

Math 129: Linear Algebra

Topics

- Matrix Operations
- Solving N equations for N unknowns
- Least Squares Curve Fitting
- Global Warming

Introduction

In Algebra, you learn how to solve one linear equation with one unknown such as

$$2(x + 3) + 5x = 10x + 20$$

In Linear Algebra, you learn how to solve N equations with N unknowns. Matlab is really good at solving this type of problem.

At its heart, Matlab is a matrix language: it allows you to do any operation you'd do with a scalar with matrices as well. Matrices have a few peculiar properties, however. That's what Math 129 is all about.

In this lecture, Matlab is used to solve problems such as solving 3 equations with 3 unknowns, such as

$$3a + 4b + 5c = 10$$

$$5a + 6b - c = 20$$

$$a + b + c = 2.$$

It is also used to come up with the 'best' solution when you have more equations than unknowns - also known as curve fitting.

Matrix Definition and Properties.

The dimension of a matrix is the number of rows and columns. A 2x3 matrix, for example, has two rows and 3 columns. For example, a 2x3 matrix can be input into Matlab as:

$$A = [1, 2, 3 ; 4, 5, 6]$$

$$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array}$$

Matrix Addition: When you add two matrices, each element added. Only matrices of like dimensions can be added. Matrix addition is commutative ($A+B = B+A$).

$$A = [1, 2, 3 ; 4, 5, 6]$$

$$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array}$$

$$B = [2, 2, 2 ; 3, 3, 3]$$

$$\begin{array}{ccc} 2 & 2 & 2 \\ 3 & 3 & 3 \end{array}$$

$$C = A + B$$

$$\begin{array}{ccc} 3 & 4 & 5 \\ 7 & 8 & 9 \end{array}$$

Matrix Multiplication: When you multiply two matrices, the inner dimension must be the same.

$$C_{2 \times 1} = A_{2 \times 3} B_{3 \times 1}$$

Element i,j of matrix C is computed as

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

$$A = [1, 2, 3 ; 4, 5, 6]$$

$$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array}$$

$$B = [7; 8; 9]$$

$$\begin{array}{c} 7 \\ 8 \\ 9 \end{array}$$

$$C = A * B$$

$$\begin{array}{c} 50 \\ 122 \end{array}$$

Note that matrix multiplication is *not* commutative:

$$AB \neq BA$$

```
C = B*A
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Zero Matrix: A zero matrix is a matrix of all zeros. The zero matrix behaves like the number zero:

- $A + 0 = A$
- $A * 0 = 0$

$$Z = \text{zeros}(2, 3)$$

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}$$

Identity Matrix: The identity matrix is a square matrix ($n \times n$) which is one on the diagonal, zero elsewhere. The identity matrix behaves like the number one:

- $A * I = A$

$$I = \text{eye}(3, 3)$$

$$\begin{array}{ccc} 1 & 0 & 0 \end{array}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Inverse: B is the inverse of A if $AB = I$

$$A = [1, 2, 3 ; 4, 5, 6; 1 \ 2 \ 1]$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 1 \end{bmatrix}$$

$$B = \text{inv}(A)$$

$$\begin{bmatrix} -1.1667 & 0.6667 & -0.5000 \\ 0.3333 & -0.3333 & 1.0000 \\ 0.5000 & 0 & -0.5000 \end{bmatrix}$$

$$A*B$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Solving N equations for N unknowns

If you have N equations to solve for N unknowns, the approach used in Math 129 is to write the system in matrix form:

$$Y_{N \times 1} = B_{N \times N} A_{N \times 1}$$

where

- A is a matrix of your N unknowns
- B is a basis function and
- Y the result for these N equations

The solution is then

$$A = B^{-1} Y$$

Example: Solve the following set of 3 equations for 3 unknowns:

$$3a + 4b + 5c = 10$$

$$5a + 6b - c = 20$$

$$a + b + c = 2$$

Step 1: Group terms and write in matrix form:

$$\begin{bmatrix} 3 & 4 & 5 \\ 5 & 6 & -1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 2 \end{bmatrix}$$

Step 2: Invert and solve

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 3 & 4 & 5 \\ 5 & 6 & -1 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 10 \\ 20 \\ 2 \end{bmatrix}$$

In Matlab:

```
B = [3,4,5;5,6,-1;1,1,1]
```

```
    3    4    5
    5    6   -1
    1    1    1
```

```
Y = [10;20;2]
```

```
    10
    20
     2
```

```
A = inv(B)*Y
```

```
-2.7500
 5.5000
-0.7500
```

Because of the way we set up the matrices, this means

- $a = -2.75$
- $b = 5.50$
- $c = -0.75$

Solving M equations for N unknowns when $M > N$

If you have more equations than unknowns, there is often times no exact solution. This is often encountered with real data where you have noise in the data. For example, take the amount of Arctic ice as recorded by the National Sea and Ice Data Center (<https://nsidc.org/arcticseaicenews/sea-ice-tools/>)

Year	min	max
2025	3.566	14.308
2024	4.280	14.952
2023	4.230	14.616
2022	4.674	14.838
2021	4.724	14.767
:	:	:
1979	7.05	16.34

Table 1: Min and max Arctic ice areas (millions km²) from the National Sea and Ice Data Center

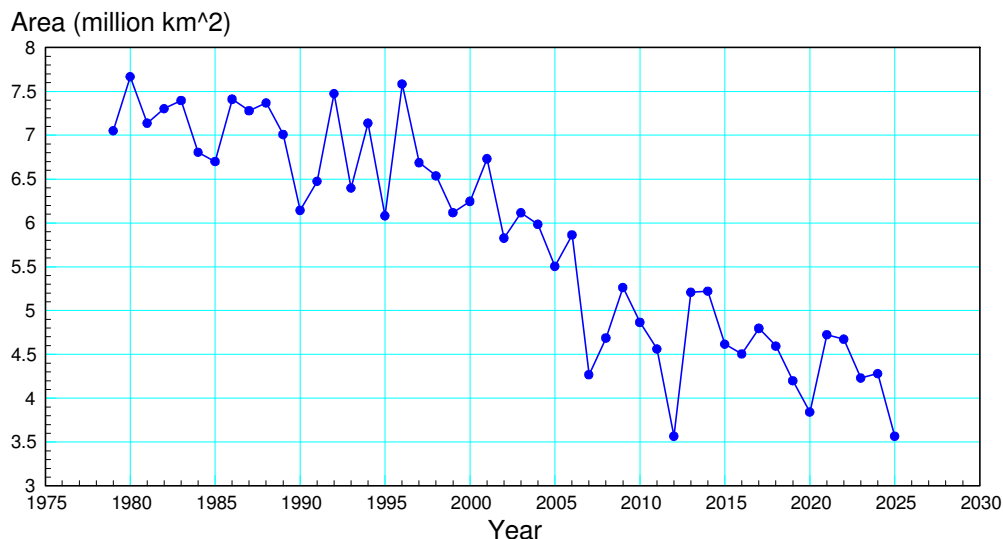


Figure 1: Minimum Arctic Sea Ice Million km2 (<http://nsidc.org/arcticseaicenews/chartic-interactive-sea-ice-graph/>)

From this data, determine a line to approximate this data:

$$y = ax + b$$

$$\text{Sea Ice Extent} = a * \text{year} + b$$

From the graph, it's clear that no line will pass through all data points - so one answer is 'no solution.'. There is definitely a trend, however. Instead of finding the line that passes through all data points, find the line which matches the data as close as possible.

One definition of 'close' is 'least squares'. Come up with a curve fit

$$\hat{y} = ax + b$$

so that the sum squared difference between the curve fit and the actual data is minimized. The solution to this problem is as follows:

First, rewrite the data in matrix form

$$y = ax + b$$

$$Y = BA$$

where

Y is the actual sea ice extent year by year

A is a matrix of the unknowns (a , b), and

B is a matrix of known functions (the basis function).

Here

$$Y_{47 \times 1} = B_{47 \times 2} A_{2 \times 1}$$

$$\begin{bmatrix} 6.9895 \\ 7.533 \\ 6.902 \\ \vdots \\ 4.988 \\ 4.387 \\ 4.083 \end{bmatrix} = \begin{bmatrix} 1979 & 1 \\ 1980 & 1 \\ 1981 & 1 \\ \vdots & \vdots \\ 2004 & 1 \\ 2015 & 1 \\ 2016 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

You can't invert a non-square matrix. However, you can pre-multiply by X transpose

$$B_{2 \times 47}^T Y_{47 \times 1} = B_{2 \times 47}^T B_{47 \times 2} A_{2 \times 1}$$

$B^T B$ is a square matrix - meaning you can usually invert it

$$A = (B^T B)^{-1} B^T Y$$

In Matlab:

```
B = [year, year.^0]

    1979    1
    1980    1
    1981    1
     :
    2023    1
    2025    1
    2025    1

Y = [ice]

    6.895
    7.533
    6.902
     :
    4.230
    4.280
    3.566

A = inv(B'*B)*B'*Y

   -0.08187
   169.72984
```

Meaning the least-squares curve fit for Arctic sea ice extent is:

$$Area \approx -0.08187 \cdot year + 169.72 \quad \text{million km}^2$$

To check, plot the curve fit ontop of the data:

```
plot(y, a, 'b.-', y, X*A, 'r')
```

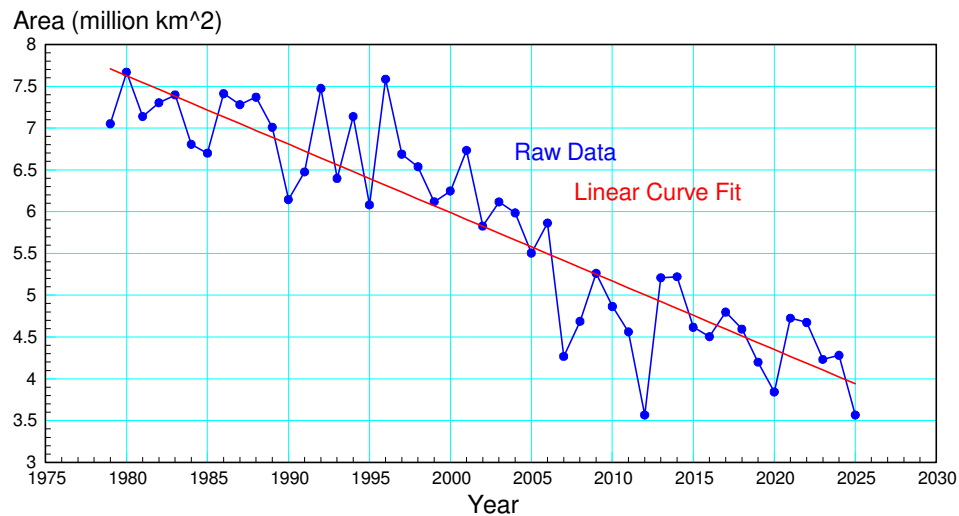


Figure 2: Minimum Arctic Sea Ice: Data (blue) and Linear Curve Fit (red)

From this data, what is the expected date when there will be no sea ice?

$$\text{Area} \approx 0 = -0.08187 \cdot \text{year} + 169.72$$

$$\text{year} = \left(\frac{169.72}{0.08187} \right) = 2073.1$$

You can also do this by finding the roots to a polynomial. For example, find the polynomial with roots at $\{-1, -2, -3\}$

```
>>poly([-1,-2,-3])
1.      6.     11.     6.
```

This is in decreasing powers of x . The polynomial with roots at $\{-1, -2, -3\}$ is:

$$x^3 + 6x^2 + 11x + 6 = 0$$

You can go the other way too: the roots of this polynomial are:

```
>>roots([1,6,11,6])
- 3.
- 2.
- 1.
```

The root (zero crossing) for

$$\text{Area} \approx -0.08187 \cdot \text{year} + 169.72$$

is

```
->roots(A)
2073.1
```

Using a linear curve fit, the data predicts that the Arctic will be ice free for the first time in 5 million years in the year 2073.

Example 2: The mean temperature in Fargo for the month of April since 1942 is:

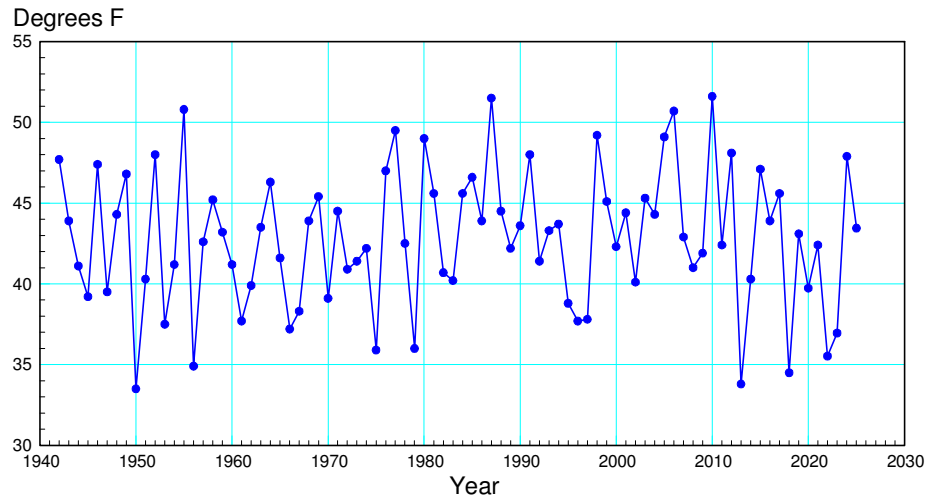


Figure 3: Average Temperature for April in Fargo.

http://weather-warehouse.com/WeatherHistory/PastWeatherData_FargoHectorIntlArpt_Fargo_ND_April.html

Is there a trend? Express this in the form of

$$y = ax + b$$

where

- y is the mean temperature and
- x is the year.

In Matlab: Go to the web site:

Copy all the data

- highlight and copy (control C)
- paste in to notepad (control V)
- remove the NR with a zero
- Copy from notepad
- Go to Matlab. Type in

```
DATA = [
    control V (paste the data)
];
y = DATA(:,1);
F = DATA(:,5);
plot(y,F, '-')
```

Again, rewrite this as

$$Y = BA$$

where B is the basis function. In Matlab:

```
B = [y, y.^0];
Y = F;
A = inv(B'*B)*B'*Y
```

```
0.0074181
28.118
```

```
plot(y,F,'.-',y,B*A,'r')
```

From this data, it appears to be warming up by 0.0074181F each year. Over 80 years, it's gotten 0.593F warmer in Fargo.

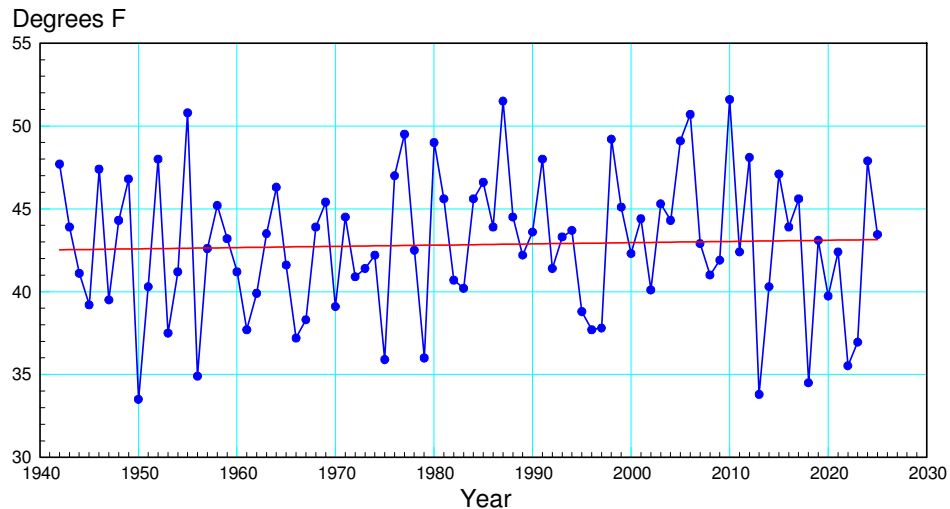


Figure 4: Mean Temperature in April in Fargo along with a Linear Curve Fit: Temperatures appear to be increasing 0.0074F/year

Weighted Least Squares

You can assign different weights to each data point. To do that, define a weighting matrix, Q .

For example, solve find A to minimize the sum squared error

$$Y = BA$$

with weightings Q

$$Q = \begin{bmatrix} q_1 & & & \\ & q_2 & & \\ & & q_3 & \\ & & & \ddots \end{bmatrix}$$

where q_i is the weighting of equation $\#i$. To solve for A , multiply both sides by Q

$$QY = QBA$$

Multiply by B^T

$$B^T QY = B^T QBA$$

Solve for A

$$A = (B^T QB)^{-1} B^T QY$$

Example: Going back to Arctic sea ice, add a weighting for the data equal to

$$w_i = \exp(i/10)$$

In Matlab:

```
Weight = exp(0.1*[1:47]);
plot(year, Weight)
```

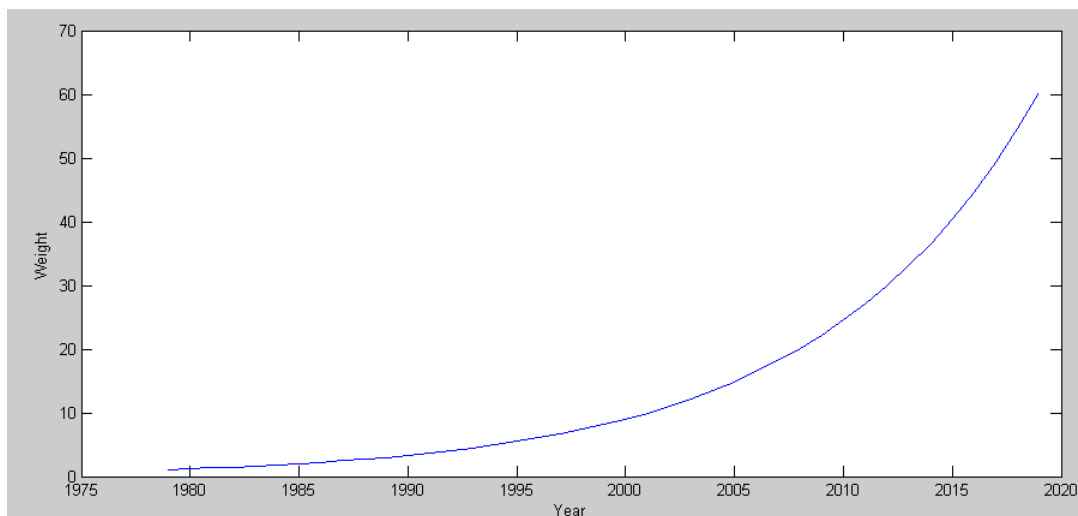


Figure 5: Weighting vs. year: Most recent data is weighted most heavily

```
Q = diag(Weight);
B = [year, year.^0];
A = inv(B'*Q*B)*B'*Q*Ice
```

```
-0.0806
167.15
```

```
plot(y,Ice,'b.-',y,B*A,'r')
```

meaning, if you weight the most recent data more, then the linear curve fit is

$$Ice \approx -0.0806 \cdot year + 167.15 \text{ km}^2$$

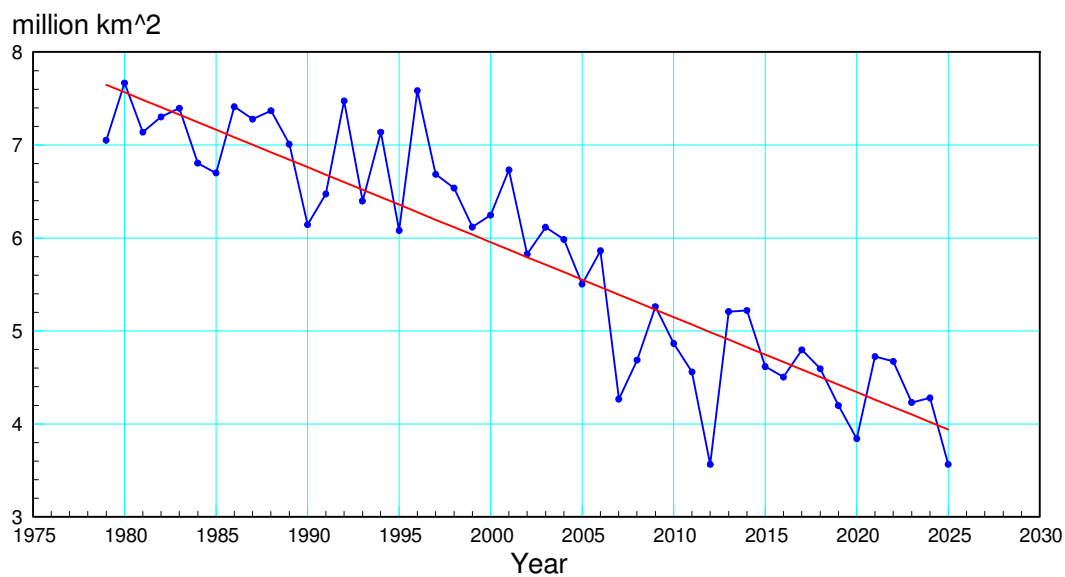


Figure 6: Ice volume each year along with a linear curve fit with exponential weighting towards more recent years

High-Order Curve Fits:

All that matters is that the system is linear in the unknowns (A). Likewise, any set of basis functions can be used for X .

Example: The Atmospheric CO2 levels measured at Mauna Loa Observatory are as follows:

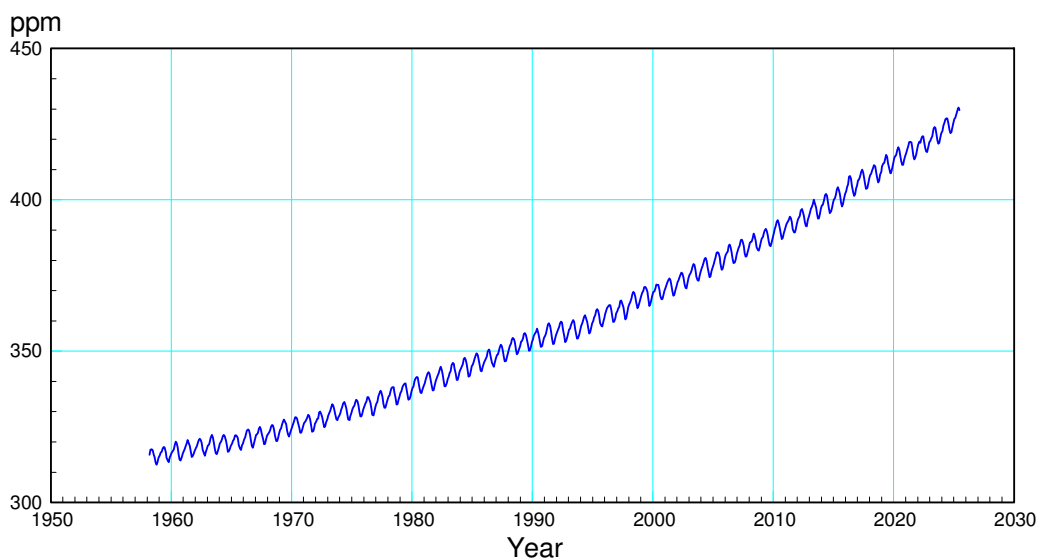


Figure 7: Global CO2 concentrations by year (<https://www.esrl.noaa.gov/gmd/ccgg/trends/full.html>)

Approximate this data as a parabolic curve fit:

$$CO_2 = ay^2 + by + c$$

Solution: Express this as

$$Y = BA$$

$$[CO_2] = \begin{bmatrix} y^2 & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

In Matlab: Go to the web site

ftp://aftp.cmdl.noaa.gov/products/trends/co2/co2_annmean_mlo.txt

Copy this data. Go in to Matlab

```
DATA = [
    paste in the data you just copied
];
```

```
y = DATA(:,3);
CO2 = DATA(:,4);
Y = CO2;
B = [y.^2, y, y.^0];
A = inv(B'*B)*B'*Y
```

```
1.3482e-002
-5.2053e+001
5.0549e+004
```

```
plot(y,CO2,'b.-',y,B*A,'r')
xlabel('Year');
ylabel('ppm');
```

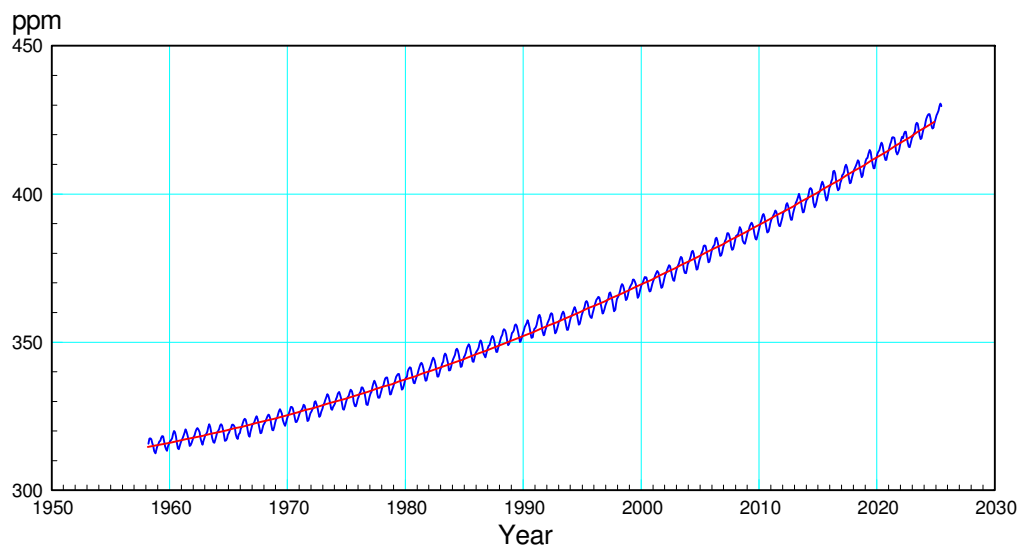


Figure 8: CO2 Levels Measured at Mauna Load (blue) and Parabolic Curve Fit (red)

From this data, when can we expect to see 2000 ppm (what was observed during the Permian extinction)?

The problem you're trying to solve is

$$ay^2 + by + c = 2000$$

Matlab doesn't have a $f(x) = 2000$ function. It *does* have a $f(x)=0$ function, i.e. `roots()`. So, change the problem to fit the solution

$$ay^2 + by + c - 2000 = 0$$

In Matlab, use the prior solution (A) had three elements:

$$A = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

To use the `roots` command, subtract a 3x1 matrix as follows:

$$\text{roots}\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix}\right)$$

or

```
roots(A - [0;0;2000])
```

```
2285.2  
1575.9
```

If nothing changes, we should hit 2000ppm of CO2 in the year 2285.

Summary

- Matlab is a matrix language. In Matlab, you can input, add, subtract, multiply, and invert matrices.
- You can solve N equations with N unknowns fairly easily in Matlab using matrices.
- Least squares allows you to do curve fitting in Matlab as well.