Mini-project 2

- Mini-project 2 presentations on April 21st.
- Please start thinking about a project you want to do.
- Please think of a project which uses material learned in this class (preferably from second half of class).

AIM seminar

Note unusual day, time, and location:

Date: 3-4 pm, Thursday, March 30, 2032 in 105 Shillman and by Zoom (Hybrid)

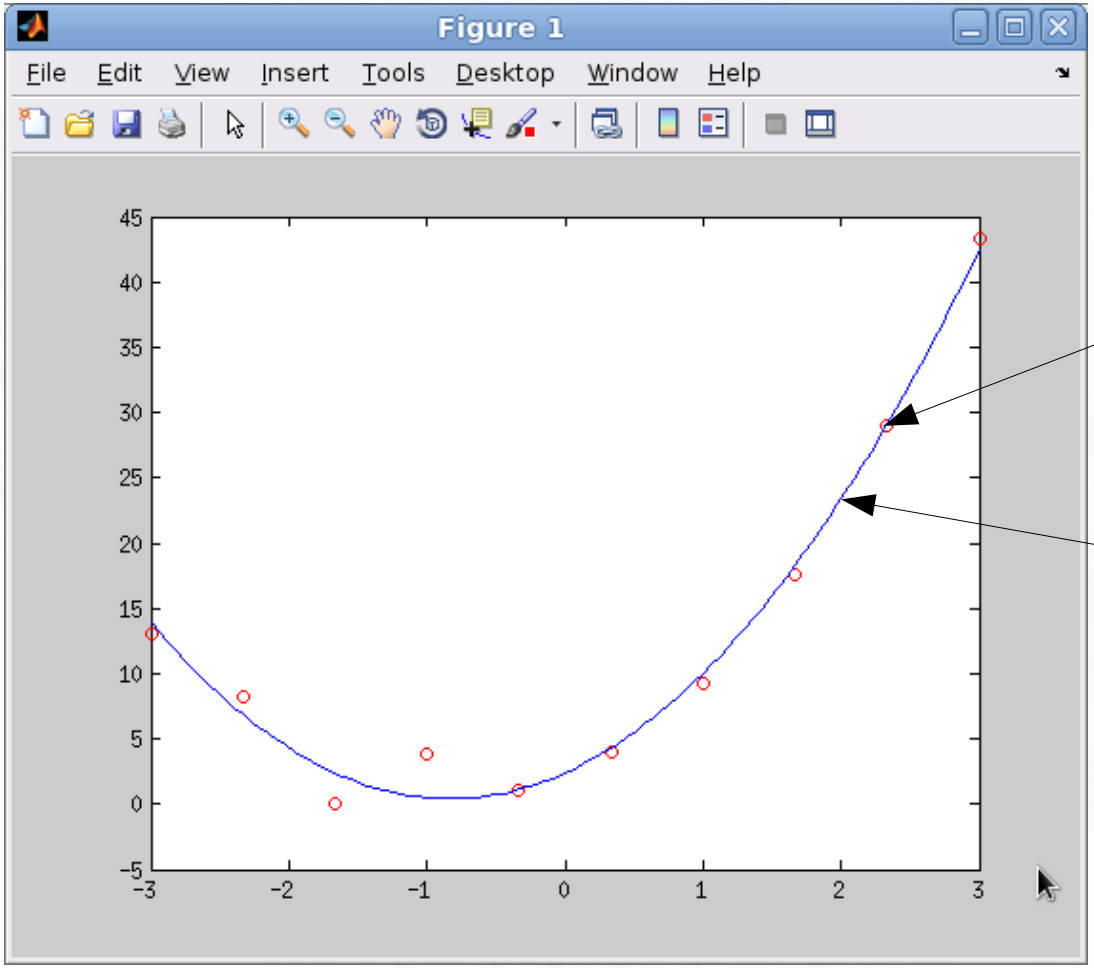
Speaker: Stuart Brorson (Northeastern University)

Title: Anomaly Detection using Linear Algebra

Abstract: Detecting anomalous events in time series is an important new application for computers. For example, if a problematic squeak or a rumble emitted by an industrial motor could be caught and repaired early, potentially millions of dollars of repair costs may be avoided. I will outline a simple anomaly detection algorithm which uses the Fourier Transform and methods drawn from Linear Algebra. I will demonstrate the algorithm running on a Beaglebone single-board computer and some inexpensive electronics. This talk will be accessible to undergrads and anybody interested in applications of applied math.

Polynomial regression

Polynomial regression



Data is assumed to be N (x_i, y_i) pairs

Goal is to find best fit quadratic:

$$y = a_0 + a_1 x + a_2 x^2$$

- For example, find best-fit quadratic to data

Polynomial regression

- We want to find line which minimizes total error
  - Quadratic: $y = a_0 + a_1 x + a_2 x^2$ Model
  - Error: $e = \sum_i (y_i - y(x_i))^2$ Prediction from model; Measured data

$$= \sum_{i=1}^{N} (y_i - [a_0 + a_1 x_i + a_2 x_i^2])^2$$

Fitting coefficients

- This involves finding $[a_0, a_1, a_2]$
- We could find a set of equations for $[a_0, a_1, a_2]$ by setting derivatives to zero:

$$\frac{\partial e}{\partial a_0} = 0 \qquad \frac{\partial e}{\partial a_1} = 0 \qquad \frac{\partial e}{\partial a_2} = 0$$

This would become messy. Also, becomes worse for higher-order polynomials.

Polynomial regression

- Taking derivatives and setting to zero

$$\frac{\partial e}{\partial a_0}=0 \qquad \frac{\partial e}{\partial a_1}=0 \qquad \frac{\partial e}{\partial a_2}=0$$

- Yields a 3x3 linear system

$$\begin{pmatrix} \sum_i 1 & \sum_i x_i & \sum_i x_i^2 \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 \\ \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i y_i x_i \\ \sum_i y_i x_i^2 \end{pmatrix}$$

- Solve for a_0 a_1 a_2 the usual way.

A better way to do it....

- Recall approach used for multiple linear regression. Write expression relating *y* and *x* pairs via *a* coefficients:

$$X\,a = y \quad \longleftrightarrow \quad \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{pmatrix}$$

Vandermonde matrix

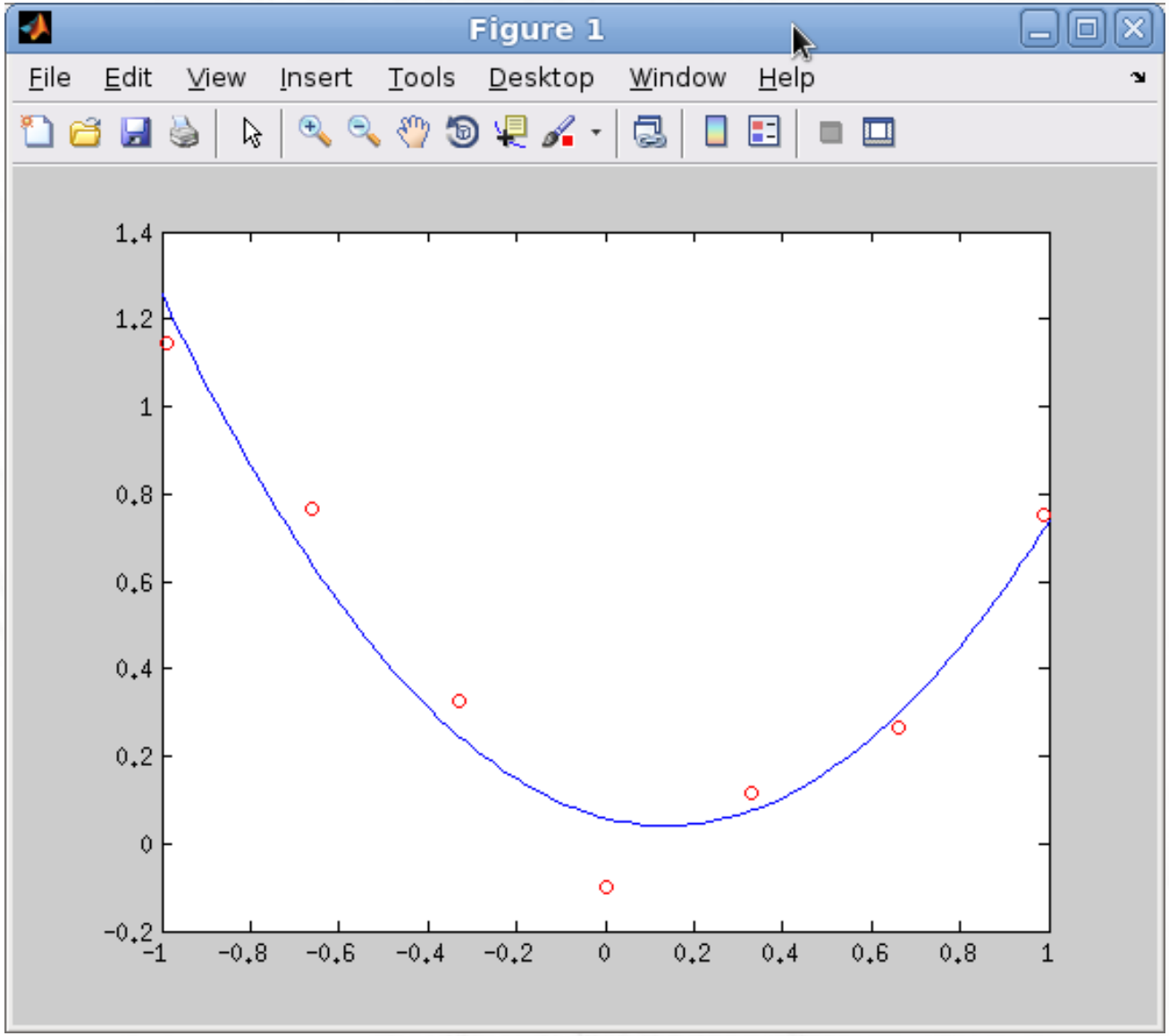
- Then, use same argument (minimizing Euclidian norm of residual) to get normal equations:

$$a = (X^T X)^{-1} (X^T y)$$

```
function A = make_polymatrix(x, M)
  % Given input points, returns Vandermonde matrix
  % whose rows are xi^n
  % Inputs are vector of sample points x and number of
  % columns to create N.

  % Output looks like this:
  % 1 x1 x1^2 x1^3 ... x1^N
  % 1 x2 x2^2 x2^3 ... x2^N
  % 1 x3 x3^2 x3^3 ... x3^N
  %                ...
  % 1 xM xM^2 xM^3 ... xM^N

  N = length(x)     % Set number of cols to create
  A = zeros(N, M);

  for row = 1:N
    for col = 1:M
      A(row, col) = x(row)^(col-1)
    end
  end

 end
```

```
function test_polyfit()

  N = 7;   % Number of sample points.
  x = linspace(-.99, .99, N)';
  y = (x-.1).*(x-.1) + .1*randn(N, 1);

  plot(x, y, 'ro')
  hold on

  M = 3;   % Order of poly to go to (order = M-1)
  A = make_polymatrix(x, M)

  a = (A'*A)\(A'*y)

  xn = linspace(-1, 1, 100);
  yn = zeros(size(xn));
  for i=1:M
    yn = yn + a(i)*xn.^(i-1)
  end

  plot(xn, yn, 'b')
```

Quadratic fit to 7 data points



How about higher-degree polys?

- Fitting with higher-degree polys is obvious extension of 2D approach.

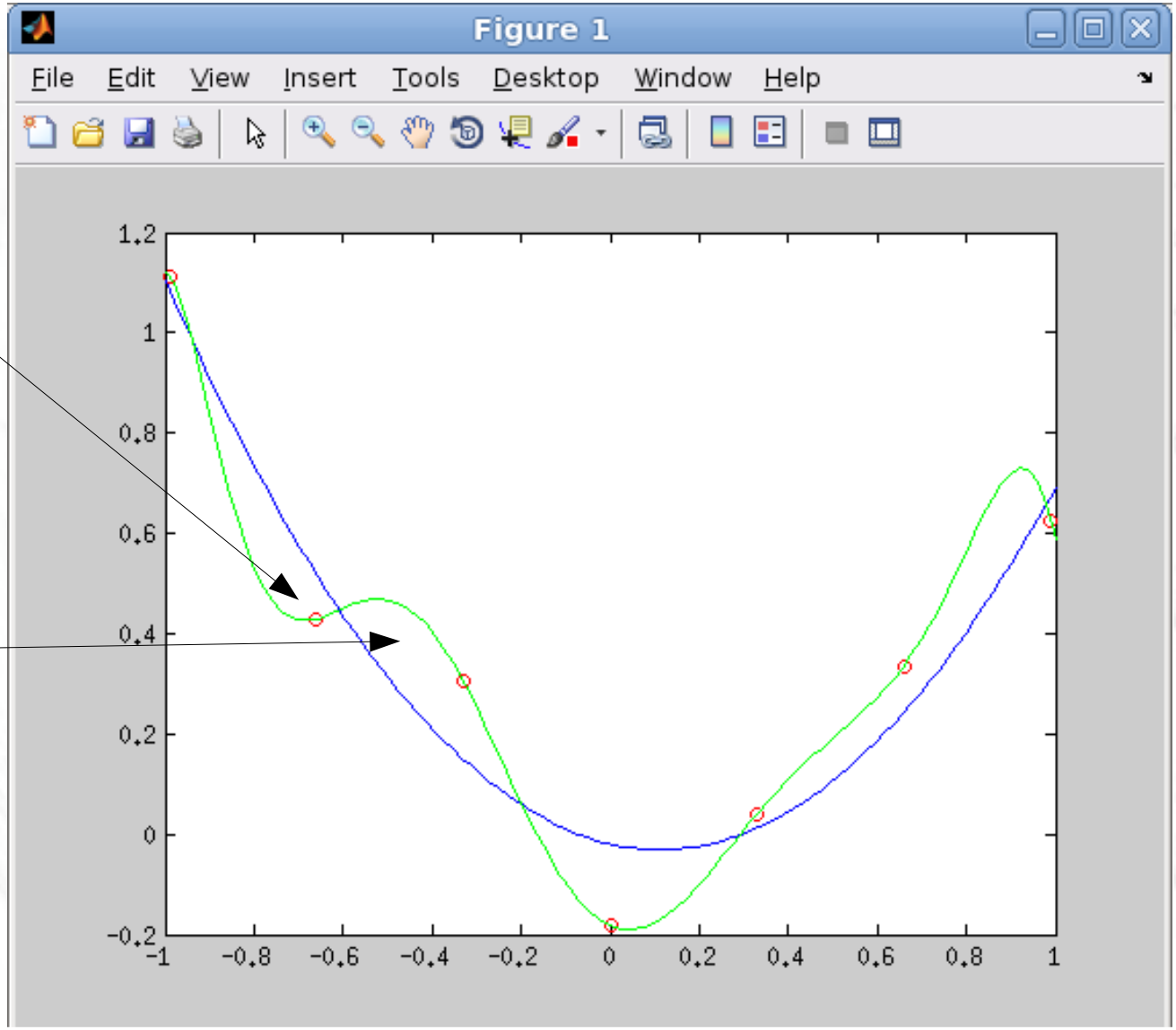
$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & \cdots \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & \cdots \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 & \cdots \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ \vdots \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{pmatrix}$$

Extend matrix with higher-order terms.

9th Degree polynomial fit to 7 data points

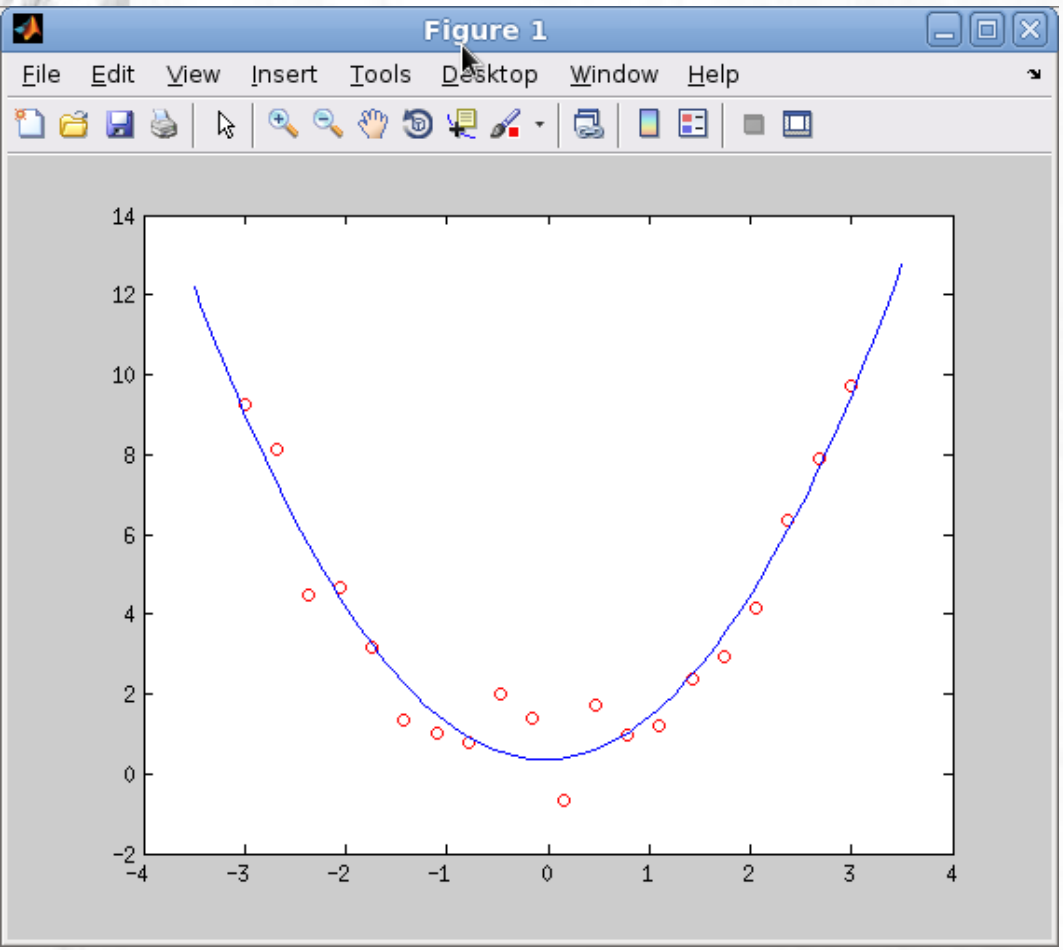
Fitting error at each datapoint is much smaller

However, fit curve looks crazy. This is because the data has been overfit.



Overfitting is bad

Matlab: polyfit() & polyval()




```
>> x = linspace(-3, 3, 20);
>> y = x.*x + 0.5*randn(size(x));
>> plot(x, y, 'ro')
>> p = polyfit(x, y, 3)

p =

    0.0003    0.9897    0.0744
0.3664

>> xn = linspace(-3.5, 3.5, 100);
>> yn = polyval(p, xn);
>> hold on
>> plot(xn, yn)
```

Remarks on polynomial fitting

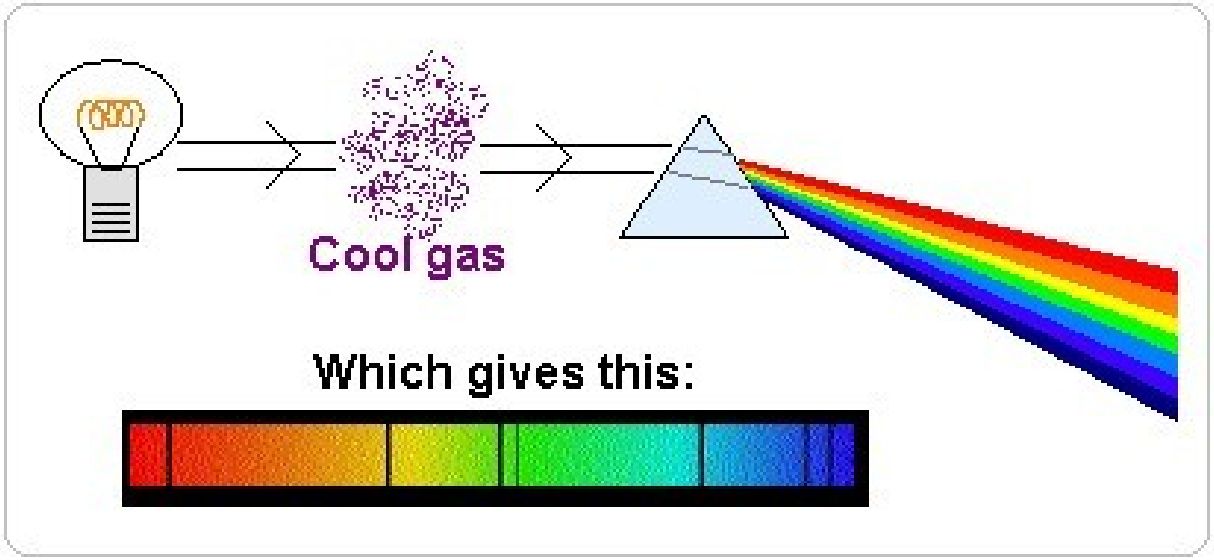
- Polynomial fitting works because the problem is “linear in the coefficients”.
  - Therefore, this is a multiple linear regression problem, which we know how to solve.
- Only fit low-order polynomials! High order polynomials lead to overfitting.
  - Most phenomena are simple anyway. If you need to fit to a high degree polynomial, you're probably doing something wrong with your model.

Spectroscopy and Chemometrics:

Real-world applications of linear regression

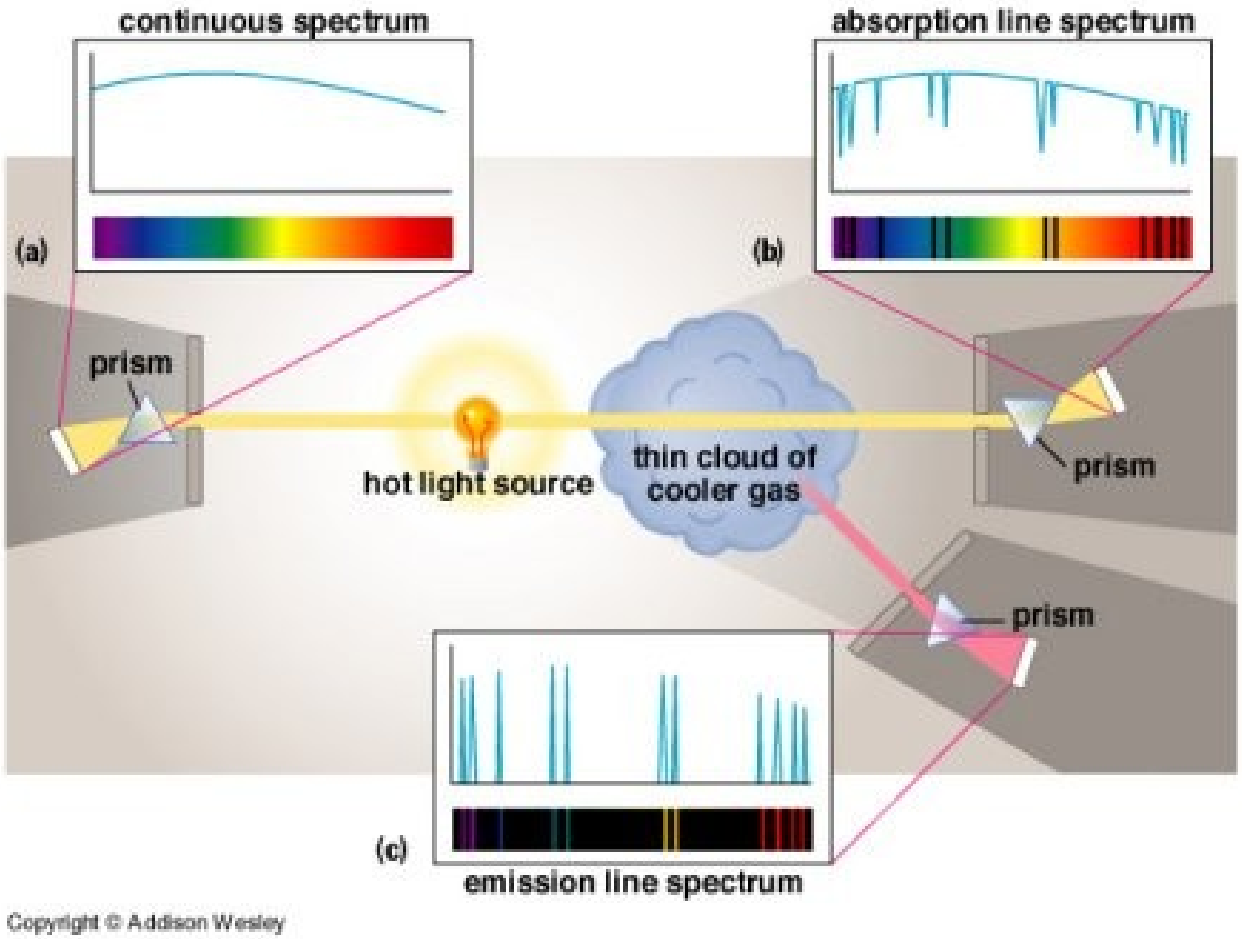
Spectroscopy and chemometrics

- Goal: from sample of unknown gas, identify what molecules are present, at what concentration.
- Use the wavelength-dependent absorption of light to extract this information (spectroscopy).



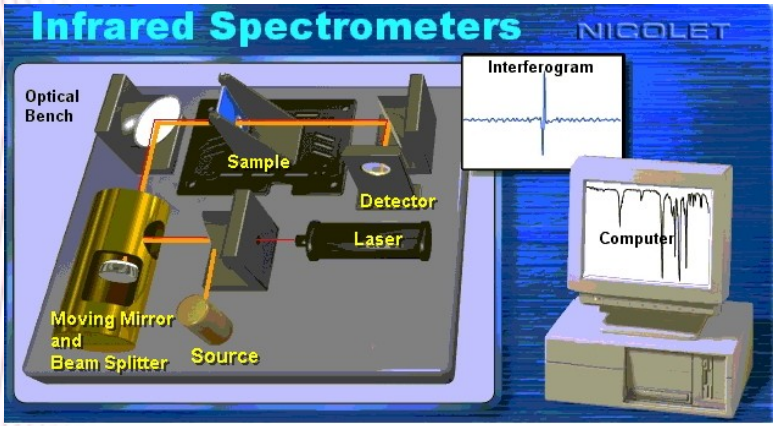
- Chemometrics is the application of math to this problem.

Spectroscopy



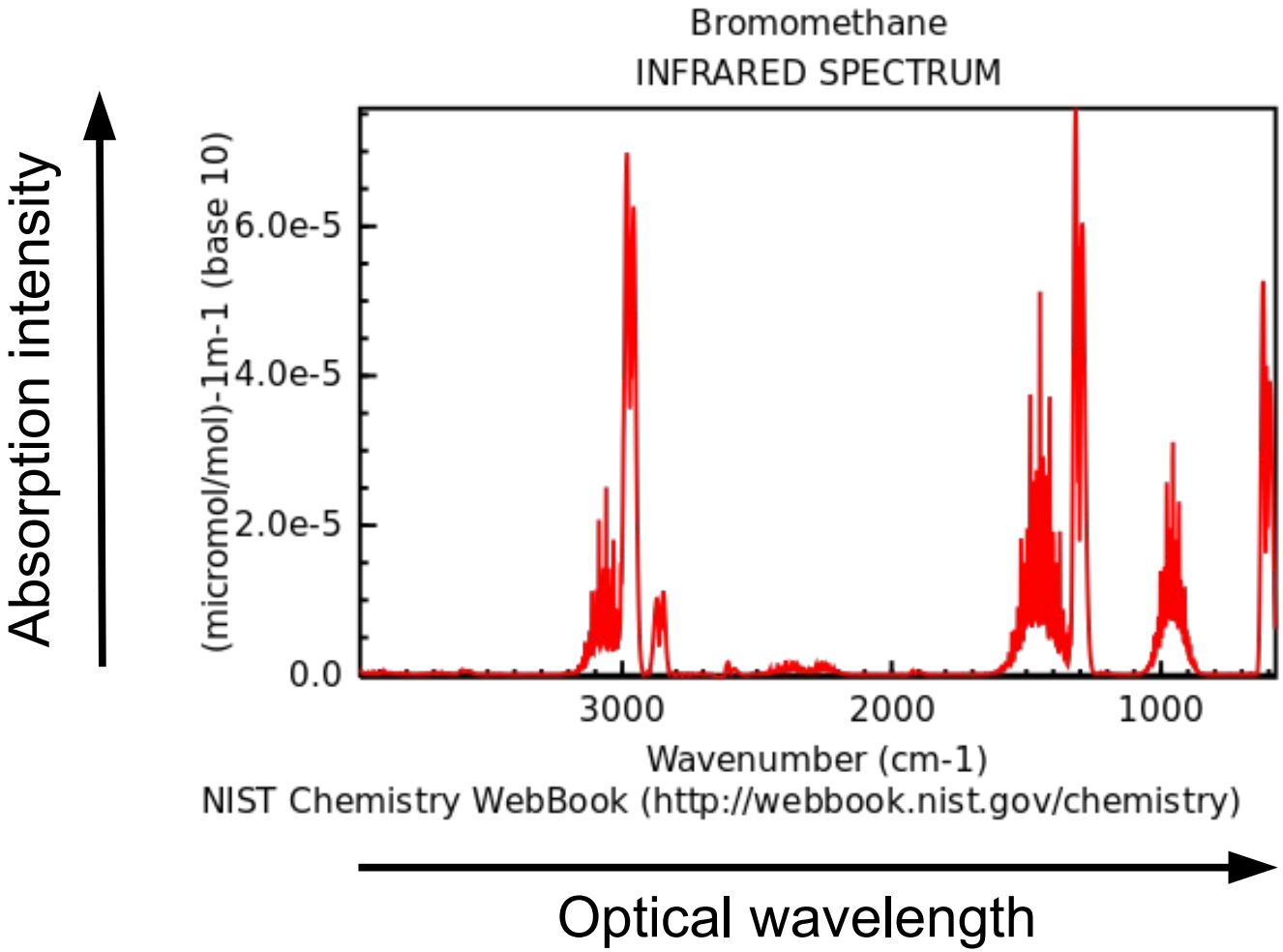
- Measure absorption spectrum using a spectrometer

Some spectrometers



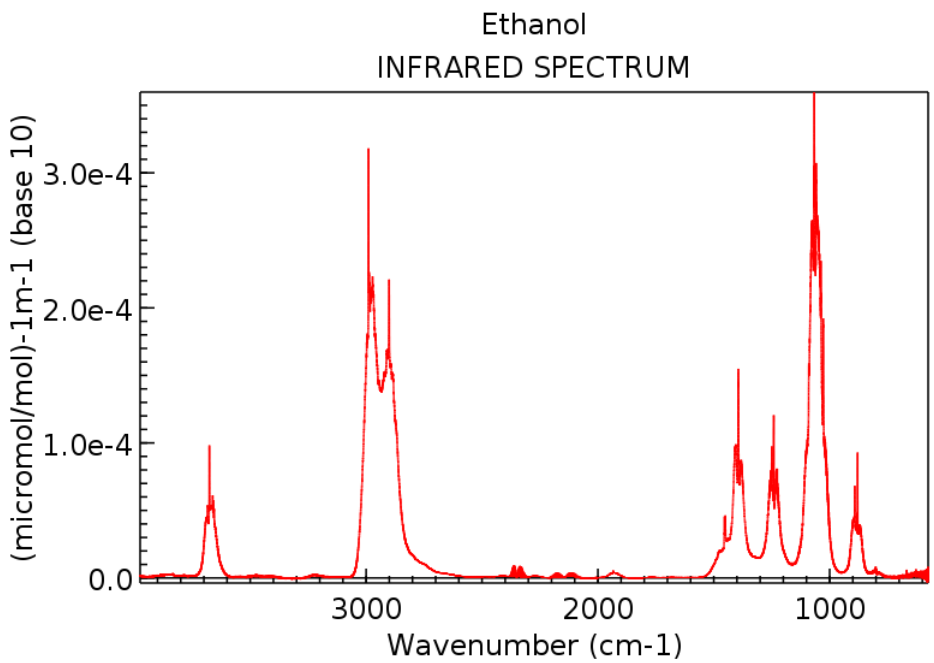
- Usually measure gasses in the infrared spectral region.

Example optical absorption spectrum

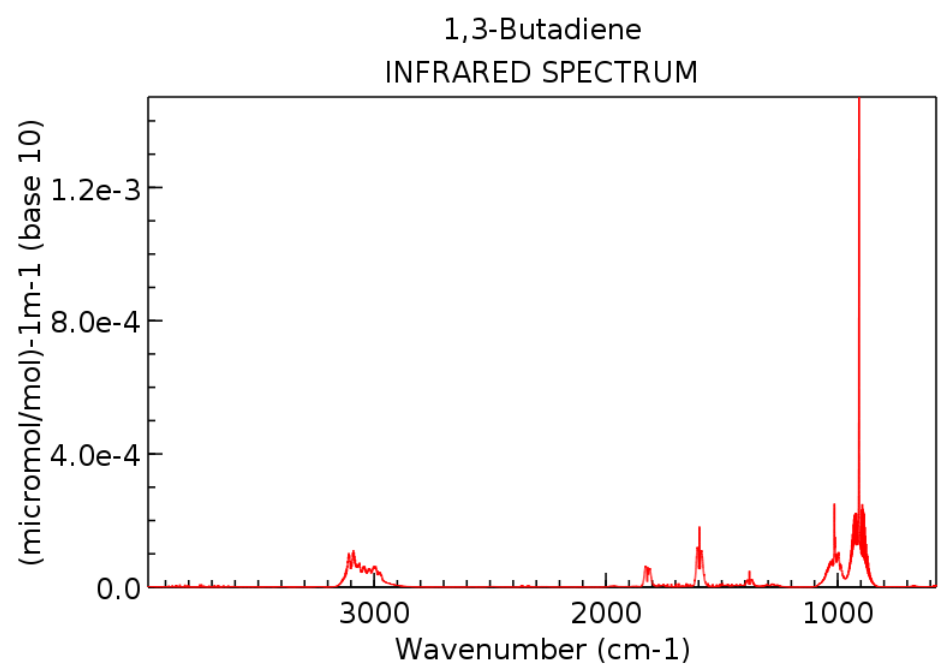


- Different gasses have different spectra.
- Therefore, the absorption spectrum is a good “fingerprint” to identify a gas.

More spectra



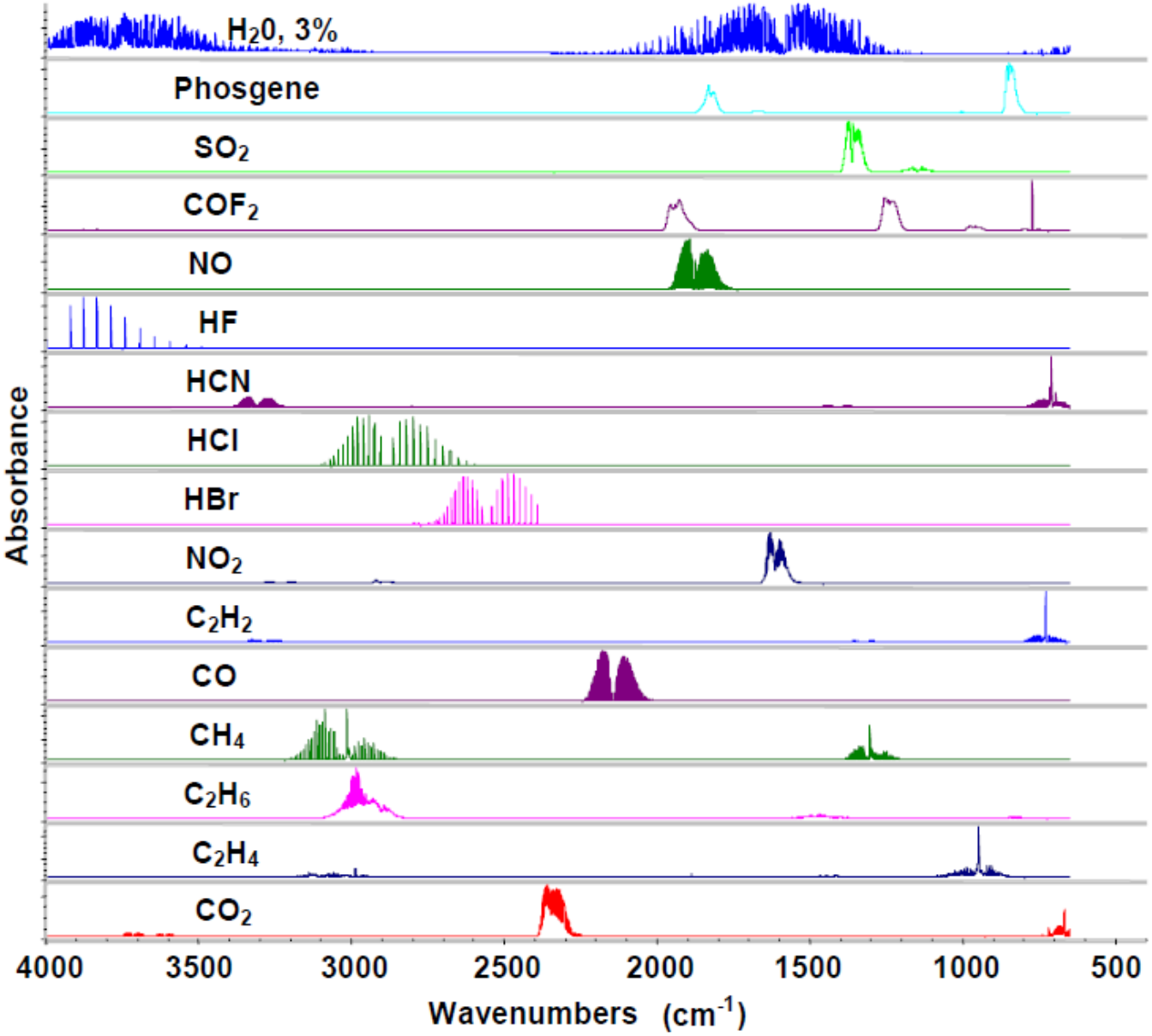
NIST Chemistry WebBook (http://webbook.nist.gov/chemistry)



NIST Chemistry WebBook (http://webbook.nist.gov/chemistry)

- Different gasses have different spectra
- Depth of absorption is proportional to gas concentration. ← Measuring concentrations is frequently the goal of spectroscopy

Combution gasses of interest



Example: our goal

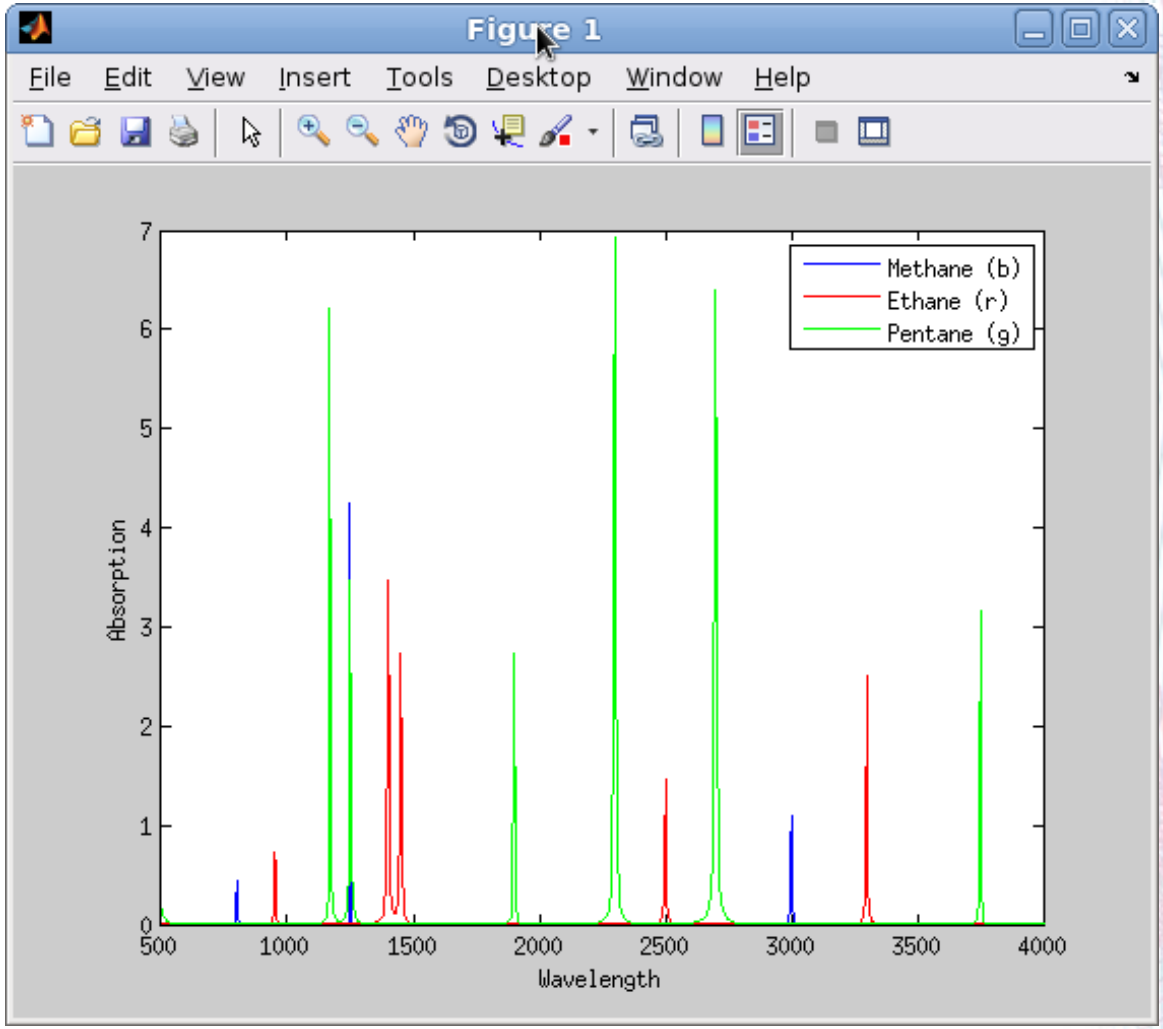
- Measure optical absorption spectrum of gas containing unknown mixture of several species:
  - Methane
  - Ethane
  - Pentane
  - Nitrogen (transparent background – no absorption)
- We know absorption spectra of methane, ethane, pentane at standard concentrations.
- Use multiple linear regression to extract concentration of each gas.

Some remarks

- We have prepared and measured calibrated spectra of methane, ethane, pentane previously (reference spectra).
  - This is entire scientific discipline itself, and has generated large datasets such as HITRAN.
- Measurement of unknown gas is not perfect: spectrum will be corrupted by some noise.
- The goal is to fit the reference spectra to the measured spectrum and extract the concentrations of each reference gas in the unknown gas.

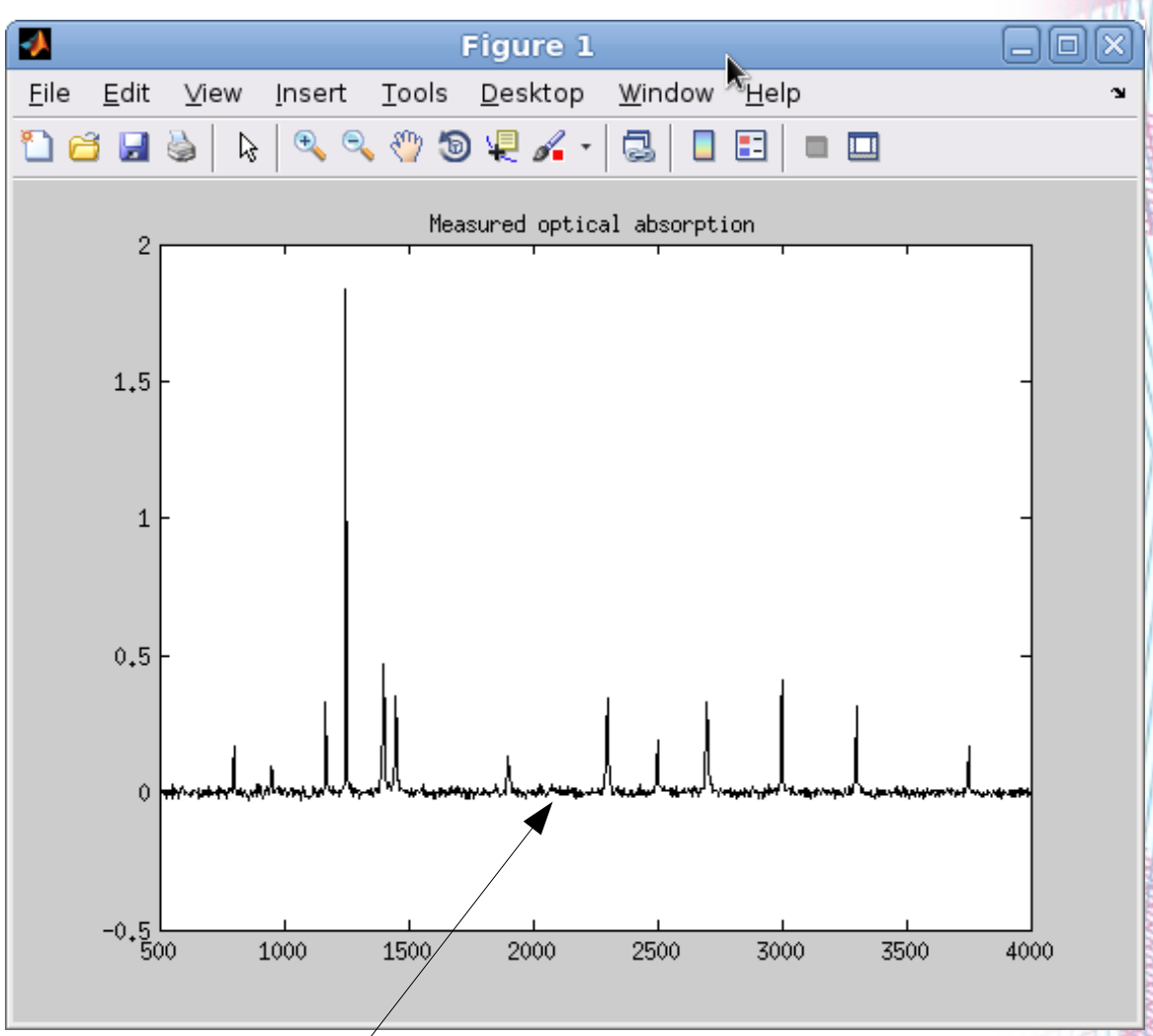
Reference spectra

- Methane
- Ethane
- Pentane
- Data is fake – just for demonstration purposes.



Example measurement

- Spectrum of mixture of methane, ethane, pentane.
- Concentrations are unknowns
- Goal of regression analysis is to extract the concentration of each molecule.



Note measurement is noisy

Mathematical treatment

- Each reference gas has a spectrum:

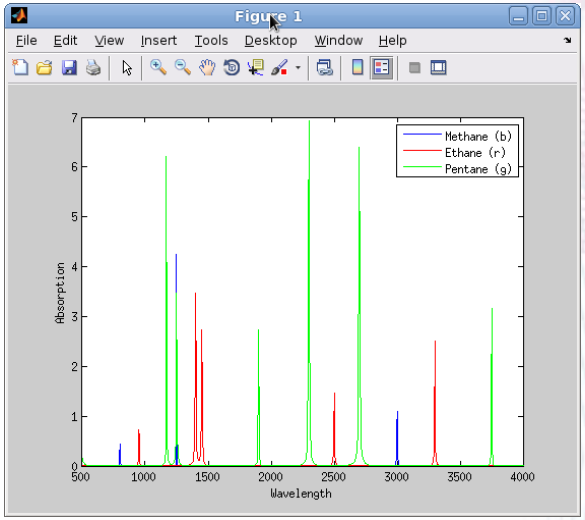
$$R_{meth}(\nu), R_{eth}(\nu), R_{pent}(\nu)$$

  - Spectrum is function of wavelength *v*

$R_{meth}(\nu)$

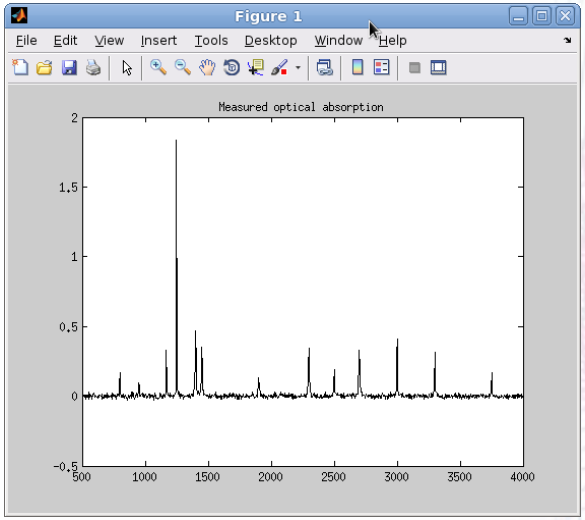
$R_{eth}(\nu)$

$R_{pent}(\nu)$



- Measured gas spectrum is linear combination of reference spectra.
  - Gas concentrations are weight coefficients

$S_{meas}(\nu)$



$$S_{meas}(\nu) = c_{meth} R_{meth}(\nu) + c_{eth} R_{eth}(\nu) + c_{pent} R_{pent}(\nu)$$

How to turn this into multiple linear regression?

- Spectrum and references measured at discrete wavelengths

$$S_{meas}(\nu_i) = c_{meth} R_{meth}(\nu_i) + c_{eth} R_{eth}(\nu_i) + c_{pent} R_{pent}(\nu_i)$$

- This can be written as

$$S^i_{meas} = c_{meth} R^i_{meth} + c_{eth} R^i_{eth} + c_{pent} R^i_{pent}$$

Superscript *i* signifies measurement at i^{th} wavelength – not a power.

- Or:

$$S^1_{meas} = c_{meth} R^1_{meth} + c_{eth} R^1_{eth} + c_{pent} R^1_{pent}$$
$$S^2_{meas} = c_{meth} R^2_{meth} + c_{eth} R^2_{eth} + c_{pent} R^2_{pent}$$
$$S^3_{meas} = c_{meth} R^3_{meth} + c_{eth} R^3_{eth} + c_{pent} R^3_{pent}$$
$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

Use multiple linear regression to get concentrations

- Written in matrix form:

Wavelength dependence runs down columns.

$$\begin{pmatrix} S^1_{meas} \\ S^2_{meas} \\ S^3_{meas} \\ \vdots \end{pmatrix} = \begin{pmatrix} R^1_{meth} & R^1_{eth} & R^1_{pent} \\ R^2_{meth} & R^2_{eth} & R^2_{pent} \\ R^3_{meth} & R^3_{eth} & R^3_{pent} \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} c_{meth} \\ c_{eth} \\ c_{pent} \end{pmatrix} \longleftrightarrow s = R\,c$$

- Use pseudoinverse (normal equations) to get concentrations:

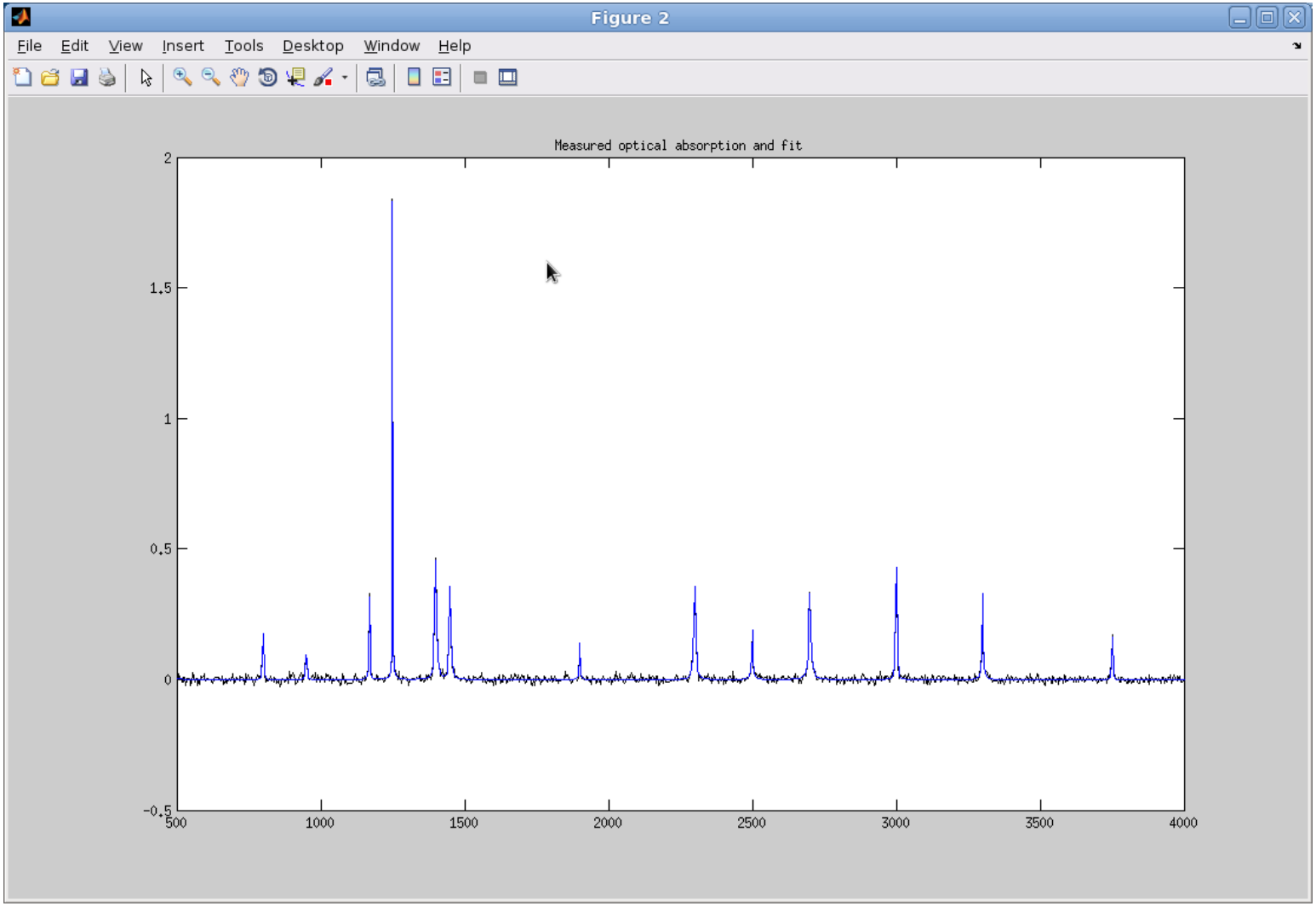
$$c = (R^T R)^{-1} R^T s$$

Matlab code to compute concentrations from spectra

```
function c = fit_spectra(S, R)
  % function performs multiple linear regression fit to input
  % spectrum.  Inputs are:
  % S = spectrum to fit.  Input is vector.
  % R = set of refence spectra.  This input is a matrix.
  %     Different reference species
  %     are held in different colums of the matrix.
  % This fcn computes the concentration of each of the reference
  % species present in the sample spectrum S.  It returns the
  % concentrations as the vector c.

  c = (R'*R)\(R'*S);

end
```

$$c = (R^T R)^{-1} R^T s$$

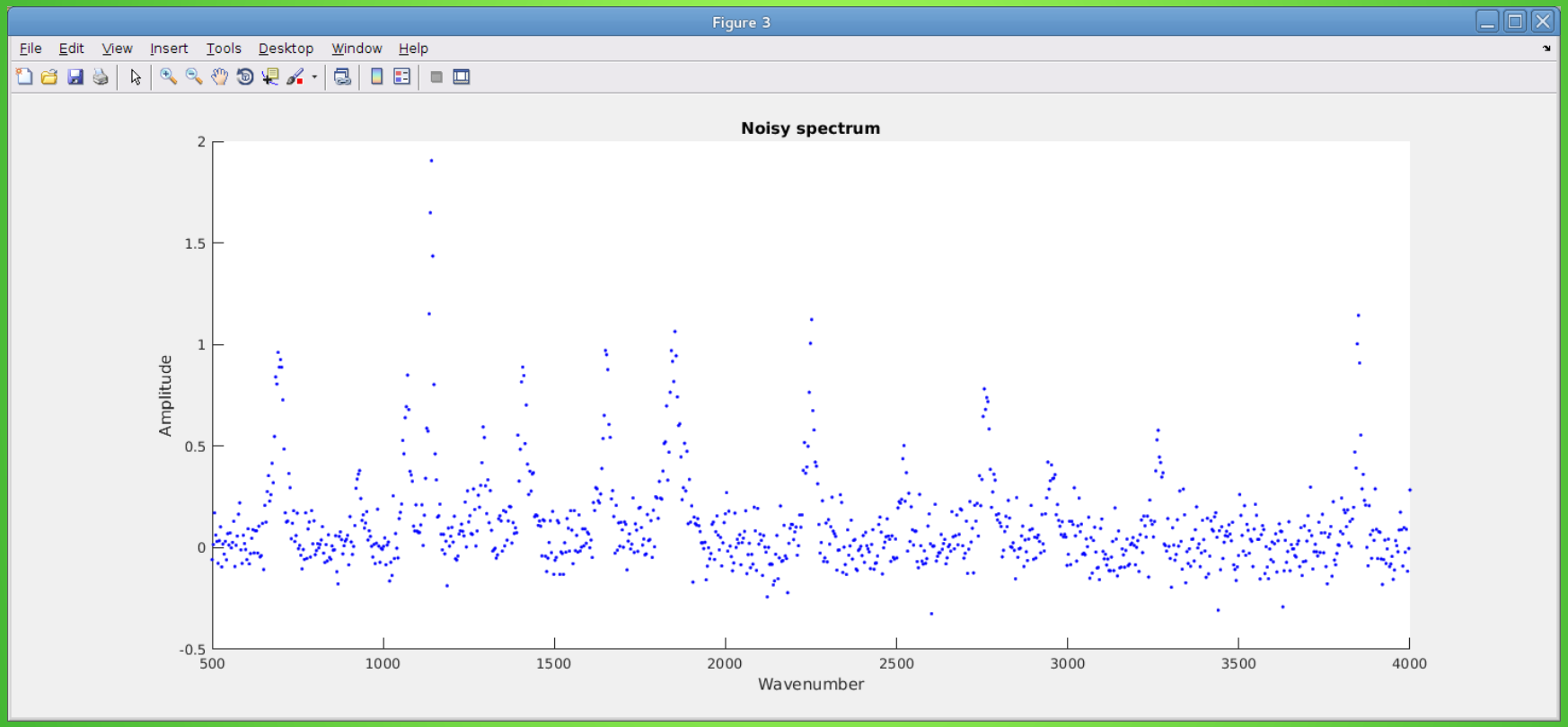
Example of fit absorption spectrum




```
>> test_fit_spectra
Reported concentrations Cmeth = 0.389912, Ceth = 0.131097, Cpent = 0.051258
Actual concentrations   Cmeth = 0.388815, Ceth = 0.132113, Cpent = 0.050529
```

Polynomial regression application: Savitzy-Golay filter

- Problem: Noisy measured spectrum.

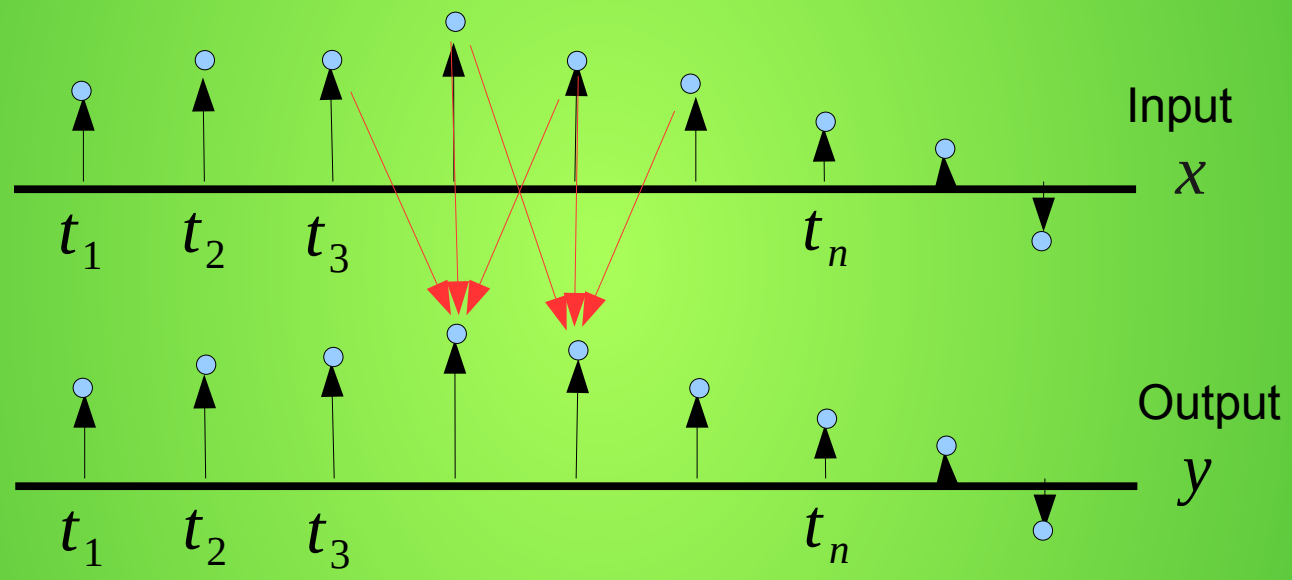


- Goal: Filter the noisy spectrum.

Recall box filter (class 2)

- Moving average filter

Example: Three-point centered average.

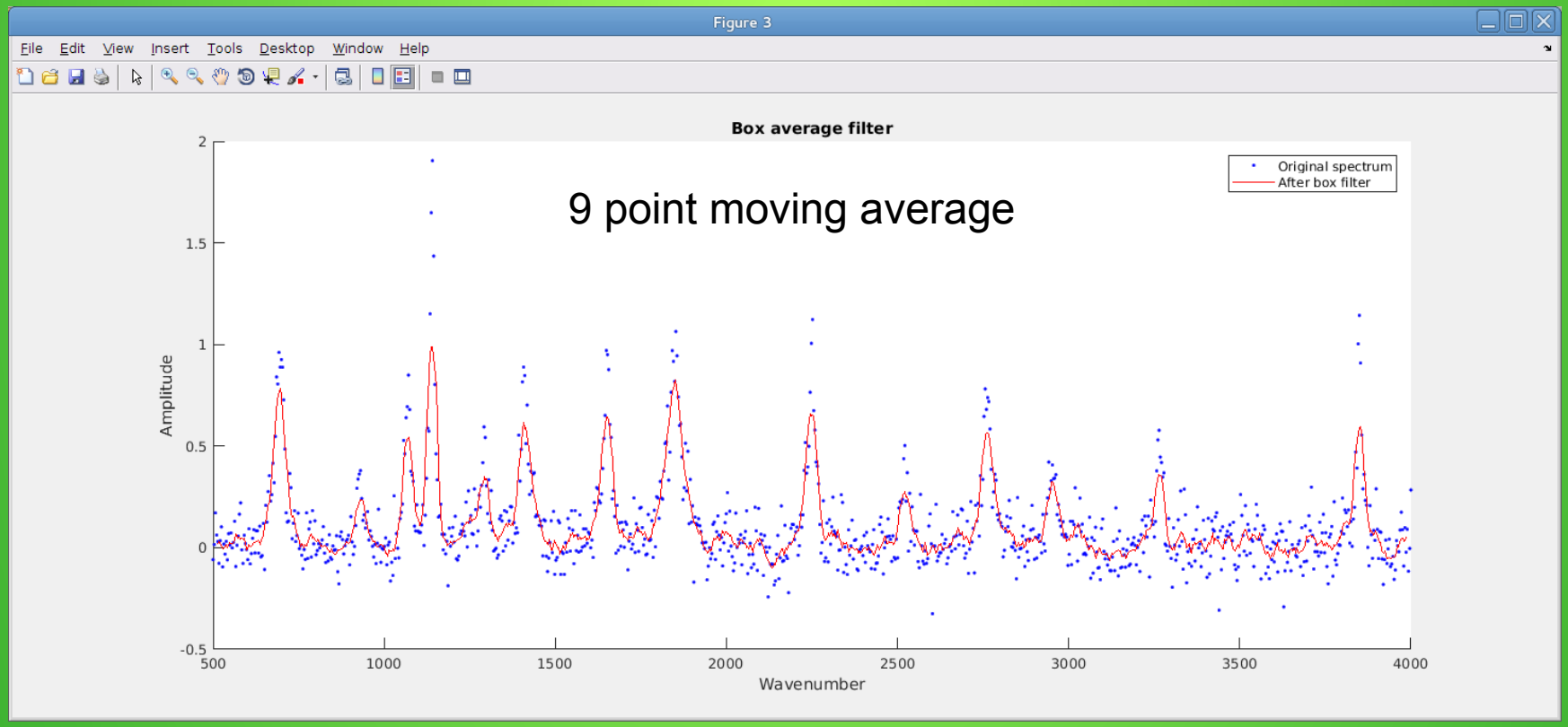


$$y_n = \frac{x_{n-1} + x_n + x_{n+1}}{3}$$

Three-point centered moving average.

Using box filter on noisy spectrum

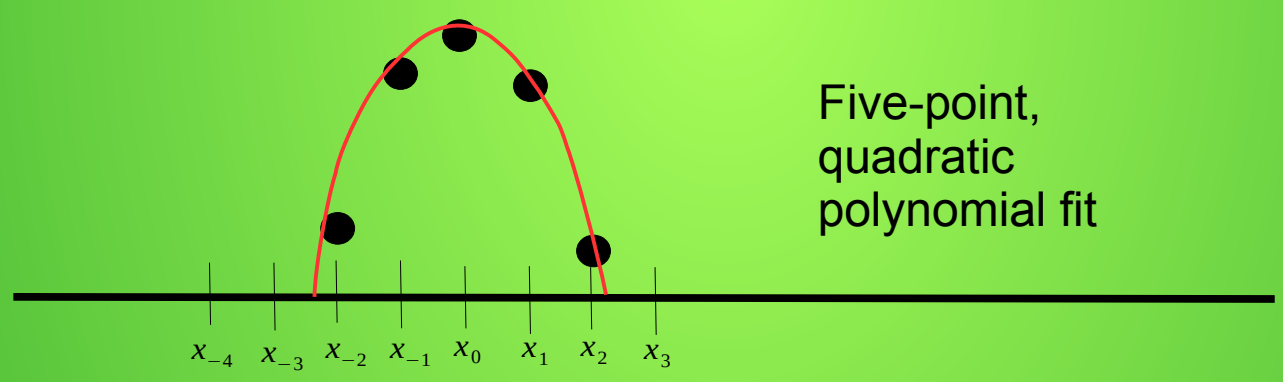
- Data is smoothed.
- But peaks are pushed down.
  - Averaging more points increases smoothing.
  - But averaging more points pushes down on peaks.



Another idea: Moving polynomial fit

- Consider N point moving poly fit (N = odd).
- Fit quadratic or quartic to N points (polynomial regression).

$$y = a_0 + a_1(x - x_0) + a_2(x - x_0)^2$$



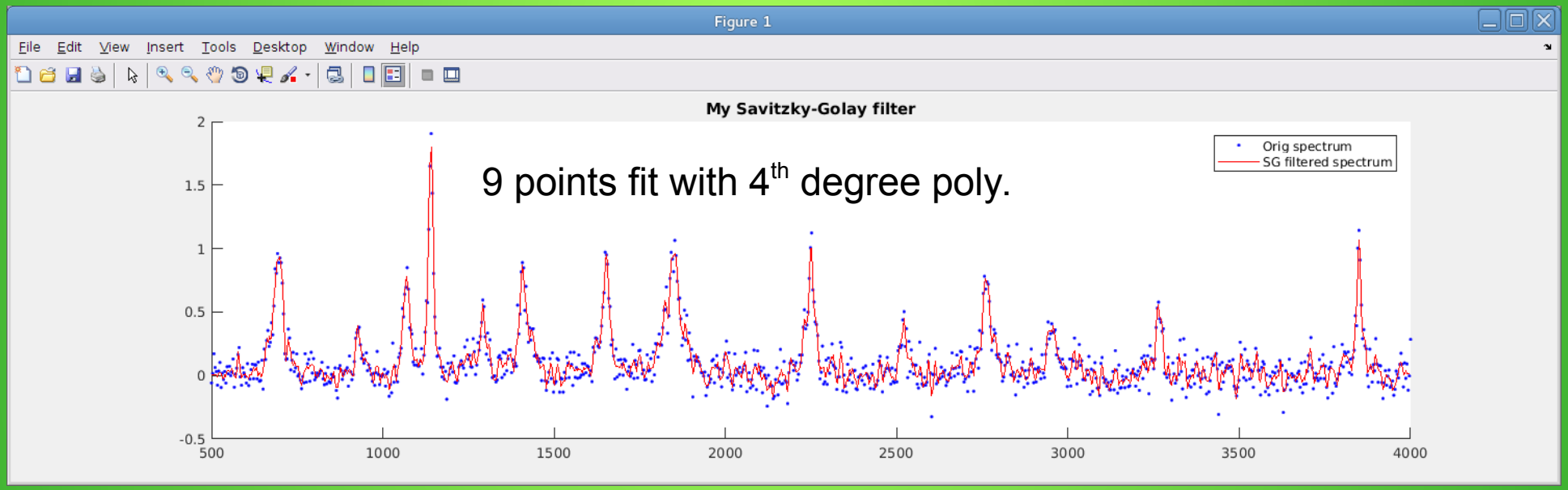
- Then choose $y_n = a_0$ (constant term).

Savitzky-Golay filter

- Sweep through all points.
- Perform N-point polynomial fit at each data point. (Small N, maybe 5, 7, or 9.)
  - Use any of the techniques described earlier, particularly Vandermonde matrix solve.
- Poly degree typically quadratic or quartic.
  - Because you are fitting peaks.
- Desired output value (filtered value) is the constant term of polynomial for each fit.

Savtzky-Golay filter

- 9 points in fit.
- Fit 4th degree polynomial

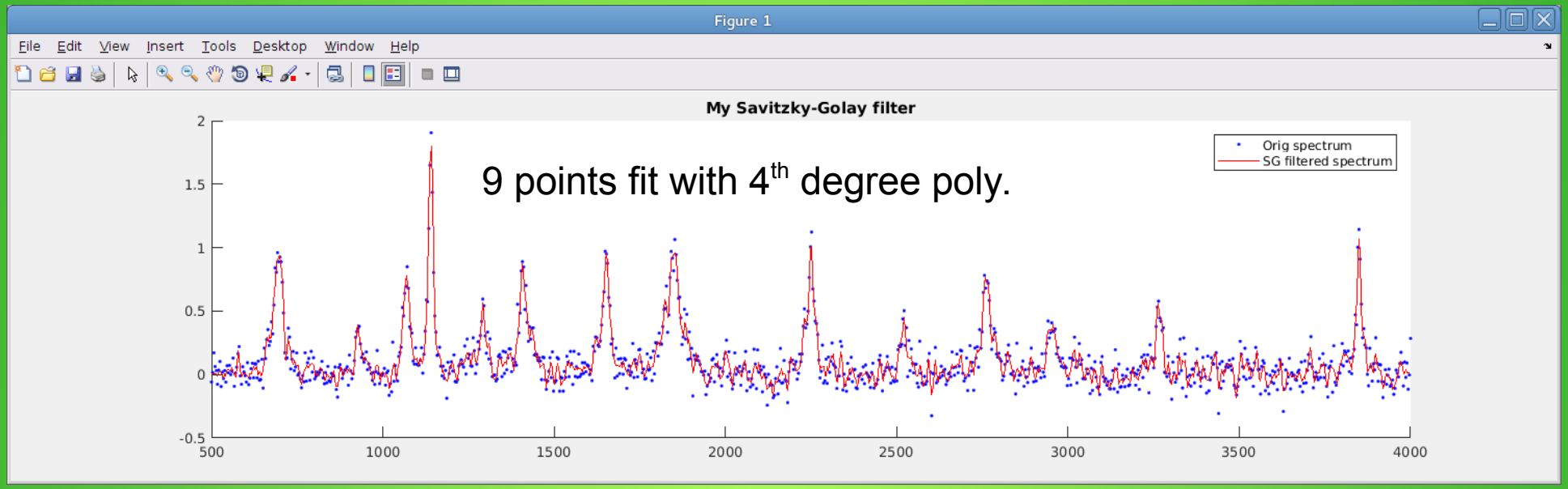


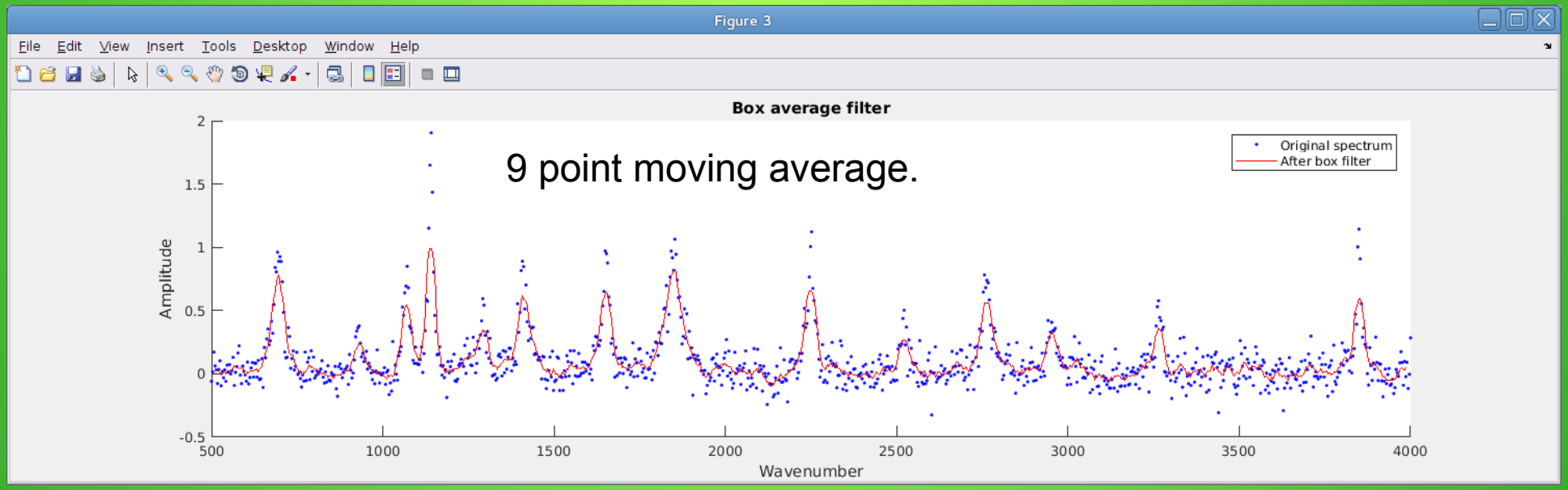
```
function [tf, yf] = savitzky_golay_filter(t, y, Npts, Norder)
  % Compute number of points to the left & right
  Noffset = (Npts-1)/2

  % Initialize output variables
  Nx = length(y);
  yf = zeros(Nx, 1);
  tf = t;

  % Do fits inside sliding window.
  for i = (Noffset+1):(Nx-Noffset)
    start = i-Noffset;
    stop = i+Noffset;
    ysamp = y(start:stop)
    tsamp = t(start:stop)-t(i)  % Center the x axis

    % Fit to Nth degree poly using polyfit
    P = polyfit(tsamp, ysamp, Norder)
    % Constant term is P(end)
    yf(i) = P(end);
  end

  % Deal with ends by padding with last computed value on each side.
  for i = 1:Noffset
    yf(i) = yf(Noffset+1);
    yf(Nx-i+1) = yf(Nx-Noffset-1);
  end

end
```

Matlab polynomial fit built-in function

Filtered output value is 0th degree term in poly fit.

Savitzky-Golay vs. Box Filter

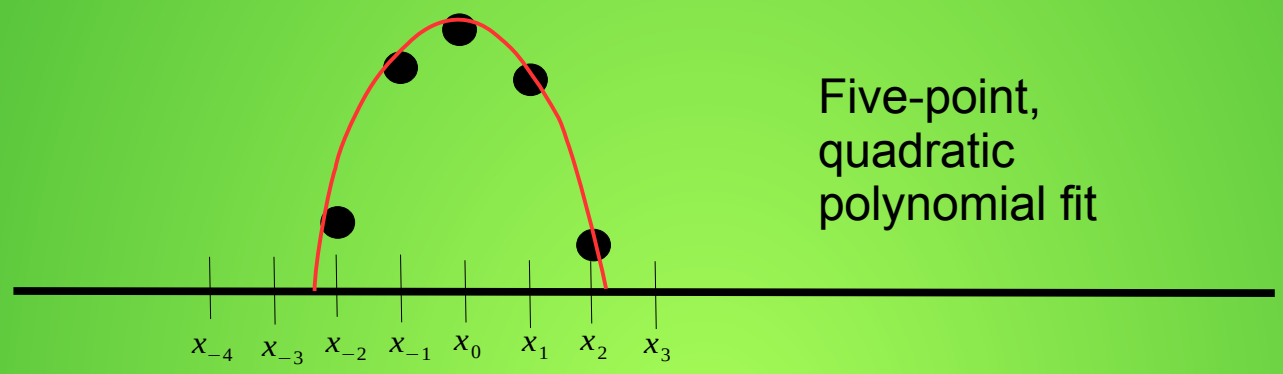




Remarks

- Trade-off between smoothness, peak decrease, and number of points in regression fit.
  - Requires you to “play around” with the algorithm to find best fit for your data.
- Dealing with ends.
  - Set ends to constant value (e.g. last value).
  - Assume input is cyclic and wrap-around poly fit.
  - Mirror the data points at end of domain.
- Repeating polynomial fit at each point is inefficient.
- Since we get all polynomial coefficients at each point, we can use this method to compute derivatives of data.

Consider performing poly fit to equally-spaced data



Five-point, quadratic polynomial fit

- 5 point fit to quadratic:

$$y = a_0 + a_1(x - x_0) + a_2(x - x_0)^2$$

- Vandermonde system to solve for *a* coefficients:

$$\begin{pmatrix} 1 & (x_{-2} - x_0) & (x_{-2} - x_0)^2 \\ 1 & (x_{-1} - x_0) & (x_{-1} - x_0)^2 \\ 1 & 0 & 0 \\ 1 & (x_1 - x_0) & (x_1 - x_0)^2 \\ 1 & (x_2 - x_0) & (x_2 - x_0)^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

- Vandermonde system

$$\begin{pmatrix} 1 & (x_{-2}-x_0) & (x_{-2}-x_0)^2 \\ 1 & (x_{-1}-x_0) & (x_{-1}-x_0)^2 \\ 1 & 0 & 0 \\ 1 & (x_1-x_0) & (x_1-x_0)^2 \\ 1 & (x_2-x_0) & (x_2-x_0)^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

- Sample points are equally spaced:

$$(x_{-2}-x_0)=-2h \qquad (x_{-1}-x_0)=-h \qquad \text{etc.}$$

- Therefore, Vandermonde system becomes:

$$\begin{pmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{pmatrix} \quad \Leftrightarrow \quad X\,a = y$$

Note I am ignoring h for now.

Computing *a* coefficients

- System to solve (non-square): $X a = y$
- Form normal equations: $X^T X a = X^T y$
- Solve for *a* coefficients: $a = (X^T X)^{-1} X^T y$

```
>> X

X =

     1    -2     4
     1    -1     1
     1     0     0
     1     1     1
     1     2     4

>> 35*inv(X'*X)*X'

ans =

   -3.0000   12.0000   17.0000   12.0000   -3.0000
   -7.0000   -3.5000         0    3.5000    7.0000
    5.0000   -2.5000   -5.0000   -2.5000    5.0000
```

The point: For every new set of y values, I can use this matrix to get the a coefficients, including the constant term.

- Computation for *a* coefficients:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \frac{1}{35} \begin{pmatrix} -3 & 12 & 17 & 12 & -3 \\ -7 & -3.5 & 0 & 3.5 & 7 \\ 5 & -2.5 & -5 & -2.5 & 5 \end{pmatrix} \begin{pmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

- Recall Savitzky-Golay filter identifies constant term with filtered value:

$$\tilde{y}_n = a_0$$

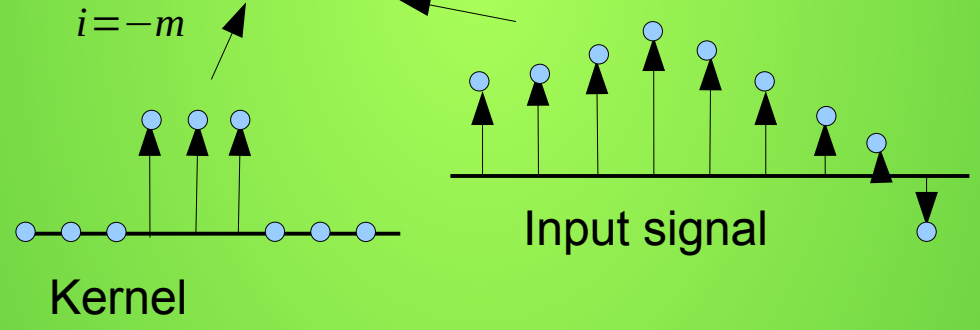
$$= \frac{1}{35}\left(-3\,y_{-2} + 12\,y_{-1} + 17\,y_0 + 12\,y_1 - 3\,y_2\right)$$

Savitzky-Golay is a convolution filter

$$\tilde{y}_n = \frac{1}{35}\left(-3\,y_{-2} + 12\,y_{-1} + 17\,y_0 + 12\,y_1 - 3\,y_2\right)$$

- Recall convolution filters from class 2:

$$\tilde{y}_n = \sum_{i=-m}^{m} w_{n-i}\, y_i$$



- 5 point quadratic SG filter kernel:

$$w = \frac{1}{35}\left(-3, 12, 17, 12, -3\right)$$

Computing derivatives

- Recall computation for *a* coefficients:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \frac{1}{35} \begin{pmatrix} -3 & 12 & 17 & 12 & -3 \\ -7 & -3.5 & 0 & 3.5 & 7 \\ 5 & -2.5 & -5 & -2.5 & 5 \end{pmatrix} \begin{pmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

- Coefficient a_1 is linear term in expansion

$$y = a_0 + a_1 (x - x_0) + a_2 (x - x_0)^2$$

- That means we can also get a 1st derivative
  - ... and 2nd derivative via a_2 term.

Derivatives

- First derivative

$$y'_n = \frac{1}{35}\left(-7\,y_{-2} - 3.5\,y_{-1} + 3.5\,y_1 + 7\,y_2\right)$$

- Second derivative

$$y''_n = \frac{2}{35}\left(5\,y_{-2} - 2.5\,y_{-1} - 5\,y_0 - 2.5\,y_1 + 5\,y_2\right)$$

Last slide:
Regression analysis is a big deal

Testing for Toxic Industrial Chemicals

Monitor the following 25 gases simultaneously -

1. Acrolein – (0.25 ppm)
2. Acrylonitrile - (0.35 ppm)
3. Ammonia - (0.13 ppm)
4. Arsine - (0.02 ppm)
5. Benzene - (0.12 ppm)
6. Boron trichloride – (0.01 ppm)
7. Carbon dioxide – (< 10 ppm)
8. Carbon monoxide – (0.25 ppm)
9. Carbon Disulfide - (0.17 ppm)
10. Dichloromethane – (0.13 ppm)
11. Ethylene oxide - (0.17 ppm)
12. Formaldehyde – (0.09 ppm)
13. Hydrogen chloride – (0.20 ppm)
14. Hydrogen cyanide – (0.35 ppm)
15. Hydrogen fluoride – (0.30 ppm)
16. Methane - (0.06 ppm)
17. Methyl Mercaptan – (0.42 ppm)
18. Nitrogen dioxide – (0.37 ppm)
19. Nitrous oxide – (0.02 ppm)
20. Phosgene – (0.02 ppm)
21. Phosphine – (0.20 ppm)
22. Sulfur dioxide – (0.03 ppm)
23. Sulfuryl fluoride – (0.03 ppm)
24. Toluene – (0.13 ppm)
25. Water Vapour

A. Dual Functionality of FTIR
Protecting health

Session summary

- Polynomial regression
  - Polynomial regression works because the problem is linear in the coefficients.
  - Use low-order polys. Don't overfit.
- Applications in chemometrics:
  - Multivariate linear regression to determine species concentration in optical spectra.
  - Savitzky-Golay filtering of optical spectra for noise reduction.