DOE/NV/25946--825

# Least-Squares PDV Error Estimation

**Jerome Blair, Great Basin**
**Eric Machorro, NSTec**
Institute for Adv. Technology PDV Workshop
Austin, Texas November 5-6, 2009

Approved for public release. Distribution unlimited.

*This work was done by National Security Technologies, LLC, under Contract No. DE-AC52-06NA25946 with the U.S. Department of Energy.*

National Security Technologies LLC
Vision • Service • Partnership

# Abstract

A new method for analyzing PDV data is discussed. It is based on using preliminary frequency estimates garnered from a windowed spectrogram and an interpolated FFT algorithm. A novel third step produces a least squares estimate with mathematically precise error bounds that meet the Cramer-Rao least variance lower bound. This approach can be used to study phenomena at the sub-nanosecond scale by allowing the window length to change adaptively to account for signal speed and noise level, while minimizing the expected errors.

# Principal authors of the software

Jerry Blair

William Kuhlow

Eric Machorro

*Thanks to:*

*David Holtkamp at LANL who supplied several utility-routines that were used in testing the software.*

*Ted Strand at LLNL who supplied some very helpful and interesting example data to “stress test” these algorithms.*

# Overview of PDV Analysis Algorithm

1) Filter raw data to remove time varying offset.
2) Estimate noise level from 1st 1000 points.
3) Perform spectrogram of length $N=2^p$ with cosine window
4) Find peaks (old algorithm)
5) Refine peaks with interpolated FFT algorithm
6) Refine further with least squares
7) Derive error estimates & show data with error bars
8) *Adaptively vary interval length to lower error*

Under construction

# Generating the Spectrogram

- For each selected time interval of length $N$
  - Multiply measured signal by cosine window.
  - Calculate magnitudes of Fourier coefficients of result.

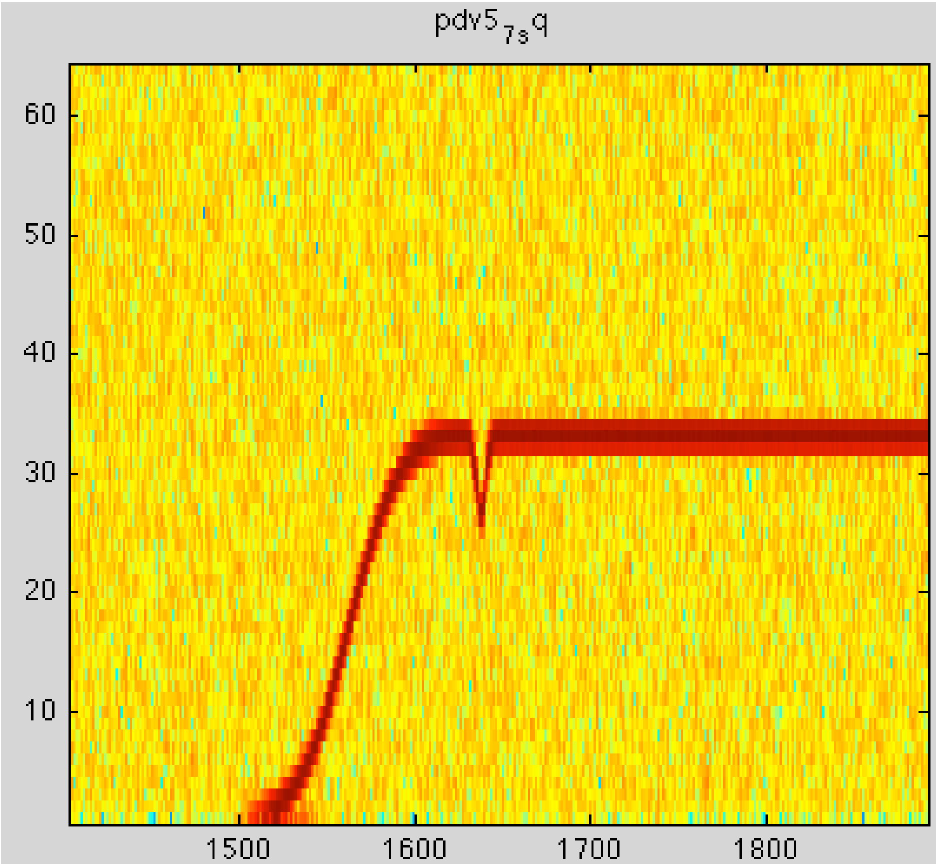
- Perform for intervals centered at $(1/2)N$, $N$, $(3/2)N$, …

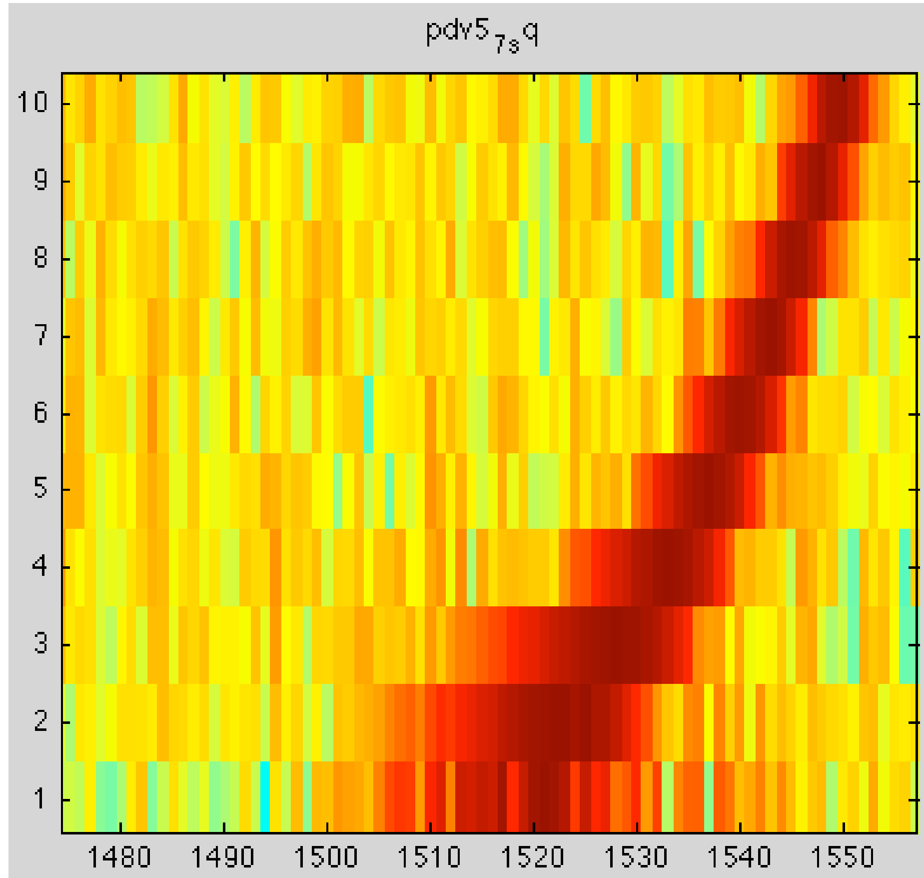
- Result is a vector of Fourier coefficients
  - Length of each vector in $N/2$
  - Time spacing of vectors is $(N/2)\Delta t$

- Plot and save result

*In next version, this has changed to a time spacing/overlap of $(1/8)N\,\Delta t$*

# Example Spectrograms

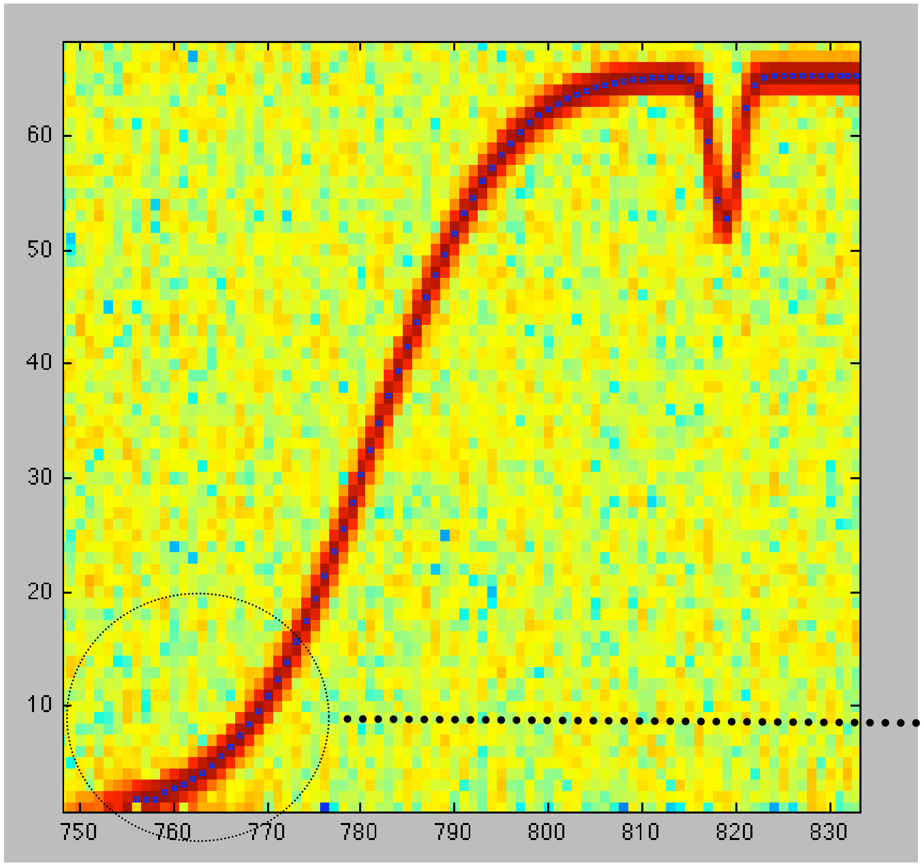


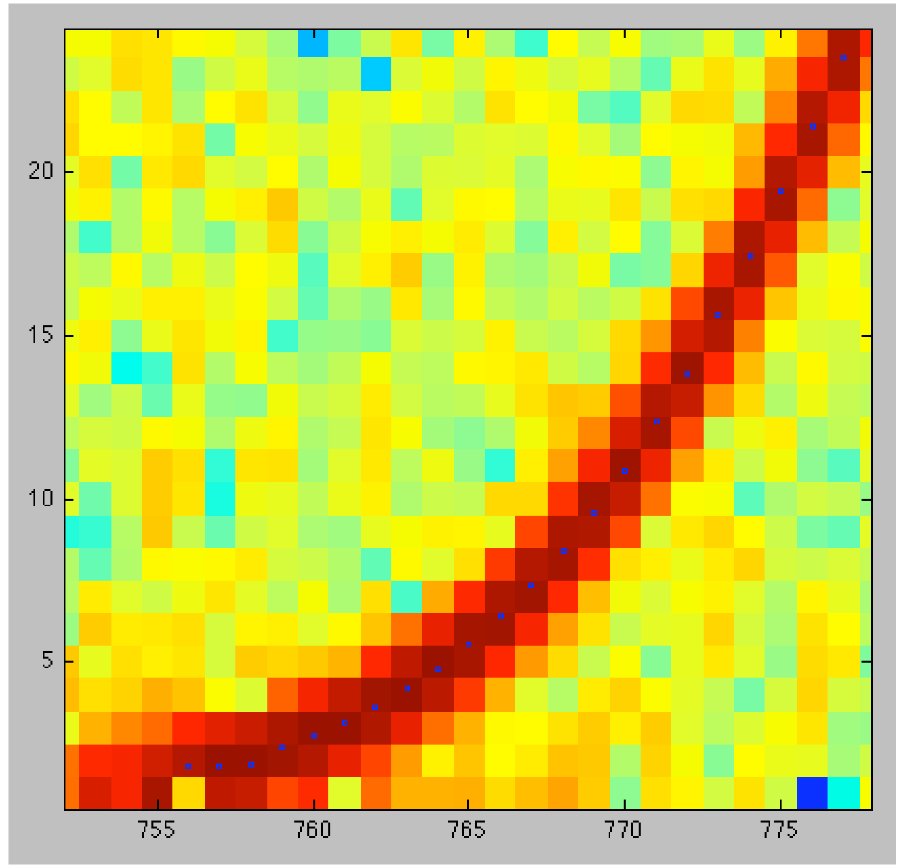




*N* = 128 for 64 vertical slices

Horizontal slices are separated by 64$\Delta t$.

# Find and Refine Peaks




- For each vertical column, find pixel with maximum intensity.
- Interpolated Fast Fourier-Transform (IpFFT) uses neighboring intensity value to give fractional pixel value.

# Least-Squares Estimation

- Uses original signal as a function of time.
- One frequency estimate for each spectrogram estimate—using the same interval of data.
- Interpolated FFT estimate (based on the spectrogram) is the starting value for an iterative non-linear least squares routine.

# Fitting the data to the model

$$y(t) = A(t+\delta t)\sin(X(t+\delta t)) + \eta(\mathrm{t})$$

for $t \in \left[ t_c - \frac{N\Delta t}{2}, t_c + \frac{N\Delta t}{2} \right]$

with

$$X(t) = x_0 + x_1(t - t_c) + x_2(t - t_c)^2,$$

$$A(t) = A_0 + A_1(t - t_c),$$

$$\eta(t) = noise, \quad \delta t = time\ jitter,$$

and $f(t_c) = \frac{x_1}{2\pi}$ the frequency estimate.

# Least-Squares Estimation

$$\text{Fit } y(t) = A(t)\sin(X(t))$$

with

$$X(t) = x_0 + x_1(t - t_c) + x_2(t - t_c)^2,$$
$$A(t) = A_0 + A_1(t - t_c).$$

- Spectrogram gives initial estimate for $x_0$. The spectrogram can also also provide estimates for $x_1$ ,$x_2$ , and $A_0$.
- Use linear least squares to yield an initial estimate for the *remaining parameters.*
- Use those starting values to begin the nonlinear least squares routine.

National Security Technologies LLC
Vision • Service • Partnership

# Error Estimates

Key assumptions are that the true signal has two continuous derivatives and that the oscilloscope noise is white.

**Fitting error**, due to signal not fitting the model

$$E_{2\sigma} \cong \sqrt{c_1^2\sigma_{noise}^2 + c_2^2\ddot{v}(t)^2 + c_3^2\ddot{A}(t)^2}$$

**Random error**, due to oscilloscope noise and time-jitter.

# Random & Fitting Error Estimates

$$E_{2\sigma} \cong \sqrt{c_1^2 \sigma_{noise}^2 + c_2^2 \ddot{v}(t)^2 + c_3^2 \ddot{A}(t)^2}$$

**Random error**, due to oscilloscope noise and time jitter.

**Fitting error**, 2nd derivative estimated with divided difference of velocity

- Random error based on oscilloscope noise being white (from tests of oscilloscopes.) The term $\sigma_{noise}$ is the standard deviation of the random effects that include oscilloscope noise and time-jitter.

# Error Estimates - cont'd

$$E_{2\sigma} \cong \sqrt{c_1^2 \sigma_{noise}^2 + c_2^2 \ddot{v}(t)^2 + \ldots}$$

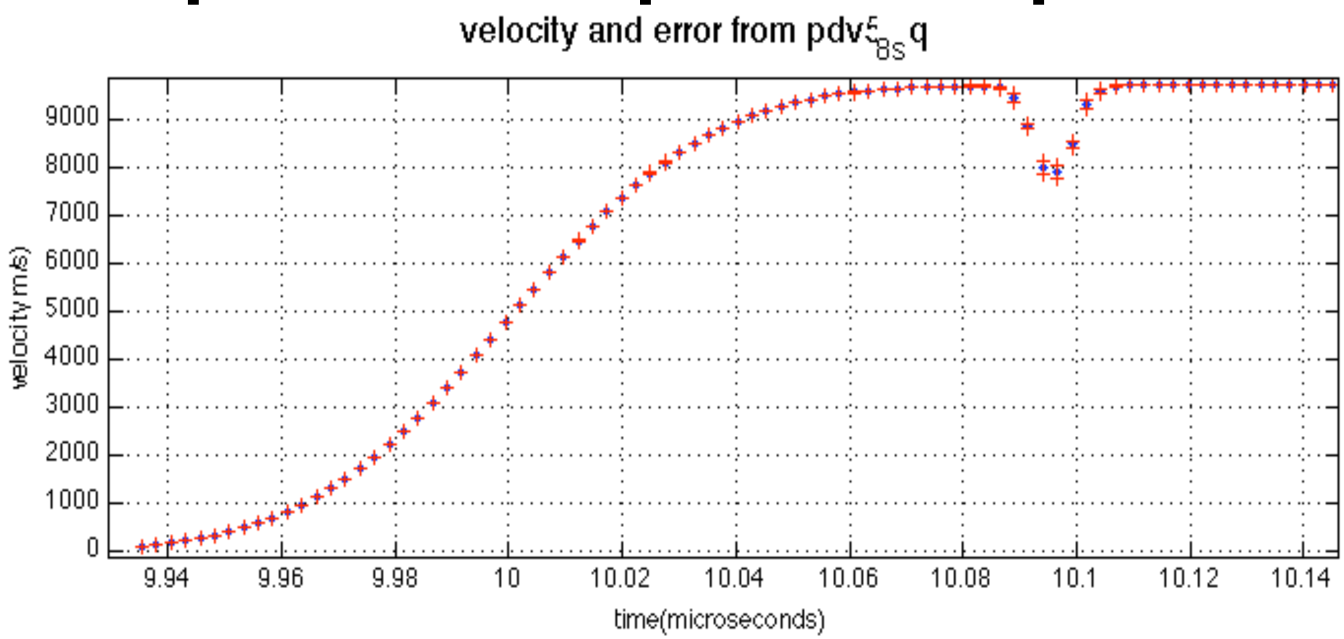
Note that

$$c_1 \cong \frac{C_1 \Delta t^{-3/2} N^{-3/2}}{A_0}, \text{ and } c_2 \cong C_2 \Delta t^2 N^2.$$

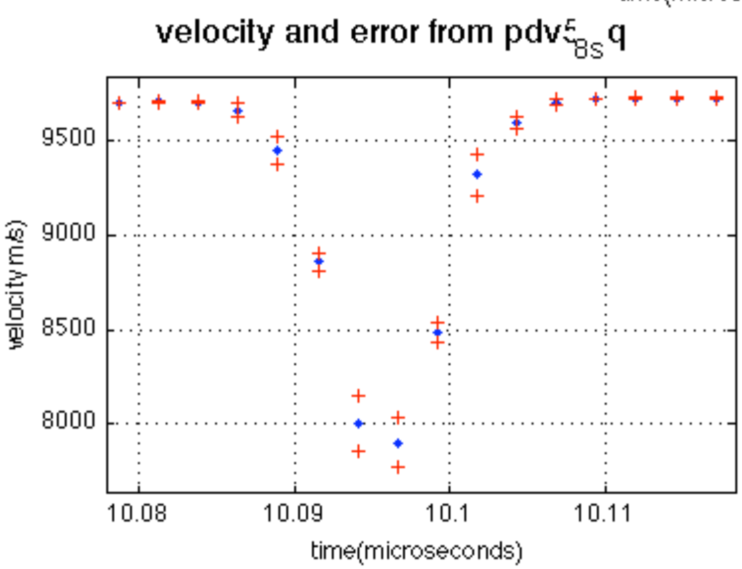
These approximations can be useful to know but are not used by the analysis software.

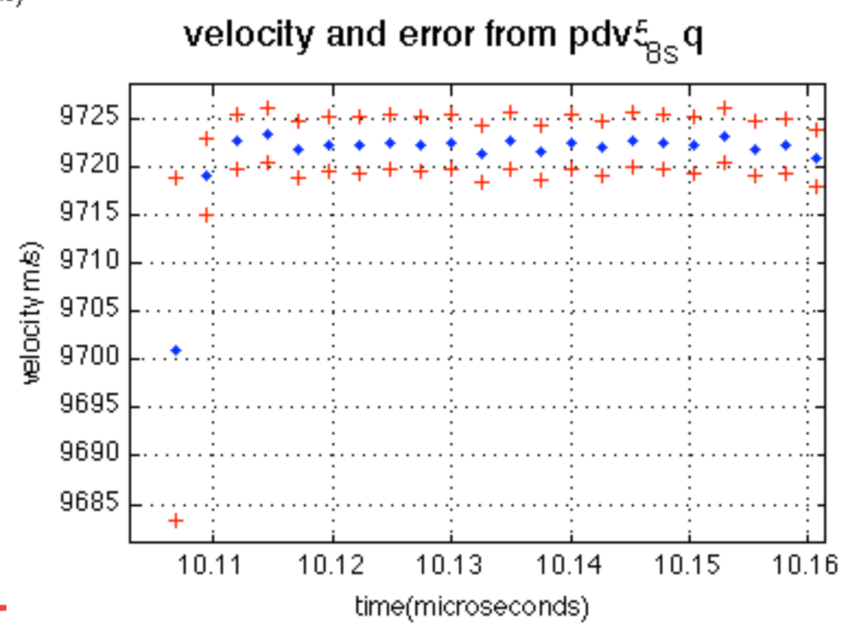
# Some examples … well ... okay … 1 example

# Example Least-Squares Output

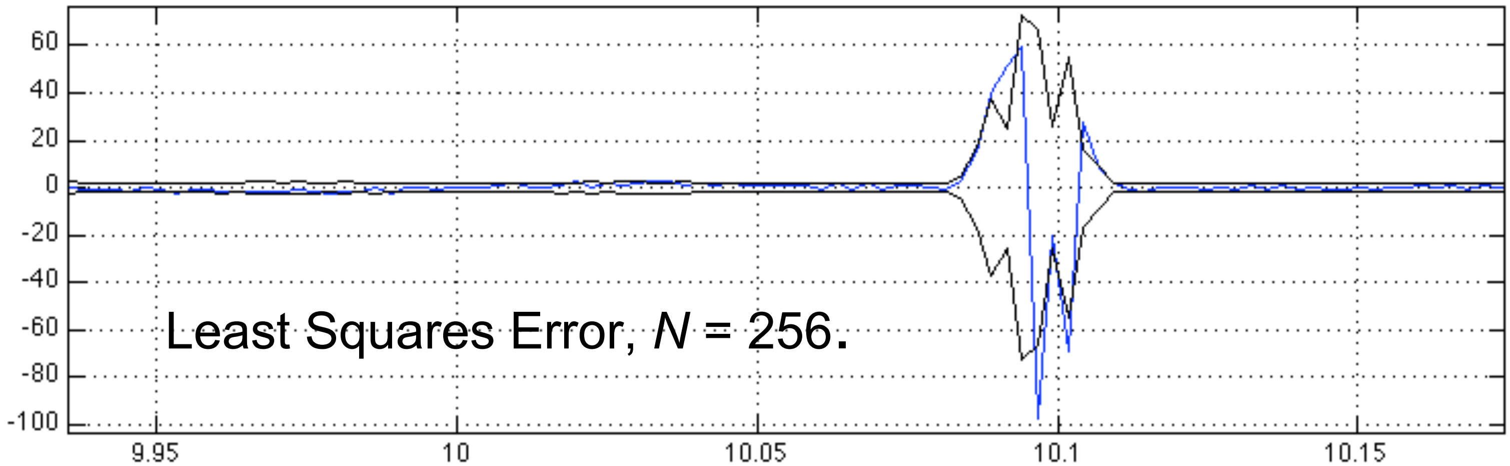


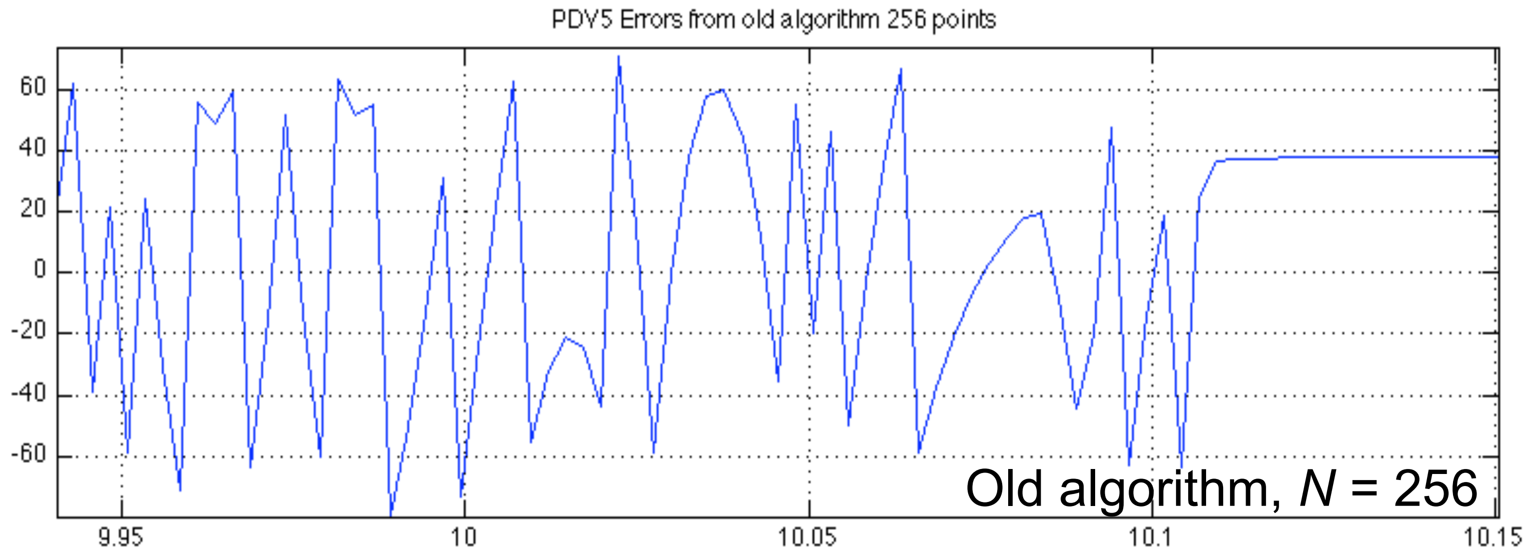

*Subinterval of*
*N* = 256 pts



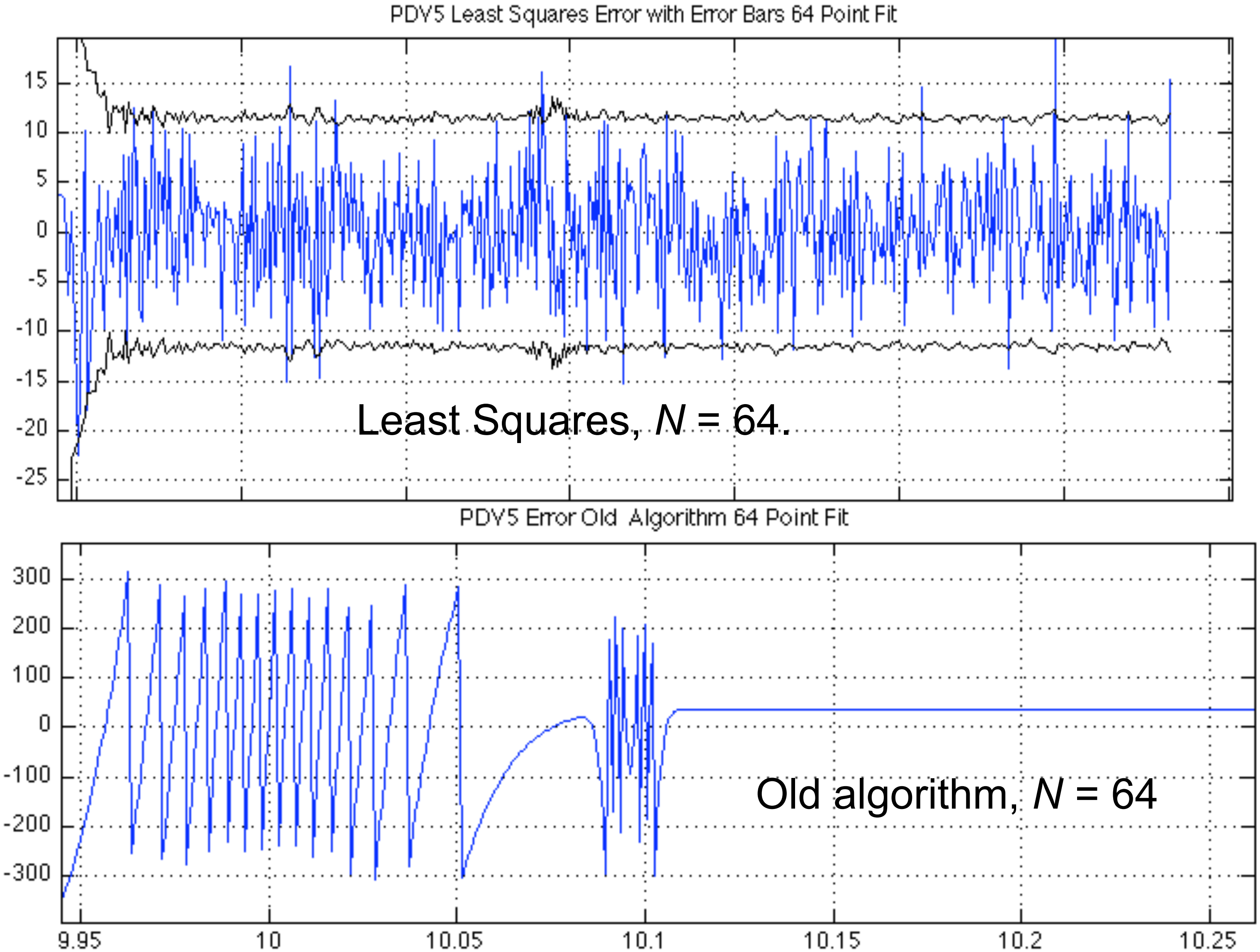
# Comparison of Errors



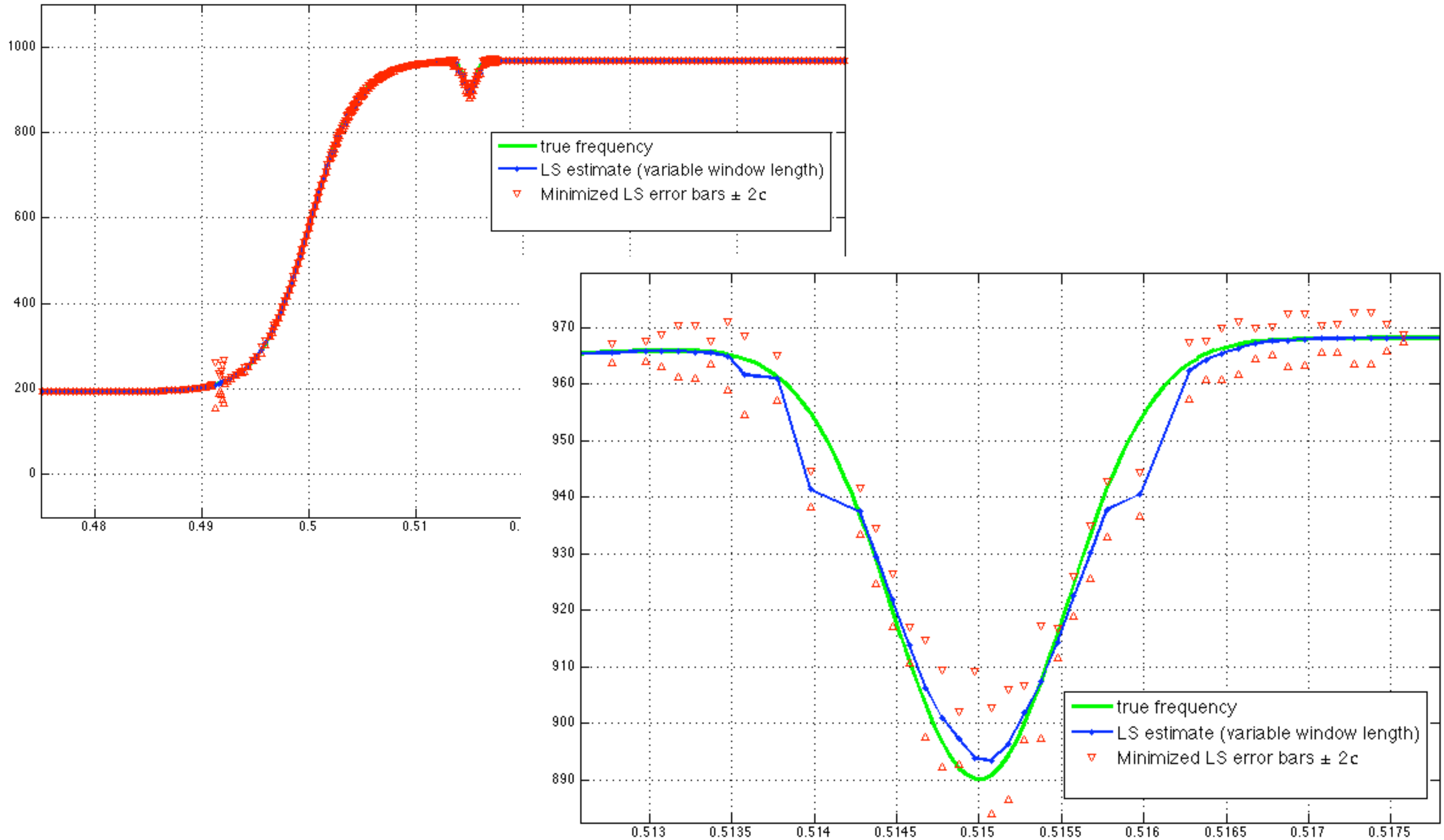




National Security Technologies LLC
Vision • Service • Partnership

# Shorter Window Size



# A preview of upcoming attractions …



# Summary

- New GUI-based software produces error bars that account for measurement noise, time-jitter, and fitting errors.
- Method produces the best, least-squares, unbiased estimate of the frequency. Mathematically impossible to do better on average for a given window length.
- Adaptively changing the window length, even more accurate error bars can be calculated. Determining the best window length to capture the phenomena of interest will no longer be an issue.

# Issues to fix … for adaptive windowing

- Non-linear least-squares routines can often find “wrong” solution
  - These routines are very sensitive to the initial IpFFt-based estimate.
  - Constrained minimization routines commercially are available.
  - Use residual sizes and error-bar length to “toss out” bad estimates.

- Computational time for larger data sets can be quite long. In testing, one example took 2 hrs to complete.
  - It’s anticipated that this can be sped up considerably.
  - Fixing the window length (however) is much faster.