Chapter 5
STEVIN Can Praat

David Weenink

5.1 Introduction

Appropriate tools are indispensable for the scientist to perform his/her work. This holds true for speech science as well. Many tools exist in this field like, for example, Audacity [1], CSL [4], the MATLAB Signal Processing Toolbox [11] or Wavesurfer [14], to name a few. The Praat program [2] is an extensive application for language, music and speech research that is used by many scientists and students around the globe. A conservative guess from our website download statistics would indicate at least 20,000 users world-wide. Some characteristics that explain its success right from the beginning, are the wide range of features, the user-friendliness and the scriptability, i.e. the possibility to create ones own processing for a series of inputs. The other aspect that adds to the enthusiastic and widespread use is the careful support available. This encompasses user help on diverse levels online, quick response to any questions by email, immediate handling of incidents and solving of problems, and last but not least, an infrastructure for user groups. The knowledge that the Praat program entails, is in this means passed on to many colleagues and students. Also, users have a way to relate to one another and share their insights with regard to the possibilities the Praat program offers. The Praat software is freely available for most current computer platforms like Linux, Windows and Macintosh; it is not available on mobile devices. The manuals, FAQ and help menu are included in the package; the user group is available on the internet.[1] Despite the multitude of features already present in the application, some important functionality was

[1] http://groups.yahoo.com/group/praat-users

D. Weenink (✉)
University of Amsterdam and SpeechMinded, Amsterdam, The Netherlands
e-mail: David.Weenink@uva.nl

P. Spyns and J. Odijk (eds.), *Essential Speech and Language Technology for Dutch*, Theory and Applications of Natural Language Processing,
DOI 10.1007/978-3-642-30910-6_5, © The Author(s) 2013

still missing. We have proposed to develop a number of improvements and added functionality that now has become freely available for speech scientists via the Praat program. This project matched the STEVIN objectives since it delivers important tools to all speech scientists who need state of the art technology to tackle the newest ideas and the largest datasets. The improvements that we have added to the Praat program are the following:

- KlattGrid: an acoustic synthesiser modeled after the Klatt synthesiser.
- VowelEditor: a sound-follows-mouse type editor by which vowel-like sounds can be generated from mouse gestures in the two dimensional formant plane.
- Robust formant frequency analysis.
- Availability of the mathematical functions from the GNU Scientific Library.
- Search and replace with regular expressions.
- Software band filter analysis.

In the rest of this chapter we will discuss these additions in somewhat more detail.[2]

5.2 The KlattGrid Acoustic Synthesiser

Although current speech synthesis is more oriented towards unit synthesis there is still need for a formant based synthesiser. A formant based speech synthesiser is a fundamental tool for those fields of speech research where detailed control of speech parameters is essential. For example, research on adult learner's vowel contrast in second language acquisition may require tight control over speech stimuli parameters while this also holds true for the investigation of vowel categorisation development of infants [6]. For the synthesis of different voices and voice characteristics and to model emotive speech formant based synthesis systems are still in use [12].

A very well known and widely used formant based speech synthesiser is the Klatt synthesiser [7,8]. One reason for its popularity is that the FORTRAN reference code was freely available as well as several C language implementations. In Fig. 5.1 we show a schematic diagram of this synthesiser with the vocal tract section realised with filters in cascade. Since a KlattGrid is based on the same design this is also the diagram of a KlattGrid. The synthesiser essentially consists of four parts:

1. The *phonation part* generates voicing as well as aspiration. It is represented by the top left dotted box labeled with the number 1 in its top right corner.
2. The *coupling part* models coupling between the phonation part and the next part, the vocal tract. In the figure it is indicated by the dotted box labeled with the number 2.
3. The *vocal tract part* filters the sound generated by the phonation part. The top right dotted box labeled 3 shows this part as a cascade of formant and

[2]The sections on the KlattGrid is a modified version of the [15] article.

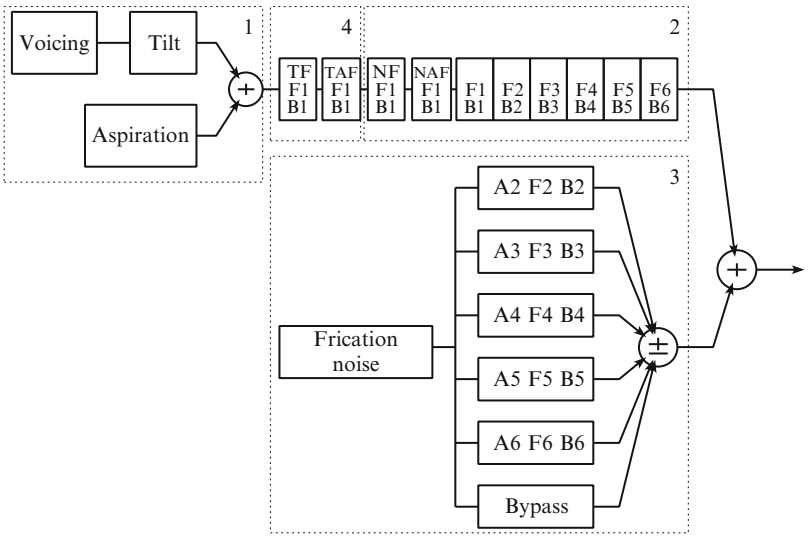

Fig. 5.1 Schematic diagram of a KlattGrid/Klatt synthesiser with the vocal tract part realised as filters in cascade

antiformant filters. The vocal tract part can also be modeled with formant filters that are in parallel instead of in cascade.

4. The *frication part* generates frication noise and is represented by the dotted box labeled 4.

A number of implementations of the Klatt synthesiser exist nowadays. However, they all show some of the limitations of the original design that originates from times that computer memory and processing power were relatively scarce. Necessarily, compromises had to be made at that time in order to achieve reasonable performance.

We present the KlattGrid speech synthesiser which is based on the original description of Klatt [7,8]. There are several new, not necessarily innovative, aspects in the KlattGrid in comparison with some other Klatt-type synthesisers.

- A Klatt synthesiser is frame-based, i.e. parameters are modeled to be constant during the interval of a frame, typically some 5 or 10 ms. As a consequence, instants of parameter change have to be synchronised on a frame basis. This poses some difficulty in modeling events where timing is important such as a rapidly changing amplitude for plosive bursts. We have removed this limitation by modeling *all* parameters in a KlattGrid as *tiers*. A tier represents a parameter contour as a function of time by (*time*, *value*) points. Parameter values at any time can be calculated from these time stamps by some kind of interpolation. For example, a formant frequency tier with two (*time*, *frequency*) points, namely 800 Hz at a time of 0.1 s and 300 Hz at 0.3 s, is to be interpreted as a formant frequency contour that is constant at 800 Hz for all times before 0.1 s, constant

at 300 Hz for all times after 0.3 s and linearly interpolated for all times between 0.1 and 0.3 s (i.e. 675 Hz at 0.15 s, 550 Hz at 0.2 s, and so on). By leaving the frame-based approach of previous synthesisers, all parameter timings become transparent and only moments of parameter change have to be specified.
- In a Klatt synthesiser one can normally define some six to eight oral formants and one nasal and one tracheal formant/antiformant pair. In a KlattGrid any number of oral formants, nasal formants and nasal antiformants, tracheal formants and tracheal antiformants are possible.
- In a Klatt synthesiser there is only one set of formant frequencies that has to be shared between the vocal tract part and the frication part. In a KlattGrid the formant frequencies in the frication part and the vocal tract part have been completely decoupled from one another.
- In the Klatt synthesiser the glottal flow function has to be specified beforehand. A KlattGrid allows varying the form of the glottal flow function as a function of times.
- In the Klatt synthesiser only the frequency and bandwidth of the first formant can be modified during the open phase. In a KlattGrid there is no limit to the number of formants and bandwidths that can be modified during the open phase of the glottis.
- In Klatt's synthesiser all amplitude parameters have been quantised to 1 dB levels beforehand. In a KlattGrid there is no such quantisation. All amplitudes are represented according to the exact specifications. Quantisation only takes place on the final samples of a sound when it has to be played or saved to disk (playing with 16-bit precision, for example). Of course sampling frequencies can be chosen freely.
- A KlattGrid is fully integrated into the speech analysis program Praat [2]. This makes the synthesiser available on the major desktop operating systems of today: Linux, Windows and Mac OS X. At the same time all scripting, visualisations and analysis methods of the Praat program become directly available for the synthesised sounds.

More details on the KlattGrid can be found in the following sections which will describe the four parts of the synthesiser in more detail. This description will be a summary of the synthesiser parameters and how they were implemented.

5.2.1 The Phonation Part

The phonation part serves two functions:

1. It generates voicing. Part of this voicing are timings for the glottal cycle. The part responsible for these timings is shown by the box labeled "Voicing" in Fig. 5.1. The start and end times of the open phase of the glottis serve to:
    - Generate glottal flow during the open phase of the glottis.
    - Generate breathiness, i.e. noise that occurs only during the open phase of the glottis.

- Calculate when formant frequencies and bandwidths change during the open phase (if formant change information is present in the coupling part).

2. It generates aspiration. This part is indicated by the box labeled "Aspiration" in Fig. 5.1. In contrast with breathiness, aspiration may take place independently of any glottal timing.

The phonation parameter tiers do not all independently modify the glottal flow function. Some of the parameters involved have similar spectral effects, however, in this article we do not go into these details too much and only briefly summarise a tier's function in the phonation part. For an extensive overview of the effects of several glottal flow parameters on the source spectrum see for example the article of Doval et al. [5]. The following 11 tiers form the phonation part:

Pitch tier. For voiced sounds the pitch tier models the fundamental frequency as a function of time. Pitch equals the number of glottal opening/closing cycles per unit of time. In the absence of flutter and double pulsing, the pitch tier is the only determiner for the instants of glottal closure. Currently pitch interpolation happens on a linear frequency scale but other interpolation, for example on a log scale, can be added easily.

Voicing amplitude tier. The voicing amplitude regulates the maximum amplitude of the glottal flow in dB. A flow with amplitude 1 corresponds to approximately 94 dB. To produce a voiced sound it is essential that this tier is not empty.

Flutter tier. Flutter models a kind of "random" variation of the pitch and it is input as a number from zero to one. This random variation can be introduced to avoid the mechanical monotonic sound whenever the pitch remains constant during a longer time interval. The fundamental frequency is modified by a flutter component according to the following semi-periodic function that we adapted from [7]: $F_0'(t) = 0.01 \cdot \text{flutter} \cdot F_0 \cdot (\sin(2\pi 12.7t) + \sin(2\pi 7.1t) + \sin(2\pi 4.7t))$

Open phase tier. The open phase tier models the open phase of the glottis with a number between zero and one. The open phase is the fraction of one glottal period that the glottis is open. The open phase tier is an optional tier, i.e. if no points are defined then a sensible default for the open phase is taken (0.7). If the open phase becomes smaller, necessarily the high frequency content of the source spectrum will increase.

Power1 and power2 tiers. These tiers model the form of the glottal flow function during the open phase of the glottis as $\text{flow}(t) = t^{\text{power1}} - t^{\text{power2}}$, where $0 \leq t \leq 1$ is the relative time that runs from the start to the end of the open phase. For the modelation of a proper vocal tract flow it is essential that the value of power2 is always larger than the value of power1. If these tiers have no values specified by the user, default values power1 = 3 and power2 = 4 are used. Figure 5.2 will show the effect of the values in these tiers on the form of the flow and its derivative. As power2 mainly influence the falling part of the flow function, we see that the higher the value of this parameter, the faster the flow function reaches zero, i.e. the shorter the closing time of the glottis would be and, consequently, the more high frequency content the glottal spectrum will have.

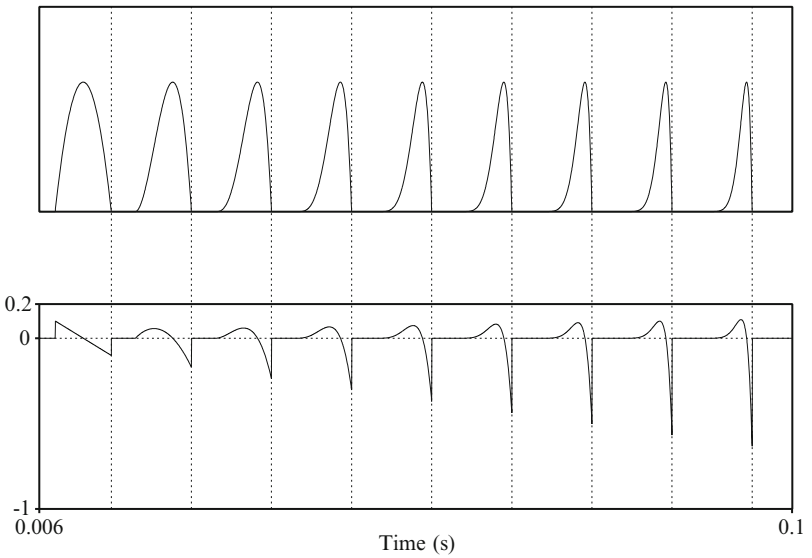

Fig. 5.2 On *top*, nine glottal pulses flow$(t) = t^{\text{power1}} - t^{\text{power2}}$, each one synthesised with a different (power1, power2) combination. Power1 increases linearly from 1, and always power2 = power1 + 1. Consequently, the first pulse on the *left* has power1 = 1 and power2 = 2, while the last one on the *right* has power1 = 9 and power2 = 10. The *bottom panel* on the *left* shows the derivatives of these flow functions. Moments of glottal closure have been indicated with a *vertical dotted line*. The open phase was held at the constant value of 0.7, the pitch was fixed at 100 Hz and the amplitude of voicing was fixed at 90 dB

Collision phase tier. The collision phase parameter models the last part of the flow function with an exponential decay function instead of a polynomial one. A value of 0.04, for example, means that the amplitude will decay by a factor of $e \approx 2.7183$ every 4 % of a period. The introduction of a collision phase will reduce the high frequency content in the glottal spectrum because of the smoother transition towards the closure.

Spectral tilt tier. Spectral tilt represents the extra number of dB's the voicing spectrum should be tilted down at 3,000 Hz [7]. This parameter is necessary to model "corner rounding", i.e. when glottal closure is non simultaneous along the length of the vocal folds. If no points are defined in this tier, spectral tilt defaults to 0 dB and no spectral modifications are made.

Aspiration amplitude tier. The aspiration amplitude tier models the (maximum) amplitude of noise generated at the glottis. The aspiration noise amplitude is, like the voicing amplitudes, specified in dB. This noise is independent of glottal timings and is generated from random uniform noise which is filtered by a very soft low-pass filter.

Breathiness amplitude tier. The breathiness amplitude tier models the maximum noise amplitude during the open phase of the glottis. The amplitude of the breathiness noise, which is plain random uniform noise, is modulated by the glottal flow. It is specified in dB.

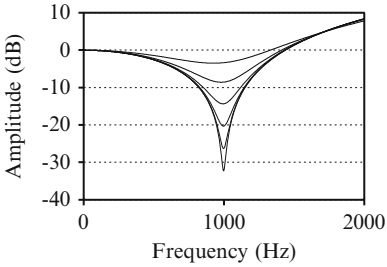

Fig. 5.3 Example of frequency responses of formant/antiformant pairs

Double pulsing tier. The double pulsing tier models diplophonia (by a number from zero to one). Whenever this parameter is greater than zero, alternate pulses are modified. A pulse is modified with this single parameter tier in two ways: it is delayed in time and its amplitude is attenuated. If the double pulsing value is maximum (=1), the time of closure of the first peak coincides with the opening time of the second one (but its amplitude will be zero).

5.2.2 The Vocal Tract Part

The sound generated by the phonation part of a KlattGrid may be modified by the filters of the vocal tract part. These filters are the oral formant filters, nasal formant filters and nasal antiformant filters. A formant filter boosts frequencies and an antiformant filter attenuates frequencies in a certain frequency region. For speech synthesis the vocal tract formant filters can be used in cascade or in parallel. Default these filters are used in cascade as is shown in Fig. 5.1 in the part numbered *3*, unless otherwise specified by the user. Each formant filter is governed by two tiers: a formant *frequency* tier and a formant *bandwidth* tier. In case of parallel synthesis an additional formant *amplitude* tier must be specified.

Formant filters are implemented in the standard way as second order recursive digital filters of the form $y_n = ax_n + by_{n-1} + cy_{n-2}$ as described in [7] (x_i represents input and y_j output). These filters are also called digital resonators. The coefficients b and c at any time instant n can be calculated from the formant frequency and bandwidth values of the corresponding tiers. The a parameter is only a scaling factor and is chosen as $a = 1 - b - c$; this makes the frequency response equal to 1 at 0 frequency. Antiformants are second order filters of the form $y_n = a'x_n + b'x_{n-1} + c'x_{n-2}$. The coefficients a', b' and c' are also determined as described in [7]. When formant filters are used in cascade all formant filters start with the same value at 0 Hz. If used in parallel this is not the case anymore since each formant's amplitude must be specified on a different tier.

As an example we show in Fig. 5.3 the frequency responses of formant/antiformant pairs where both formant and antiformant have the same "formant" frequency, namely 1,000 Hz, but different bandwidths. The bandwidth of the antiformant filter

was fixed at 25 Hz but the bandwidth of the formant filter doubles at each step. From top to bottom it starts at 50 Hz and then moves to 100, 200, 400 and 800 Hz values. A perfect spectral "dip" results without hardly any side-effect on the spectral amplitude. This shows that a combination of a formant and antiformant at the same frequency can model a spectral dip: the formant compensates for the effect on the slope of the spectrum by the antiformant. Best spectral dips are obtained when the formant bandwidth is approximately 500 Hz. For larger bandwidths the dip will not become any deeper, the flatness of the spectrum will disappear and especially the higher frequencies will be amplified substantially.

5.2.3 The Coupling Part

The coupling part of a KlattGrid models the interactions between the phonation part, i.e. the glottis, and the vocal tract. Coupling is only partly shown in Fig. 5.1, only the tracheal formants and antiformants are shown. We have displayed them in front of the vocal tract part after the phonation part because tracheal formants and antiformants are implemented as if they filter the phonation source signal.

Besides the tracheal system with its formants and antiformants the coupling part also models the change of formant frequencies and bandwidths during the open phase of the glottis. With a so-called *delta formant grid* we can specify the amount of change of any formant and/or bandwidth during the open phase of the glottis. The values in the delta tiers will be added to the values of the corresponding formant tiers but *only during the open phase of the glottis*.

In Fig. 5.4 we show two examples where extreme coupling values have been used for a clear *visual* effect. In all panels the generated voiced sounds had a constant 100 Hz pitch, an constant open phase of 0.5 to make the duration of the open and closed phase equal, and only one formant. In the left part of the figure formant bandwidth is held constant at 50 Hz while formant frequency was modified during the open phase. The oral formant frequency was set to 500 Hz. By setting a delta formant point to a value of 500 Hz we accomplish that during the start of the open phase of the glottis, the formant frequency will increase by 500–1,000 Hz. At the end of the open phase it will then decrease to the original 500 Hz value of the formant tier. To avoid instantaneous changes we let the formant frequency increase and decrease with the delta value in a short interval that is one tenth of the duration of the open phase. In a future version of the synthesiser we hope to overcome this limitation [13]. The top display at the left shows the actual first formant frequency as a function of time during the first 0.03 s of the sound. This is exactly the duration of three pitch periods; the moments of glottal closure are indicated by dotted lines. The bottom left display shows the corresponding one-formant sound signal. The 100 Hz periodicity is visible as well as the formant frequency doubling in the second part of each period: we count almost two and a half periods of this formant in the first half of a period, the closed phase, and approximately five during the second half of a period, the open phase. At the right part of the same figure we show the effect of a

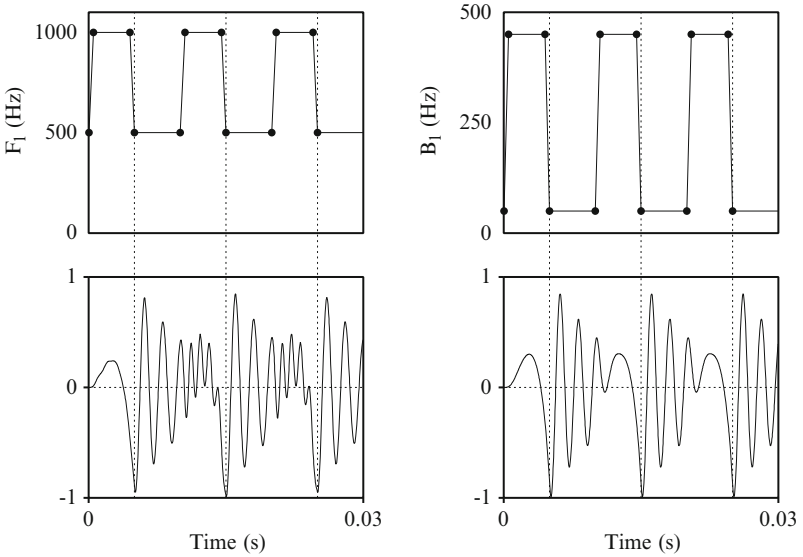

Fig. 5.4 Synthesised example of coupling between vocal tract and the glottis with some extreme formant and bandwidth coupling values. Formant and bandwidth tier at *top*, resulting sounds at *bottom*. Glottal closure times are indicated with *dotted lines*

bandwidth increase from 50 Hz during the closed phase to 450 Hz during the open phase for a one-formant vowel. As before the increase and decrease occur during the first and last one tenth of the open phase interval as is shown in the top right panel. The bottom right panel shows the corresponding synthesised sound.

5.2.4 The Frication Part

The frication part is an independent section in the synthesiser and it gives the opportunity to add the frication noise completely independent of the phonation and the vocal tract part. The frication sound is added to the output of the vocal tract part. A layout of the frication part is shown at the bottom of Fig. 5.1 in the dotted box labeled 4. The following tiers specify the frication sound:

Frication amplitude tier. This tier regulates the maximum amplitude of the noise source in dB before any filtering takes place. In Fig. 5.1 this part is represented by the rectangle labeled "Frication noise". This noise source is uniformly distributed random noise.

Formant frequency and bandwidth tiers. To shape the noise spectrum a number of parallel formant filters are available whose frequencies, bandwidths and amplitudes can be specified. In the figure we have limited the number of formants to five but in principle this number is not limited at all.

Formant amplitude tiers. Each formant is governed by a separate amplitude tier with values in dB. These formant amplitudes act like multipliers and may amplify or attenuate the formant filter input. For formant amplitudes 0 dB means an amplification of 1. Formants can be increased by giving positive dB values and decreased by giving negative values.

Bypass tier. The bypass tier regulates the amplitude of the noise that bypasses the formant filters. This noise is added straight from the noise source to the output of the formant filters. The amplitude is in dB's, where 0 dB means a multiplier of 1.

5.2.5 A KlattGrid Scripting Example

As the preceding sections have shown, the KlattGrid has a large number of parameters. It is difficult to get to grips with the many ways of changing a synthesiser's sound output. To facilitate experimenting with parameter settings, the user interface has been designed to make it easy to selectively include or exclude, in the final sound generation process, some of the parameter tiers that you have given values. For example, if the breathiness amplitude has been defined, hearing the resulting sound with or without breathiness is simply achieved by a selection or deselection of the breathiness tier option in the form that regulates this special playing mode of the KlattGrid synthesiser. The same holds true for the phonation part of the synthesiser whose sound output can be generated separately with some of its parameter tiers selectively turned on or off.

As an example of the synthesisers interface we show a simple example script to synthesise a diphthong. This script can be run in Praat's script editor. The first line of the script creates a new KlattGrid, named "kg", with start and end times of 0 and 0.3 s, respectively. The rest of the parameters on this line specify the number of filters to be used in the vocal tract part, the coupling part and the frication part and are especially important for now (additional filters can always be added to a KlattGrid afterwards).

The second line defines a pitch point of 120 Hz at time 0.1 s. The next line defines a voicing amplitude of 90 dB at time 0.1 s. Because we keep voicing and pitch constant in this example the exact times for these points are not important, as long as they are within the domain on which the kg KlattGrid is defined. With the pitch and voicing amplitude defined, there is enough information in the KlattGrid to produce a sound and we can now Play the KlattGrid (line 4).[3] During 300 ms you will hear the sound as produced by the glottal source alone. This sound normally would be filtered by a vocal tract filter. But we have not defined the vocal tract filter yet (in this case the vocal tract part will not modify the phonation sound).

[3]Despite the fact that it will play correctly, you will receive a warning because not all parameters in the KlattGrid have been specified. For example, the oral formants have not been specified thus far.

In lines 5 and 6 we add a first oral formant with a frequency of 800 Hz at time 0.1 s, and a bandwidth of 50 Hz also at time 0.1 s. The next two lines add a second oral formant at 1,200 Hz with a bandwidth of 50 Hz. If you now play the KlattGrid (line 9), it will sound like the vowel /a/, with a constant pitch of 120 Hz. Lines 10 and 11 add some dynamics to this sound; the first and second formant frequency are set to the values 350 and 600 Hz of the vowel /u/; the bandwidths have not changed and stay constant with values that were defined in lines 6 and 8. In the interval between times 0.1 and 0.3 s, formant frequency values will be interpolated. The result will now sound approximately as an /au/ diphthong.

This script shows that with only a few commands we already may create interesting sounds.

```
Create KlattGrid... kg 0 0.3 6 0 0 0 0 0 0
Add pitch point... 0.1 120
Add voicing amplitude point... 0.1 90
Play
Add oral formant frequency point... 1 0.1 800
Add oral formant bandwidth point... 1 0.1 50
Add oral formant frequency point... 2 0.1 1200
Add oral formant bandwidth point... 2 0.1 50
Play
Add oral formant frequency point... 1 0.3 350
Add oral formant frequency point... 2 0.3 600
Play
```

5.3 Vowel Editor

Although we can perfectly specify sounds with the KlattGrid synthesiser this is not always the most convenient way. Especially for simple vowel-like sounds we need a more intuitive way to specify them. For didactic and demonstration purposes, a straightforward vowel generator of the type sound-follows-mouse has been implemented. We will now present some of its functionality without going into much detail.

By pressing the left mouse button and moving around the mouse pointer in the plane formed by the first two formants, a sound with the formant frequencies of this mouse pointer trajectory will be generated whenever the left mouse button is released. In Fig. 5.5 we show an example of a vowel editor window where the trajectory is indicated with a fat solid line. The main part of the vowel editor shows the plane formed by the first and second formant where the origin is in the upper right corner. The first formant frequency runs from top to bottom while the second formant frequency runs from right to left and the frequency scales are logarithmic, not linear. The dotted lines in the formant plane mark equidistant intervals. At the bottom of the window some characteristics of the generated trajectory are displayed

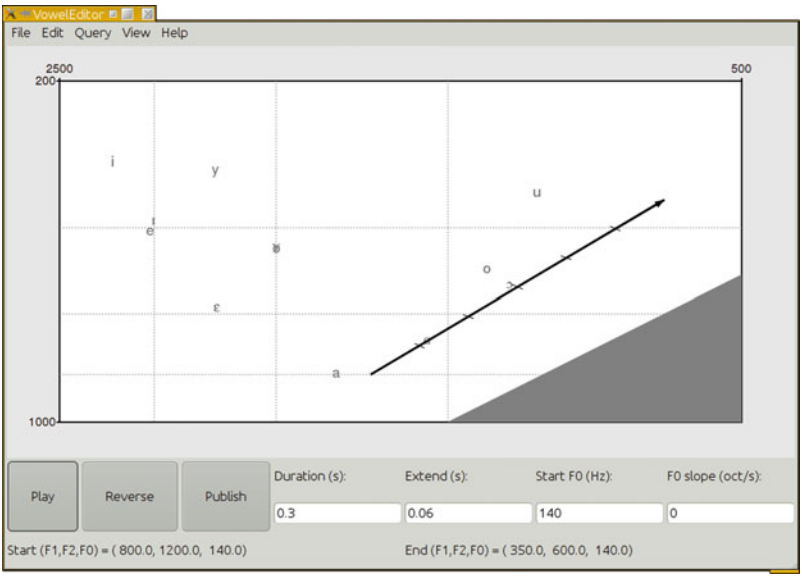

Fig. 5.5 The vowel editor's interface. The *fat line* shows a straight trajectory that starts with a first formant frequency of 800 Hz and the second formant at 1,200 Hz and that ends with a first formant of 350 Hz and the second at 600 Hz. The little bars on the trajectory mark 50 ms time intervals

such as the first two formant frequencies at the start and at the end points of the trajectory. Because the positions of the Dutch vowels in the formant plane are displayed too, we note that this trace corresponds to an /au/ type of diphthong sound. Hitting the big buttons labeled "Play" below the formant plane with the mouse generates a sound according to the trajectory specification and then plays this sound; for the displayed trajectory this will sound as /au/. Hitting the "Reverse" button will reverse the trajectory and then play it; for the displayed trajectory this will sound as /ua/. Apart from generating trajectories by moving the mouse around, trajectories can also be specified (and extended) from vowel editors menu options. For example, with the option "New trajectory . . ." from the Edit menu you can specify the first and second formant frequencies of the start and end points together with the duration of the trajectory. The start and end point will then be connected with a straight line. In fact the trajectory shown in Fig. 5.5 was specified as such. The start first and second formant frequencies and the end first and second formant frequencies had the same values as for the diphthong generated by the script in the previous section. More trajectory generating options exist but these will not be discussed here.

5.4 Robust Formant Frequency Analysis

Many phoneticians rely on formant frequency measurements for their investigations. The current formant frequency measurements in Praat are based on LPC-analysis performed by standard algorithms available in the literature [3, 10]. The formant

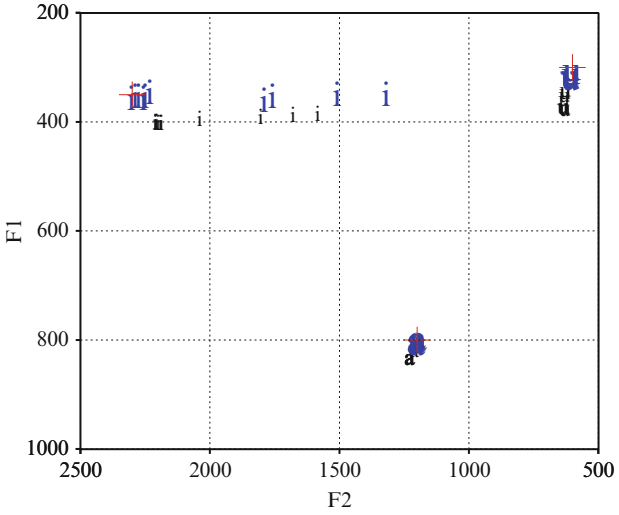

Fig. 5.6 F_1 versus F_2 measure with two different procedures on artificial vowels /u/, /i/ and /a/. The formant frequencies used for the vowel synthesis are indicated with crosses. The outcome of the standard Burg algorithm is indicated with the *smaller font* while the robust analysis is indicated with the larger sans serif font (*blue color*)

frequencies derived by these LPC-analyses algorithms can be improved upon by incorporating knowledge gained from robust statistics. The new formant analysis procedure first down-samples the sound to a frequency twice the highest formant frequency one is interested in. Default this means down-sampling to 10 kHz for male voices and down-sampling to 11 kHz for female voices. Next, LPC coefficients will be determined by the autocorrelation method. Finally, the prediction coefficients are refined by an iterative procedure which uses selective weighing of sample values, i.e. sample values with large prediction errors will be down-weighted as described in [9]. Our own tests indicate substantial improvements in formant frequency and bandwidth accuracy on artificially generated signals and significantly less variation in these measurements with analysis window position for sufficiently long windows that overlap several pitch periods. Of course, this does not immediately guarantee large measurement stability improvements for real speech signals, however, if we can improve significantly the precision of the formant frequency and band filter measurements on test signals, this mere fact should give us more confidence in the newer robust analysis method as compared to the standard methods. Although no known methods exist that always give the "correct" formant frequency values, making some of the current methods more robust with respect to analysis window position should have high priority. This is useful in all research where formant frequencies are used. Especially for high pitched voices this is potentially very interesting as it can be used in basic phonetic research where the development of vowel spaces for young children is the domain.

In Fig. 5.6 we show formant frequencies as measured on three synthetic vowels /u/, /i/ and /a/ by two different algorithms. The first is the Burg algorithm, which is

the standard in Praat for doing formant frequency analysis. The second one is our implementation of the robust LPC algorithm. The vowels were synthesised with an impulsive source of constant pitch. These three vowels are indicated with a cross in the figure. The third, fourth and fifth formant frequencies were set at values of 2,500, 3,500 and 4,500 Hz, respectively. In order to make the analysis more realistic, we varied the frequency interval for the LPC analysis from 4,000 to 5,500 Hz in ten steps. The figure therefore shows ten vowels symbols per analysis method. As we can see the outcome of the robust analysis lie closer to the target values for /u/ and /a/. Only for /i/ for the four lowest analysis frequency intervals, the values are off. However, the rest of the measurement values again lie closer. To show the possible improvement of the robust analysis more tests are necessary on which we will report elsewhere [16].

5.5 Availability of the Mathematical Functions in the GNU Scientific Library

The GNU Scientific Library (GSL) is a GPL licenced library with high quality basic elementary scientific functions.[4] It covers many aspects of mathematics such as special functions like the Bessel, gamma, Legendre and error functions, as well as permutations, linear algebra matrix operations and eigensystems. Statistical functions and distributions are also part of the library. Incorporation of this library therefore facilitates advanced functionality. The GSL library is in constant development. Praat's functionality was extended by making the complete library linkable with Praat. However, we never use the available GSL functions directly but only through a numerical interface. We have also setup an infrastructure for intensive testing of the used GSL functionality. Besides delivering information about valid input ranges of functions it gives information about numerical precision. This information can be used to test whether newer versions of the library comply with current precision constraints. The generation of intensive functional tests before actually writing the code is also heavily promoted by current software theory like Extreme Programming.[5]

5.6 Search and Replace with Regular Expressions

Before we completed this project, the search and replace functions that were available in Praat could only search and replace literal text. By implementing a regular expression engine[6] we have now substantially improved these possibilities by also

[4] http://www.gnu.org/software/gsl/

[5] http://www.extremeprogramming.org/rules/testfirst.html

[6] This regular expression engine is part of the GPL licenced programming editor *nedit*, available via http://www.nedit.org.

allowing regular expressions as patterns for search and replace. The new search and replace functions are available at all levels in Praat, in text-based data types used for annotation such as a TextGrid as well as for Praat's scripting language.

5.7 Software Band Filter Analysis

Software band filter analysis is a kind of spectral analysis and an essential part of any speech analysis program. In general band filter analysis by software proceeds by first dividing the sound in overlapping segments, making a spectrum of each segment and then binning the spectral values. The different band filter methods generally differ in how this binning is performed. For example, in Praat's implemented MelFilter object the binning is performed by overlapping triangular filters. On the mel frequency scale[7] these filters all have the same width. This type of band filter analysis will result in substantial data reduction as the hundreds of spectral values of each segment's spectrum are reduced to some 30 or less mel band filter values. The mel band filter representations can be graphically displayed or used for further processing to mel frequency cepstral coefficients. Further details can be found in the help available in the Praat program an in [16].

5.8 Conclusion

In the field of speech signal processing, Praat is the most widely used computer program. It is still gaining in acceptance. Keeping up-to-date with current techniques and user interface design is very important for all these enthusiastic current day users. Our STEVIN project has successfully implemented a number of useful additions to the Praat program that are being used by Praat's users all over the globe.

Acknowledgements I would like to thank prof. dr. Paul Boersma, project coordinator and release coordinator of the Praat program, for his enthusiasm and his constant emphasis on striving for the highest quality possible. Further thanks go to the other members of the research team prof. dr. Frans Hilgers, prof. dr. Vincent van Heuven and dr. Henk van den Heuvel for their support. Final thanks go to the reviewers for their valuable comments and suggestions.

[7] To go from frequency f in hertz to mel: $\text{mel}(f) = 2{,}595 \cdot \log(1 + f/700)$.

References

1. Audacity (Computer Program). http://audacity.sourceforge.net/
2. Boersma, P., Weenink, D.J.M.: Praat: doing phonetics by computer (computer program). http://www.praat.org/ (2012)
3. Childers, D.: Modern Spectrum Analysis. IEEE Press, New York (1978)
4. CSL Computerized Speech Lab (Computer Program). http://www.kayelemetrics.com/ (2012)
5. Doval, B., d'Alessandro, C., Henrich, N.: The spectrum of glottal flow models. Acta Acust. **92**, 1026–1046 (2006)
6. Escudero, P., Benders, T., Wanrooij, K.: Enhanced bimodal distributions facilitate the learning of second language vowels. J. Acoust. Soc. Am. Express Lett. **130**, 206–212 (2011)
7. Klatt, D.H.: Software for a cascade/parallel formant synthesizer. J. Acoust. Soc. Am. **67**, 971–995 (1980)
8. Klatt, D.H., Klatt, L.C.: Analysis, synthesis, and perception of voice quality variations among female and male talkers. J. Acoust. Soc. Am. **87**, 820–857 (1990)
9. Lee, C.H.: On robust linear prediction of speech. IEEE Trans. Acoust. Speech Signal Process. **36**, 642–649 (1988)
10. Markel, J.D., Gray Jr., A.H.: Linear Prediction of Speech. Springer Verlag, Berlin (1976)
11. MATLAB Signal Processing Toolbox (Computer Program). http://www.mathworks.es/products/signal/index.html (2012)
12. Ternström, S., Sundberg, J.: Formant-based synthesis of singing. In: Proceedings of Eurospeech 2, Antwerp, pp. 4013–4014 (2007)
13. Verhelst, W., Nilens, P.: A modified-superposition speech synthesizer and its application. In: Proceedings of ICASSP, Tokyo, pp. 2007–2010 (1986)
14. Wavesurfer (Computer Program). http://www.speech.kth.se/wavesurfer/ (2012)
15. Weenink, D.J.M.: The KlattGrid acoustic speech synthesizer. In: Proceedings Interspeech 2009, Brighton, pp. 2059–2062 (2009)
16. Weenink, D.J.M.: Speech signal processing by Praat. http://www.fon.hum.uva.nl/david/sspbook/sspbook.pdf (2012, To be published)