Processing Very Low Velocities with PDV

WCI
WEAPONS
AND COMPLEX
INTEGRATION

Ted Strand

LLNL

5th Annual PDV Workshop
Ohio State University
Columbus, OH
September 8-9, 2010
LLNL-PRES-450499

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Processing Very Low Velocities--Outline

WCI
WEAPONS
AND COMPLEX
INTEGRATION

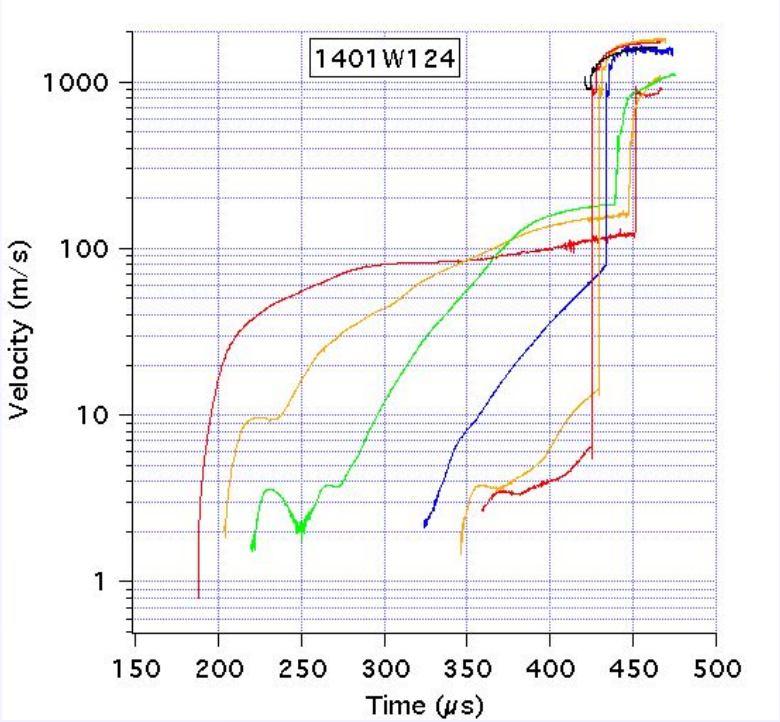
Introduction
Motivation
Method (peakfind)
Examples
Show process on sample data

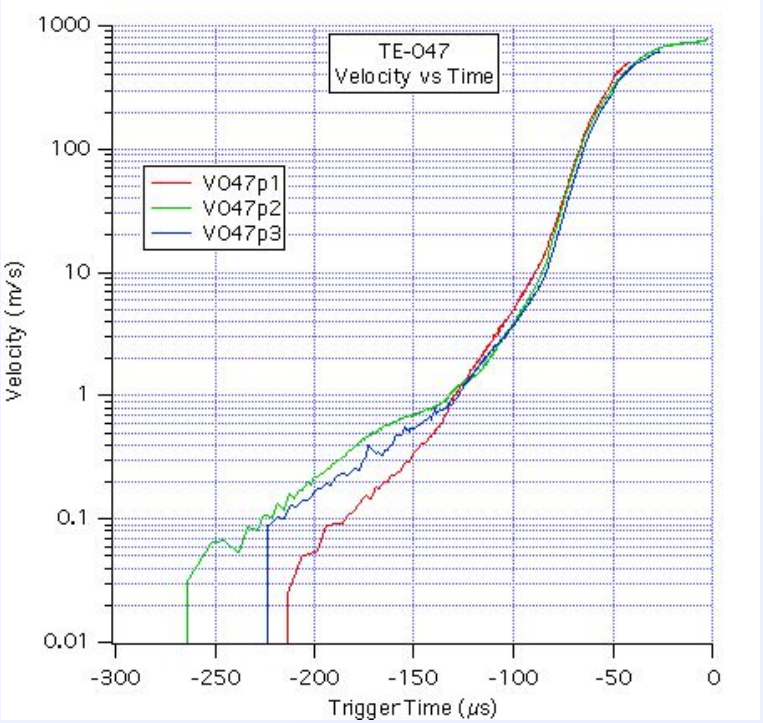
Notes:

I started this analysis using Excel spreadsheets. When I converted to Igor, I used the same method.

I will assume a digitizer recording rate of 20 GS/s throughout.

Sometimes we encounter very low velocities

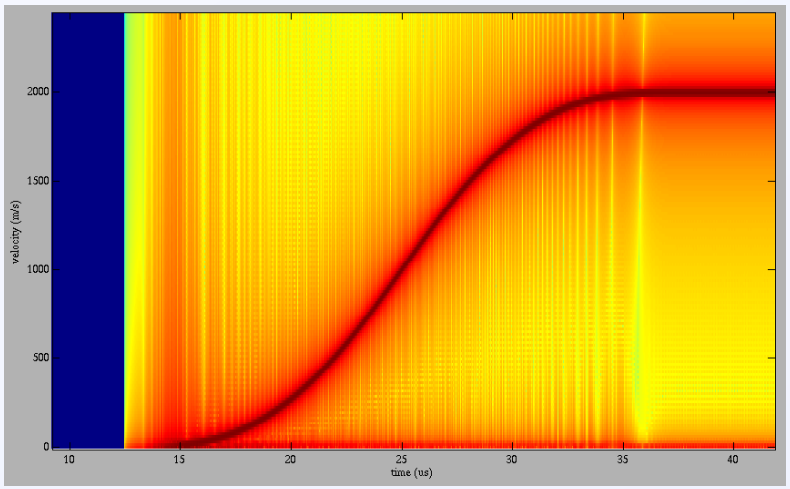




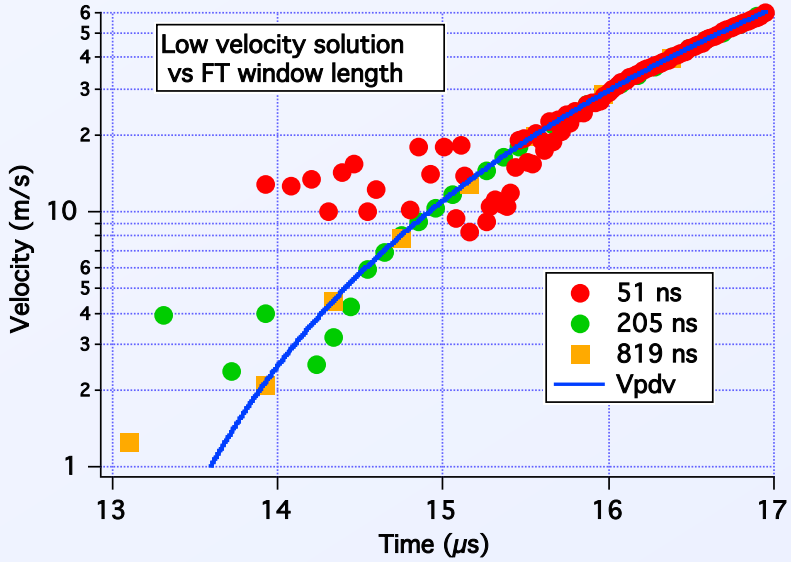
Longer FT windows can process to lower velocities, but we do not obtain very many data points

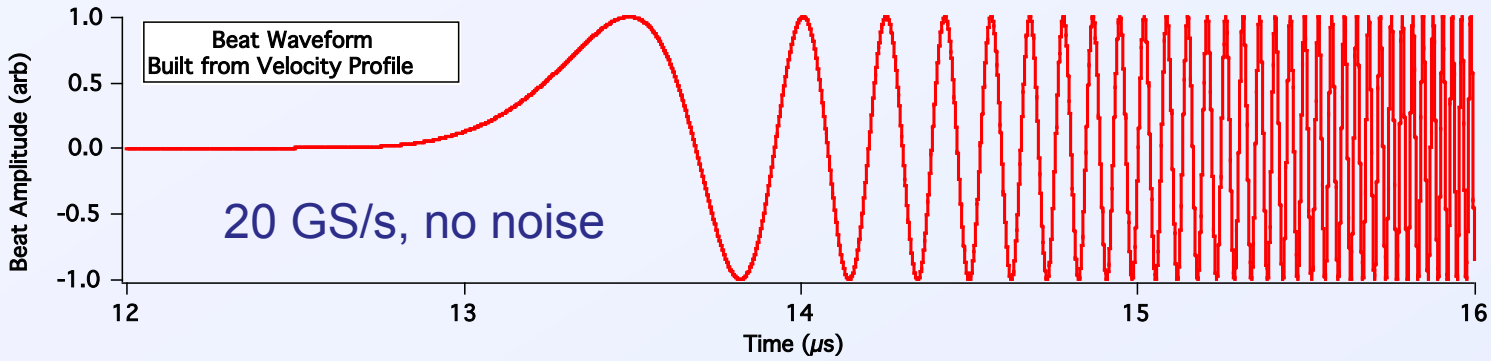
WCI WEAPONS AND COMPLEX INTEGRATION

Spectrogram with 51 ns FT window



Low velocity performance

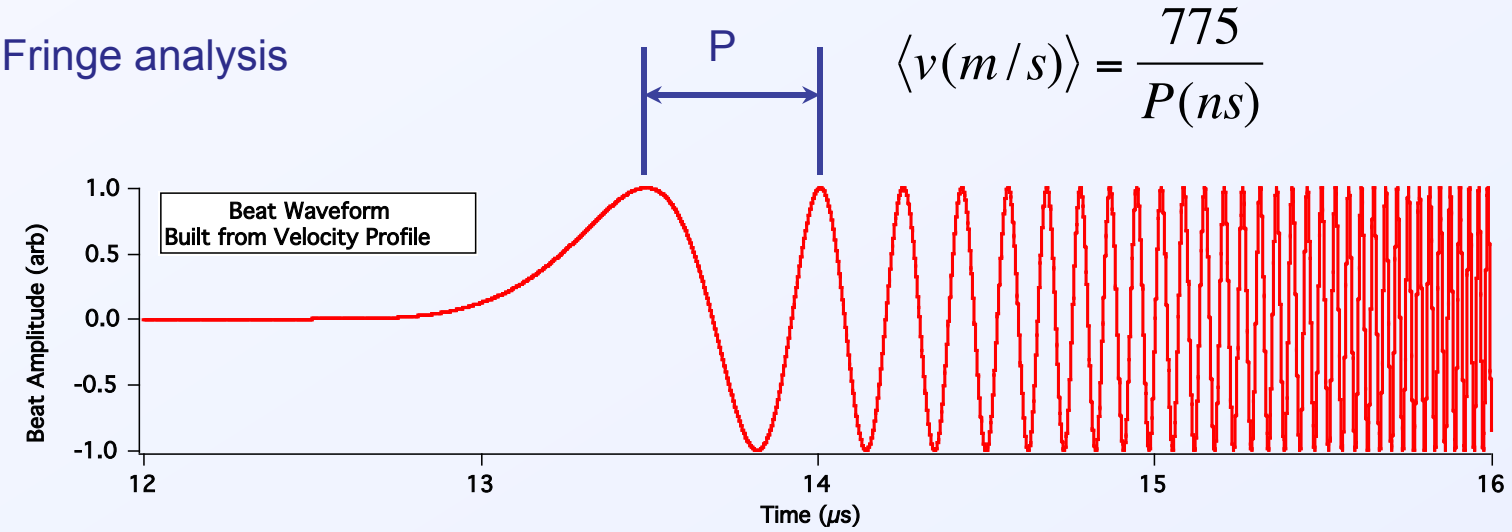




Measure beat periods to get velocity averaged over one beat cycle

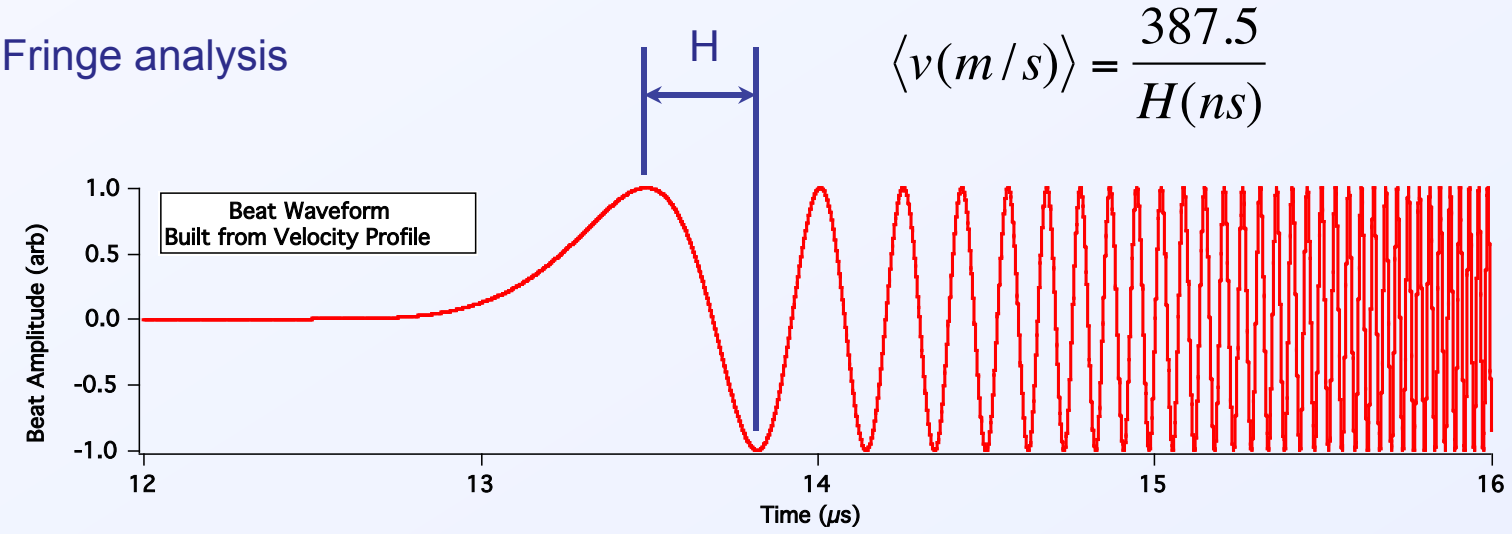
Try working in the time domain, rather than frequency domain.

Fringe analysis



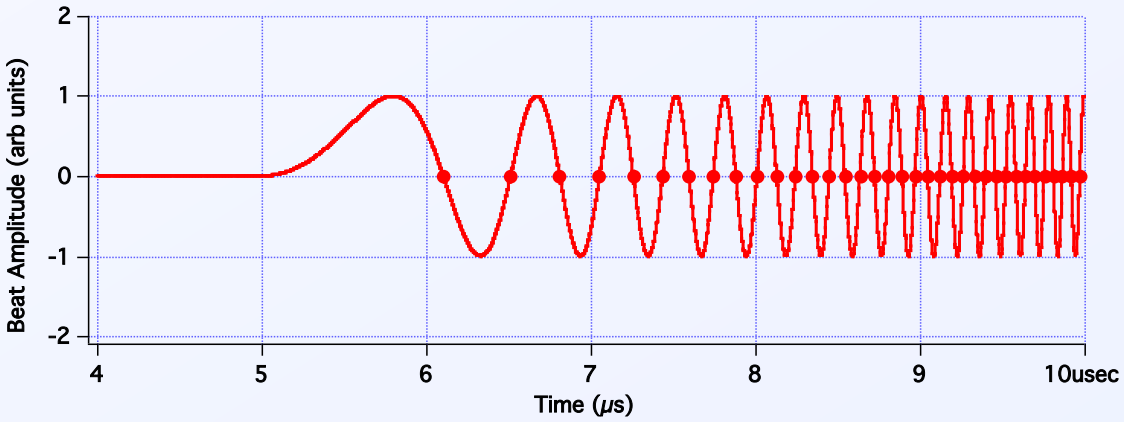
Or better, find each peak to get velocity averaged over one-half beat cycle

Fringe analysis



Get twice as many data points in VvsT file this way.

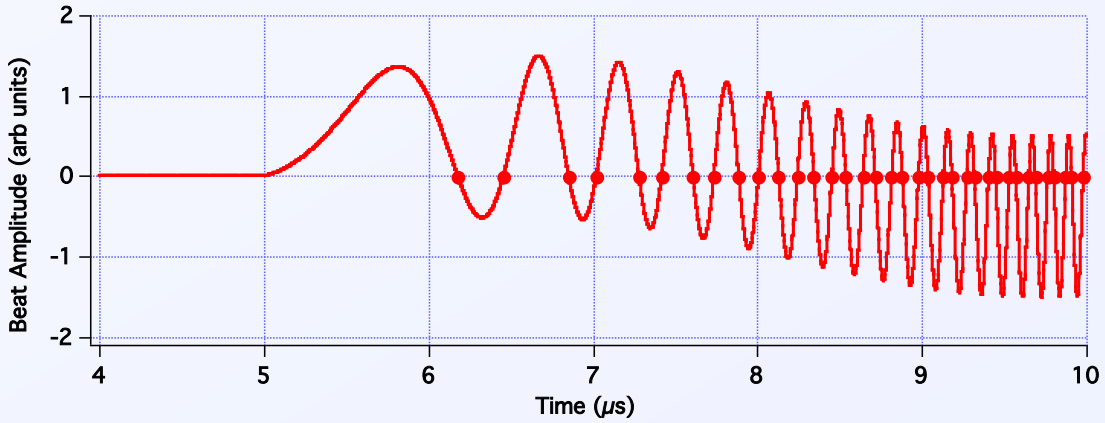
Consider a zero-crossing algorithm?

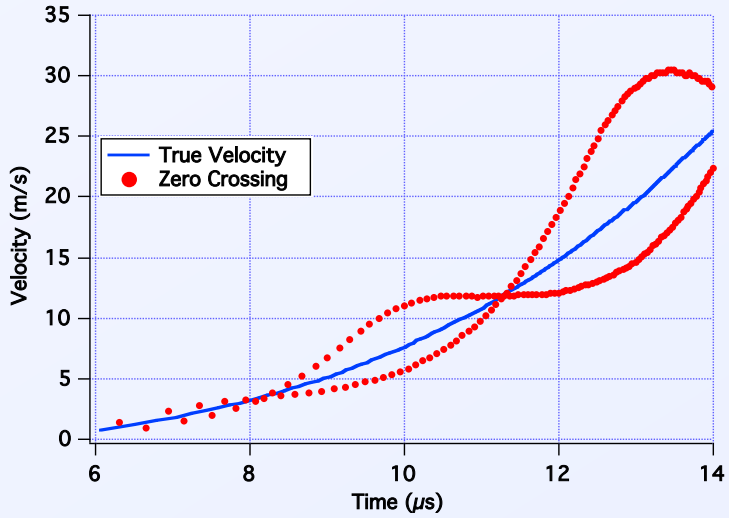


Notes:

- This is easy to do in Excel
- Miss half of the first cycle with this method.

Zero crossing will not work with baseline shifts





Every other point is too high or too low.

Baselines shifts are usually caused by intensity changes

$$I_b = I_0 + I_d + \sqrt{I_0 I_d}\,\sin(\omega_b t + \phi)$$

CW — not always CW — Beat amplitude

I_b = intensity of beat signal

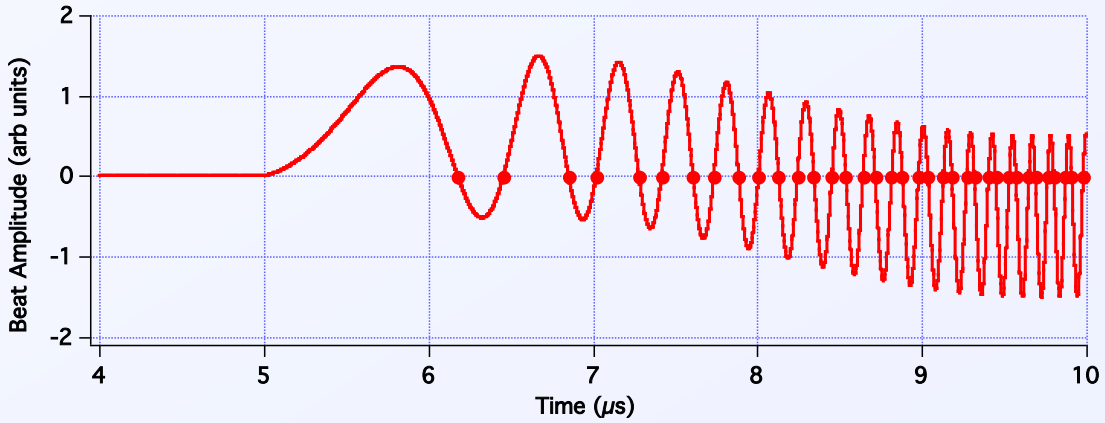
I_0 = intensity of reference signal = constant

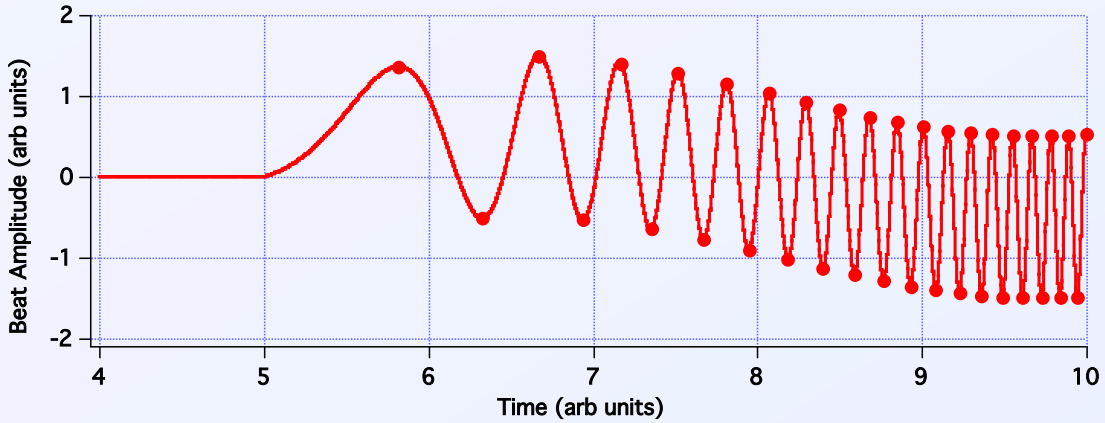
I_d = intensity of doppler-shifted signal from surface

Depends upon:

- Surface material
- Surface roughness
- Surface tilt
- Probe efficiency
- Spot diameter

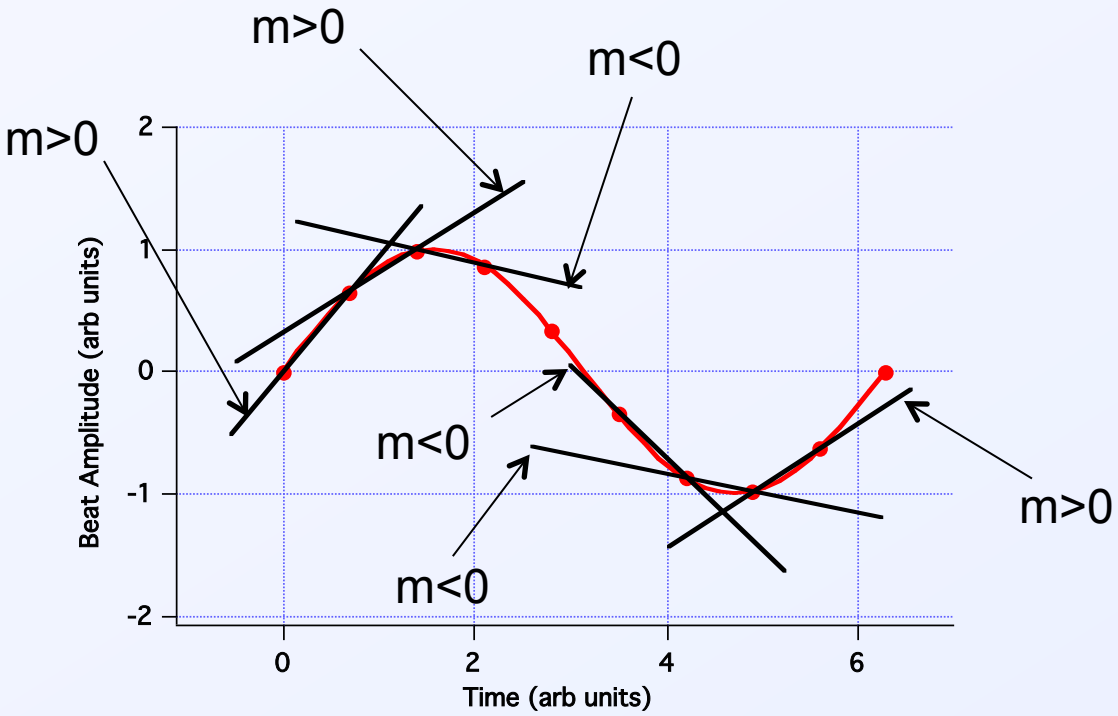
I really want to find the individual peaks





Note: miss less of the 1st cycle.

The point-to-point slope changes sign at the peaks

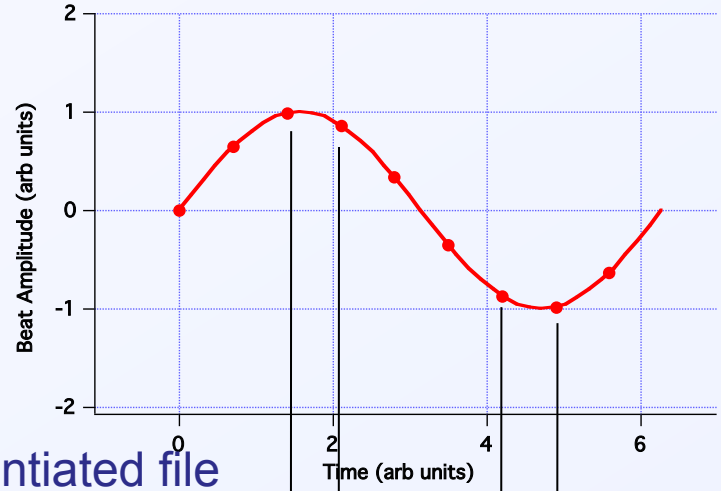


But, working with slopes is just differentiation…

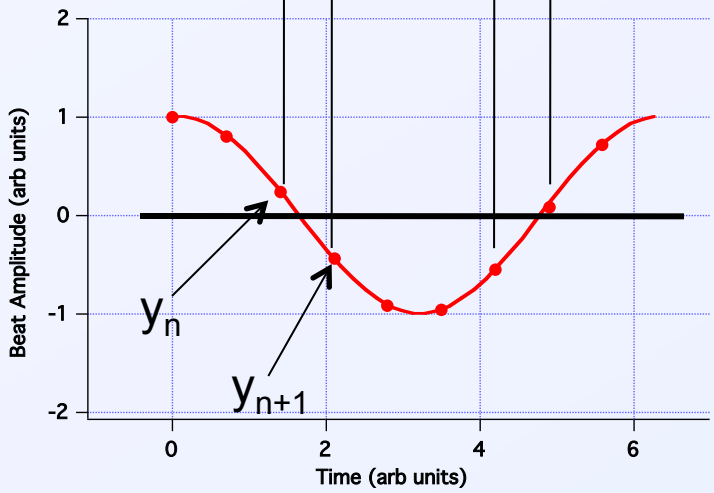
Differentiate and then use zero crossing method

WCI WEAPONS AND COMPLEX INTEGRATION

Beat waveform file



Differentiated file



To find zero crossings, I have the code scan through the diff file and look for:

$$y_n * y_{n+1} <= 0$$

Chadd May of LLNL suggested looking at the sign of adjacent points:

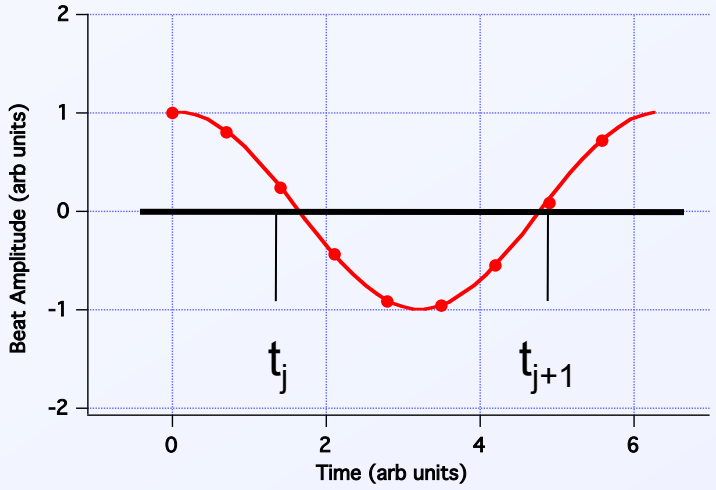
$$SIGN(y_n) \neq SIGN(y_{n+1})$$

Note:
In Excel, the SIGN function yields
SIGN(+y) = 1
SIGN(0) = 0
SIGN(-y) = -1
In Igor, the SIGN function yields
SIGN(+y) = 1
SIGN(0) = -1
SIGN(-y) = -1

Pick the point closer to the x-axis and write the time to a file

Maximum error of zero crossing method

Differentiated file



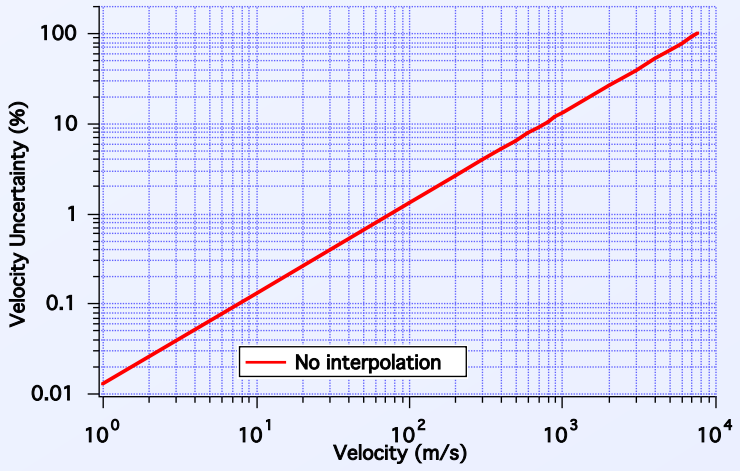
Save all the t_j values and calculate

$$\langle v(m/s)\rangle\Big|_{j}^{j+1} = \frac{387.5}{t_{j+1} - t_j}$$

What is the maximum error of this method?

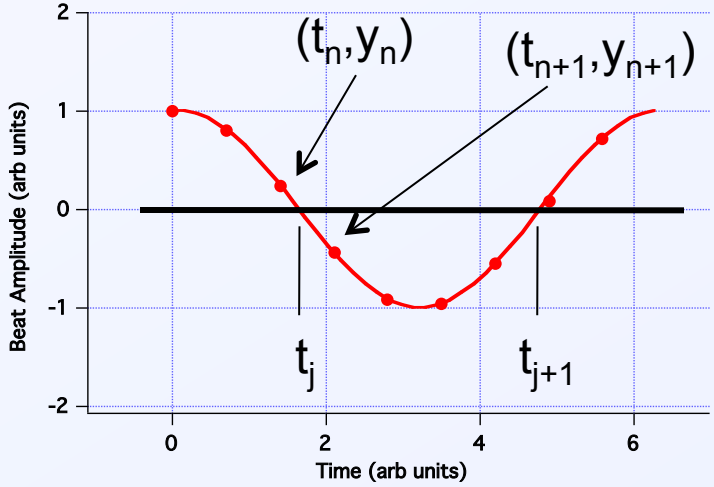
Each point could be wrong ½ digitizer sample time, so total uncertainty ΔH = 1 dig pt

$$\Delta v = \frac{v^2}{7750}\Delta H$$



Interpolate between points for better estimate

Differentiated file

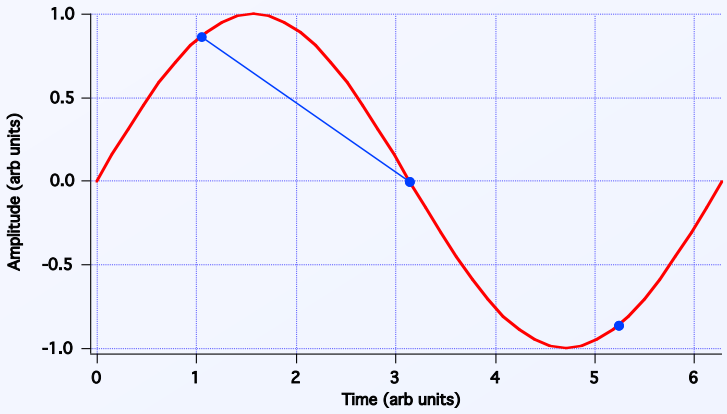


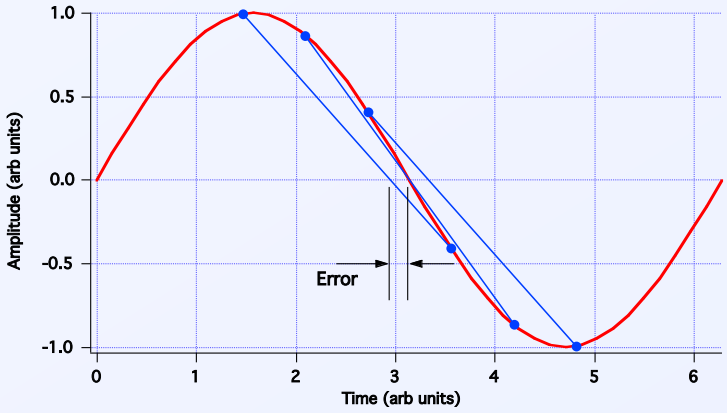
Interpolation yields:

$$t_j = t_n + \left(\frac{y_n}{y_{n+1} - y_n}\right)(t_{n+1} - t_n)$$

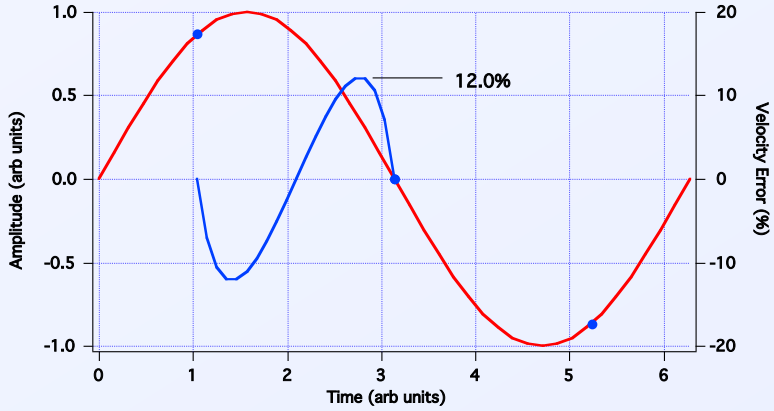
Now what is the max error vs velocity?

Use a linear interpolation to reduce max error





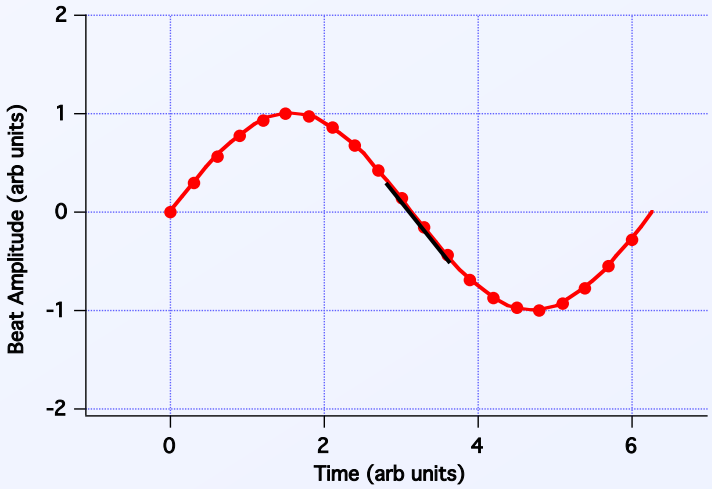
This example is for
3 points per cycle,
which is 5167 m/s.



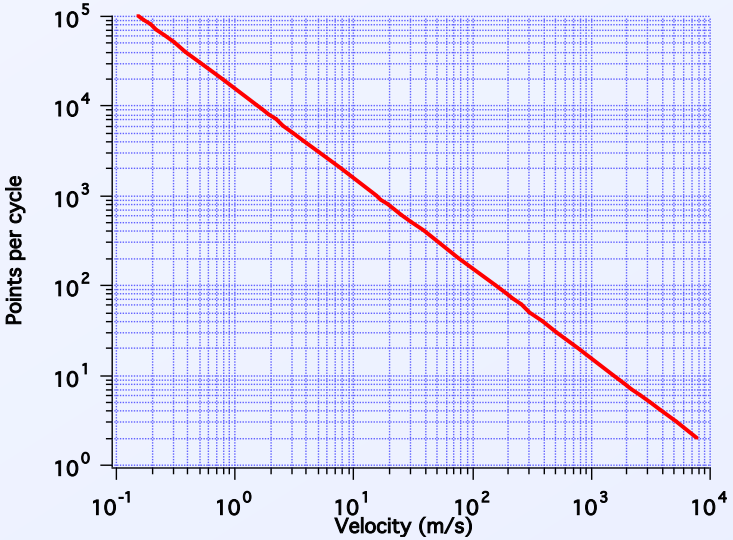
Max error is approx 12%

A linear fit becomes a very good approximation with increasing points per cycle

This shows 20 points per cycle, which is 775 m/s.

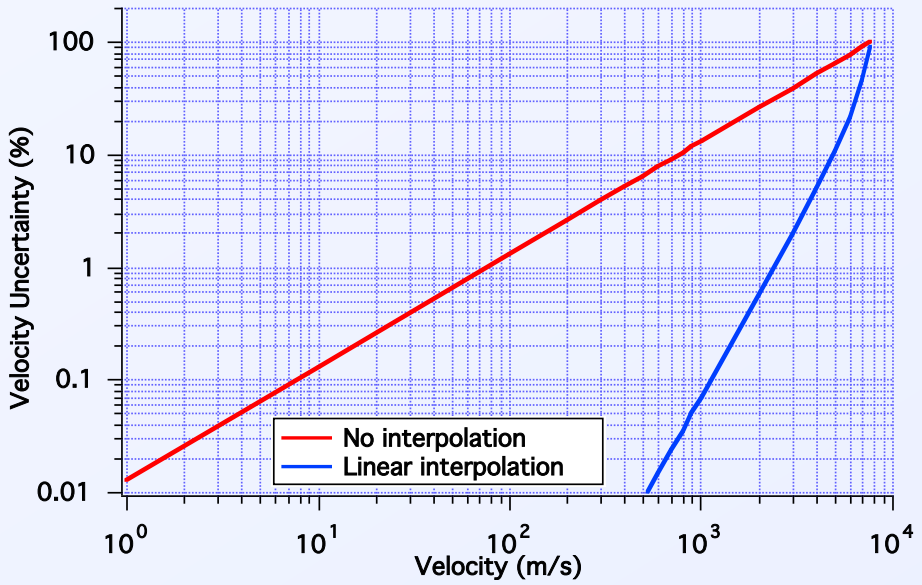


This is for 20 GS/s digitizer recording.



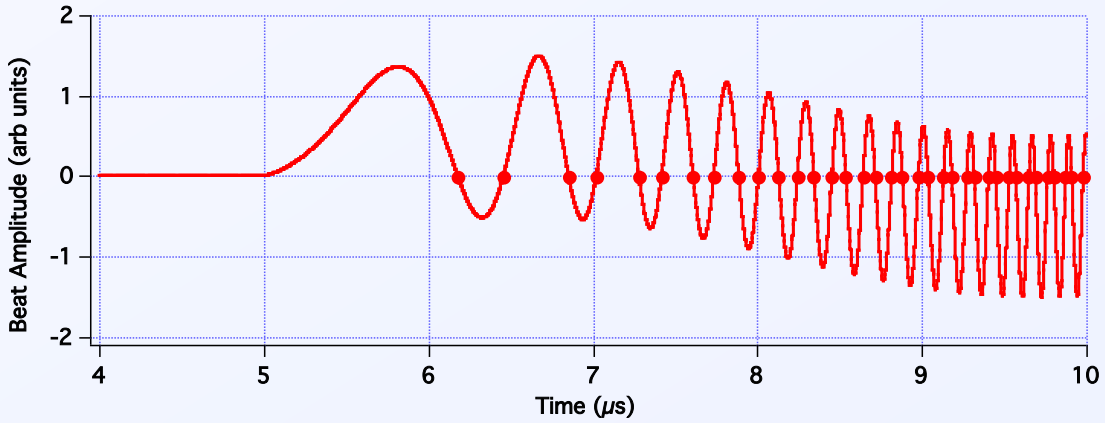
Interpolate between points for better estimate

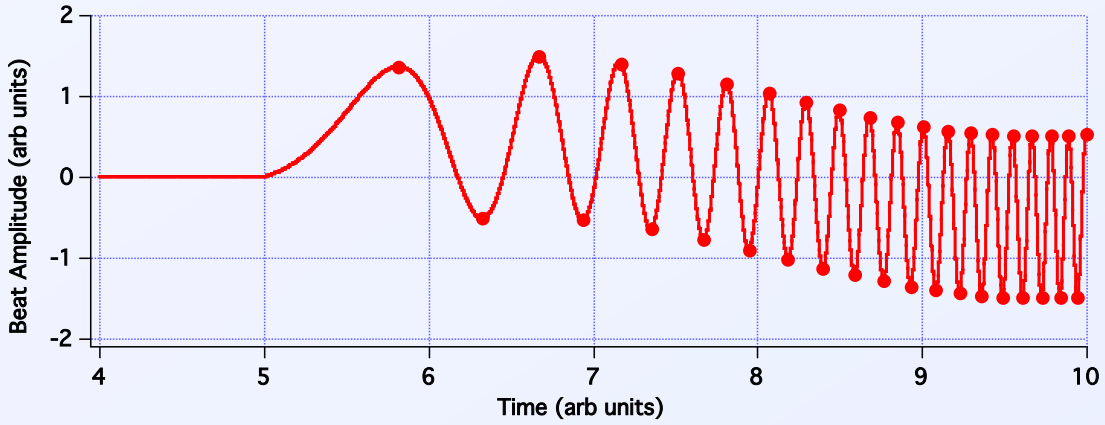
Calculate max errors
for linear interpolation
at different points per cycle



Again, this is with no noise.

What about the baseline shifts with zero crossing?





Baseline fluctuations are usually much slower than the beat frequency we are looking for

$$I_b = I_0 + I_d + \sqrt{I_0 I_d}\,\sin(\omega_b t + \phi)$$

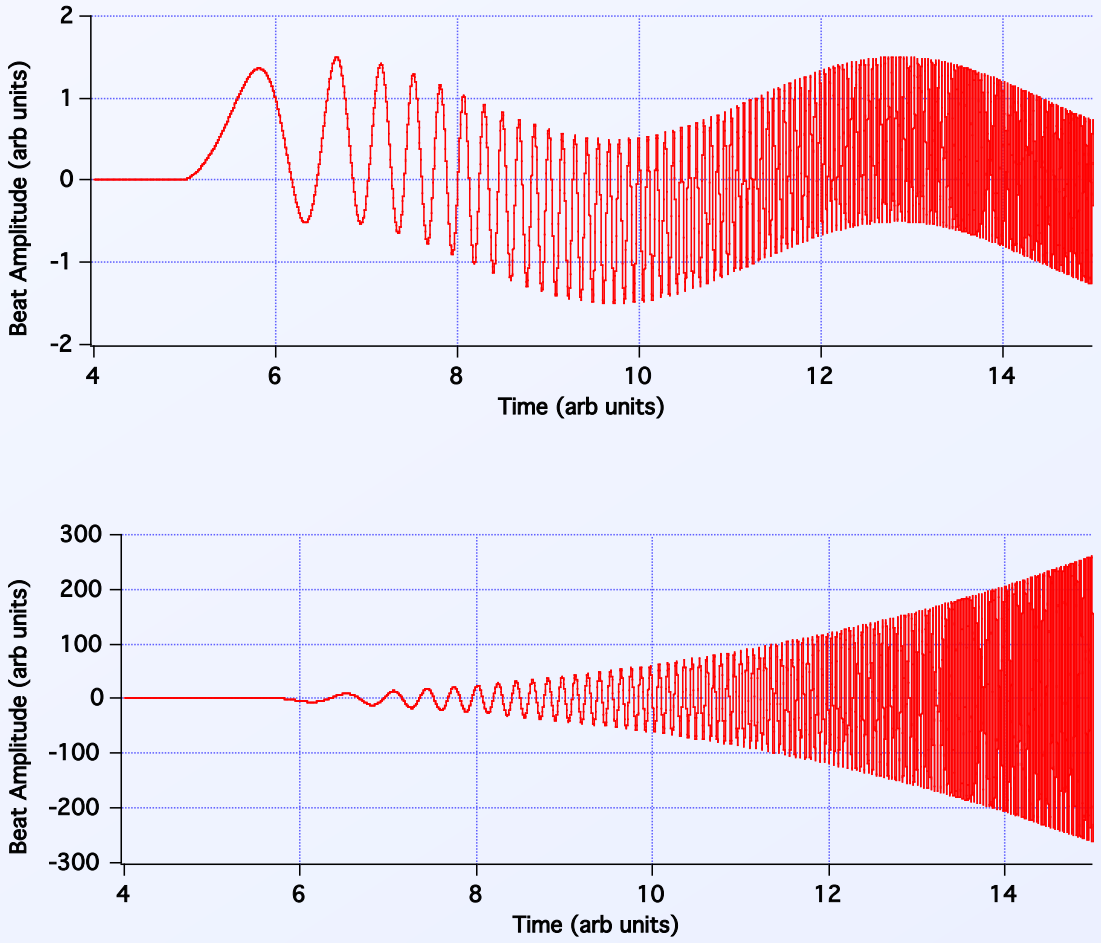
CW

nearly CW

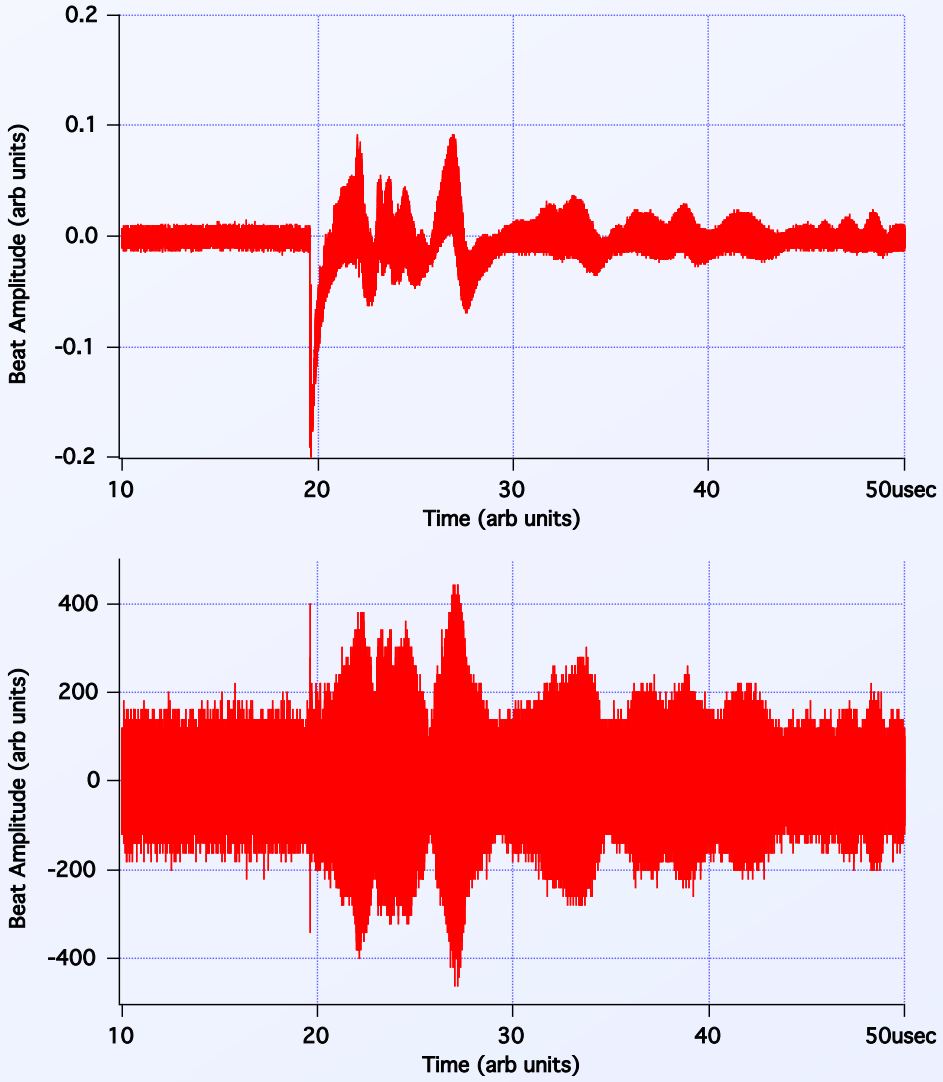
Beat amplitude

$$\frac{dI_b}{dt} = \omega_b \sqrt{I_0 I_d}\,\cos(\omega_b t + \phi)$$

Differentiation does a good job of eliminating baseline shifts



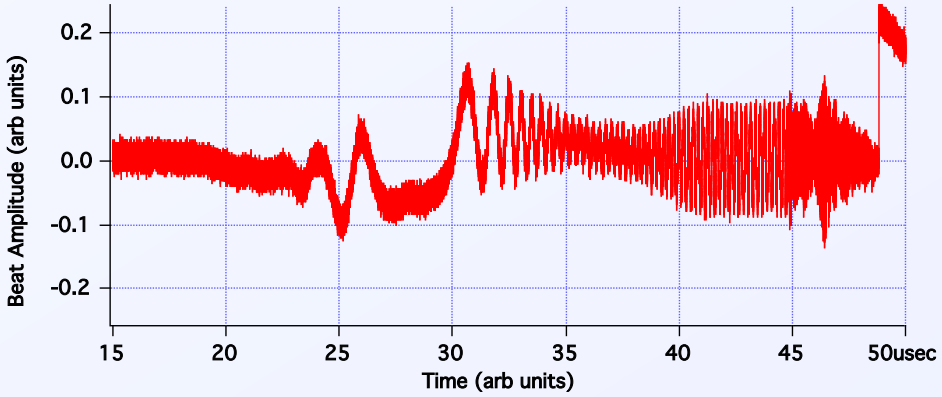
Example #1 of real data (all high velocity)



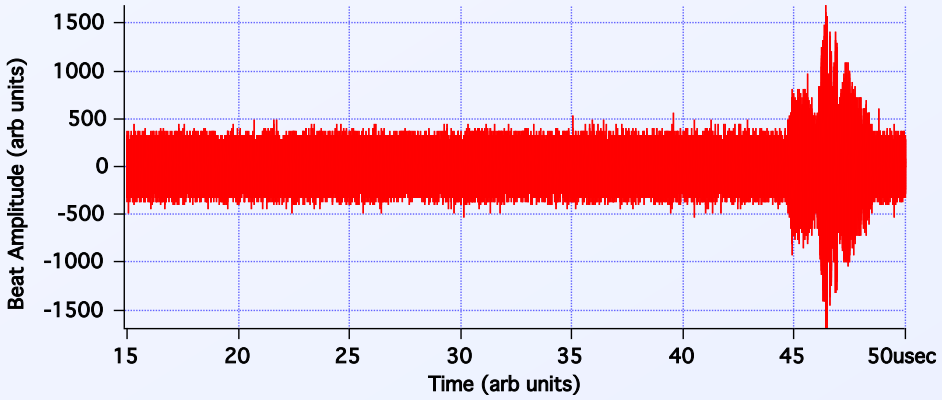
Differentiation increases noise

Example #2 of real data (low velocity plus high velocity)

Original file



Differentiated file



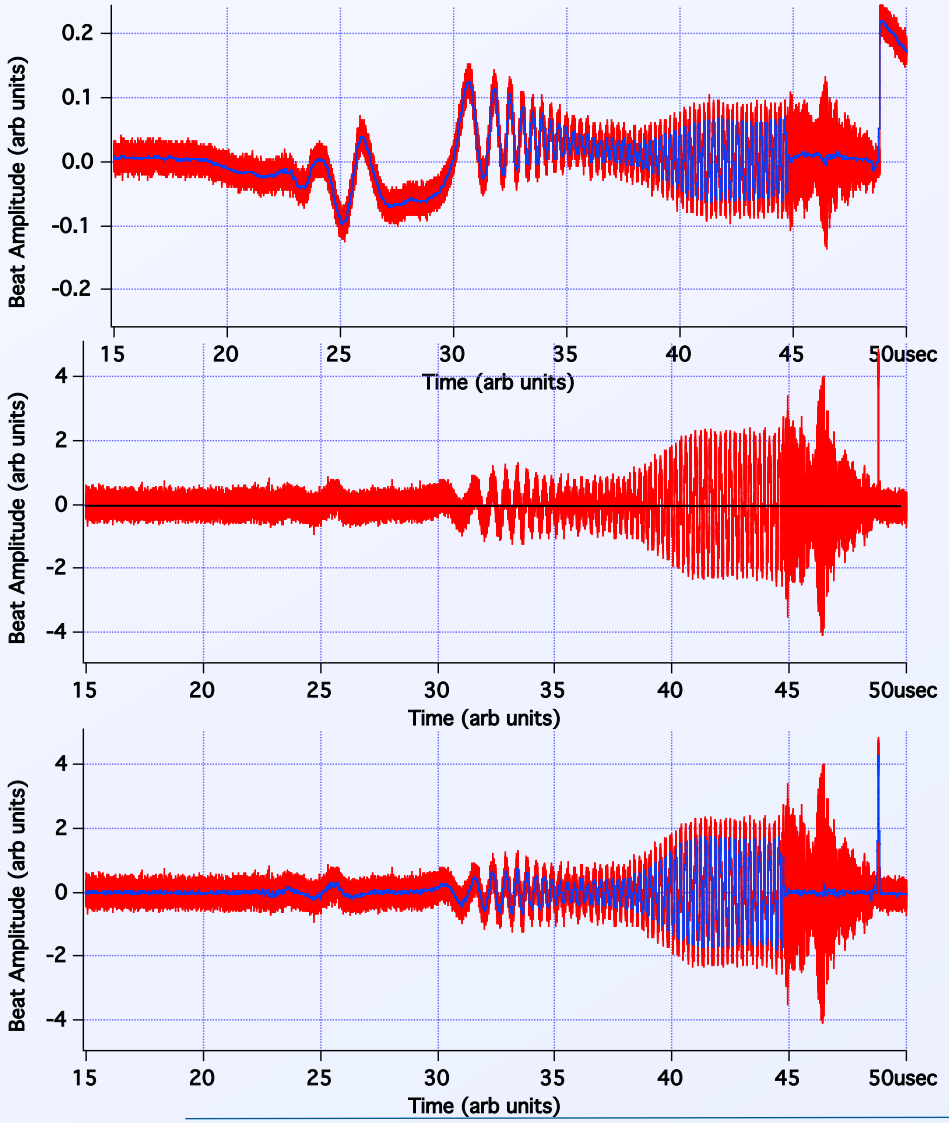
The beat waveform has a low frequency compared to the noise.

Example #2 of real data

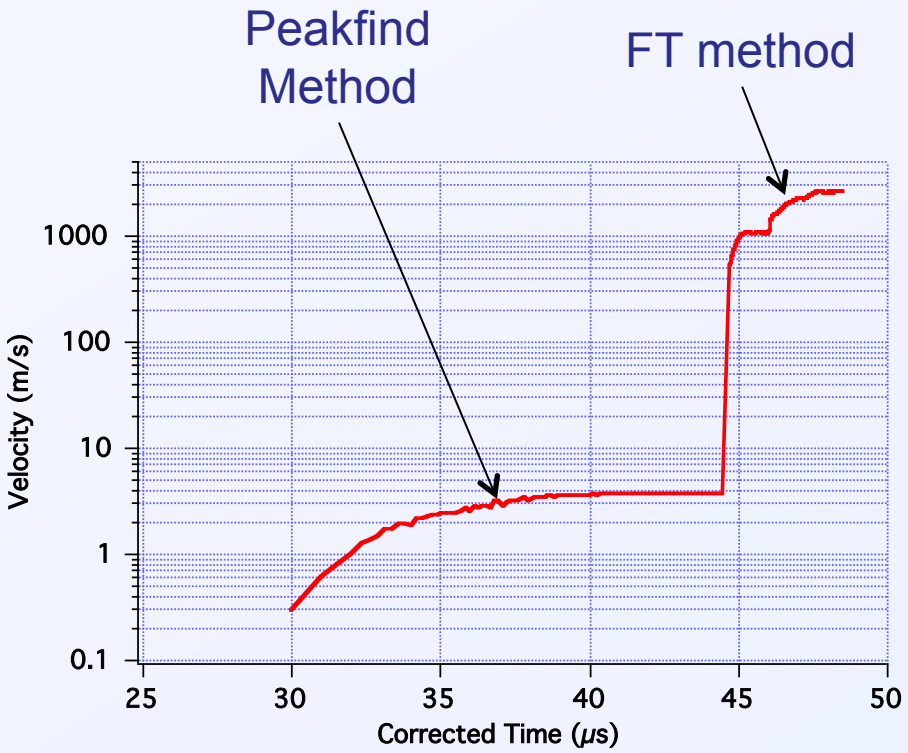
Need to do some smoothing first so that the noise is minimized.

Differentiate the smoothed data.

Smooth the differentiated data to find only 1 point per half cycle.



This is the final velocity profile for Example #2



Main steps in the Peakfind Code

WCI WEAPONS AND COMPLEX INTEGRATION

Smooth

Differentiate

Smooth again

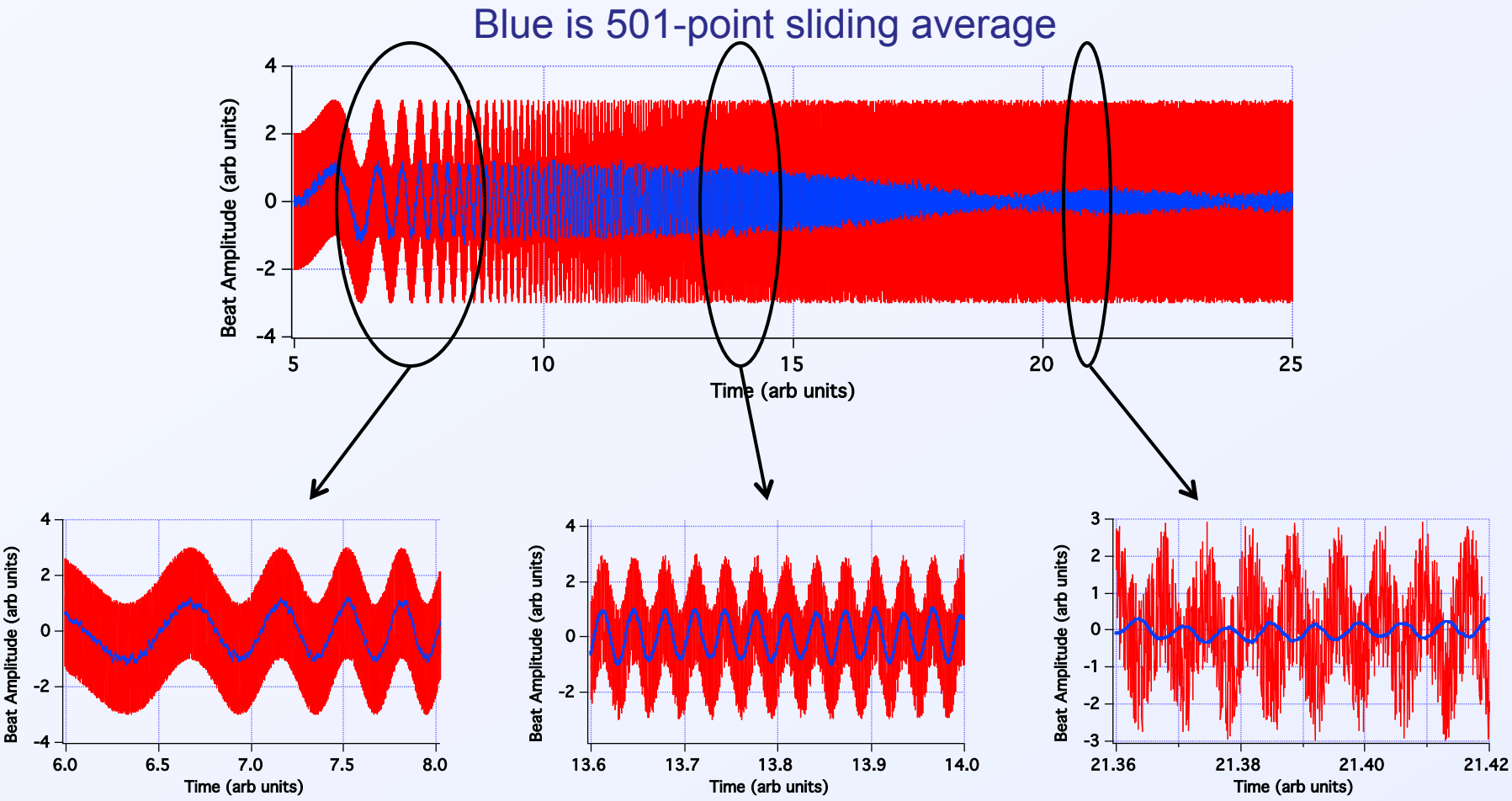
```
//Generate working wave and find peaks with smoothing
Duplicate InputWave WorkBeat
Print "Number of points in sliding average is", Npts
Smooth/B Npts, WorkBeat
Differentiate WorkBeat
Smooth/B Npts, WorkBeat
FindLevels/B=(Npts) WorkBeat,0

//Find delta T between zeros and convert to velocity
Make/D/O/N=(V_LevelsFound) W_FindLevels, VelZeroCross, Tave
variable i
for (i=0;i<V_LevelsFound;i+=1)
    VelZeroCross[i]=775/2/(W_FindLevels(i+1)-W_FindLevels(i))*1e-9
    Tave[i]=(W_FindLevels(i+1)+W_FindLevels(i))/2
    endfor
```

Find zero crossings, compile times

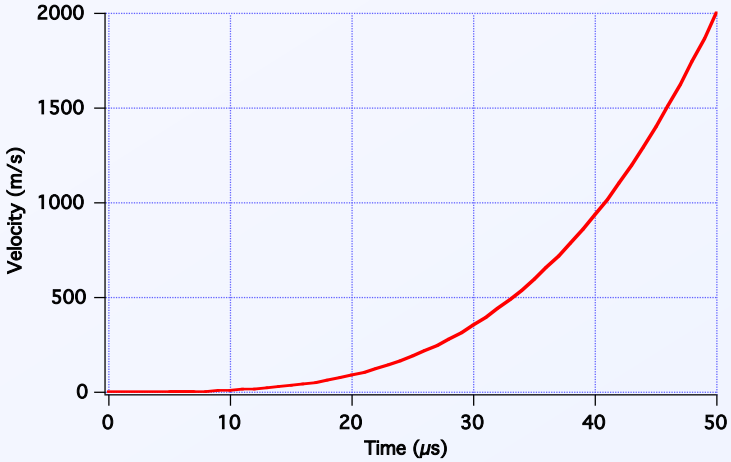
Calculate velocities averaged over each half cycle

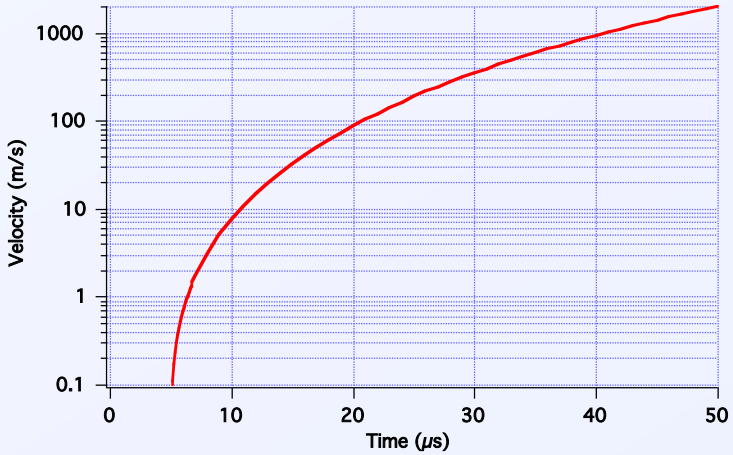
The only “knob” that the code has is the amount of smoothing.
I use sliding averages in the code because of the Excel work.

Need to do the right amount of smoothing

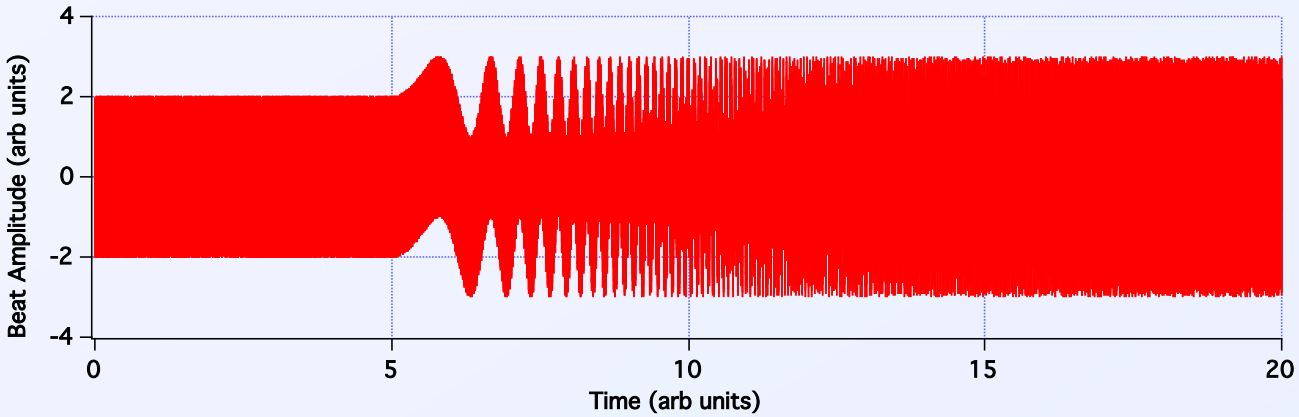


Let's run the Peakfind code on this velocity profile



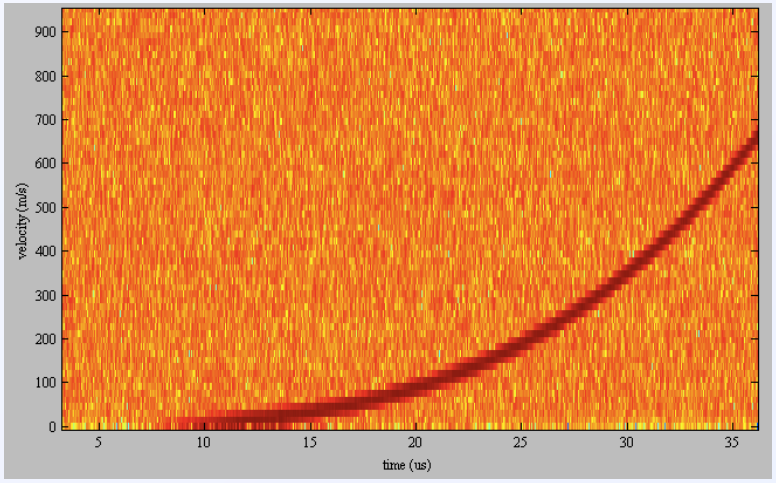


Generate a beat waveform with S:N = 1:2

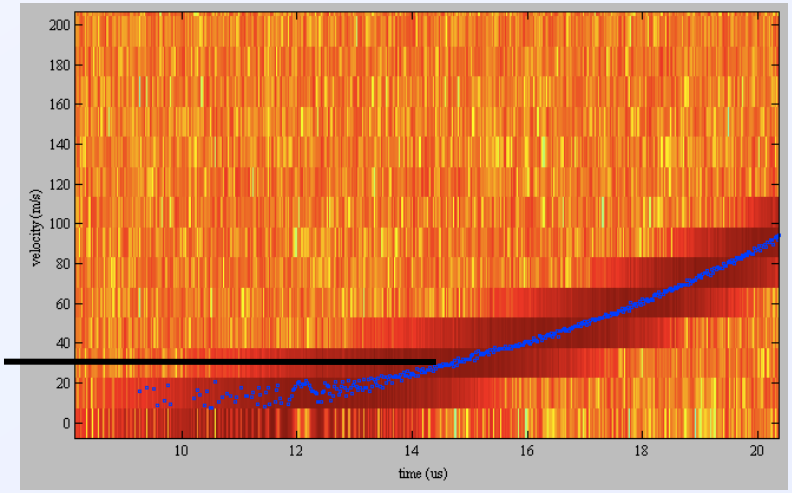


First I process with the FT code as usual

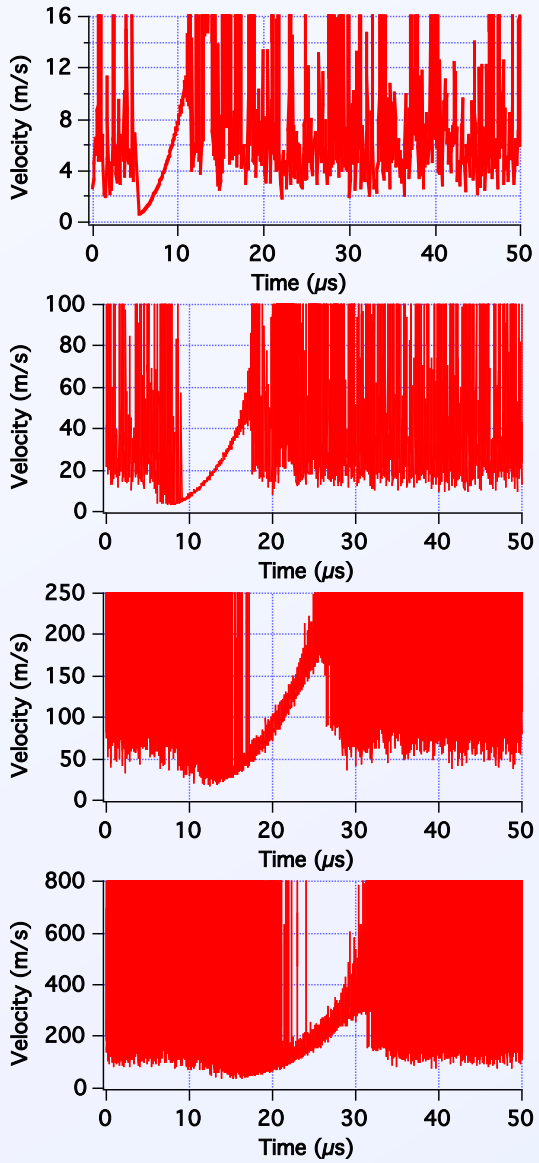
Perform the Fourier transform analysis with 51 ns windows.
We note that the noise increases below around 30 m/s.



30



Run the Peakfind code with different amounts of smoothing

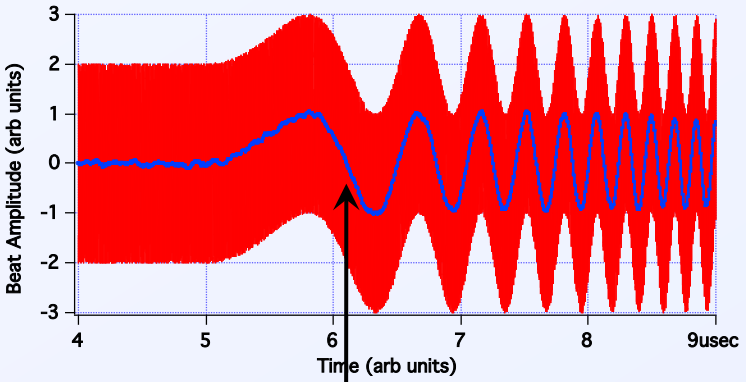


1001 pts

201 pts

41 pts

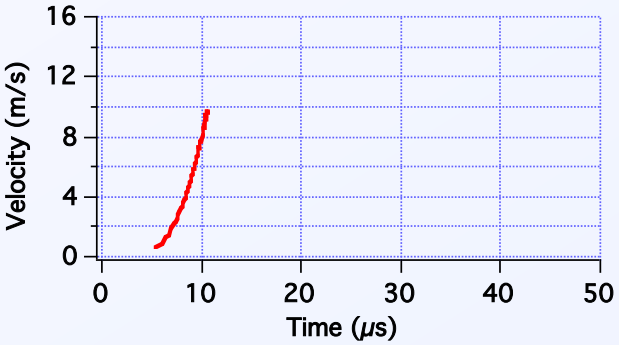
21 pts

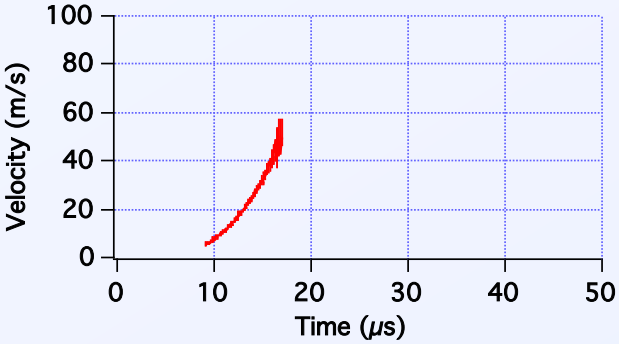


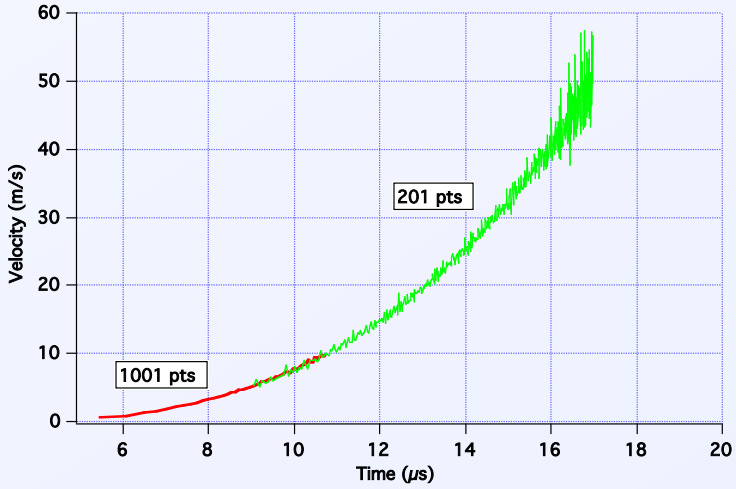
Expect to find 1st point around 6.1 μs

Remember that we want a method to get below 30 m/s.

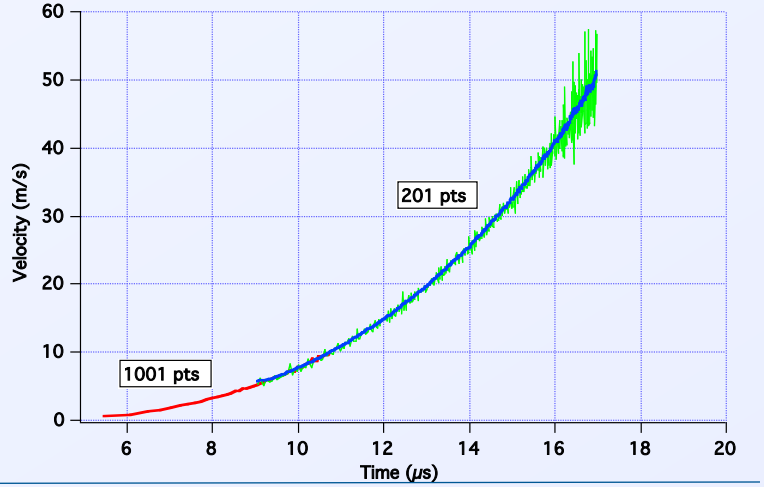
Delete the points that are obviously too noisy and plot together



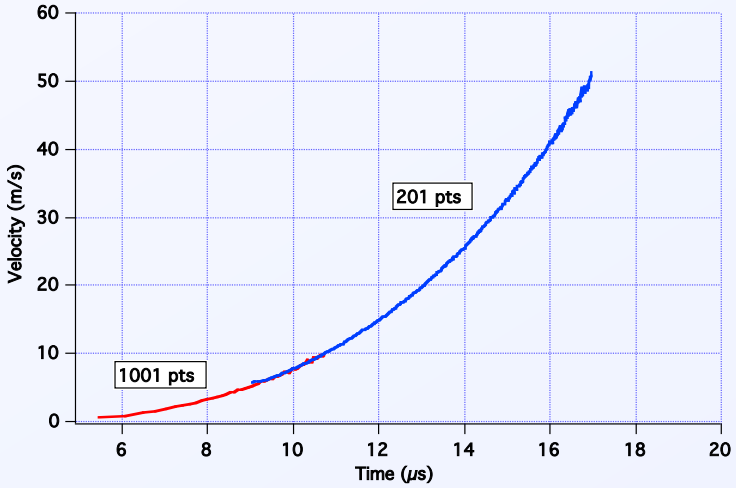


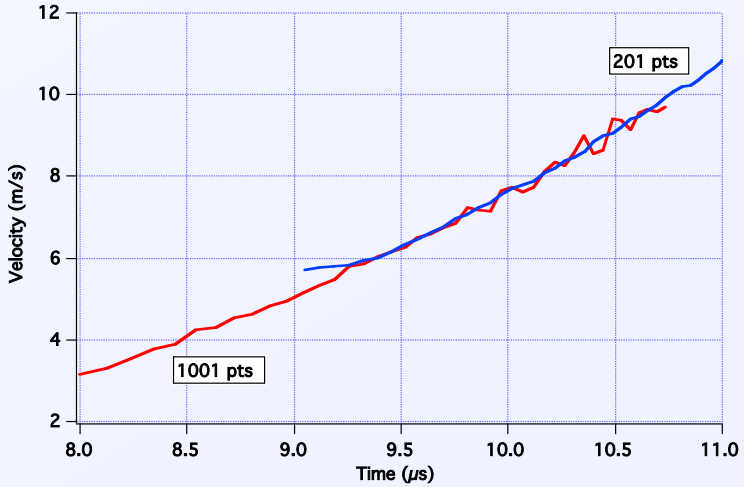


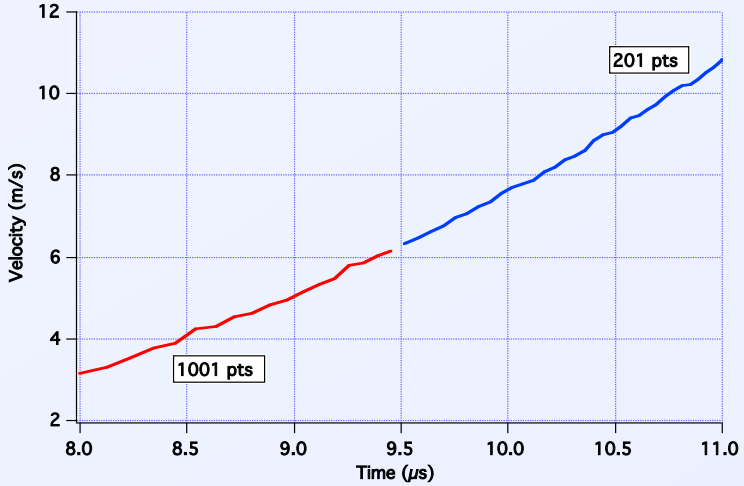
Maybe do a little additional smoothing on the noisier one (11 pt sliding ave).



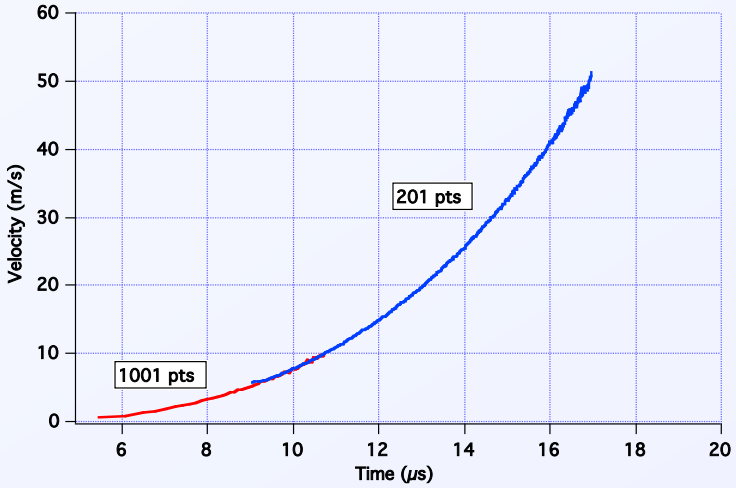
Clean up the overlap region

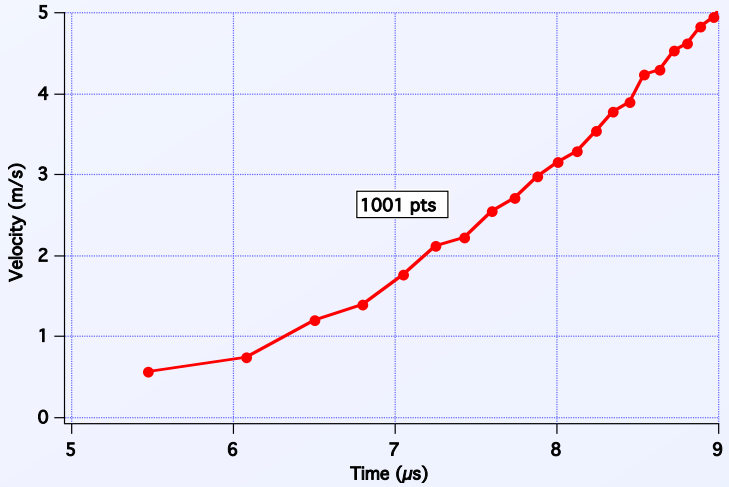


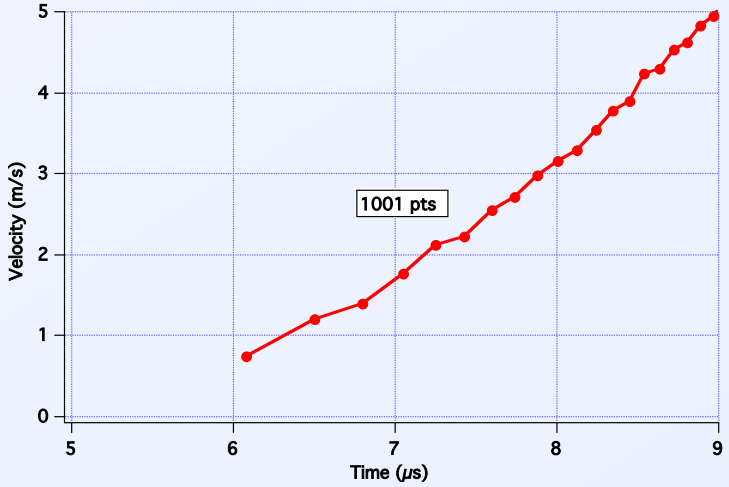




Check the beginning of the velocity profile



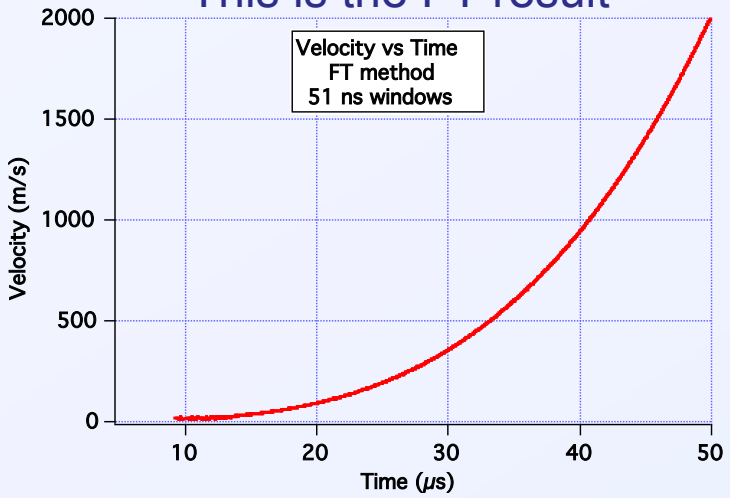




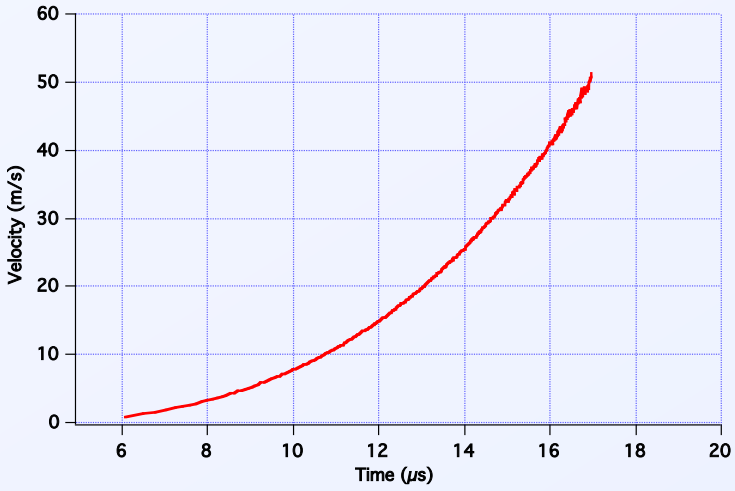
Remember: 1st point should be around 6.1 μs

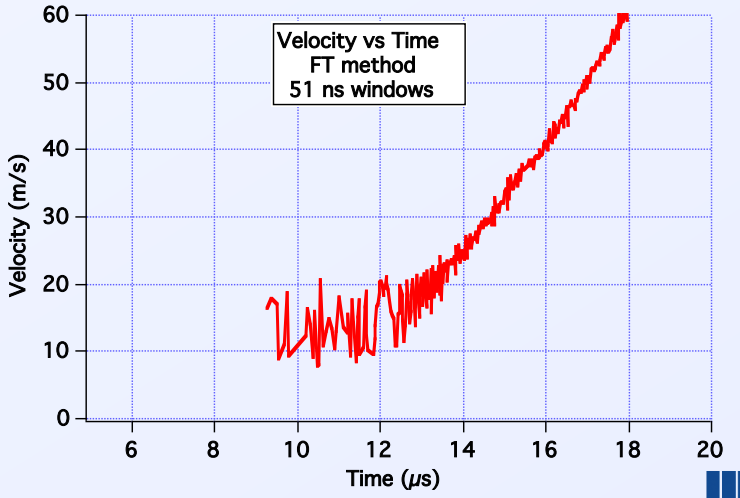
Combine the peakfind curve with the FT curve

This is the FT result

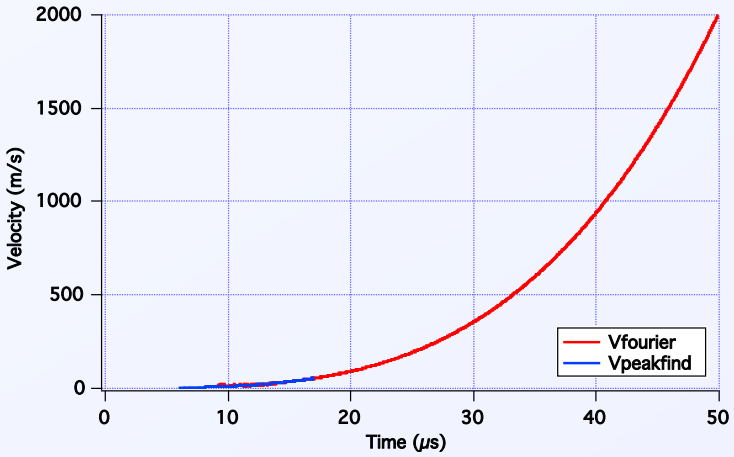


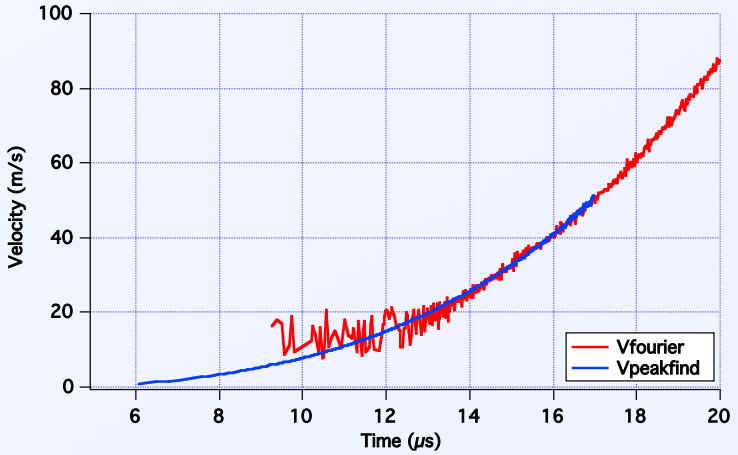
Combine the sm1001 curve
with the sm201 curve
to obtain the final peakfind profile

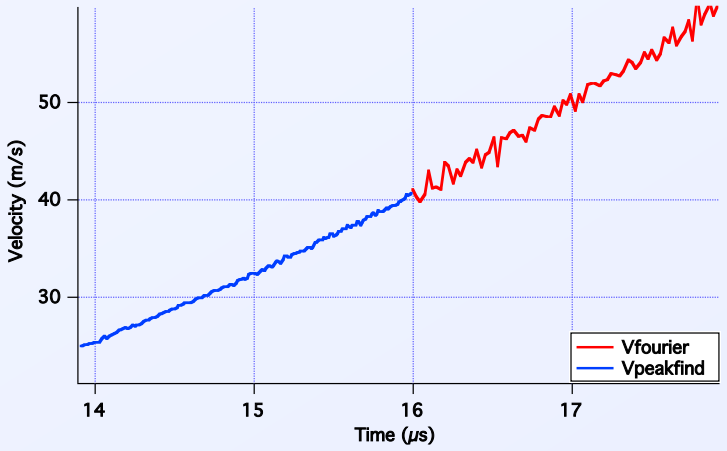




Connect peakfind results with FT results for final profile

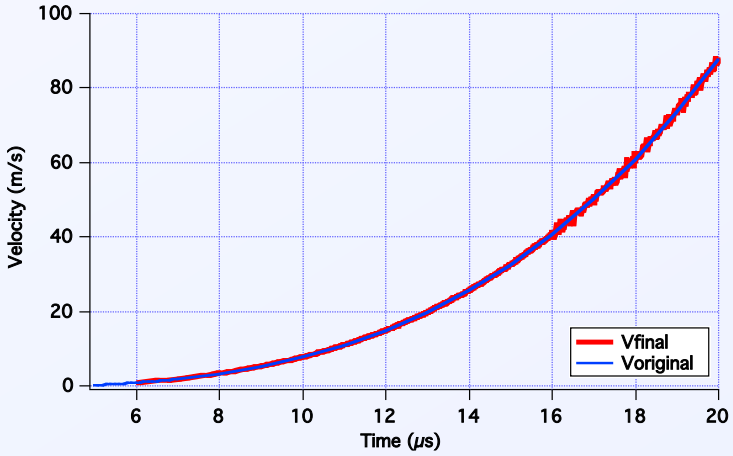


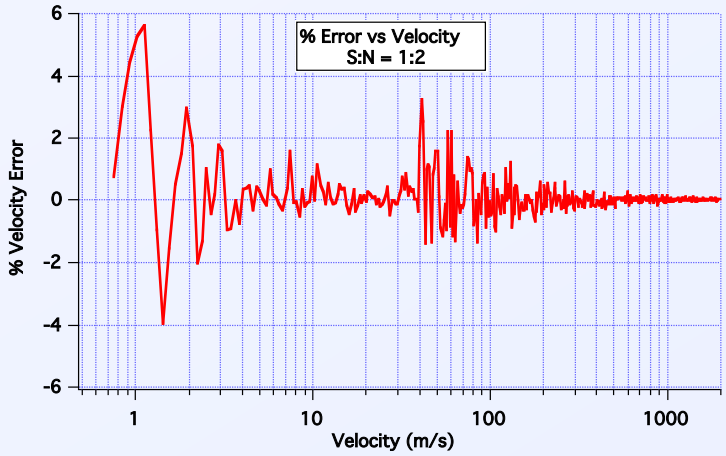




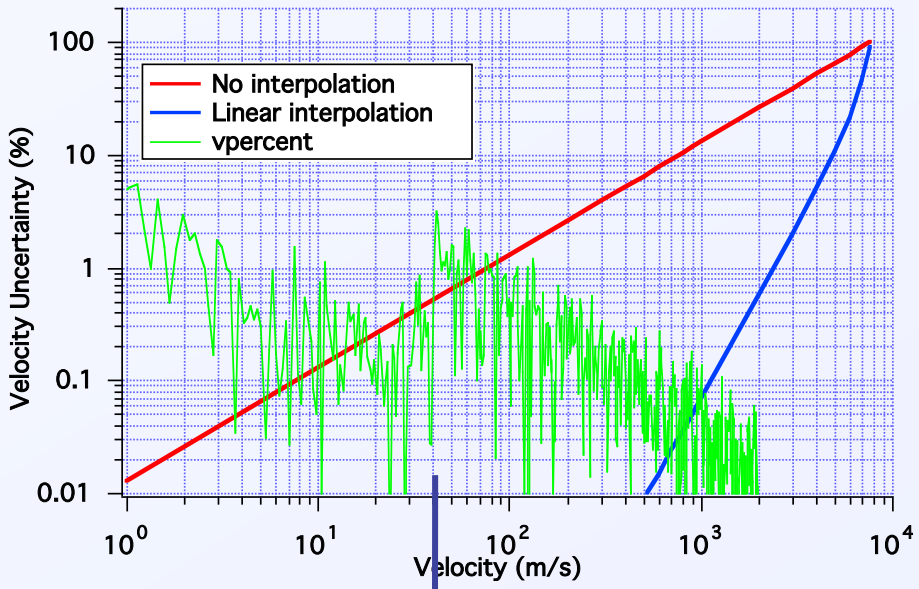
Blue = peakfind method
Red = FT method

Compare final VvsT profile with original profile





Summary of calculated errors



Peakfind Method

FT method

(absolute value of % errors from peakfind method)

Red is peakfind method on noise-free data with no linear interpolation.

Blue is peakfind method on noise-free method with linear interpolation.

Green is processed result on data with S:N = 2:1.

Conclusion

I show how I process very low (< 10 m/s) velocities using a peakfinding method.

I generally achieve sub-1% errors except at very low velocities and except with very noisy data.

I developed this code starting with Excel spreadsheets and then translated it into Igor.

I am sure there are better ways to do this:

Use filters rather than sliding averages, for example.

If someone comes up with a more user-friendly method, I would love to hear about it.