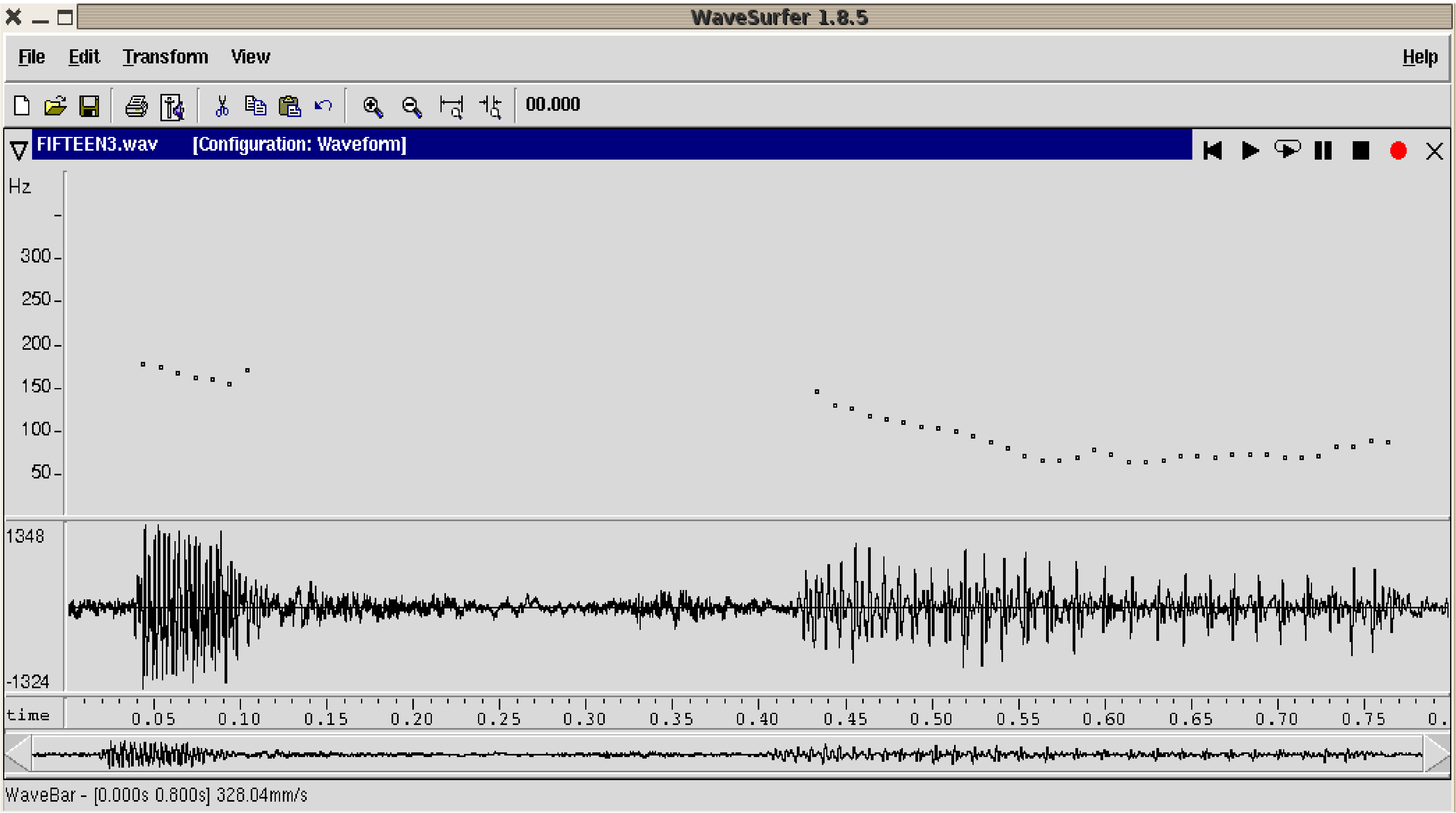
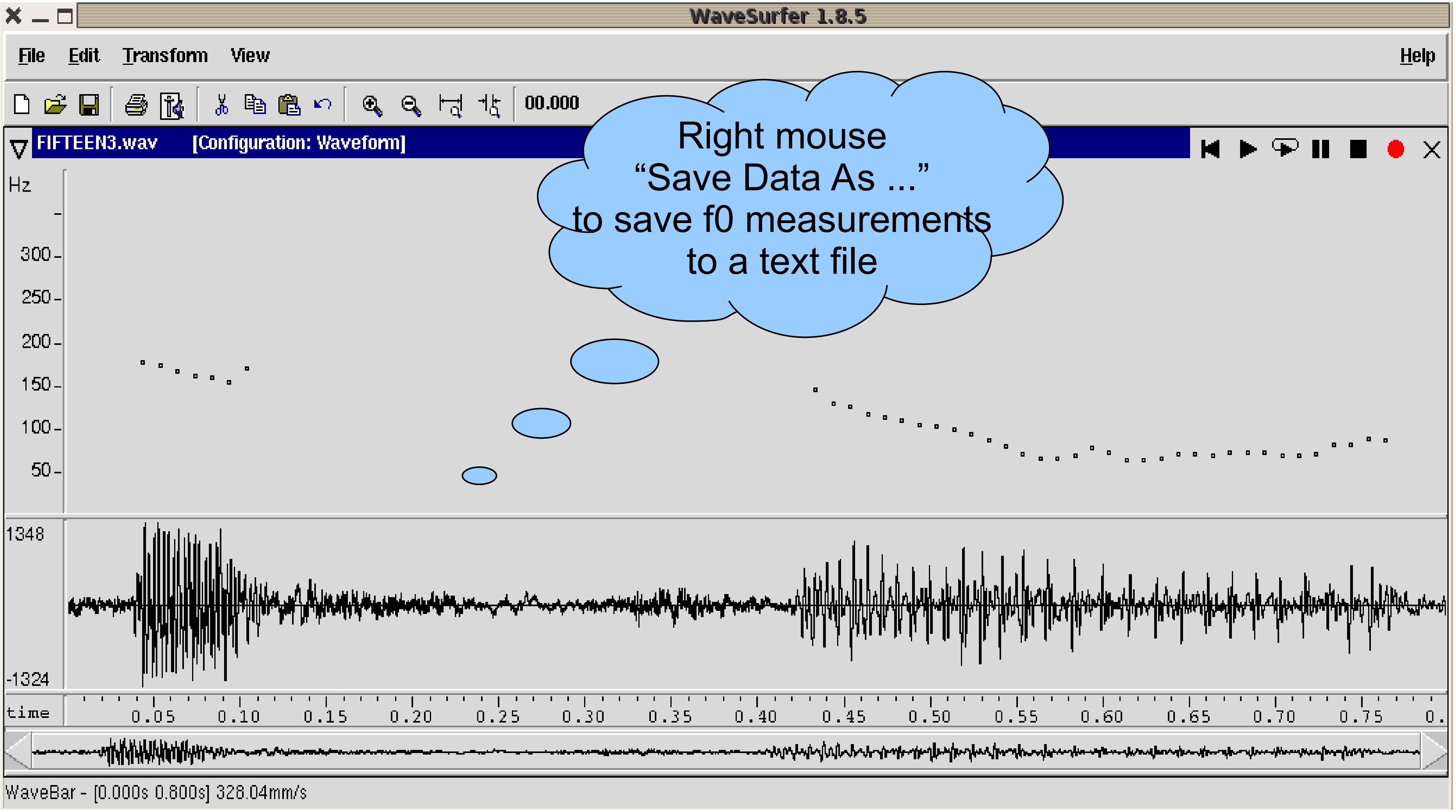
Intonation of FIFTEEN

- Using wavesurfer, ESPS get_f0 to obtain f_0 time-series



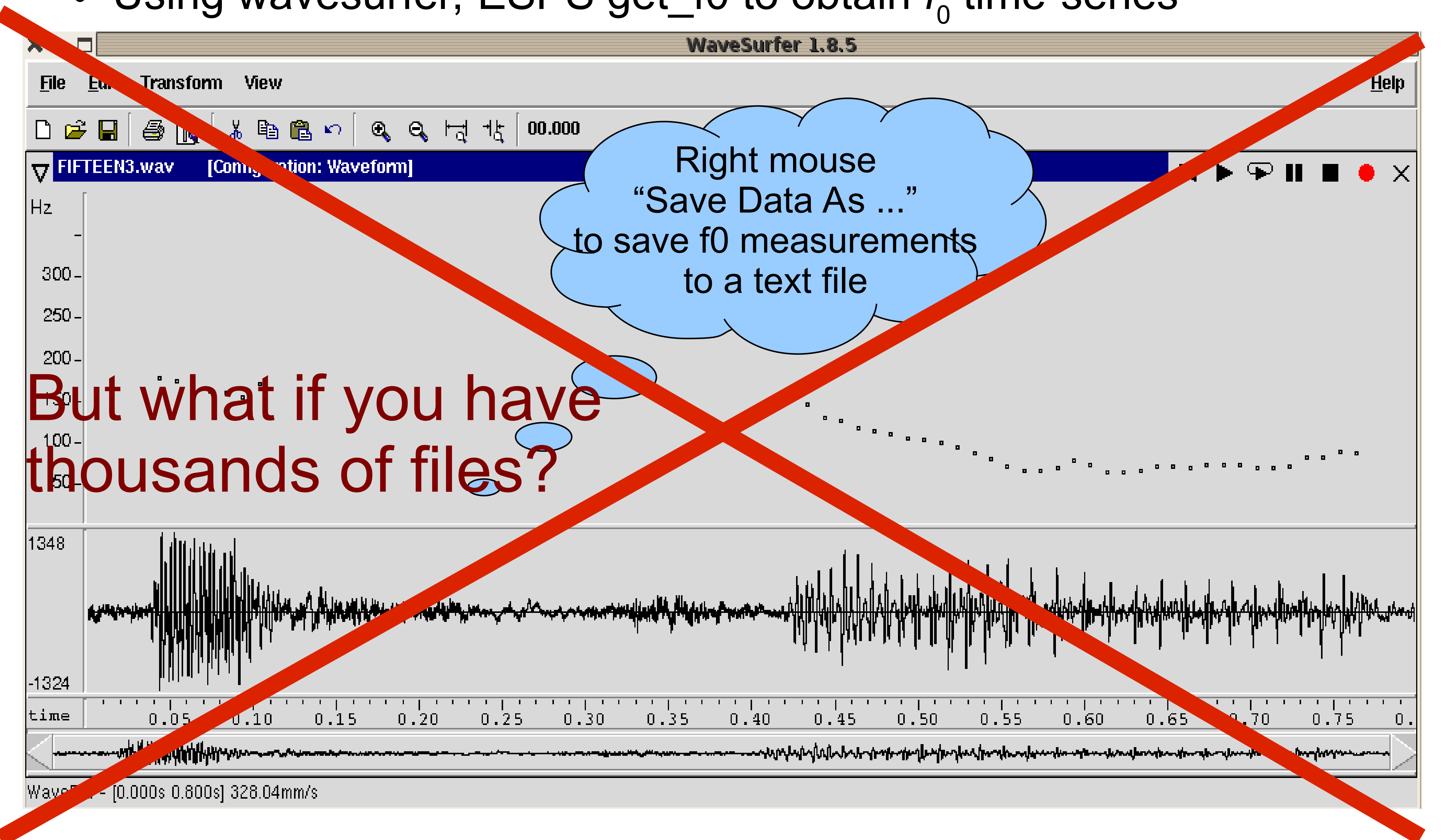
Intonation of FIFTEEN

- Using wavesurfer, ESPS get_f0 to obtain f_0 time-series



Intonation of FIFTEEN

- Using wavesurfer, ESPS get_f0 to obtain f_0 time-series



Intonation of FIFTEEN

- wavesurfer uses the ESPS get_f0 command to obtain f_0 time-series
- syntax: get_f0 [options] *input_file output_file*

ESPS is a package of UNIX-like commands and programming libraries for speech signal processing.

You can download a recent .deb package for ESPS from http://www.phon.ox.ac.uk/releases

David Talkin's paper on get_f0 is here: http://www.ee.columbia.edu/~dpwe/papers/Talkin95-rapt.pdf

UNIVERSITY OF OXFORD

Intonation of FIFTEEN

- Using wavesurfer, ESPS `get_f0` to obtain f_0 time-series
- syntax: `get_f0` [options] *input_file output_file*

```
for i in *.wav
> do get_f0 $i $i.f0
> pplain $i.f0 >$i.f0.csv
> done
```

On one line, that's:

```
for i in *.wav; do get_f0 $i $i.f0; pplain $i.f0 >$i.f0.csv; done
```

UNIVERSITY OF OXFORD

- These .csv files are simple ASCII text files like this: ------------→

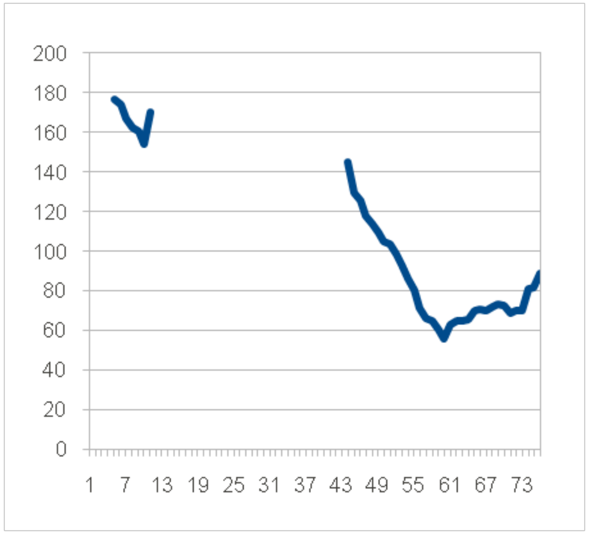
- The first column is f_0, the second is voicing; ignore the other two

```
0 0 0 0.272821
0 0 34.4253 0.551759
0 0 44.9999 0.641592
0 0 242.326 0.515299
176.894 1 401.017 0.553706
174.434 1 399.113 0.931412
167.352 1 378.998 0.951326
162.623 1 358.704 0.927735
160.734 1 356.843 0.931884
154.345 1 250.132 0.617554
170.107 1 159.65 0.843205
0 0 82.2662 0.494668
0 0 92.7429 0.730789
0 0 110.42 0.433576
0 0 71.1711 0.53332
0 0 59.6541 0.419894
0 0 62.9074 0.538716
0 0 53.2908 0.319767
0 0 47.034 0.288603
0 0 38.5281 0.346631
0 0 47.2128 0.452121
0 0 50.4091 0.43822
0 0 44.2441 0.568226
--More--(28%)
```

UNIVERSITY OF OXFORD

Intonation of FIFTEEN3

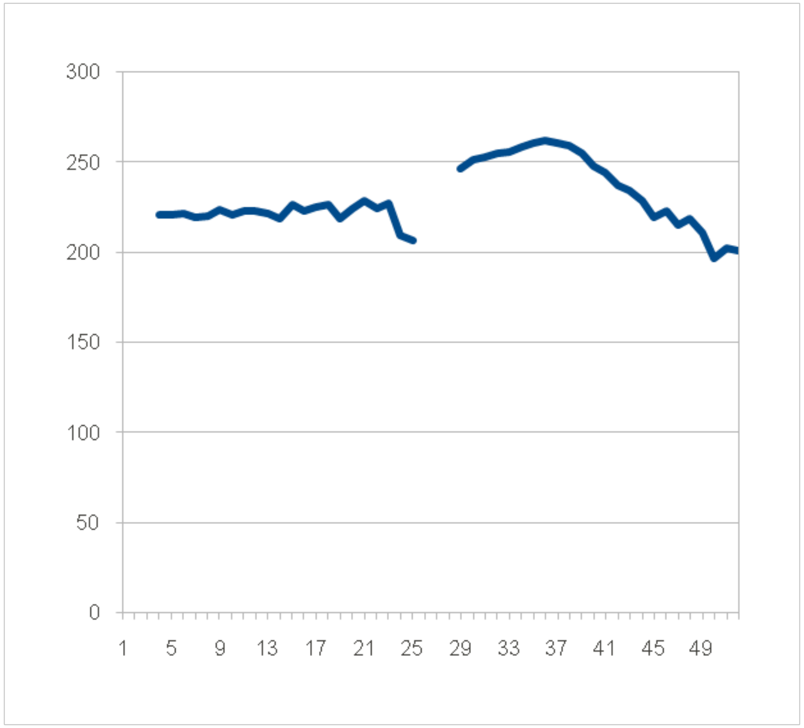
- You can open a .csv file in a spreadsheet programme, and plot the data (see, you don't have to have Praat or wavesurfer to draw speech parameters)



UNIVERSITY OF OXFORD

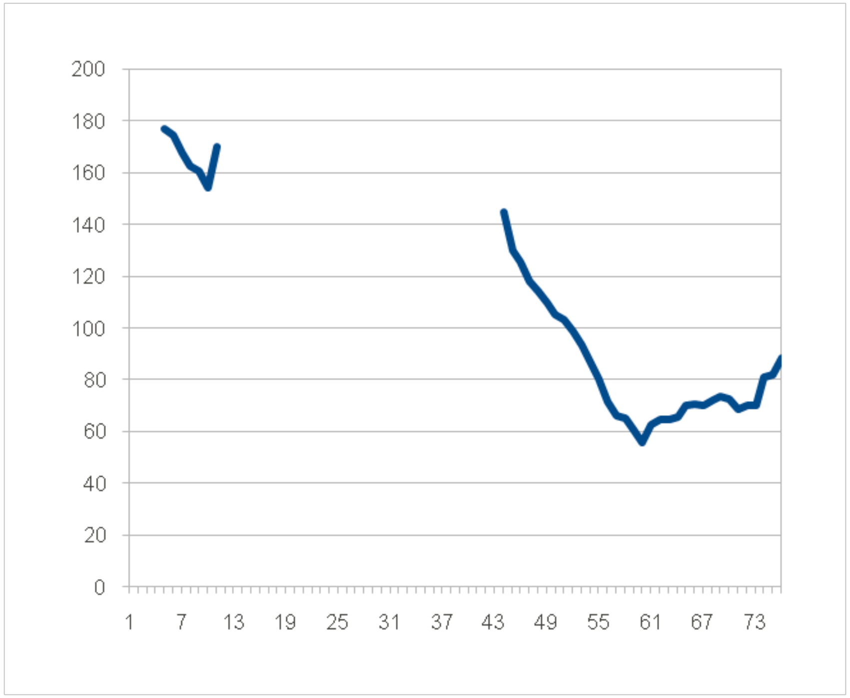
Intonation of FIFTEEN165

- You can open a .csv file in a spreadsheet programme, and plot the data (see, you don't have to have Praat or wavesurfer to draw speech parameters)



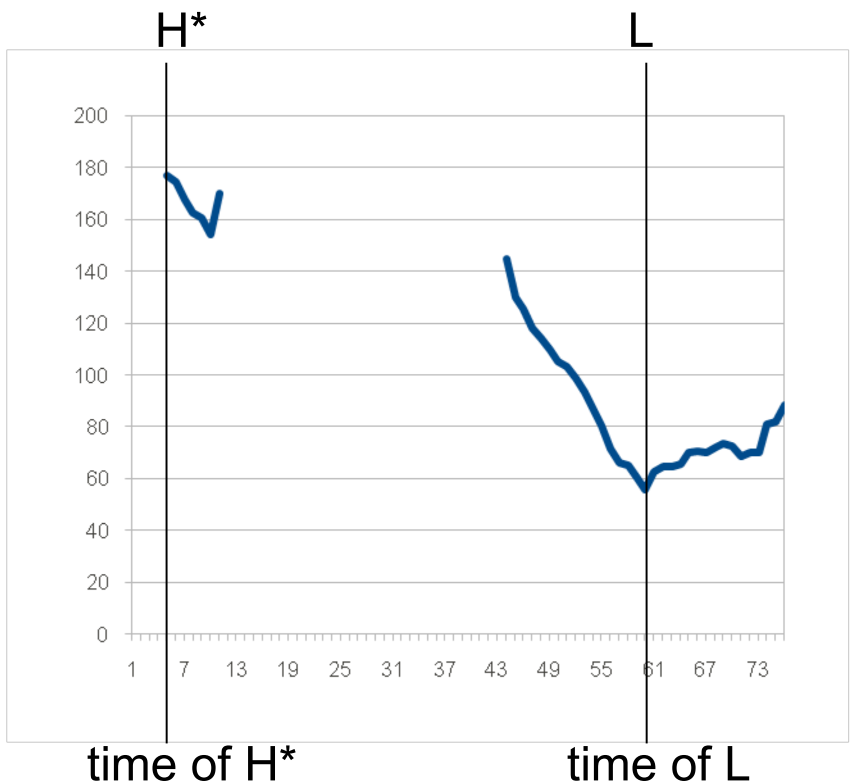
How you *could* analyse intonation ...

- Many approaches to phonetic analysis focus on particular points of interest in the time series, e.g.



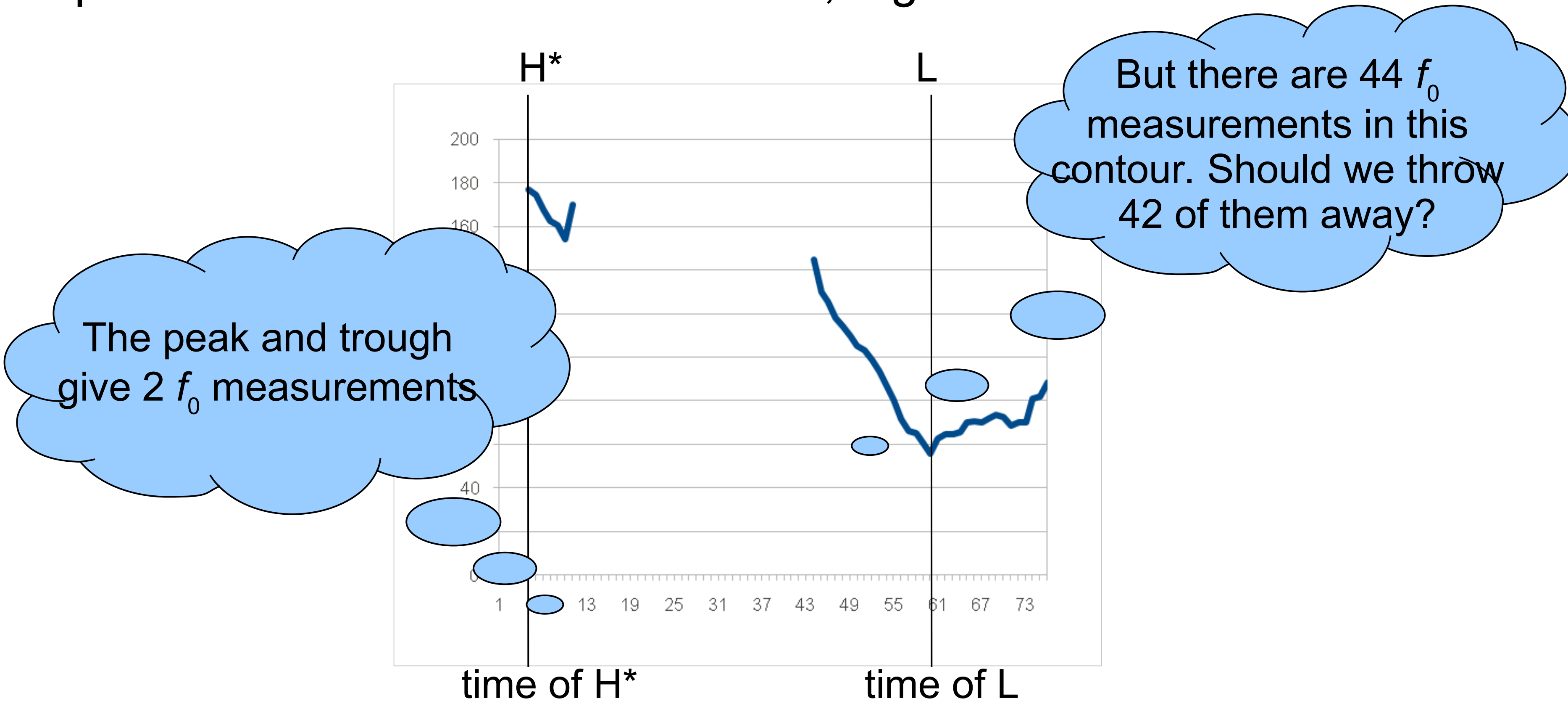
How you *could* analyse intonation ...

- Many approaches to phonetic analysis focus on particular points of interest in the time series, e.g.



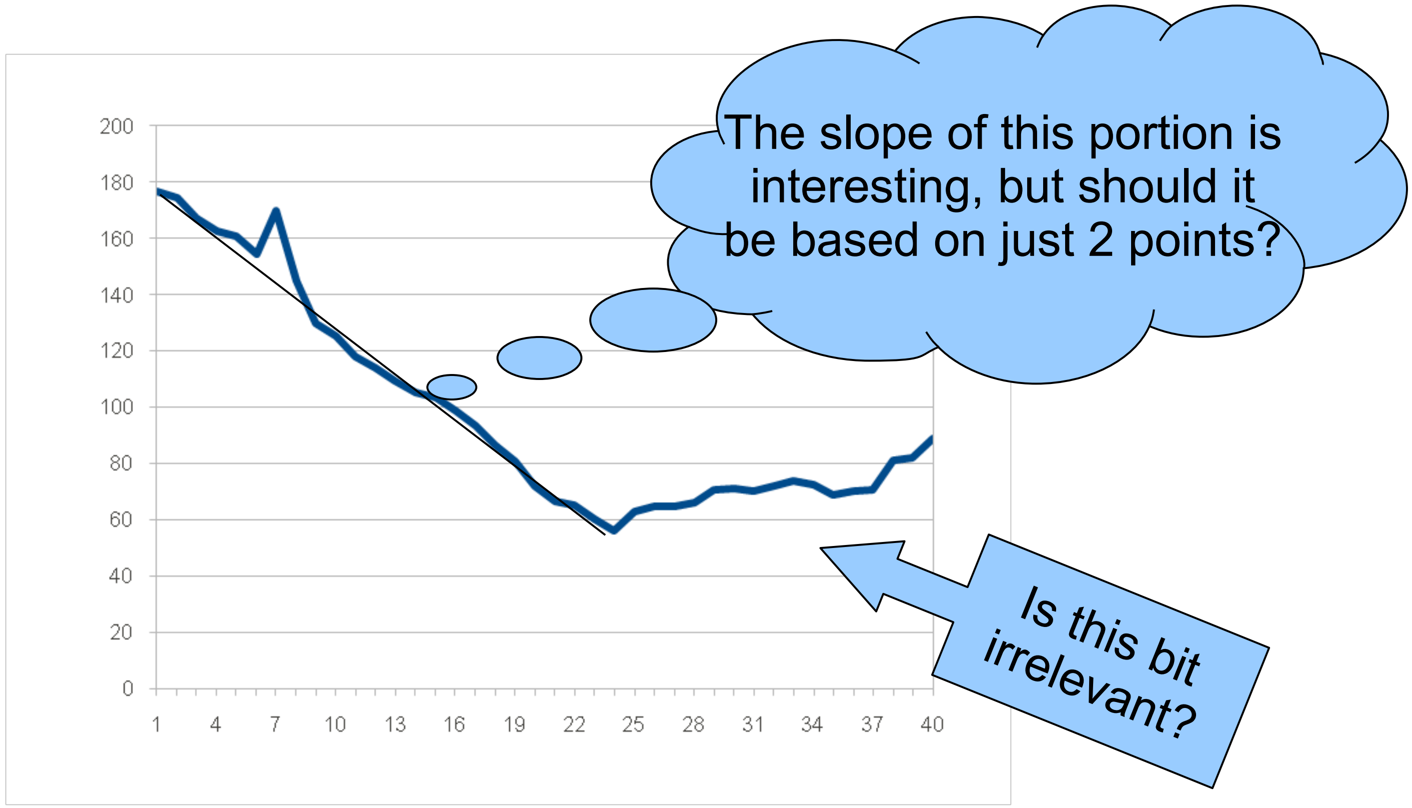
How you *could* analyse intonation ...

- Many approaches to phonetic analysis focus on particular points of interest in the time series, e.g.



Intonation of FIFTEEN3

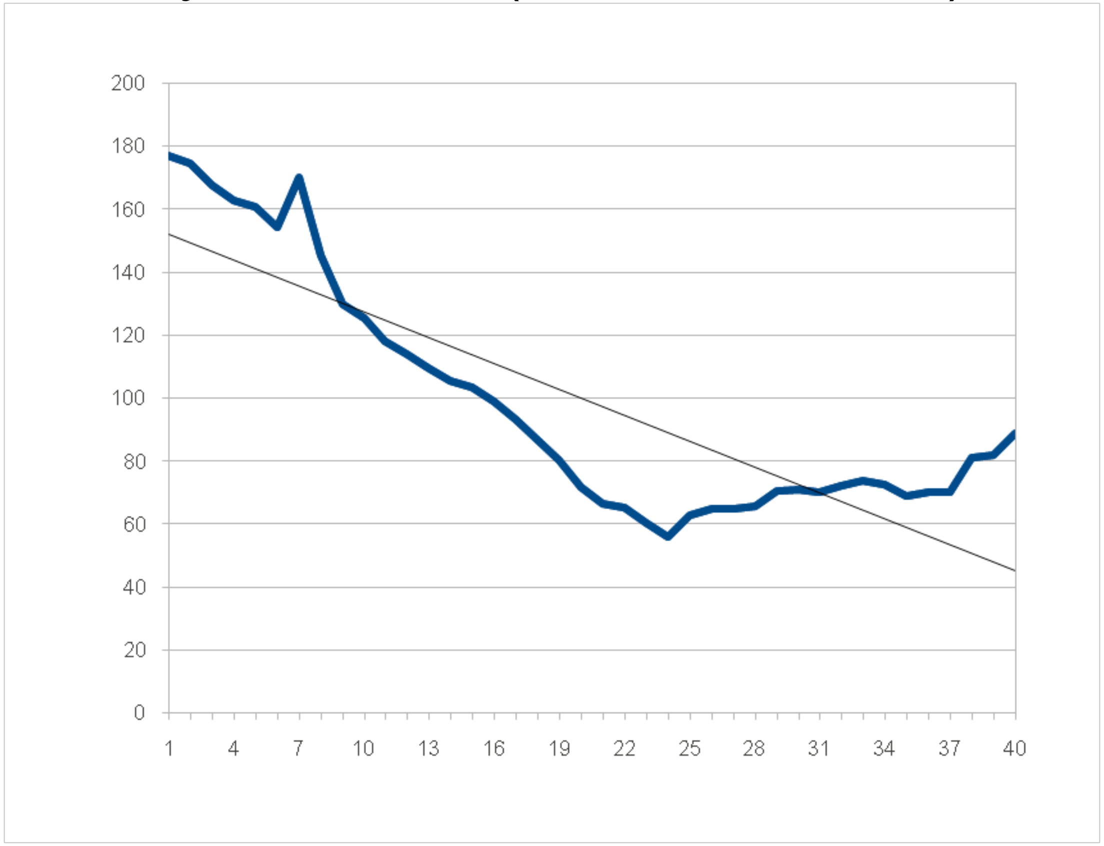
- With the discontinuity (voiceless portion) excised:



UNIVERSITY OF OXFORD

Intonation of FIFTEEN3

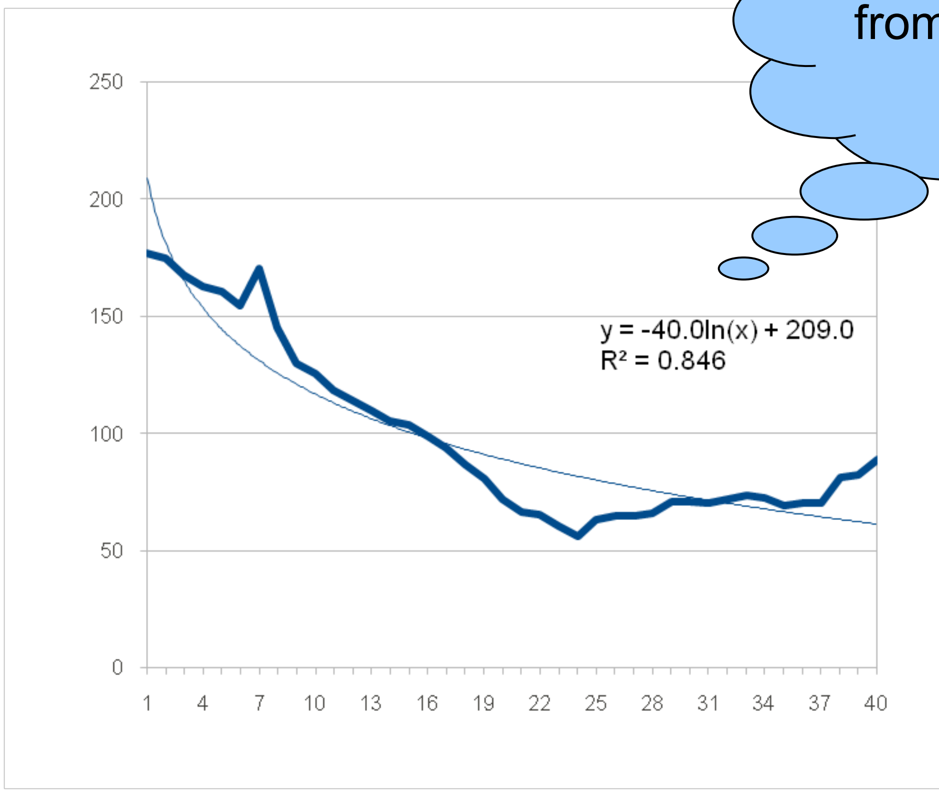
- Cf. a linear regression line; it may not look quite so good, but it's actually a better fit (to the *whole* line)



Intonation of FIFTEEN3

- A logarithmic regression curve fits better

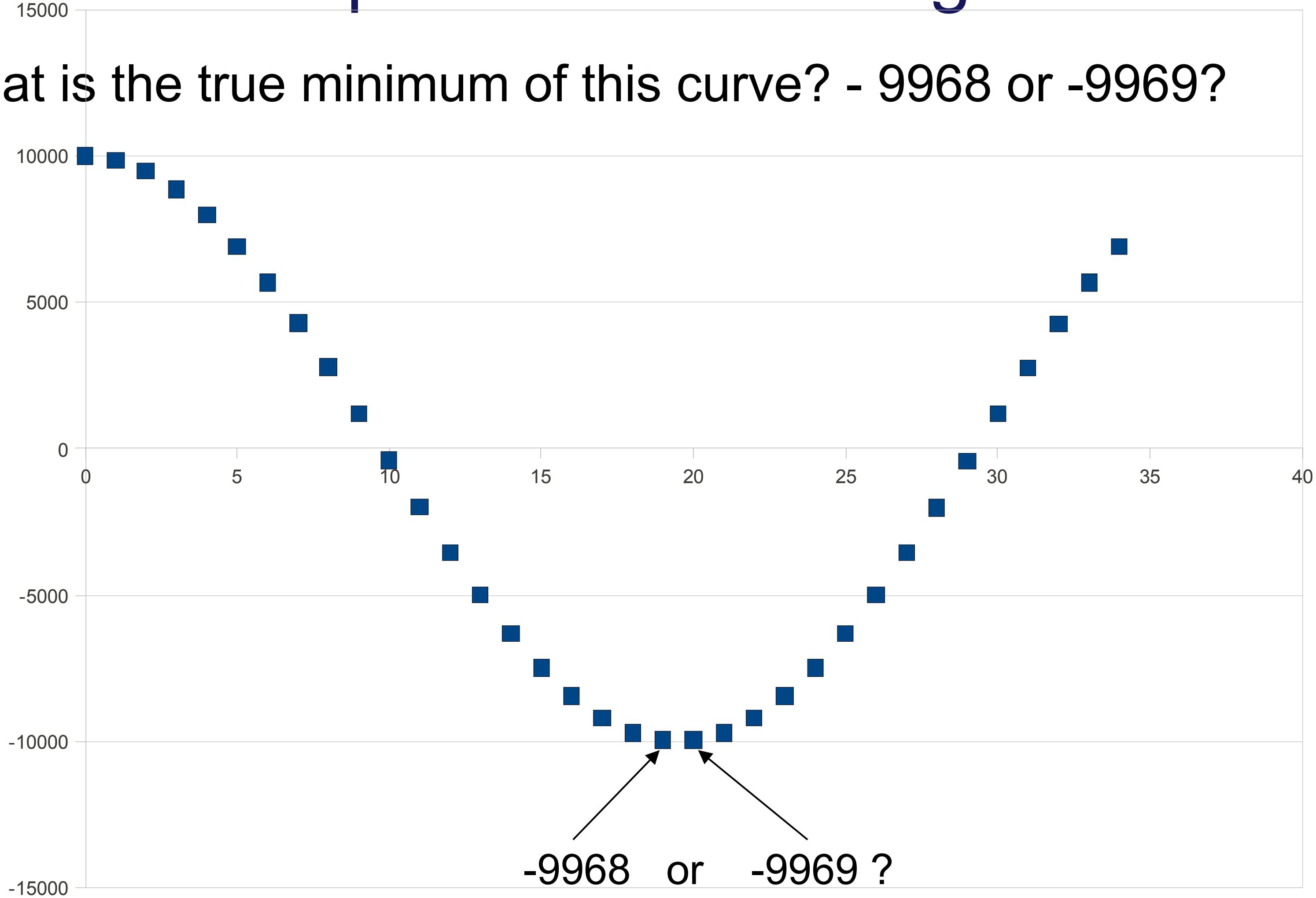
Could we discover anything from the details of this equation?



UNIVERSITY OF OXFORD

Why single-point measurements of sampled data are not great

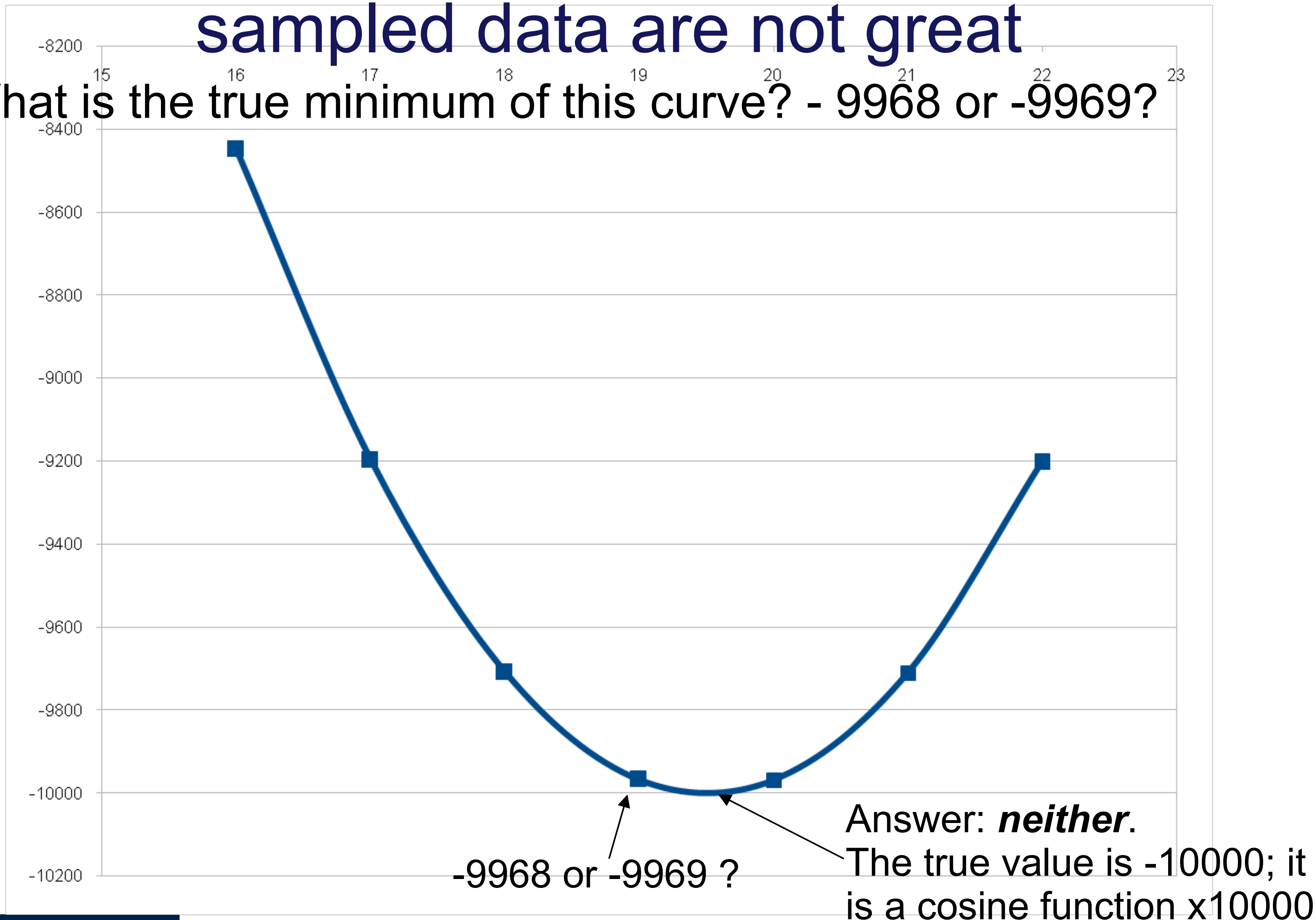
- What is the true minimum of this curve? - 9968 or -9969?



UNIVERSITY OF OXFORD

Why single-point measurements of sampled data are not great

What is the true minimum of this curve? - 9968 or -9969?



UNIVERSITY OF OXFORD

Functional Data Analysis

- Modelling sampled data using (continuous) functions

- General approach:

  – Possibly smooth the data a bit, to iron out irrelevant wiggles
  – Possibly normalize the data
  – Registration: some sort of time alignment of the individual tokens (not always necessary)

Functional Data Analysis

Choose a general kind of (basis) function that looks like your data

- For periodic data: Fourier series
- For nonperiodic data: B-splines
  sometimes: Orthogonal Polynomials (Example 1)
- *others are possible*

- Probability density functions
- E.g., for normally-distributed data: Gaussians (Example 2)

Fit the function to the data
i.e. find the parameters of the function that minimizes the differences between the function and the data

UNIVERSITY OF OXFORD

Orthogonal polynomials in Octave/Matlab

- Put numeric data into Matlab's vector notation:

```
f0 = load('FIFTEEN3.wav.f0.csv');
y = f0(:,1);
y = y(y>0);
```

```
y =
[176.894;
174.434;
167.352;
162.623;
160.734;
...
88.6733];
```

- Normalize it:
- (Better:

```
yn = y/mean(y)-1;
ym = yn/max(abs(yn));
```

- Normalize the time dimension to the interval [-1 1], and turn it into a column vector:

```
x = ((1:length(yn))-length(yn)/2)/(length(yn)/2);
x = x';
```

UNIVERSITY OF OXFORD

Orthogonal polynomials in Octave/Matlab

- Fit the normalized data to a polynomial (e.g. a cubic)

$$y = a_1 x^3 + a_2 x^2 + a_3 x + a_4$$

```
[a,S] = polyfit(x,yn,3);
```

Output values: a = 0.19321 0.63340 -0.70280 -0.19866

- The fitted function is given by

```
fit = getfield(S,'yf');
```

and restored to the original units (e.g. Hz)

```
ysynth = mean(y)*(fit+1);
```

UNIVERSITY OF OXFORD

Orthogonal polynomials in Octave/Matlab

- Fit the normalized data to a polynomial (e.g. a cubic)

$$y = a_1 x^3 + a_2 x^2 + a_3 x + a_4$$

average height

```
[a,S] = polyfit(x,yn,3);
```

Output values: a = 0.19321 0.63340 -0.70280 -0.19866

UNIVERSITY OF OXFORD

Orthogonal polynomials in Octave/Matlab

- Fit the normalized data to a polynomial (e.g. a cubic)

$$y = a_1 x^3 + a_2 x^2 + a_3 x + a_4$$

slope (steepness anddirection)

average height

Output values: a = 0.19321 0.63340 -0.70280 -0.19866

UNIVERSITY OF OXFORD

Orthogonal polynomials in Octave/Matlab

- Fit the normalized data to a polynomial (e.g. a cubic)

$$y = a_1 x^3 + a_2 x^2 + a_3 x + a_4$$

$a_2 x^2$	$a_3 x$	a_4
breadth of curvature	slope (steepness anddirection)	average height

Output values: a = 0.19321 0.63340 -0.70280 -0.19866

Orthogonal polynomials in Octave/Matlab

- Fit the normalized data to a polynomial (e.g. a cubic)

$$y = a_1 x^3 + a_2 x^2 + a_3 x + a_4$$

$a_1 x^3$	$a_2 x^2$	$a_3 x$	a_4
amount of ~ -shaped wiggle	breadth of curvature	slope (steepness and direction)	average height

Output values: a = 0.19321 0.63340 -0.70280 -0.19866

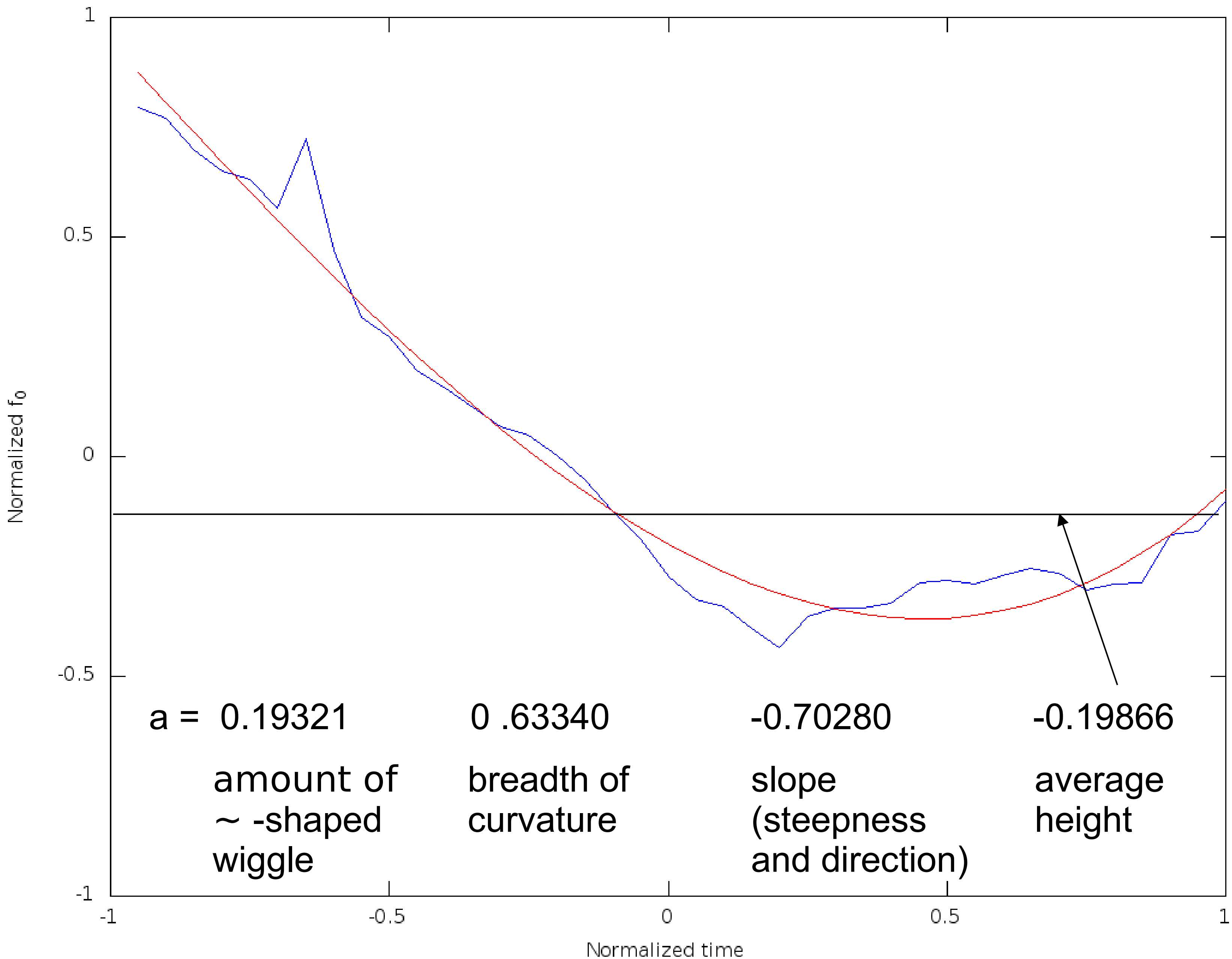
•The fitted function is given by

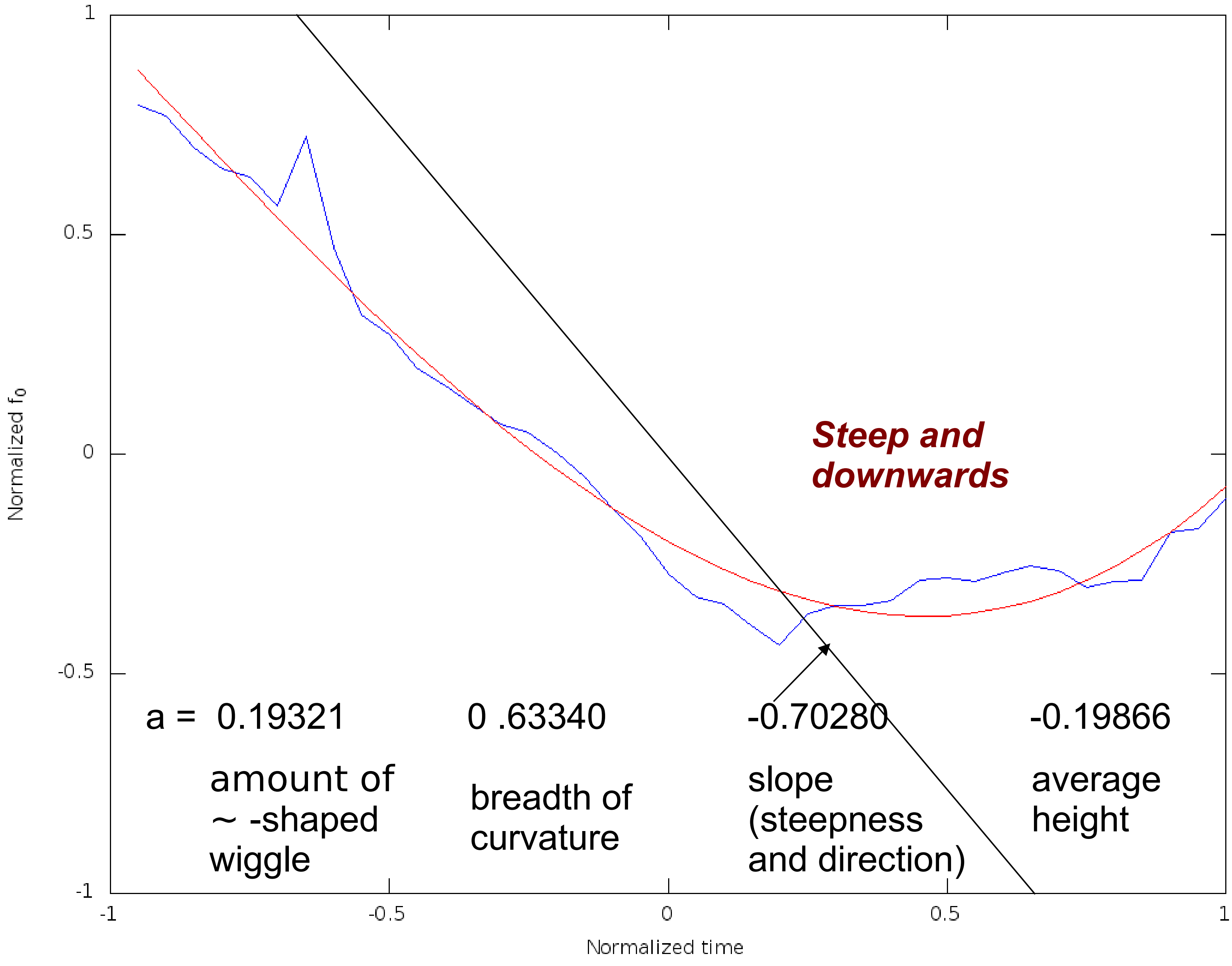
```
fit = getfield(S,'yf');
```

and can be restored to the original units (e.g. Hz) by

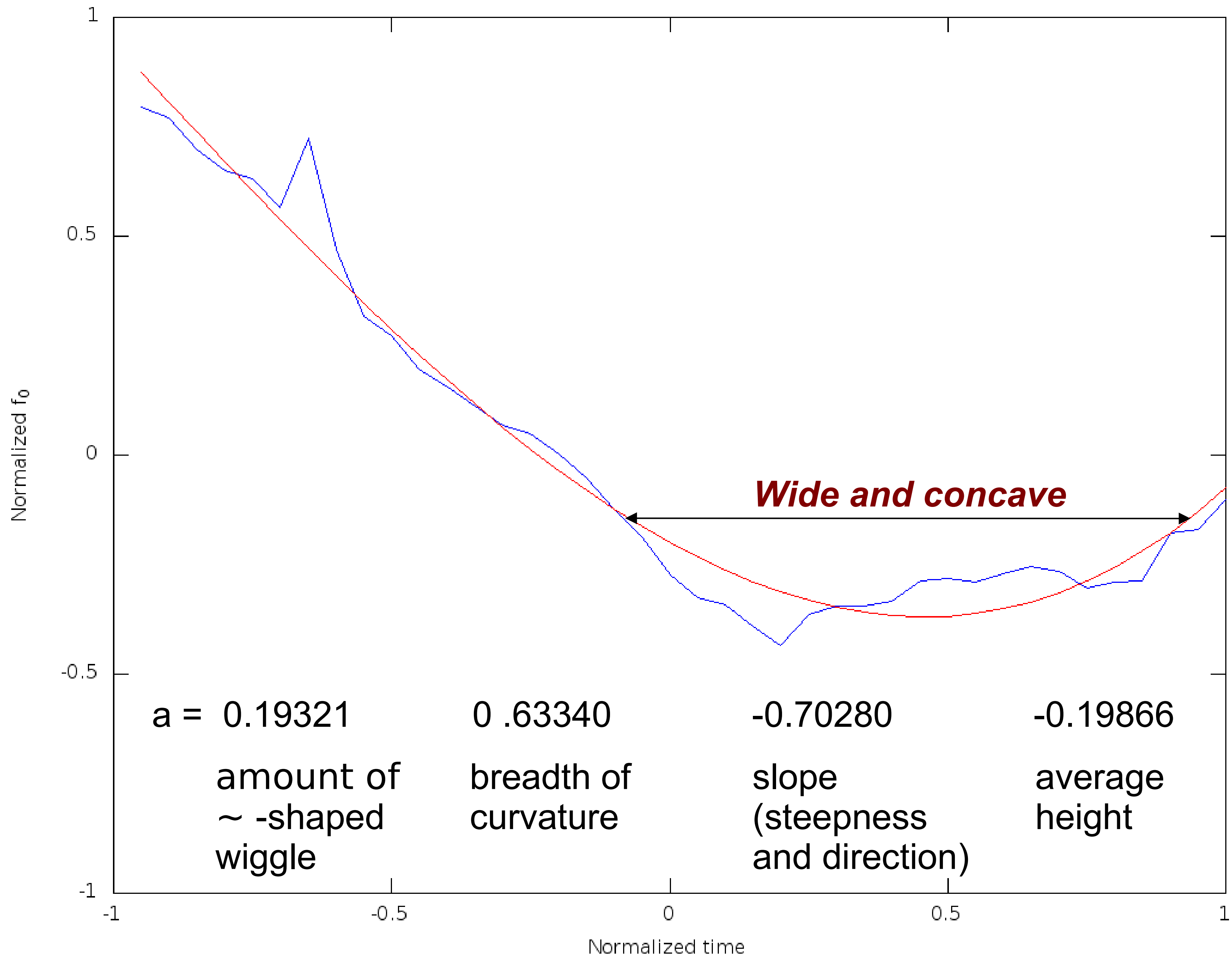
```
ysynth = mean(y)*(fit+1);
```

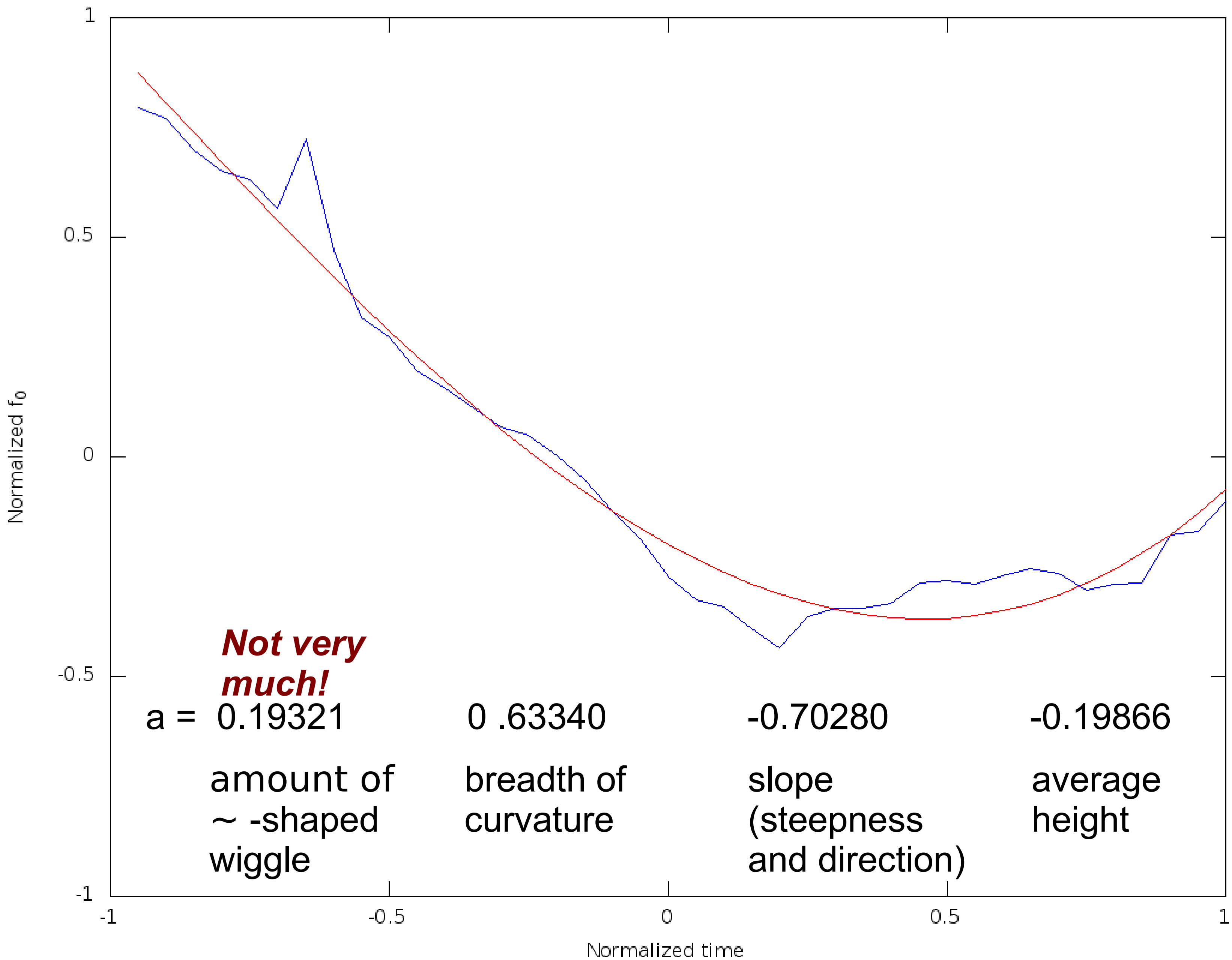
UNIVERSITY OF OXFORD

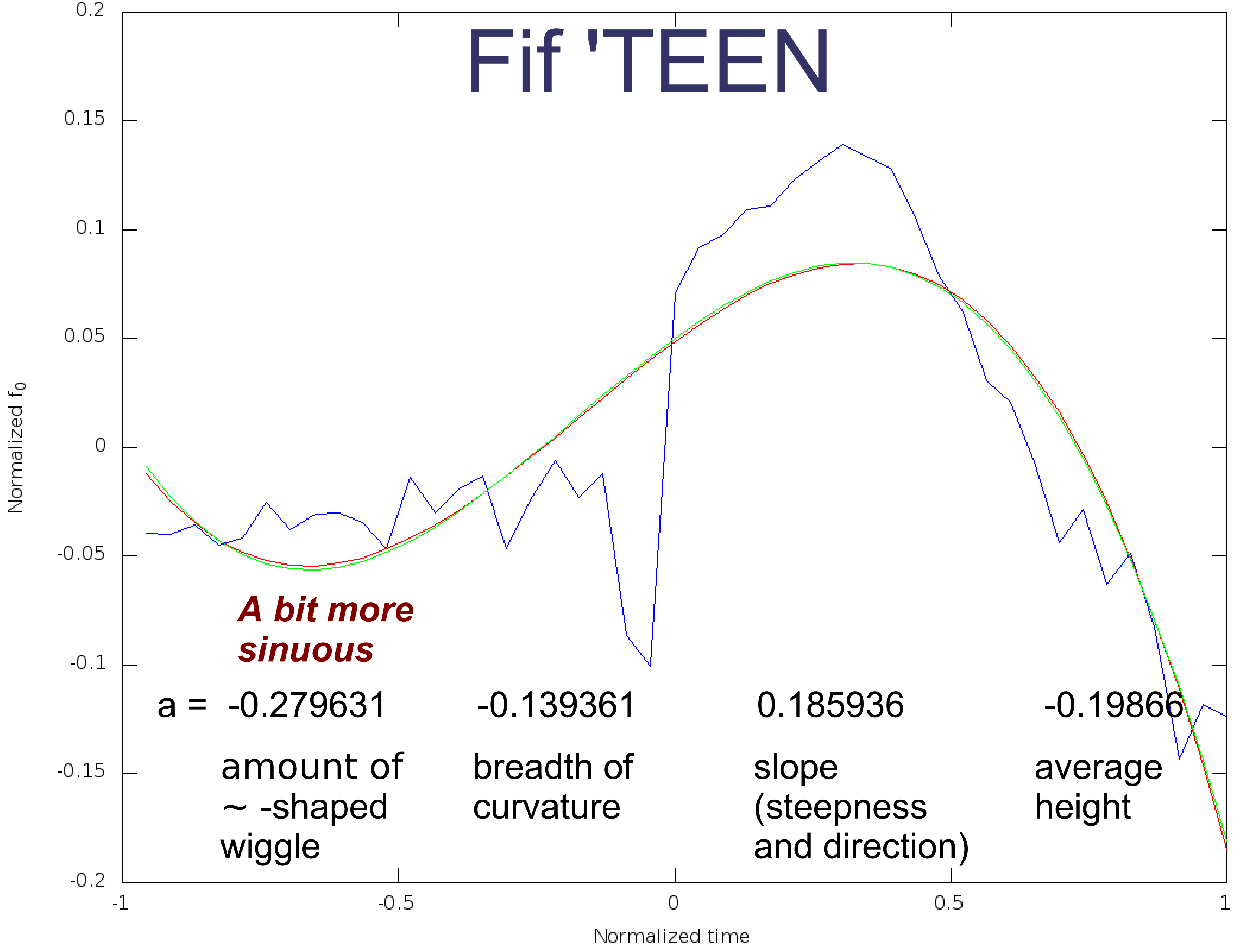

1
0.5
0
-0.5
-1
Normalized f0
-1
-0.5
0
0.5
1
Normalized time
a = 0.19321
0 .63340
-0.70280
-0.19866
amount of ~ -shaped wiggle
breadth of curvature
slope (steepness and direction)
average height

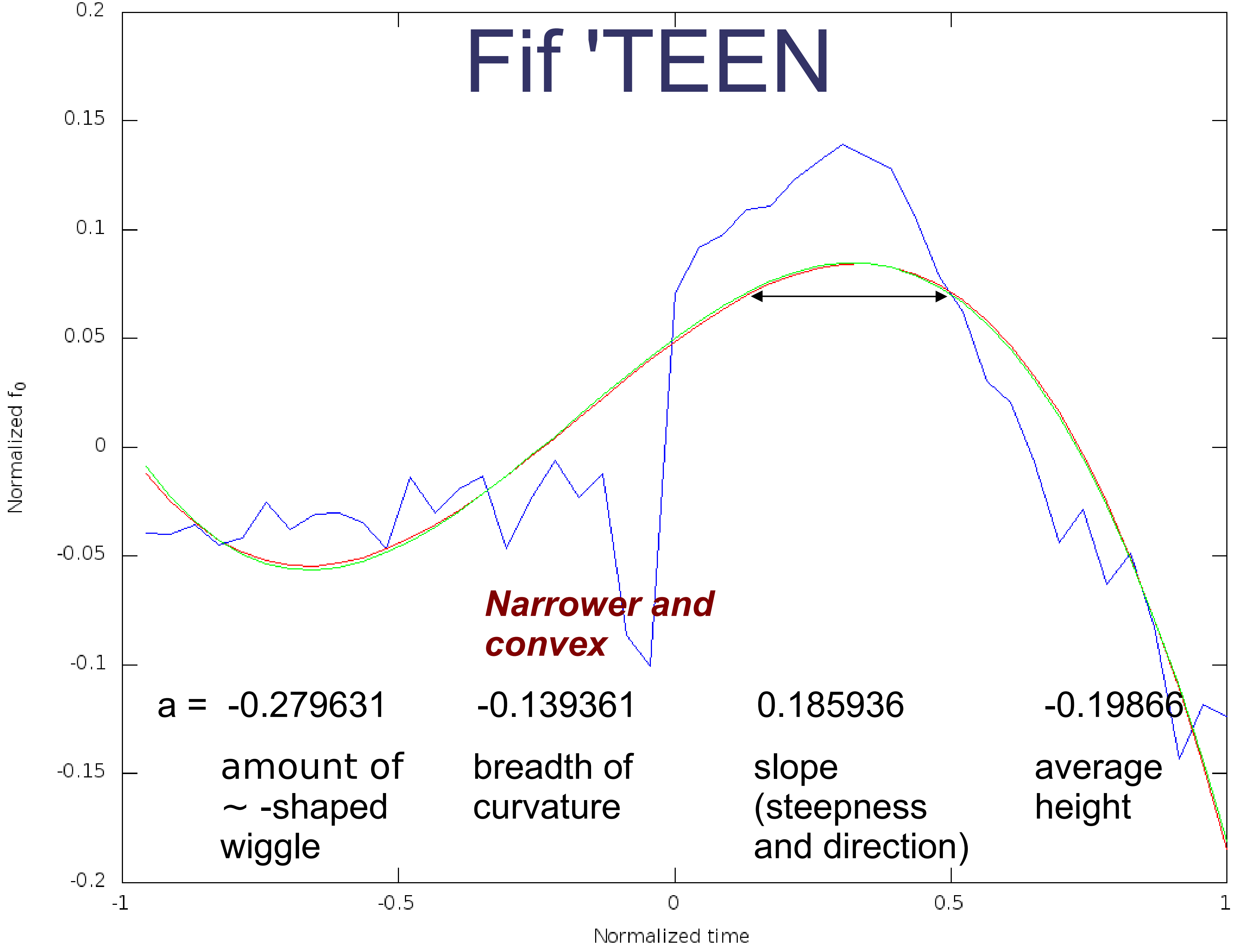


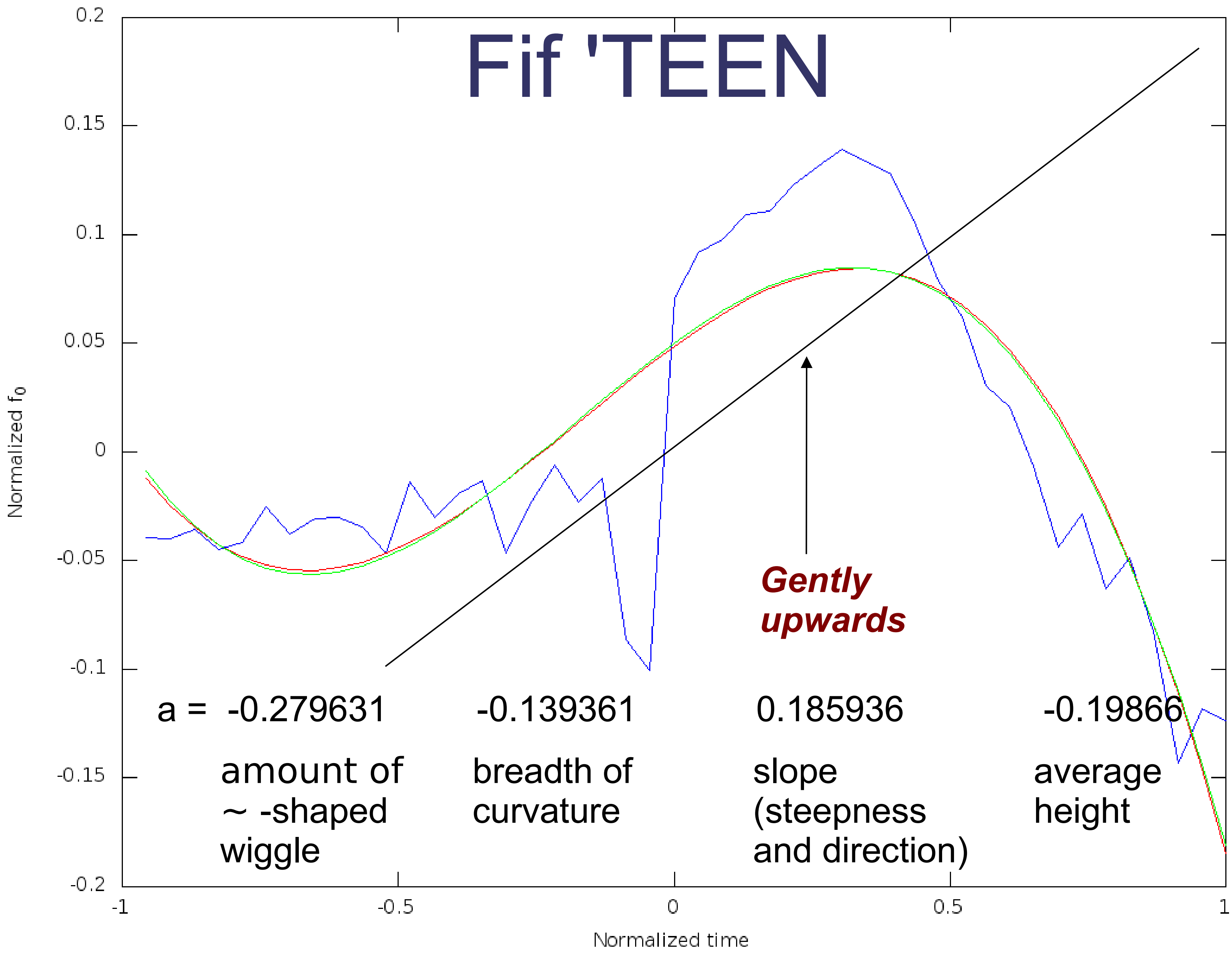
Normalized f_0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
Normalized time
Steep and downwards
a = 0.19321
0 .63340
-0.70280
-0.19866
amount of ~ -shaped wiggle
breadth of curvature
slope (steepness and direction)
average height


Wide and concave
a = 0.19321
0 .63340
-0.70280
-0.19866
amount of ~ -shaped wiggle
breadth of curvature
slope (steepness and direction)
average height
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
Normalized f_0
Normalized time

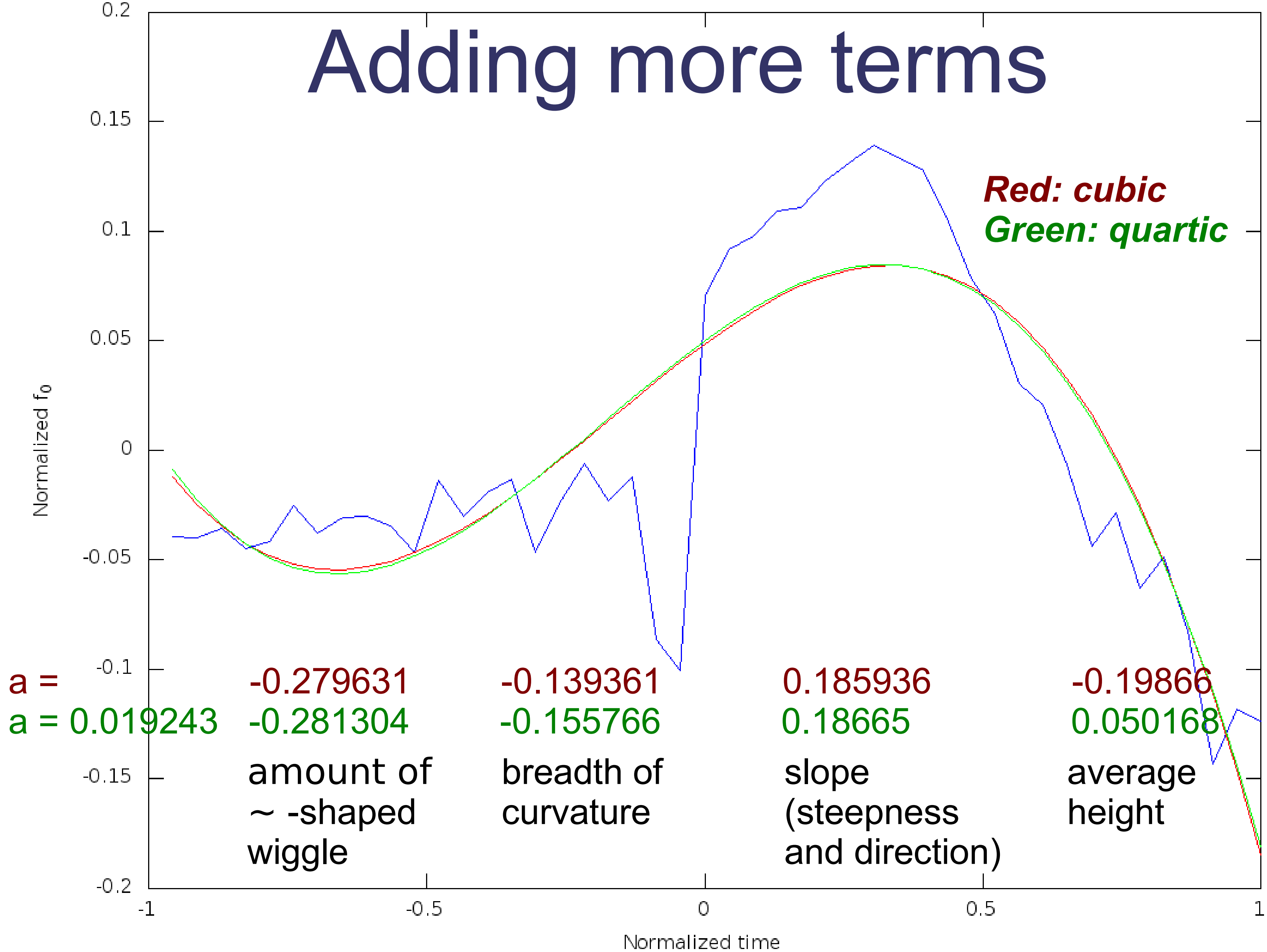

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
Normalized f_0
Normalized time
Not very much!
a = 0.19321
0 .63340
-0.70280
-0.19866
amount of ~ -shaped wiggle
breadth of curvature
slope (steepness and direction)
average height


Fif 'TEEN
A bit more sinuous
a = -0.279631
-0.139361
0.185936
-0.19866
amount of ~ -shaped wiggle
breadth of curvature
slope (steepness and direction)
average height
Normalized f_0
Normalized time
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-1
-0.5
0
0.5
1


Fif 'TEEN
Narrower and convex
a = -0.279631
-0.139361
0.185936
-0.19866
amount of ~ -shaped wiggle
breadth of curvature
slope (steepness and direction)
average height
Normalized f0
Normalized time
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-1
-0.5
0
0.5
1


Fif 'TEEN
Gently upwards
a = -0.279631
amount of ~ -shaped wiggle
-0.139361
breadth of curvature
0.185936
slope (steepness and direction)
-0.19866
average height
Normalized f_0
Normalized time
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-1
-0.5
0
0.5
1

Adding more terms



Orthogonalisation

- Translate polynomial coeffients into *orthogonal* (Legendre) polynomial coeffs:

```
c = [0.4*a(1) 2*a(2)/3 a(3)+6*a(1)/5 a(4)]
```

%% a = 0.19321 0.63340 -0.70280 -0.19866
%% c = 0.077284 0.422269 -0.470948 -0.198659

Loop over all the “good” files

```
for i = 2:172
   eval(['fid = fopen(''FIFTEEN',int2str(i),'.wav.f0.csv'');']);
   if (fid ~= -1) %% checks file FIFTEEN i ... exists
      eval(['f0 = load(''FIFTEEN',int2str(i),'.wav.f0.csv'');']);
      y = f0(:,1);
      y = y(y>0);
      yn = y/mean(y)-1;
      x = ((1:length(yn))-length(yn)/2)/(length(yn)/2);
      x = x';
      [a,S] = polyfit(x,yn,3);
      c = [0.4*a(1) 2*a(2)/3 a(3)+6*a(1)/5 a(4)];
      C(i,:) = [i c];
   end
end
save('coeffs.csv','C');
```

UNIVERSITY OF OXFORD

Now do your statistics

- Rather than applying statistical tests to the raw data, examine the means, variances etc of the coefficients of the functions you're using to model the data