer to the procedure which generates
326 *                the waveform of the residual signal.
327 *=====
328 WriteResFile(OutPulseData, FileName, ResWaveProc)
329 {
330     Create an AIFF file named FileName for the residual data and create a
331     buffer to speed up writing to the file.
332
333     Until all samples have been processed do :
334     {
335         If not in gap between nodes, get a sample from the residual signal of
336         the current node by calling ResWaveProc. Otherwise make the sample
337         value zero.
338
339         Write the residual sample to the buffer and goto the next node.
340         If the buffer is full, write it to file.
341     }
342     Close file and exit routine.
343 }
344 EOF : WriteResFile()
345
346
347 /*=====
348 * Name:          WriteLPCsFile
349 *
350 * Description:   Write LPC data to file.
351 *
352 * Arguments:    OutPulseData = This structure contains inputdata for the
353 *                generation of the file.
354 *                FileName     = Name of outputfile for the LPC data.
355 *=====
356 WriteLPCsFile(OutPulseData, FileName)
357 {
358     Create a LVSA file named FileName for the LPC data.
359
360     Check to see if the file should start with a dummy frame. The start time of
361     a LPC frame should be equal to the time of the node to which the frame
362     belongs. If the time of the first node is greater than zero, a dummy frame
363     is needed. If so do :
364     {
365         Allocate memory for dummy LPC frame.
366         Fill dummy frame with dummy data.
367         Write dummy frame to file.
368     }

```

```

369     For nodes of OutPulseData do :
370     {
371         Write LPC frame of current node to file.
372     }
373     Close file and exit routine.
374 }
375 EOF : WriteLPCsFile()
376
377
378 /*=====
379 *   Name:           WriteFile
380 *
381 *   Description:    Write speech signal to file.
382 *
383 *   Arguments:      OutPulseData = This structure contains inputdata for the
384 *                   generation of the file.
385 *                   LPCAdjFac   = Multiplication factor for the LPC bandwidth
386 *                   change.
387 *                   FileName    = Name of outputfile for the speech signal.
388 *                   ResWaveProc = Pointer to the procedure which generates
389 *                   the waveform of the residual signal.
390 *=====*/
391 WriteFile(OutPulseData, LPCAdjFac, FileName, ResWaveProc)
392 {
393     Create an AIFF file named FileName for speech data and create a buffer to
394     speed up writing to the file.
395
396     For all samples of the residual signal do :
397     {
398         If not in gap between nodes, get a sample from the residual signal of
399         the current node by calling ResWaveProc. Otherwise make the sample
400         value zero.
401
402         Calculate what the current LPC adjustment factor should be (to
403         change the bandwidth of the all-pole filter). Use LPCAdjFac.
404
405         Setup LPC Filter; calculate the coefficients.
406         Setup de-emphasis Filter; calculate the coefficient.
407
408         Apply the filters to the residual sample and store the resulting
409         speech sample in the buffer. First apply the LPC filter, then
410         the de-emphasis filter.
411
412         If the buffer is full, write it to file.
413     }
414     Close file and exit routine.
415 }
416 EOF : WriteFile()
417
418
419 /*=====
420 *   Name:           OutPulseData2OutputFiles
421 *
422 *   Description:    Write data from OutPulseData to files.
423 *
424 *   Arguments:      OutPulseData = This structure contains inputdata for
425 *                   the generation of the files.
426 *                   LPCAdjFac   = Multiplication factor for LPC bandwidth
427 *                   change.
428 *                   OutResGnFileName = Name of outputfile for residual gain
429 *                   data.
430 *                   OutResFileName = Name of outputfile for residual data.
431 *                   OutLPCsFileName = Name of outputfile for LPC data.
432 *                   OutFileName   = Name of outputfile for speech data.
433 *                   ResWaveProc   = Pointer to the procedure which generates
434 *                   the waveform of the residual signal.
435 *=====*/
436 OutPulseData2OutputFiles(OutPulseData, LPCAdjFac, OutResGnFileName,
437     OutResFileName, OutLPCsFileName, OutFileName, ResWaveProc)
438 {
439     Call WriteResGnFile(OutPulseData, OutResGnFileName).
440     Call WriteResFile(OutPulseData, OutResFileName, ResWaveProc).
441     Call WriteLPCsFile(OutPulseData, OutLPCsFileName).
442     Call WriteFile(OutPulseData, LPCAdjFac, OutFileName, ResWaveProc).
443 }
444 EOF : OutPulseData2OutputFiles()
445
446
447 /*=====
448 *   Name:           main
449 *
450 *   Description:    A basic (& not very user-friendly) interface for the
451 *                   routines above. Use only for test purposes.
452 *=====*/
453 main()
454 {
455     Put some program information on the screen (name, version and copyrights).
456
457     Ask if male voice should be used or female voice.
458     Ask if gain interpolation should be used.
459     Ask if pitch interpolation should be used.
460     Ask if pitch contour in LPC file should be used. Is not, then

```

```
461 the original pitch contour will be maintained.
462 Ask for time multiplication factor (float).
463 Ask for pitchperiod multiplication factor (float).
464 Ask for LPC adjustment factor (to change the bandwidth).
465 Ask if glottal pulse excitation or impulse excitation should be used.
466
467 If glottal pulse excitation should be used do :
468 {
469     Ask for Tp-factor (Tp = Tp-factor * pitchperiod). See ResWave_GlottalPulse.
470     Ask for Te-factor (Te = Te-factor * pitchperiod). See ResWave_GlottalPulse.
471     Ask for Ta-factor (Ta = Ta-factor * pitchperiod). See ResWave_GlottalPulse.
472     Ask for alfa. See ResWave_GlottalPulse.
473 }
474 Setup filenames & print what parameter values will be used.
475
476 Call InputFiles2InPulseData; read inputfiles and put data in the
477 structure InPulseData.
478
479 If call to InputFiles2InPulseData was ok, call ProsodicManipulation;
480 create structure OutPulseData from InPulseData, using the given parameters.
481
482 If call to ProsodicManipulation was ok, call OutPulseData2OutputFile (with
483 the right parameters) to write data from the structure OutPulseData to
484 the outputfiles.
485
486 Release structures.
487 If there was an error, say so.
488 }
489 EOF : main()
```



```

1
2 ProsMan.Nat Version Oct 28 1994.
3 Prosodic manipulations using a natural residual signal.
4 Copyright (c) IPO 1994. All Rights Reserved.
5
6 Male or female [M/F]? : m
7 Use gain interpolation [Y/N]? : Y
8 Use pitch interpolation? [Y/N]? : Y
9 Use pitch contour in LPC file [Y/N]? : n
10 Time multiplication factor X (0.2 <= X <= 5). : 1
11 Pitchperiod multiplication factor X (0.2 <= X <= 5). : 1
12 Windowwidth multiplication factor X (0 <= X <= 1). : 1
13 LPC adj factor X (bandwidth change) (0.8 <= X <= 1). : 0.96
14
15 Using the following parameter values :
16 -----
17 Male / Female = Male
18 UseGainInterpolation = Yes
19 UsePitchInterpolation = Yes
20 UsePitchcontour = No
21 TimeMulVal = 1.000000
22 PitchPeriodMulval = 1.000000
23 WindowWidthMulVal = 1.000000
24 LPCAdjFac = 0.960000
25 -----
26
27 Entering routine 'InputFiles2InPulseData'.
28 {
29   Entering routine 'ReadResGnFile'.
30   {
31     Opening ResGnFile 'dicmaleresgn.aiff'.
32     NrOfSamples = 36660.
33     Created buffer of length 20000.
34     Current sample at time 0. Found 1 excitation instants.
35     Closing ResGnFile.
36   } ['ReadResGnFile']
37
38   Entering routine 'ReadLPCsFile'.
39   {
40     Opening LPCsFile 'dicmaleap.lvsa'.
41     Number of parameter frames = 731.
42     Current node = 0 (time = 31). Duration of frame 1 from 50 to 100.
43     Closing LPCsFile.
44   } ['ReadLPCsFile']
45
46   Entering routine 'ReadFile'.
47   {
48     Opening File 'dicmale.aiff'.
49     NrOfSamples = 36608.
50     Created buffer of length 20000.
51     Current node = 0 (time = 31). ResTimeOffset = -12, WindowWidth = 25.
52     Closing File.
53     Buffer misses = 2, buffer hits = 560.
54   } ['ReadFile']
55
56 } ['InputFiles2InPulseData']
57
58 Entering routine 'ProsodicManipulation'.
59 {
60   HP filter coefficient Alfa = 0.945093.
61   InNode = 0, OutNode = 0.
62   MaxResGainScaling = 1.000000, MinResGainScaling = 1.000000.
63 } ['ProsodicManipulation']
64
65 Entering routine 'OutPulseData2OutputFiles'.
66 {
67   Entering routine 'WriteResGnFile'.
68   {
69     Creating ResGnFile 'out.dicmaleresgn.aiff'.
70     NrOfSamples = 36629.
71     Created buffer of size 20000.
72     Current node = 0.
73     Closing ResGnFile.
74   } ['WriteResGnFile']
75
76   Entering routine 'WriteResFile'.
77   {
78     Creating ResFile 'out.dicmaleres.aiff'.
79     Current node = 0.
80     Total size of gaps = 5485.
81     Closing ResFile.
82   } ['WriteResFile']
83
84   Entering routine 'WriteLPCsFile'.
85   {
86     Creating LPCsFile 'out.dicmaleap.lvsa'.
87     Current node = 0.
88     Closing LPCsFile.
89   } ['WriteLPCsFile']
90
91   Entering routine 'WriteFile'.
92   {

```

## Appendix-III : Output of PROG-1

```
93         Creating File 'out.dicmale.aiff'.
94         Current node = 1.
95         Closing File.
96     } ['WriteFile']
97
98 } ['OutPulseData2OutputFiles']
99
100 Entering routine 'ReleasePulseData'.
101 {
102     Current node = 0.
103 } ['ReleasePulseData']
104
105 Entering routine 'ReleasePulseData'.
106 {
107     Current node = 0.
108 } ['ReleasePulseData']
109
110 Ok. Ready.
```

```

1
2 ProsMan.Syn Version Oct 28 1994.
3 Prosodic manipulations using a synthetic residual signal.
4 Copyright (c) IPO 1994. All Rights Reserved.
5
6 Male or Female [M/F]? : m
7 Impulse or Glottal pulse excitation [I/G]? : g
8 Use gain interpolation [Y/N]? : Y
9 Use pitch interpolation? [Y/N]? : Y
10 Use pitch contour in LPC file [Y/N]? : n
11 Time multiplication factor X (0.2 <= X <= 5). : 1
12 Pitchperiod multiplication factor X (0.2 <= X <= 5). : 1
13 LPC adj factor X (bandwidth change) (0.8 <= X <= 1). : 0.96
14 Tp (as factor of pitchperiod) (0 < Tp < 1). : 0.3
15 Te (as factor of pitchperiod) (0.400 < Tp < 1). : 0.4
16 Ta (as factor of pitchperiod) (0 < Ta < 1). : 0.01
17 Alfa (as in LF model) (0 <= Tp <= 3). : 0.2
18
19 Using the following parameter values :
20 -----
21 Male / Female = Male
22 UseGainInterpolation = Yes
23 UsePitchInterpolation = Yes
24 UsePitchcontour = No
25 TimeMulVal = 1.000000
26 PitchPeriodMulval = 1.000000
27 LPCAdjFac = 0.960000
28 Impulse / Glottal pulse = Glottal pulse
29 Tp (as factor of T0) = 0.300000
30 Te (as factor of T0) = 0.400000
31 Ta (as factor of T0) = 0.010000
32 Alfa = 0.200000
33 -----
34
35 Entering routine 'InputFiles2InPulseData'.
36 {
37   Entering routine 'ReadResGnFile'.
38   {
39     Opening ResGnFile 'dicmaleresgn.aiff'.
40     NrOfSamples = 36660.
41     Created buffer of length 20000.
42     Current sample at time 31. Found 1 excitation instants.
43     Closing ResGnFile.
44   } ['ReadResGnFile']
45
46   Entering routine 'ReadLPCsFile'.
47   {
48     Opening LPCsFile 'dicmaleap.lvsa'.
49     Number of parameter frames = 731.
50     Current node = 0 (time = 31). Duration of frame 1 from 50 to 100.
51     Closing LPCsFile.
52   } ['ReadLPCsFile']
53 } ['InputFiles2InPulseData']
54
55 Entering routine 'ProsodicManipulation'.
56 {
57   InNode = 0, OutNode = 0.
58 } ['ProsodicManipulation']
59
60 Entering routine 'OutPulseData2OutputFiles'.
61 {
62   Entering routine 'WriteResGnFile'.
63   {
64     Creating ResGnFile 'out.dicmaleresgn.aiff'.
65     NrOfSamples = 36629.
66     Created buffer of size 20000.
67     Current node = 0.
68     Closing ResGnFile.
69   } ['WriteResGnFile']
70
71   Entering routine 'WriteResFile'.
72   {
73     Creating ResFile 'out.dicmaleres.aiff'.
74     Current node = 0.
75     Total size of gaps = 31.
76     Closing ResFile.
77   } ['WriteResFile']
78
79   Entering routine 'WriteLPCsFile'.
80   {
81     Creating LPCsFile 'out.dicmaleap.lvsa'.
82     Current node = 0.
83     Closing LPCsFile.
84   } ['WriteLPCsFile']
85
86   Entering routine 'WriteFile'.
87   {
88     Creating File 'out.dicmale.aiff'.
89     Current node = 1.
90     Closing File.
91   } ['WriteFile']
92

```

## Appendix-IV : Output of PROG-2

```
93
94   } ['OutPulseData2OutputFiles']
95
96   Entering routine 'ReleasePulseData'.
97   {
98     Current node = 0.
99   } ['ReleasePulseData']
100
101   Entering routine 'ReleasePulseData'.
102   {
103     Current node = 0.
104   } ['ReleasePulseData']
105
106   Ok. Ready.
```