Math 541 - Numerical Analysis

Approximation Theory
Discrete Least Squares Approximation

Joseph M. Mahaffy,
⟨jmahaffy@mail.sdsu.edu⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

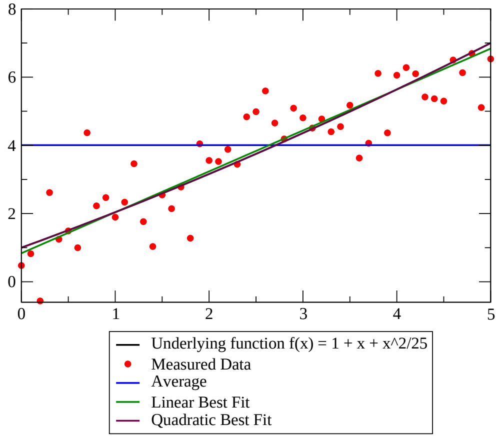
http://jmahaffy.sdsu.edu

Spring 2018

Outline

Introduction: Matching a Few Parameters to a Lot of Data.

Sometimes we get a lot of data, ***many observations***, and want to fit it to a simple model.



PDF link: code

Why a Low Dimensional Model?

Low dimensional models (*e.g.* low degree polynomials) are **easy to work with**, and are quite **well behaved** (high degree polynomials can be quite oscillatory.)

All measurements are **noisy**, to some degree. Often, we want to use a large number of measurements in order to "average out" random noise.

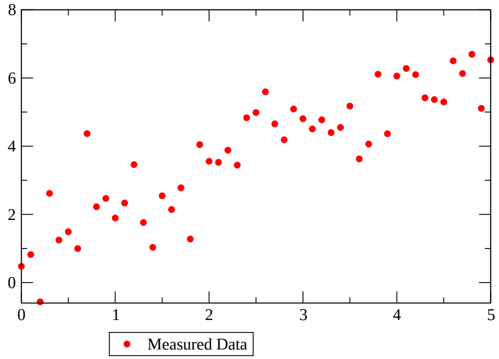
Approximation Theory looks at two problems:

[1] Given a data set, find the best fit for a model (*i.e.* in a class of functions, find the one that best represents the data.)

[2] Find a simpler model approximating a given function.

Interpolation: A Bad Idea?

We can probably agree that trying to interpolate this data set:



with a 50th degree polynomial is not the best idea in the world...
Even fitting a cubic spline to this data gives wild oscillations!
[*I tried, and it was not pretty!*]

Defining "Best Fit" — the Residual.

We are going to ***relax*** the requirement that the approximating function must pass through all the data points.

Now we need a measurement of how well our approximation fits the data. — A definition of "best fit."

If $f(x_i)$ are the measured function values, and $a(x_i)$ are the values of our approximating functions, we can define a function, $r(x_i) = f(x_i) - a(x_i)$ which measures the deviation **(residual)** at x_i. Notice that $\tilde{\mathbf{r}} = \{r(x_0), r(x_1), \dots, r(x_n)\}^T$ is a vector.

Notation: From now on, $f_i = f(x_i)$, $a_i = a(x_i)$, and $r_i = r(x_i)$. Further, $\tilde{\mathbf{f}} = \{f_0, f_1, \dots, f_n\}^T$, $\tilde{\mathbf{a}} = \{a_0, a_1, \dots, a_n\}^T$, and $\tilde{\mathbf{r}} = \{r_0, r_1, \dots, r_n\}^T$.

What is the Size of the Residual?

There are many possible choices, *e.g.*

- The abs-sum of the deviations:

$$E_1 = \sum_{i=0}^{n} |r_i| \quad \Leftrightarrow \quad E_1 = \|\tilde{\mathbf{r}}\|_1$$

- The sum-of-the-squares of the deviations:

$$E_2 = \sqrt{\sum_{i=0}^{n} |r_i|^2} \quad \Leftrightarrow \quad E_2 = \|\tilde{\mathbf{r}}\|_2$$

- The largest of the deviations:

$$E_\infty = \max_{0 \le i \le n} |r_i| \quad \Leftrightarrow \quad E_\infty = \|\tilde{\mathbf{r}}\|_\infty$$

In most cases, the **sum-of-the-squares** version is the easiest to work

Discrete Least Squares Approximation

We have chosen the sum-of-squares measurement for errors. Lets find the **constant** that best fits the data, *minimize*

$$E(C) = \sum_{i=0}^{n} (f_i - C)^2.$$

If C^* is a minimizer, then $E'(C^*) = 0$ [**derivative at a max/min is zero**]

$$E'(C) = -\sum_{i=0}^{n} 2(f_i - C) = \underbrace{-2\sum_{i=0}^{n} f_i + 2(n+1)C,}_{\textbf{Set =0, and solve for C}} \quad E''(C) = \underbrace{2(n+1)}_{\textbf{Positive}}$$

hence

$$\mathbf{C^* = \frac{1}{n+1}\sum_{i=0}^{n} f_i}, \quad \text{it is a min since } E''(C^*) = 2(n+1) > 0.$$

is the constant that best the fits the data. (***Note:*** C^* is the average.)

Discrete Least Squares: Linear Approximation.

The form of Least Squares you are most likely to see: **Find the Linear Function, $p_1(x) = a_0 + a_1 x$, that best fits the data.** The error $E(a_0, a_1)$ we need to minimize is:

$$E(a_0, a_1) = \sum_{i=0}^{n} \left[(a_0 + a_1 x_i) - f_i\right]^2 .$$

The first partial derivatives with respect to a_0 and a_1 better be zero at the minimum:

$$\begin{aligned} \frac{\partial}{\partial a_0} E(a_0, a_1) &= 2\sum_{i=0}^{n} \left[(a_0 + a_1 x_i) - f_i\right] &= 0 \\ \frac{\partial}{\partial a_1} E(a_0, a_1) &= 2\sum_{i=0}^{n} x_i \left[(a_0 + a_1 x_i) - f_i\right] &= 0. \end{aligned}$$

We "massage" these expressions to get the **Normal Equations**...

Linear Approximation: The Normal Equations $p_1(x)$

$$\begin{cases} \mathbf{a_0}(n+1) + \mathbf{a_1}\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} f_i \\ \mathbf{a_0}\sum_{i=0}^{n} x_i + \mathbf{a_1}\sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} x_i f_i. \end{cases}$$

Since **everything except** $\mathbf{a_0}$ and $\mathbf{a_1}$ is known, this is a 2-by-2 system of equations.

$$\begin{bmatrix} (n+1) & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} f_i \\ \sum_{i=0}^{n} x_i f_i \end{bmatrix}.$$

Alternate Linear Least Squares

For the **Linear Least Squares model**, $p_1(x) = a_0 + a_1 x$, where the error

$$E(a_0, a_1) = \sum_{i=0}^{n} \left[(a_0 + a_1 x_i) - f_i\right]^2$$

is minimized for data set $(x_i, f_i), i = 0, ..., n$, there is an easy formulation without matrices often stated in Statistics texts

Define the averages

$$\bar{x} = \frac{1}{n+1} \sum_{i=0}^{n} x_i \qquad \text{and} \qquad \bar{f} = \frac{1}{n+1} \sum_{i=0}^{n} f_i.$$

The best fitting slope and intercept are

$$a_1 = \frac{\sum_{i=0}^{n}(x_i - \bar{x}) f_i}{\sum_{i=0}^{n}(x_i - \bar{x})^2} \qquad \text{and} \qquad a_0 = \bar{f} - a_1 \bar{x}.$$

Quadratic Model, $p_2(x)$

For the quadratic polynomial $p_2(x) = a_0 + a_1 x + a_2 x^2$, the error is given by

$$E(a_0, a_1, a_2) = \sum_{i=0}^{n} \left[a_0 + a_1 x_i + a_2 x_i^2 - f_i \right]^2$$

At the minimum (best model) we must have

$$\begin{aligned}
\frac{\partial}{\partial a_0} E(a_0, a_1, a_2) &= 2 \sum_{i=0}^{n} \left[(a_0 + a_1 x_i + a_2 x_i^2) - f_i \right] &= 0 \\
\frac{\partial}{\partial a_1} E(a_0, a_1, a_2) &= 2 \sum_{i=0}^{n} x_i \left[(a_0 + a_1 x_i + a_2 x_i^2) - f_i \right] &= 0 \\
\frac{\partial}{\partial a_2} E(a_0, a_1, a_2) &= 2 \sum_{i=0}^{n} x_i^2 \left[(a_0 + a_1 x_i + a_2 x_i^2) - f_i \right] &= 0.
\end{aligned}$$

Quadratic Model: The Normal Equations $p_2(x)$

Similarly for the quadratic polynomial $p_2(x) = a_0 + a_1 x + a_2 x^2$, the normal equations are:

$$\left\{\begin{array}{lllllll} \mathbf{a_0}(n+1) & + & \mathbf{a_1}\sum_{i=0}^{n} x_i & + & \mathbf{a_2}\sum_{i=0}^{n} x_i^2 & = & \sum_{i=0}^{n} f_i \\ \mathbf{a_0}\sum_{i=0}^{n} x_i & + & \mathbf{a_1}\sum_{i=0}^{n} x_i^2 & + & \mathbf{a_2}\sum_{i=0}^{n} x_i^3 & = & \sum_{i=0}^{n} x_i f_i. \\ \mathbf{a_0}\sum_{i=0}^{n} x_i^2 & + & \mathbf{a_1}\sum_{i=0}^{n} x_i^3 & + & \mathbf{a_2}\sum_{i=0}^{n} x_i^4 & = & \sum_{i=0}^{n} x_i^2 f_i. \end{array}\right.$$

Note: Even though the model is quadratic, the resulting (normal) equations are **linear**. — The model is linear in its parameters, a_0, a_1, and a_2.

The Normal Equations — As Matrix Equations.

We rewrite the Normal Equations as:

$$\begin{bmatrix} (n+1) & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 \\ \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} f_i \\ \sum_{i=0}^{n} x_i f_i. \\ \sum_{i=0}^{n} x_i^2 f_i. \end{bmatrix}.$$

It is not immediately obvious, but this expression can be written in the form $\mathbf{A^T A \tilde{a} = A^T \tilde{f}}$. Where the matrix A is very easy to write in terms of x_i. [Jump Forward].

The Polynomial Equations in Matrix Form $p_m(x)$

We can express the mth degree polynomial, $p_m(x)$, evaluated at the points x_i:

$$a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m = f_i, \quad i = 0, \dots, n$$

as a product of an $(n+1)$-by-$(m+1)$ matrix, A and the $(m+1)$-by-1 vector $\tilde{\mathbf{a}}$ and the result is the $(n+1)$-by-1 vector $\tilde{\mathbf{f}}$, usually $n \gg m$:

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^m \\ 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ 1 & x_3 & x_3^2 & \cdots & x_3^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}}_{\tilde{\mathbf{a}}} = \underbrace{\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix}}_{\tilde{\mathbf{f}}}.$$

Building a Solvable System from $A\tilde{\mathbf{a}} = \tilde{\mathbf{f}}$

We cannot immediately solve the linear system

$$A\tilde{\mathbf{a}} = \tilde{\mathbf{f}}$$

when A is a rectangular matrix $(n+1)$-by-$(m+1)$, $m \neq n$.

We can generate a solvable system by multiplying both the left- and right-hand-side by A^T, *i.e.*

$$\mathbf{A^T A}\tilde{\mathbf{a}} = \mathbf{A^T}\tilde{\mathbf{f}}$$

Now, the matrix $A^T A$ is a square $(m+1)$-by-$(m+1)$ matrix, and $A^T\tilde{\mathbf{f}}$ an $(m+1)$-by-1 vector.

Let's take a closer look at $A^T A$, and $A^T\tilde{\mathbf{f}}$...

Computing $A^T A$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ x_0 & x_1 & x_2 & x_3 & \dots & x_n \\ x_0^2 & x_1^2 & x_2^2 & x_3^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^m & x_1^m & x_2^m & x_3^m & \dots & x_n^m \end{bmatrix} \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^m \\ 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ 1 & x_3 & x_3^2 & \cdots & x_3^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}$$

$$= \begin{bmatrix} n+1 & \sum_{i=0}^n x_i^1 & \cdots & \sum_{i=0}^n x_i^m \\ \sum_{i=0}^n x_i^1 & \sum_{i=0}^n x_i^2 & \cdots & \sum_{i=0}^n x_i^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^n x_i^m & \sum_{i=0}^n x_i^{m+1} & \cdots & \sum_{i=0}^n x_i^{2m} \end{bmatrix}.$$

Computing $A^T f$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ x_0 & x_1 & x_2 & x_3 & \dots & x_n \\ x_0^2 & x_1^2 & x_2^2 & x_3^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^m & x_1^m & x_2^m & x_3^m & \dots & x_n^m \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^n f_i \\ \sum_{i=0}^n x_i f_i \\ \sum_{i=0}^n x_i^2 f_i \\ \vdots \\ \sum_{i=0}^n x_i^m f_i. \end{bmatrix}$$

We have recovered the Normal Equations...

[Jump Back].

Discrete Least Squares: A Simple, Powerful Method.

Given the data set $(\tilde{\mathbf{x}}, \tilde{\mathbf{f}})$, where $\tilde{\mathbf{x}} = \{x_0, x_1, \dots, x_n\}^T$ and $\tilde{\mathbf{f}} = \{f_0, f_1, \dots, f_n\}^T$, we can quickly find the best polynomial fit for **any** specified polynomial degree!

Notation: Let $\tilde{\mathbf{x}}^j$ be the vector $\{x_0^j, x_1^j, \dots, x_n^j\}^T$.

E.g. to compute the best fitting polynomial of degree 3, $p_3(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, define:

$$A = \begin{bmatrix} | & | & | & | \\ \tilde{\mathbf{1}} & \tilde{\mathbf{x}} & \tilde{\mathbf{x}}^2 & \tilde{\mathbf{x}}^3 \\ | & | & | & | \end{bmatrix}, \quad \text{and compute} \quad \underbrace{\tilde{\mathbf{a}} = (A^TA)^{-1}(A^T\tilde{\mathbf{f}}).}_{\text{not necessarily like this}}$$

Discrete Least Squares: Matlab Example.

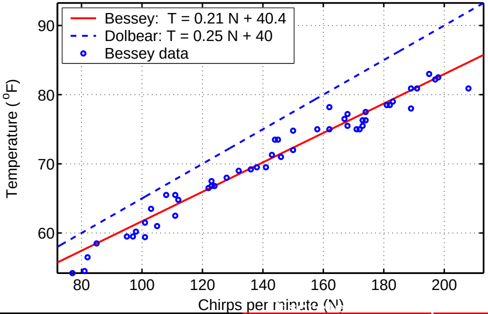
I used this code to generate the data for the plots on slide 2.

```
x  = (0:0.1:5)';              % The x-vector
f  = 1+x+x.^2/25;             % The underlying function
n  = randn(size(x));          % Random perturbations
fn = f+n;                     % Add randomness
A  = [x ones(size(x))];       % Build A for linear fit
%a  = (A'*A)\(A'*fn);         % Solve:  Using Normal Eqns.
a  = A\fn;                    % Solve:  Better, Equivalent
p1 = polyval(a,x);            % Evaluate
A  = [x.^2 x ones(size(x))];  % A for quadratic fit
%a  = (A'*A)\(A'*fn);         % Solve:  Using Normal Eqns.
a  = A\fn;                    % Solve:  Better, Equivalent
p2 = polyval(a,x);            % Evaluate
```

Cricket Thermometer Application Example source: Bessey and Bessey, 1897

There is a folk method of approximating the temperature (in Fahrenheit). This entered the scientific literature in 1896 by Dolbear with data collected by the Bessey brothers in 1898.

The temperature is approximated from the rate of crickets chirping by taking the number of chirps/min dividing by 4 and adding 40.



Cricket Data Analysis

C. A. Bessey and E. A. Bessey collected data on eight different crickets that they observed in Lincoln, Nebraska during August and September, 1897. The number of chirps/min was N and the temperature was T.

Create matrices

$$A_1 = \begin{pmatrix} 1 & N_1 \\ 1 & N_2 \\ \vdots & \vdots \end{pmatrix} \qquad A_2 = \begin{pmatrix} 1 & N_1 & N_1^2 \\ 1 & N_2 & N_2^2 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 1 & N_1 & N_1^2 & N_1^3 \\ 1 & N_2 & N_2^2 & N_2^3 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \qquad A_4 = \begin{pmatrix} 1 & N_1 & N_1^2 & N_1^3 & N_1^4 \\ 1 & N_2 & N_2^2 & N_2^3 & N_2^4 \\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

Cricket A_n Matrices

How do we efficiently create the A_n matrices from the previous slide?

The data for the number of chirps/min stored as a vector,

$$N = [N_1, N_2, ..., N_m]^T,$$

so we use the MatLab function below with $x = N$ and n entered as the degree of the polynomial fit desired

```
function A = vanA(x,n)
%Least Squares Matrix for x and n poly
A = [ones(length(x),1)];
for i = 1:n
    A = [A,x.^i];
end
end
```

The output forms the matrices on the previous slide

Cricket Linear Model

If you compute the matrix *which you never should!*

$$A_1^T A_1 = \begin{pmatrix} 52 & 7447 \\ 7447 & 1133259 \end{pmatrix},$$

it has eigenvalues

$$\lambda_1 = 3.0633 \quad \text{and} \quad \lambda_2 = 1,133,308,$$

which gives the condition number

$$\texttt{cond}(A_1^T A_1) = \frac{\lambda_2}{\lambda_1} = 3.6996 \times 10^5.$$

Whereas

$$\texttt{cond}(A_1) = 608.2462.$$

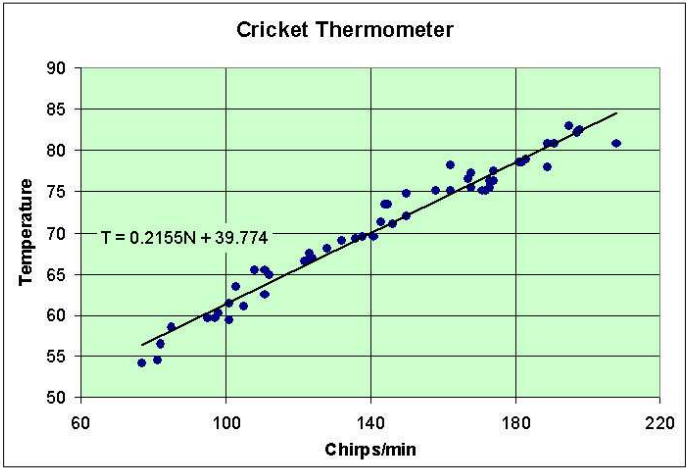
In Matlab

```
A₁\T
```

gives the parameters for best linear model

$$T_1(N) = 0.2155\, N + 39.7441.$$

Polynomial Fits to the Data: Linear



Linear Fit

Cricket Quadratic Model

Similarly, the matrix

$$A_2^TA_2 = \begin{pmatrix} 52 & 7447 & 1133259 \\ 7447 & 1133259 & 1.8113\times 10^8 \\ 1133259 & 1.8113\times 10^8 & 3.0084\times 10^{10} \end{pmatrix},$$

has eigenvalues

$$\lambda_1 = 0.1957, \quad \lambda_2 = 42,706, \quad \lambda_3 = 3.00853\times 10^{10}$$

which gives the condition number

$$\texttt{cond}(A_2^TA_2) = \frac{\lambda_3}{\lambda_1} = 1.5371\times 10^{11}.$$

Whereas,

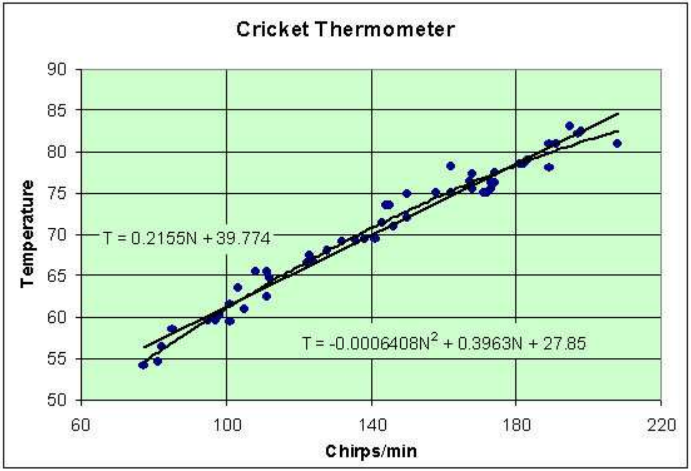
$$\texttt{cond}(A_2) = 3.9206\times 10^5,$$

and

$$\texttt{A}_2\backslash\texttt{T},$$

gives the parameters for best quadratic model

Polynomial Fits to the Data: Quadratic



Quadratic Fit

Cricket Cubic and Quartic Models

The condition numbers for the cubic and quartic rapidly get larger with

$$\texttt{cond}(A_3^T A_3) = 6.3648 \times 10^{16} \quad \text{and} \quad \texttt{cond}(A_4^T A_4) = 1.1218 \times 10^{23}$$

These last two condition numbers suggest that any coefficients obtained are highly suspect.

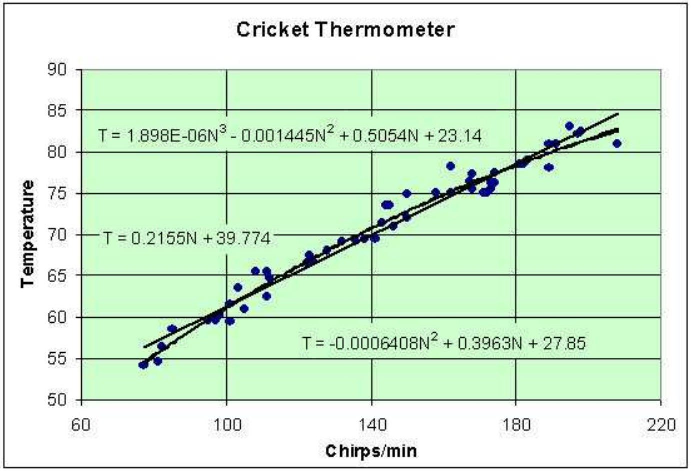
However, if done right, we are "only" subject to the condition numbers

$$\texttt{cond}(A_3) = 2.522 \times 10^{8}, \qquad \texttt{cond}(A_4) = 1.738 \times 10^{11}.$$

The best cubic and quartic models are given by

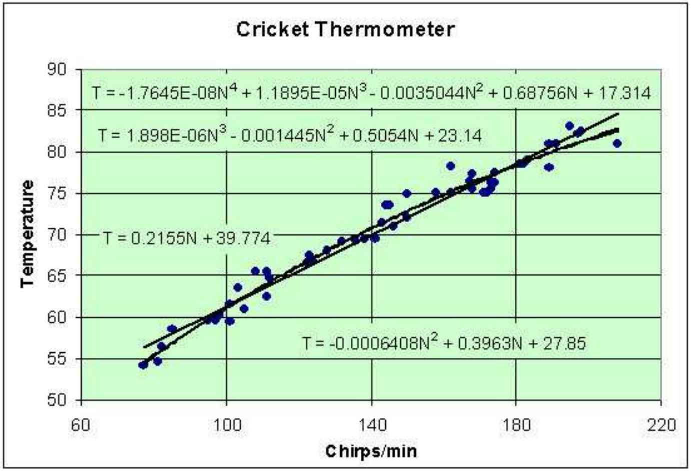
$$\begin{aligned} T_3(N) &= 0.0000018977\,N^3 - 0.001445\,N^2 + 0.50540\,N + 23.138 \\ T_4(N) &= -0.00000001765\,N^4 + 0.00001190\,N^3 - 0.003504\,N^2 \\ &= +0.6876\,N + 17.314 \end{aligned}$$

Polynomial Fits to the Data: Cubic



Cubic Fit

Polynomial Fits to the Data: Quartic



Quartic Fit

Best Cricket Model

So how does one select the best model?

Visually, one can see that the linear model does a very good job, and one only obtains a slight improvement with a quadratic. Is it worth the added complication for the slight improvement.

It is clear that the sum of square errors (SSE) will improve as the number of parameters increase.

From statistics, it is hotly debated how much penalty one should pay for adding parameters.

Best Cricket Model - Analysis

Bayesian Information Criterion

Let n be the number of data points, SSE be the sum of square errors, and let k be the number of parameters in the model.

$$BIC = n\ln(SSE/n) + k\ln(n).$$

Akaike Information Criterion

$$AIC = 2k + n(\ln(2\pi SSE/n) + 1).$$

Best Cricket Model - Analysis Continued

The table below shows the by the Akaike information criterion that we should take a quadratic model, while using a Bayesian Information Criterion we should use a cubic model.

	Linear	Quadratic	Cubic	Quartic
SSE	108.8	79.08	78.74	**78.70**
BIC	46.3	33.65	**33.43**	37.35
AIC	189.97	**175.37**	177.14	179.12

Returning to the original statement, we do fairly well by using the folk formula, despite the rest of this analysis!

What About Different Norms?

The **Least Squares Analysis** is based on ***minimizing*** the **sum of square errors**, which uses setting the **partial derivatives equal to zero**

What about different norms?

1. The best linear function fit in the **1-norm** satisfies

$$\min_{a_0,a_1}\left(\sum_{i=0}^{n}|(a_1x_i+a_0)-y_i|\right)$$

2. The best linear function fit in the **∞-norm** satisfies

$$\min_{a_0,a_1}\left(\max_i|(a_1x_i+a_0)-y_i|\right)$$

MatLab for Different Linear Fits

The previous slide shows the ***minimization*** problems in different norms. Below is the MatLab code for solving this.

```
function [a,err] = normfit(x,y,xmin,xmax)
%Take data and find best 1, 2, and infinity norm ...
    fits to a line
p = polyfit(x,y,1);
err(1) = (sum(((p(1)*x + p(2)) - y).^2)).^0.5;
a(1) = p(1); a(2) = p(2);
[q,err1] = fminsearch(@norm1err,p,[],x,y);
err(2) = err1;
a(3) = q(1); a(4) = q(2);
[w,errinf] = fminsearch(@norminferr,p,[],x,y);
err(3) = errinf;
a(5) = w(1); a(6) = w(2);
```

Next slide has the norm functions

MatLab for Different Linear Fits

Below are the MatLab code for the norm functions.

```
function err1 = norm1err(p,x,y)
%Compute 1-norm residual error for line with data
err1 = sum(abs((p(1)*x + p(2)) - y));
end
```

```
function errinf = norminferr(p,x,y)
%Compute infinity-norm residual error for line ...
    with data
errinf = max(abs((p(1)*x + p(2)) - y));
end
```

Different Norm Fits

The MatLab program uses the `polyfit` to find the ***least squares best fit*** to the Cricket data

It gives the best fitting **linear model** as before as

$$T = 0.215476N + 39.77407$$

and the **2-norm**

$$\|Error\|_2 = \min_{a_0,a_1} \left(\sum_{i=0}^{n} ((a_1 N_i + a_0) - T_i)^2 \right)^{1/2} = 10.4315$$

Different Norm Fits

The MatLab program uses the nonlinear solver `fminsearch` to find the ***best fit*** to the Cricket data in the **1-norm** and **∞-norm**

Best **linear models** are

$$T = 0.215255N + 39.79479$$

with **1-norm**

$$\|Error\|_1 = \min_{a_0,a_1}\left(\sum_{i=0}^{n}|(a_1N_i + a_0) - T_i|\right) = 57.2170$$

and

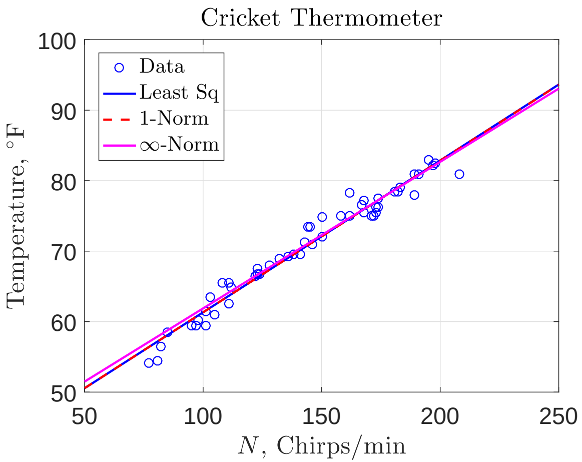
$$T = 0.207874N + 41.09332$$

with **∞-norm**

$$\|Error\|_\infty = \min_{a_0,a_1}\left(\max_i |(a_1N_i + a_0) - T_i|\right) = 3.4311$$

Graphing the Cricket Thermometer Models

The best fitting models with the 3 different norms are shown below



Weighted Least Squares

Often we know that some observations are better than others, so these data should be weighted more heavily

The least squares error becomes

$$\|r\|_w^2 = \sum_{i=0}^{m} w_i r_i^2$$

It becomes harder to count the chirps as the chirping rate increases, so assume the error is proportional to the chirp rate or $e_i = kN_i$. An appropriate weight is

$$w_i = \frac{100}{N_i}$$

The residual becomes

$$\|r\|_w^2 = \sum_{i=0}^{m} w_i \left[(a_1 N_i + a_0) - T_i\right]^2$$

Weighted Least Squares

With the residual above the **normal equations** become

$$\begin{aligned} a_0\sum_{i=0}^{m} w_i + a_1\sum_{i=0}^{m} w_iN_i &= \sum_{i=0}^{m} w_iT_i \\ a_0\sum_{i=0}^{m} w_iN_i + a_1\sum_{i=0}^{m} w_iN_i^2 &= \sum_{i=0}^{m} w_iN_iT_i \end{aligned}$$

Recall from before we defined A and create the diagonal weighting matrix W

$$A_1 = \begin{pmatrix} 1 & N_1 \\ 1 & N_2 \\ \vdots & \vdots \\ 1 & N_m \end{pmatrix} \qquad \text{and} \qquad W = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_m \end{pmatrix}$$

Weighted Least Squares

With these definitions it follows that we can write the matrix equation

$$W\begin{pmatrix} 1 & N_1 \\ \vdots & \vdots \\ 1 & N_m \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = W\begin{pmatrix} T_1 \\ \vdots \\ T_m \end{pmatrix}$$

Following the theory from before, we can generate the solvable system

$$(\mathbf{WA})^{\mathbf{T}}(\mathbf{WA})\tilde{\mathbf{a}} = (\mathbf{WA})^{\mathbf{T}}\mathbf{W}\tilde{\mathbf{T}}$$

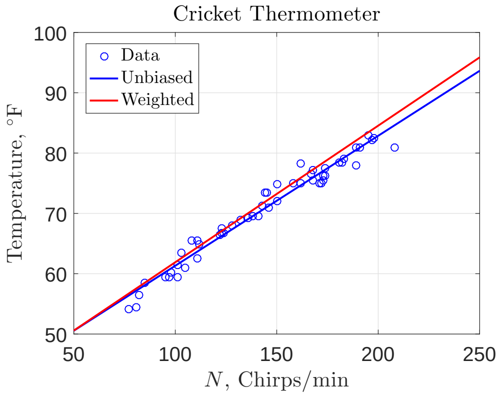
This is readily solved in MatLab with the backslash operator

```
(W*A)\(W*T)
```

Weighted Least Squares Model

The **unbiased model** and **weighted model** ($w_i = 100/N_i$) satisfy:

$$T = 0.215476N + 39.77407 \quad \text{and} \quad T = 0.226451N + 39.25686$$



U. S. Population Models

Source: Census Data

In this example we examine various population models for the U. S. population

Year	Pop (M)	Year	Pop (M)	Year	Pop (M)
1790	3.929	1870	39.818	1950	150.697
1800	5.308	1880	50.189	1960	179.323
1810	7.240	1890	62.948	1970	203.302
1820	9.638	1900	76.212	1980	226.546
1830	12.866	1910	92.228	1990	248.710
1840	17.069	1920	106.022	2000	281.422
1850	23.192	1930	122.775	2010	308.746
1860	31.443	1940	132.165		

We'll use $t = 0$ as 1790

U. S. Population Analysis

Create a vector from the years after 1790, t, and the population (in millions), P

Create matrices

$$A_1 = \begin{pmatrix} 1 & t_1 \\ \vdots & \vdots \end{pmatrix} \quad A_2 = \begin{pmatrix} 1 & t_1 & t_1^2 \\ \vdots & \vdots & \vdots \end{pmatrix} \quad A_3 = \begin{pmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

As before, we use the MatLab backslash operator to find the coefficients for polynomial models

$$\mathtt{A}_i\backslash\mathtt{P},$$

where the matrices A_i are formed from `vanA.m` (or alternately, coefficients come from `polyfit`)

Polynomial U. S. Population Models

With the coefficients found on previous slides we obtain the **Linear Model**:

$$P_1(t) = 1.359627t - 45.568207,$$

the **Quadratic Model**:

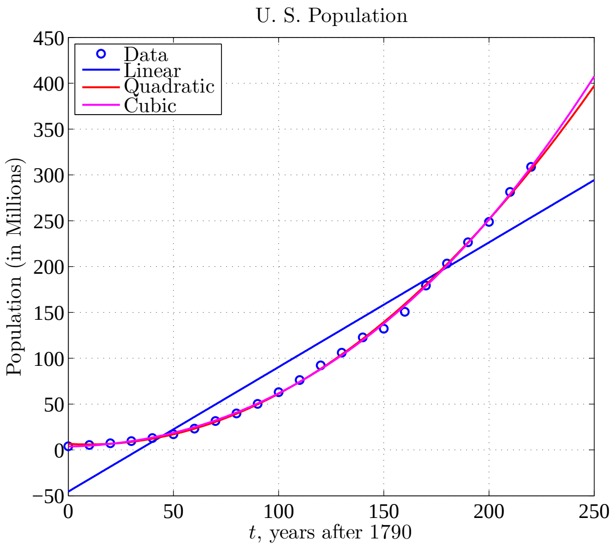
$$P_2(t) = 0.00677199t^2 - 0.130210t + 6.576104,$$

the **Cubic Model**:

$$P_3(t) = 0.00000623775t^3 + 0.00471353t^2 + 0.0469419t + 3.694263.$$

Clearly, the **Linear Model** has serious problems with its negative intercept

Polynomial U. S. Population Models



Polynomial Population - Analysis

The models are compared to data, finding the ***norm*** of the error and ***sum of square errors (SSE)***

The table below shows that both the Akaike information criterion and the Bayesian information criterion suggest that we should use a cubic model.

	Linear	Quadratic	Cubic
$\|Err\|$	128.20	13.83	12.05
SSE	16434.8	191.27	145.20
BIC	157.42	58.13	54.92
AIC	220.42	119.99	115.65

The graph and the two information criteria show that the **Quadratic** and **Cubic** models do quite well matching the data

However, **polynomials** are only qualitative fits and **NOT** based on dynamics of populations

Exponential Models

The most basic population model is the **Malthusian growth model**, which comes from the ***linear differential equation***

$$\frac{dP}{dt} = rP, \qquad P(0) = P_0.$$

This model says that the population growth is proportional to the population, and this works well for most uncrowded populations

This has the solution

$$P(t) = P_0 e^{rt}$$

First, re-cast the problem as a set of linear equations:

$$P_0 e^{rt_i} = P_i, \quad i = 0, \dots, m$$

compute the natural logarithm on both sides:

$$\underbrace{\ln P_0}_{a_0} + \underbrace{r}_{a_1} t_i = \underbrace{\ln P_i}_{f_i}.$$

Malthusian Growth Model

From the **Malthusian growth model**,

$$P(t) = P_0 e^{rt}$$

we obtained:

$$P_0 e^{rt_i} = P_i, \quad i = 0, \dots, m,$$

which with natural logarithms becomes a **linear model**:

$$\ln P_0 + rt_i = a_0 + a_1 t_i = \ln P_i.$$

Now, apply a ***linear least squares fit*** to the problem and obtain coefficients (a_0, a_1), where

$$P_0 = e^{a_0} \qquad \text{and} \qquad r = a_1.$$

Note: This does **not** give the least squares fit to the original problem!!! (It gives us an estimate on the log-scale.)

Malthusian Growth Model

Using the U. S. population data and fitting the $\ln(P)$, we find

$$a_0 = 1.843853 \qquad \text{and} \qquad a_1 = 0.0196234,$$

so the best fitting **Malthusian growth model** is

$$P(t) = 6.320851e^{0.0196234t}$$

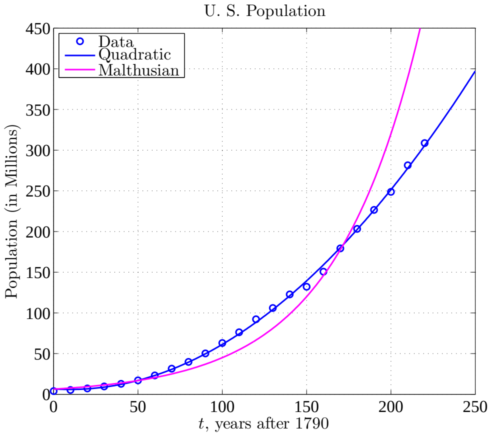
We find the ***norm*** of the error and ***sum of square errors (SSE)*** for this model are

$$\|Err\| = 220.66606 \qquad \text{and} \qquad SSE = 48693.51,$$

which is even worse than the **linear model** fit

This large error is largely caused by the bias of the logarithmic scale to over emphasize the early points.

Malthusian Growth Model - Graph



The graph shows the ***log fit*** parameters perform poorly for recent history. However, this was the linear fit to the logarithm of the population data.

Nonlinear Least Squares - Malthusian Growth

The **nonlinear sum of square errors** satisfies

$$E(P_0, r) = \sum_{i=0}^{n} \left(P_0 e^{rt_i} - P_i\right)^2$$

Taking partial derivatives gives

$$\begin{aligned}
\frac{\partial E}{\partial P_0} &= 2\sum_{i=0}^{n} \left(P_0 e^{rt_i} - P_i\right) e^{rt_i} \\
\frac{\partial E}{\partial r} &= 2\sum_{i=0}^{n} \left(P_0 e^{rt_i} - P_i\right) P_0 t_i e^{rt_i}
\end{aligned}$$

The minimum occurs when these partial derivatives are **zero**

This requires solving two nonlinear equations for the parameters, P_0 and r

Nonlinear Least Squares - Malthusian Growth

Setting the partial derivative equations to **zero** gives the nonlinear system to solve for the parameters, P_0 and r:

$$\begin{aligned} P_0 \sum_{i=0}^{n} e^{2rt_i} &= \sum_{i=0}^{n} P_i e^{rt_i} \\ P_0 \sum_{i=0}^{n} t_i e^{2rt_i} &= \sum_{i=0}^{n} P_i t_i e^{rt_i} \end{aligned}$$

Note that the first equation could be solved for P_0 easily, which could be substituted into the second equation

The resulting highly nonlinear equation could be solved by Newton's method or one of our other routine for solving $f(x) = 0$

Nonlinear Least Squares with MatLab

Recall that the function we want to minimize is the **nonlinear sum of square errors**:

$$E(P_0, r) = \sum_{i=0}^{n} \left(P_0 e^{rt_i} - P_i\right)^2$$

In MatLab

```
function LS = mallstsq(p,t,y)
%Least Squares sum of square errors to Malthusian ...
    growth model
LS = sum((p(1)*exp(p(2)*t)-y).^2);
end
```

MatLab has a function `fminsearch`, which handles multidimensional unconstrained nonlinear minimization (Nelder-Mead), so we type

`[p,err]=fminsearch(@mallstsq,p0,[],t,pop)`

Nonlinear Least Squares with MatLab

The output from the MatLab command

```
[p,err]=fminsearch(@mallstsq,p0,[],t,pop)
```

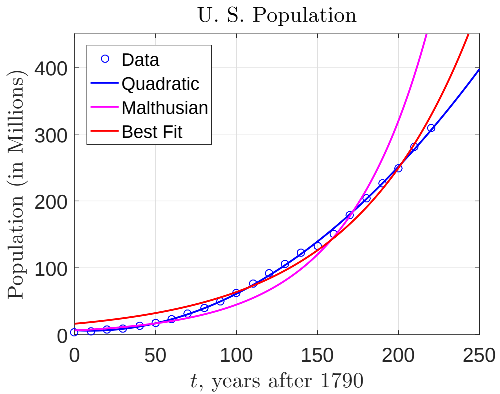
is $p = [16.345612, 0.0136284]$ and $err = 2875.53$

This gives the **Nonlinear Least Squares Best Model**

$$P(t) = 16.345612e^{0.0136284t},$$

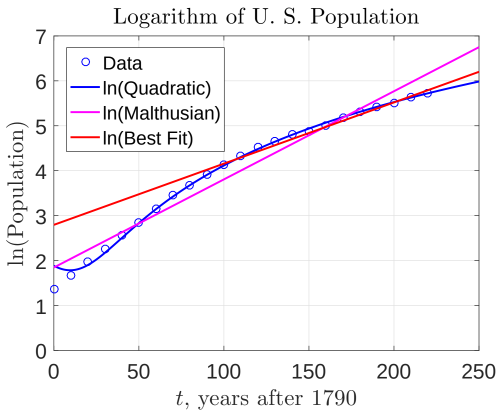
with a substantially improved sum of square error = 2875.53. This is worse than the Quadratic fit, but quite good for only **2 parameters**.

Malthusian Growth Model



We see the best fitting Malthusian growth model tracks the population much better than the one using logarithms and a linear fit.

Malthusian Growth Model



This graph shows the models on a logarithmic scale allowing a better understanding of what is happening.

U. S. Population Model Summary

1. The polynomial fits are the best for these population data, but they fail to capture any dynamics
2. The modified problem (Linear fit to logarithms of the population data) does **not** give the least squares fit to the original problem. It weights the early data too heavily, leaving a poor fit
3. **Nonlinear Least Squares Best model** gives a reasonable fit and correlates to what is known about populations
   - Finding the true least squares fit requires knowing how to find roots and/or minima/maxima of non-linear systems of equations
   - Introduced the MatLab `fminsearch` to solve this, but it remains a **Black Box** at this time
   - What we need: `Math 693a` — **Numerical Optimization Techniques.**

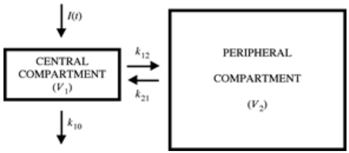
Pharmokinetics - Introduction Modeling

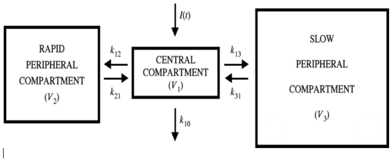
Pharmokinetic Modeling

- Primarily uses **Compartmental Models**
  - **Two compartment** examines fast ***distribution phase*** and slow ***elimination phase***
  - **Multi-compartment** examines multiple time scale distribution phases and an elimination phase
- Compartmental Models usually composed of **linear differential equations**
- Parameters are notoriously hard to estimate
- Controversial how to refine model
  - How many compartments make sense?
  - Are nonlinear interactions more important?

Pharmokinetics - Diagram

Diagrams for Compartmental Models[1]




I(t)
RAPID PERIPHERAL COMPARTMENT (V2)
k12
k21
CENTRAL COMPARTMENT (V1)
k13
k31
k10
SLOW PERIPHERAL COMPARTMENT (V3)

$$C_2(t) = a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t}$$

$$C_3(t) = a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t} + a_3 e^{-\lambda_3 t}$$

$I(t)$ is the injection amount, V_i is the volume of each compartment, k_{ij} is the flow between compartments and exterior (or metabolism), C_i is the concentration is each compartment, and the parameters to be found are a_i and λ_i

[1] Steven L. Shafer, Pharmacokinetics and Pharmacodynamics: The Anesthesiologists Perspective, (downloaded from Google accessed 12/03/2016)

Pharmokinetics - Description/Definitions

The **Drug Compartmental Model** generally only measures the **plasma concentration** of the drug

$$C(t) = \sum_{i=1}^{m} a_i e^{-\lambda_i t}$$

- The parameters a_i and λ_i reflect information about the volumes of the compartments, kinetics of the drug, and flow between compartments
- The smallest λ_i relates to the ***terminal or elimination phase*** of the drug, key to how long a drug is active in the body
- The largest λ_i is called the ***rapid distribution phase*** of the drug, reflecting the initial movement of the drug to the tissues or filtering by the kidneys
- For the 3 compartment model the intermediate λ_i reflects the drug entering well perfused organs, such as the liver

Pharmokinetics - Finding Parameters

Consider the model

$$C(t) = \sum_{i=1}^{m} a_i e^{-\lambda_i t}$$

- If the parameters λ_i are known, then our **Least Squares** techniques from above are readily used
- Nonlinear least squares methods can be applied
  - Often poor fits, as square errors have similar values over wide range of parameters
  - Use weighted least squares, such as logarithmic scale, to improve fit
  - Use **Exponential Peeling**, starting with the terminal, λ_i, values to sequentially obtain a good fit
- Demonstrate some of these techniques with an **Example**

Pharmokinetics - Fentanyl Dog Study

A study including 6 dogs were injected with the opioid ^{3}H-fentanyl citrate

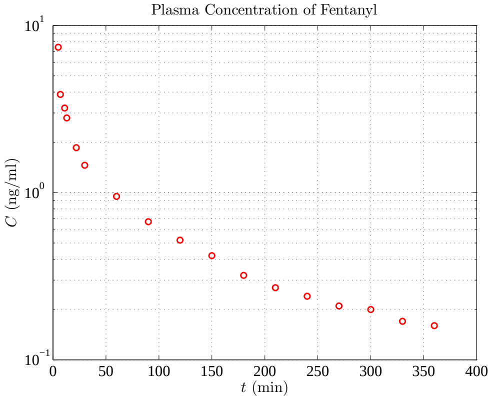
Below is a table[2] of the time evolution of plasma concentration (ng/ml), where t is in min

t	C	t	C	t	C
5	7.42	60	0.95	240	0.24
7	3.87	90	0.67	270	0.21
11	3.21	120	0.52	300	0.20
13	2.80	150	0.42	330	0.17
22	1.86	180	0.32	360	0.16
30	1.46	210	0.27		

[2] Murphy, M. R., Olson, W. A., and Hug, Jr, C. C., Pharmacokinetics of ^{3}H-Fentanyl in the dog anesthetized with enflurane, *Anethesiology*, **50**: 13-19, 1979

Pharmokinetics - Fentanyl Dog Study

The data are graphed using a logarithmic scale, which shows this is not simple exponential decay (not a straight line)



Observe that the tail is almost linear, while the first part has a significant steep drop

Drug Study - 2-Compartment Model

2-Compartment Model is the sum of 2 decaying exponentials

$$C_2(t) = a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t}$$

We will later show (**exponential peeling**) that a good choice is $\lambda_1 = 0.1601$ and $\lambda_2 = 0.003794$

The set $\{\phi_1(t), \phi_2(t)\} = \{e^{-\lambda_1 t}, e^{-\lambda_2 t}\}$ creates a good basis for a ***least squares analysis***

$$A_2 = \begin{pmatrix} e^{-\lambda_1 t_1} & e^{-\lambda_2 t_1} \\ \vdots & \vdots \\ e^{-\lambda_1 t_m} & e^{-\lambda_2 t_m} \end{pmatrix}$$

with plasma drug concentration, $c_d = [c_1, \cdots, c_m]^T$, so our least squares technique finds a_1 and a_2 with the MatLab command

$$\texttt{A}_2 \backslash \texttt{c}_d$$

Drug Study - 3-Compartment Model

3-Compartment Model is the sum of 3 decaying exponentials

$$C_3(t) = b_1 e^{-\lambda_1 t} + b_2 e^{-\lambda_2 t} + b_3 e^{-\lambda_3 t}$$

We will later show that a good choice is $\lambda_1 = 0.1601$, $\lambda_2 = 0.02078$, and $\lambda_3 = 0.003794$

The set $\{\phi_1(t), \phi_2(t), \phi_3(t)\} = \{e^{-\lambda_1 t}, e^{-\lambda_2 t}, e^{-\lambda_3 t}\}$ creates a good basis for a ***least squares analysis***

$$B_3 = \begin{pmatrix} e^{-\lambda_1 t_1} & e^{-\lambda_2 t_1} & e^{-\lambda_3 t_1} \\ \vdots & \vdots & \vdots \\ e^{-\lambda_1 t_m} & e^{-\lambda_2 t_m} & e^{-\lambda_3 t_m} \end{pmatrix}$$

with drug data, $c_d = [c_1, \cdots, c_m]^T$, so our least squares technique finds b_1, b_2, and b_3 with the MatLab command

$$\texttt{B}_3\backslash\texttt{c}_d$$

Drug Study - Compartment Models

The program to obtain the coefficients is

```
function A = vanAexp(t,L)
%Least Squares Matrix for column data t and
%L vector for columns of exp(-L(i)t) functions
n = length(L);
A = [exp(-L(1)*t)];
for i = 2:n
    y = [exp(-L(i)*t)];
    A = [A,y];
end
end
```

So `A = vanAexp(t,L)` with `a = A\`c_d, where t and c_d are the column time and plasma drug data and L is the appropriate vector of λ_i's, produces the desired ***least squares best fitting*** coefficients a

Drug Study - Compartment Models

The **2-Compartment Model** is

$$C_2(t) = a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t},$$

where $[a_1, a_2] = [12.46300, 1.06818]$ and $[\lambda_1, \lambda_2] = [0.1601, 0.003794]$
The SSE between $C_2(t)$ and the data is 2.88615, while the SSE between $\ln(C_2(t))$ and $\ln(data)$ is 2.75940.

The **3-Compartment Model** is

$$C_3(t) = b_1 e^{-\lambda_1 t} + b_2 e^{-\lambda_2 t} + b_3 e^{-\lambda_3 t},$$

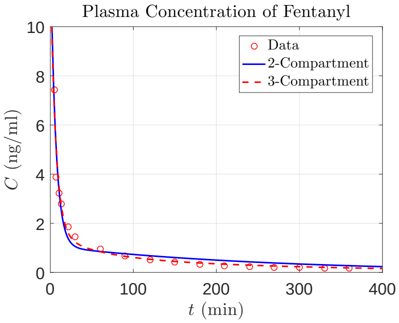
where $[b_1, b_2, b_3] = [11.02014, 0.94518, 0.71575]$ and $[\lambda_1, \lambda_2, \lambda_3] = [0.1601, 0.02078, 0.003794]$
The SSE between $C_3(t)$ and the data is 2.54328, while the SSE between $\ln(C_3(t))$ and $\ln(data)$ is 0.41208, substantially better.

Compartmental Model Fits

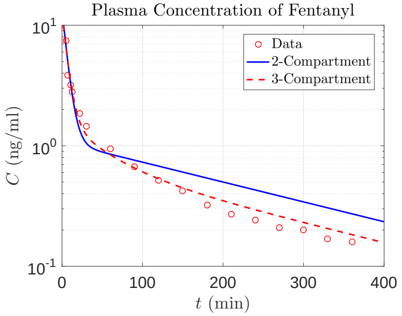
```
load 'dog'
t = dog(:,1); d = dog(:,2);
L2 =[0.1601,0.003794];
A2 = vanAexp(t,L2);
a2 = A2\d
model2d = a2(1)*exp(-L2(1)*t)+a2(2)*exp(-L2(2)*t);
sse2 = sum((d-model2d).^2)
lnsse2 = sum((log(d)-log(model2d)).^2)
L3 =[0.1601,0.02078,0.003794];
A3 = vanAexp(t,L3);
a3 = A3\d
model3d = a3(1)*exp(-L3(1)*t) + ...
     a3(2)*exp(-L3(2)*t) + a3(3)*exp(-L3(3)*t);
sse3 = sum((d-model3d).^2)
lnsse3 = sum((log(d)-log(model3d)).^2)
```

Graphs of Least Squares Fits

Below are graphs of the data and the **least squares best fit** of the **2** and **3-compartment models**



Linear Scale



Semilog Scale

Drug Study - Least Squares Fit

The **Compartment Model** was fit with the sum of 2 or 3 decaying exponentials, using our ***least squares best fit*** routine

- The linear scale graph shows a very good model fit of the data
  - The early rapid decline and late slow elimination match data very well
  - The data compares very well with both models
  - 2-compartment model has a corner at the phase transitions
- The semilog fit shows the deficiencies better
  - The early ***rapid distribution phase*** still matches extremely well
  - The late ***elimination phase*** parallels the data, showing the correct rate of decay, but a vertical shift
  - The **3-compartment model** is clearly a better fit
- The problem with the transition shows the dynamics of the drug absorption and decay are more **complex**

Exponential Peeling - Basis Set

The ***linear least squares*** method above requires a reasonable set of exponentially decaying functions to use as a **basis set**

Earlier discussion of the graphes pointed to stretches where the data appear almost linear

The **basis set** is derived from **Exponential Peeling**

- One subjectively decides which data at the tail are almost linear on a log scale
- This tail is fit with an exponential
- This exponential tail is subtracted from the data, then process is repeated

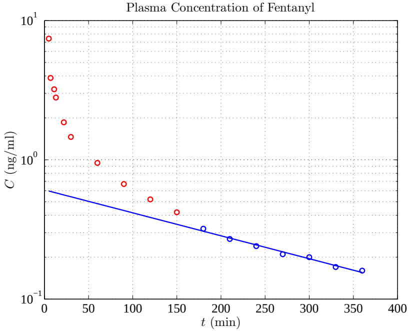
Exponential Peeling

Exponential Peeling

- Graph data with either a semilog scale or log of the data
- Observe the **linear tail** and **select data** related to the tail (**Subjective!**)
- Find the ***linear least squares best fit*** to the log of the data, obtaining $c_i(t) = a_i e^{\lambda_i t}$, fitting the tail
- Subtract $c_i(t) = a_i e^{\lambda_i t}$ from the data
- Repeat to obtain other **decay rates**

Exponential Peeling

Graph of the **fentanyl drug** data

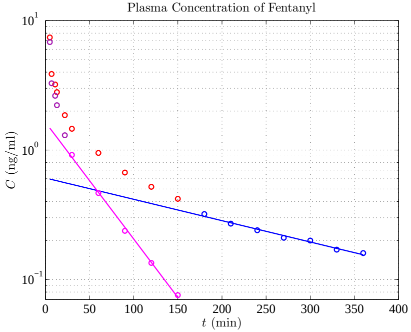


The ***terminal/elimination phase*** on the right of the graph almost forms a line ($t \in [180, 360]$)

With a linear fit to the log of the data we produce the best fitting function $c_3(t) = 0.6083e^{-0.003794t}$

Exponential Peeling

Graph of the **fentanyl drug** data



The *middle distribution phase* on the right of the data after subtracting the ***terminal phase*** almost forms a line $(t \in [30, 150])$

With a linear fit to the log of the data we produce the best fitting function $c_2(t) = 1.6396e^{-0.02078t}$

Exponential Peeling

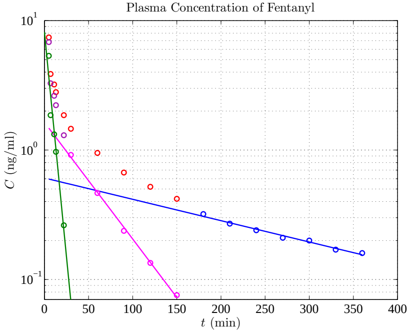
```
% Plots of Dog data (2 Exponential Peeling Fits)
load 'dog'; t = dog(:,1); d = dog(:,2);
% Divide data into 3 phases (rapid, middle, terminal)
t1 = t(1:5,:);t2 = t(6:10,:);t3 = t(11:17,:);
d1 = d(1:5,:);d2 = d(6:10,:);d3 = d(11:17,:);
% log of terminal data for exponential fit
ld3=log(d(11:17,:)); LA3 = vanA(t3,1);
aa3 = LA3\ld3;  % Least squares fit to log data
% Subtract terminal exponential from rapid and ...
    middle data
d1a = d1 - exp(aa3(1))*exp(aa3(2)*t1);
d2a = d2 - exp(aa3(1))*exp(aa3(2)*t2);
% log of modified middle data
ld2=log(d2a); LA2 = vanA(t2,1);
aa2 = LA2\ld2; % Least squares fit to log data
t4 = t(1:10,:); % Extended domain for graphing
```

Exponential Peeling

```
%Plot Data and Models
tt = linspace(0,400,200);
semilogy(t1,d1,'ro'); % Rapid phase data
hold on
semilogy(t2,d2,'ro'); % Middle phase data
semilogy(t3,d3,'bo'); % Terminal phase data
% Graph best terminal phase exponential model
semilogy(t,exp(aa3(1))*exp(aa3(2)*t),'b-');
% Graph data - terminal phase model
semilogy(t1,d1a,'o','color',[0.65,0.1,0.7]);
semilogy(t2,d2a,'mo');
% Graph best middle phase exponential model
semilogy(t4,exp(aa2(1))*exp(aa2(2)*t4),'m-');
grid
ylim([0.07,10]);
```

Exponential Peeling

Graph of the **fentanyl drug** data

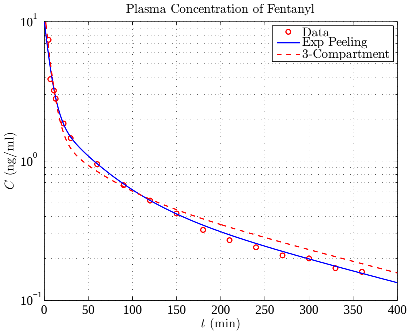


The *rapid distribution phase* from the data after subtracting the two other ***phases*** almost forms a line ($t \in [5, 22]$)

With a linear fit to the log of the data we produce the best fitting function $c_1(t) = 8.1514e^{-0.1601t}$

3-Compartment Models

3-Compartment Models: Used ***Least Squares*** and ***Exponential Peeling*** methods



Least squares model and Exponential peeling model:

$$C(t) = 11.02014e^{-0.1601t} + 0.94518e^{-0.02078t} + 0.71575e^{-0.003794t}$$
$$C(t) = 8.1514e^{-0.1601t} + 1.6396e^{-0.02078t} + 0.6083e^{-0.003794t}$$

Nonlinear Least Squares

The Least squares model minimizes:

$$J(a_1,a_2,a_3) = \min_{\{a_1,a_2,a_3\}} \left(C_i - (a_1e^{-0.1601t} + a_2e^{-0.02078t} + a_3e^{-0.003794t})\right)^2,$$

where the λ_i values are fixed

This minimum is $J(a_1,a_2,a_3) = 2.5433$, giving:

$$C(t) = 11.02014e^{-0.1601t} + 0.94518e^{-0.02078t} + 0.71575e^{-0.003794t}.$$

The Exponential peeling model gives

$$C(t) = 8.1514e^{-0.1601t} + 1.6396e^{-0.02078t} + 0.6083e^{-0.003794t},$$

where $J(a_1,a_2,a_3) = 3.4859$

Nonlinear Least Squares

With exponentially decaying data it makes more sense to minimize the logarithm of the data
Thus, we want the model that minimizes:

$$J_2(a_1, a_2, a_3) = \min_{\{a_1, a_2, a_3\}} \left(\ln(C_i) - \ln(a_1 e^{-0.1601t} + a_2 e^{-0.02078t} + a_3 e^{-0.003794t})\right)^2,$$

where the λ_i values are fixed

In this case, the Least squares model has a **sum of square errors** of

$$J_2(11.02014, 0.94518, 0.71575) = 0.41208.$$

The Exponential peeling model has a **sum of square errors** of

$$J_2(8.1514, 1.6396, 0.6083) = 0.12778,$$

which is clearly closer to the minimum (as was seen on the graph).

Nonlinear Least Squares

The Exponential peeling model gives very good parameters, so can be used for a **Nonlinear Least Squares** using MatLab's `fminsearch`

We want the model that minimizes:

$$J_3(p) = \min_p \left(\ln(C_i) - \ln(p_1 e^{-p_2 t} + p_3 e^{-p_4 t} + p_5 e^{-p_6 t})\right)^2$$

for our 6-dimensional parameter p, as the exponential peeling used guesses for peeling off exponentials

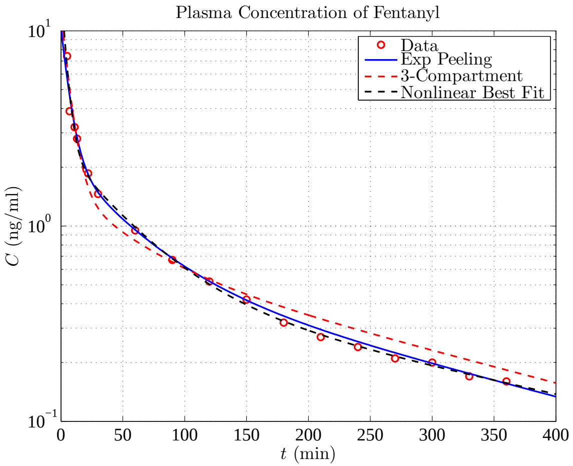
With `fminsearch` for this nonlinear least squares functional and the initial guess from the exponential peeling, we obtain the model

$$C(t) = 14.1472 e^{-0.24226t} + 2.0063 e^{-0.020695t} + 0.4928 e^{-0.0031957t}$$

This $J_2(p) = 0.08607$, which improves on the exponential peeling

It also shows the exponential peeling overestimates the rate of elimination, as has been noted in the literature

Fentanyl Drug Models



Nonlinear Least Squares

The Exponential peeling model gives very good parameters, so can be used for a **Nonlinear Least Squares** using MatLab's `fminsearch`

The standard **least squares fit** uses our 6-dimensional parameter p

$$J_3(p) = \min_p \left(C_i - (p_1 e^{-p_2 t} + p_3 e^{-p_4 t} + p_5 e^{-p_6 t})\right)^2$$

With `fminsearch` for this nonlinear least squares functional gives

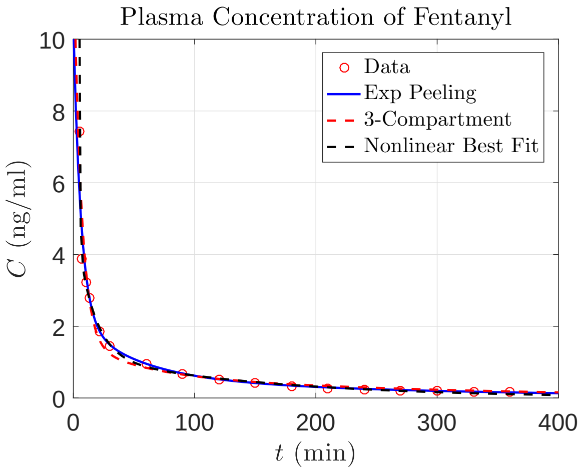
$$C(t) = 2364.1e^{-1.3165t} + 4.2249e^{-0.07100t} + 1.21415e^{-0.006696t}$$

This $J_3(p) = 0.05247$, which improves substantially over the other models

However, the leading coefficients relate to the modeling volumes of the compartments (important in the physiology), and these are unreasonable because of the heavy weighting of the initial data

From a modeling perspective, this model is **NOT** as good!

Fentanyl Drug Models



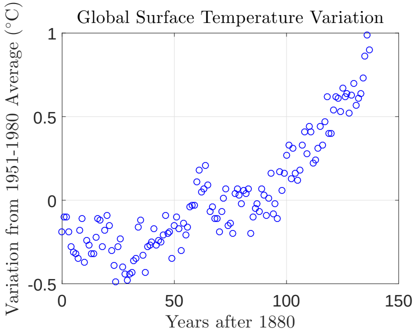
Climate Modeling

Source: NASA

Global Climate Change is a ***HOT*** area of study.

The last **3 years** have seen the hottest *Global average temperatures* on Earth.

Below is a graph showing the ***variation in temperature ($^\circ C$)*** from the ***average global surface temperature from 1951-1980***.



Climate Modeling - Polynomial Fits

The **NASA hyperlink** gives access to the data on the previous slide.

Variables are created for the t data (in years after 1880) and the V data representing the variation from the ***average global surface temperature*** from 1951-1980.

Observation of the data suggests that a low degree (perhaps ***quadratic***) polynomial fits the data reasonably well.

From before, we create the Vandermonde matrices for the appropriate polynomial fits

```matlab
function A = vanA(x,n)
%Least Squares Matrix for x and n poly
A = [ones(length(x),1)];
for i = 1:n
    A = [A,x.^i];
end
end
```

Climate Modeling - Polynomial Fits

The program below illustrates how to obtain the best ***quadratic fit*** to the data $(climt, climd)$.

In addition to the best fitting coefficients, it computes the ***least sum of square errors*** and plots the model.

```
A2 = vanA(climt',2);
a2 = A2\climd';
qfit = a2(1) + a2(2)*tt + a2(3)*tt.^2;
qfite = a2(1) + a2(2)*climt + a2(3)*climt.^2;
err2 = qfite - climd;
lst2 = err2*err2'
plot(tt,qfit,'-','color',[0,0.5,0],'linewidth',1.5);
```

Climate Modeling - Polynomial Fits

The best linear, quadratic, and cubic fits are given by:

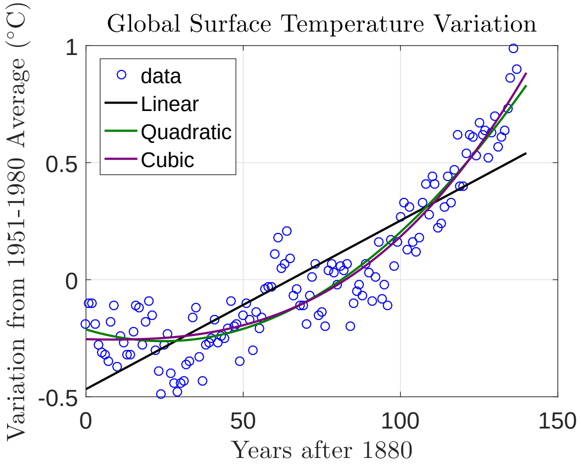
Linear:	$V = 0.0072034\,t - 0.46757.$
Quadratic:	$V = 0.000082216\,t^2 - 0.0040601\,t - 0.21226.$
Cubic:	$V = 3.3119\times10^{-7}\,t^3 + 1.4157\times10^{-5}\,t^2 - 0.00034401\,t - 0.25391.$

The ***least sum of square errors*** for these fits are

Linear:	$SSE = 3.7290.$
Quadratic:	$SSE = 1.8500..$
Cubic:	$SSE = 1.8127..$

Climate Modeling - Polynomial Fits

The best linear, quadratic, and cubic fits give the following graph:



Climate Modeling - Trigonometric

The graph shows very stochastic behavior, but could there be underlying periodic phenomenon?

The sun has an approximately **11 year** cycle, so the frequency of the trig functions would be:

$$\omega = \frac{2\pi}{11} \approx 0.5712.$$

The polynomial study showed that a ***quadratic model*** fit the data quite well, so consider a quadratic model with trig functions:

$$V(t) = a_0 + a_1 t + a_2 t^2 + a_3 \cos(\omega t) + a_4 \sin(\omega t).$$

Climate Modeling - Trignometric

For a model of the form:

$$V(t) = a_0 + a_1 t + a_2 t^2 + a_3 \cos(\omega t) + a_4 \sin(\omega t).$$

We create a Vandermonde matrix for this model

```
function A = vanA2cosin(x)
%Least Squares Matrix for x and n poly
A = [ones(length(x),1)];
A = [A,x,x.^2,cos(2*pi*x/11),sin(2*pi*x/11)];
end
```

Climate Modeling - Trigonometric

The program below illustrates how to obtain the best ***quadratic model with trig functions*** to the data $(climt, climd)$.

In addition to the best fitting coefficients, it computes the ***least sum of square errors*** and plots the model.

```
Acs = vanA2cosin(climt');
acs = Acs\climd';
model3 = acs(1) + acs(2)*tt + acs(3)*tt.^2 ...
          + acs(4)*cos(2*pi*tt/11) + ...
              acs(5)*sin(2*pi*tt/11);
model3e = acs(1) + acs(2)*climt + acs(3)*climt.^2 ...
          + acs(4)*cos(2*pi*climt/11) + ...
              acs(5)*sin(2*pi*climt/11);
errm = model3e - climd;
lstm = errm*errm'
plot(tt,model3,'r-','linewidth',1.5);
```

Climate Modeling - Trigonometric

The best quadratic and trigonometric models are given by:

Quadratic: $V = 0.000082216\,t^2 - 0.0040601\,t - 0.21226.$

Trig: $V = 0.000081757\,t^2 - 0.0039972\,t - 0.21399 + 0.0024086\cos(\omega t) + 0.011336\sin(\omega t).$

The ***least sum of square errors*** for these fits are

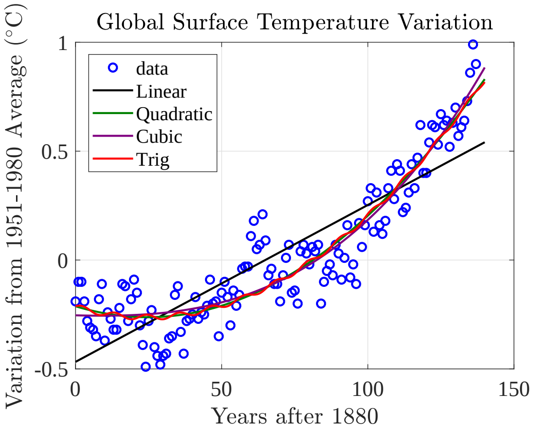
Quadratic: $SSE = 1.8500.$

Trig: $SSE = 1.8409.$

which show only a small difference.

Climate Modeling - Trigonometric

The best linear, quadratic, cubic, and trig model fits give the following graph:



Climate Modeling - Models

It remains to compare the models developed with the Bayesian and Akaike Information Criteria.

Below is a Table of all the models developed above:

	Linear	Quadratic	Cubic	Trigonometric
SSE	3.72902	1.85005	1.81274	1.84088
BIC	-488.48	-580.28	-578.16	-571.11
AIC	-102.71	-197.43	-198.25	-194.12

The **BIC** indicates that the ***quadratic model*** is the best model.

The **AIC** indicates that the ***cubic model*** is the best model.

The addition of ***trigonometric functions*** does NOT create a significantly better model.