Misc stuff

MATLAB

Saving and reading your workspace

```
>> eig_mov
>> whos
  Name          Size                Bytes  Class       Attributes
  D             2x2                    32  double
  V             2x2                    32  double
  . . .
  cnt           1x1                     8  double
  x             2x361                5776  double
>> save eig_mov_ex.mat
```

Saves workspace in file stuff.mat

```
>> clear
>> whos
>> load eig_mov_ex.mat
>> whos
  Name          Size                Bytes  Class       Attributes
  D             2x2                    32  double
  V             2x2                    32  double
  . . .
  cnt           1x1                     8  double
  x             2x361                5776  double
>>
```

Saving what you type

```
>>diary everythingItype.txt
```

Saves everything you type

```
>>dairy
```

To turn it off

Garbage collection

Any system such as MATLAB that maintains an environment with variables continually being created and destroyed must have a form of "garbage collection" to remove dead (or unused) space.

Unfortunately, MATLAB has no automatic garbage collection mechanism.

The function clear allows the user to manage his workspace and do his own house cleaning.

Even that is not enough, since other temporary arrays might be created and destroyed whenever M-files are run.

In place of garbage collection, there is a MATLAB function called

pack

which saves all the variables in the entire workspace, clears the workspace, and then loads the saved variables.

This is time-consuming, but it is the best way to get some room to work if memory limits start to hinder your progress.

Supressing "Current plot held", etc. messages

be more specific - hold ('on'), etc.

Importance of thinking through how to program something

http://www.joelonsoftware.com/articles/fog0000000319.html

(so the world is not stuck with your mistake forever).

Random numbers

How random are they?

Compare – run multiple times
Restart Matlab – run multiple times
Mac vs PC

(seed)

Linear Algebra (a la Matlab) Review

MATLAB

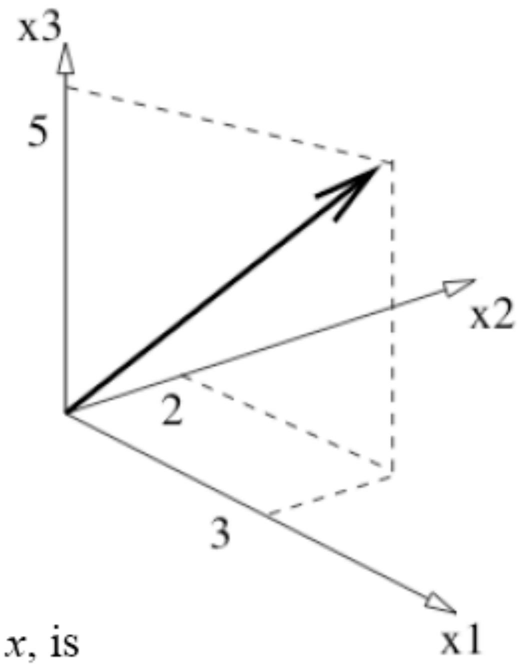
Acknowledgement
This lecture borrows heavily from online lectures/ppt files posted by

David Jacobs at Univ. of Maryland
Tim Marks at UCSD
Joseph Bradley at Carnegie Mellon

Vectors

```
>> a=[2 3 5];
>> norm(a)
6.1644
>> norm(a)^2
38.000
```

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \quad \text{e.g.} \quad \boldsymbol{x} = \begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix}$$



The **length** of x, a.k.a. the **norm** or **2-norm** of x, is

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

e.g.,

$$\|x\| = \sqrt{3^2 + 2^2 + 5^2} = \sqrt{38}$$

Matrices

$$U = [u_{mn}] = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ & \cdots & & \\ u_{m1} & u_{m2} & \cdots & u_{mn} \end{bmatrix}$$

The general matrix consists of *m* rows and *n* columns. It is also known as an *m* x *n* (read *m* by *n*) array.

Each individual number, u_{ij}, of the array is called the *element*

Elements u_{ij} where m=n is called the principal diagonal

Transpose of a Matrix

```
>> a=[6 1;2 5]
a =
     6     1
     2     5
>> a'
ans =
     6     2
     1     5
```

Transpose:

$$C_{m\times n} = A^T{}_{n\times m}$$

$$c_{ij} = a_{ji}$$

$$(A+B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

Examples:

$$\begin{bmatrix} 6 & 2 \\ 1 & 5 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 2 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 2 \\ 1 & 5 \\ 3 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 & 3 \\ 2 & 5 & 8 \end{bmatrix}$$

If $A^T = A$, we say *A* is **symmetric**.

Notice how Matlab looks like math.

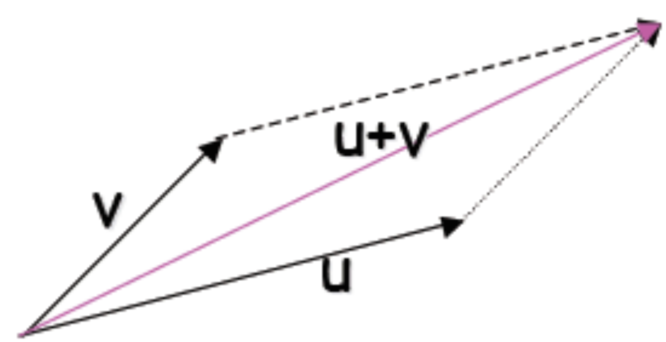
Matrix & Vector Addition

```
>> a=[1; 2]
a =
     1
     2
>> b=[3; 4]
b =
     3
     4
>> c=a+b
c =
     4
     6
>>
```

Vector/Matrix addition is associative and commutative

(A+B)+C=A+(B C);   A+B=B+A

$$u + v = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \end{bmatrix}$$



NO LOOPS
Looks like Math – just add them.

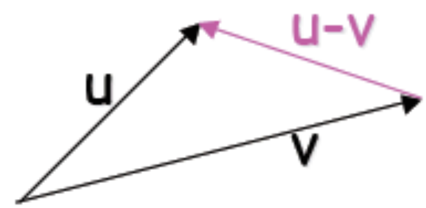
Matrix and Vector Subtraction

Same as addition

Vector/Matrix subtraction is also associative and commutative

(A-B)-C =A-(B- C); A-B=B-A

$$u - v = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 - v_1 \\ u_2 - v_2 \end{bmatrix}$$



Matrix and Vector Scaling

$$\boldsymbol{z} = \alpha \boldsymbol{x}$$

for a scalar α then

$$\boldsymbol{z} = \alpha \begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix} = \begin{pmatrix} 3\alpha \\ 5\alpha \\ 2\alpha \end{pmatrix}$$

```
>> x=[1 2 3]
x =
     1     2     3
>> y=3*x
y =
     3     6     9
>>
```

For addition and subtraction, the size of the matrices must be the same

$A_{nm} + B_{nm} = C_{nm}$

For scalar multiplication, the size of A_{nm} does not matter

All three of these operations do not differ from their ordinary number counterparts

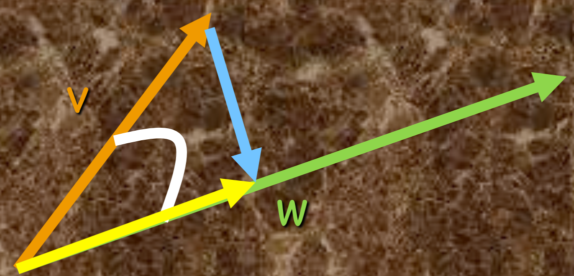
The operators work element-by-element through the array, $a_{mn}+b_{mn}=c_{mn}$

Vector Multiplication: inner or dot product

The inner product of vector multiplication is a SCALAR

$$v \cdot w = (x_1, x_2) \cdot (y_1, y_2) = \| v \| \cdot \| w \| \cos \alpha$$

$$v \cdot w = (x_1, x_2) \cdot (y_1, y_2) = x_1 y_1 + x_2 y_2$$



Projection of one vector (orange) onto another

(green, result - projection - yellow).

Dot product is zero for perpendicular vectors.

The inner/dot product can be represented as row matrix multiplied by a column matrix. A row matrix can be multiplied by a column matrix, in that order, only if they each have the same number of elements!

```
>> x=[1 2]
x =
     1     2
>> y=[2 1]
y =
     2     1
>> x*y'
ans =
     4
>> y=[-2 1]
y =
    -2     1
```

```
>> x*y'
ans =
     0
>> y=[2 -1]
y =
     2    -1
>> x*y'
ans =
     0
>>
```

$$a = \begin{bmatrix} 6 \\ 2 \\ -3 \end{bmatrix} \qquad b = \begin{bmatrix} 4 \\ 1 \\ 5 \end{bmatrix}$$

$$\begin{aligned} a \cdot b &= a^T b \\ &= \begin{bmatrix} 6 & 2 & -3 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \\ 5 \end{bmatrix} \\ &= 6 \cdot 4 + 2 \cdot 1 + (-3) \cdot 5 \\ &= 11 \end{aligned}$$

Several ways to properly calculate the dot product of two vectors

```
>>sum(a.*b)
```

element by element multiplication (.*), then sum the results – based on definition.

Or making it look like matrix multiplication

```
>>a'*b
>>a*b'
```

Or using matlab function

```
>>c=dot(a,b)
```

The outer product

A column vector multiplied by a row vector. The outer product of vector multiplication is a MATRIX.

```
>> a=[6 2 -3]
a =
     6     2    -3
>> b=[4;1;5]
b =
     4
     1
     5
>> c=b*a
c =
    24     8   -12
     6     2    -3
    30    10   -15
>>
```

Matrix Multiplication

Two matrices can be multiplied together if and only if the number of columns in the first equals the number of rows in the second.

$$C_{n \times p} = A_{n \times m} B_{m \times p}$$

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

In MATLAB, the * symbol represents matrix multiplication :

```
>> a=[1 2 3;3 2 1]
a =
     1     2     3
     3     2     1
>> b=[4 5;10 2;2 10]
b =
     4     5
    10     2
     2    10
>> c=a*b
c =
    30    39
    34    29
>>
```

```
>> a(1,☺
ans =
     1     2     3
>> b(:,1)
ans =
     4
    10
     2
>> a(1,:)*b(:,1)
ans =
    30
>> a(1,:)*b(:,2)
ans =
    39
>> a(2,:)*b(:,2)
ans =
    29
>> a(2,:)*b(:,1)
ans =
    34
```

- Matrix multiplication is not commutative!

$$A_{n\times n}B_{n\times n} \neq B_{n\times n}A_{n\times n}$$

```
>> c=a*b
c =
   30   39
   34   29
>> c=b*a
c =
   19   18   17
   16   24   32
   32   24   16
```

- Matrix multiplication is distributive and associative

A(B+C) = AB + BC

(AB)C = A(BC)

Matrices can represent sets of equations

$$a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n=b_1$$

$$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n=b_2$$

$$\ldots$$

$$a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n=b_n$$

What's the matrix representation?

$$A=\begin{bmatrix} a11 & a12 & \ldots & a1n \\ a21 & a22 & \ldots & a2n \\ & \ldots & & \\ an1 & an2 & \ldots & ann \end{bmatrix} \quad x=\begin{bmatrix} x1 \\ x2 \\ . \\ xn \end{bmatrix} \quad b=\begin{bmatrix} b1 \\ b2 \\ . \\ bn \end{bmatrix}$$

$$Ax=b$$

Determinant of a Matrix

```
>> a=magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
>> det(a)
Ans
   -360
>>
```

Determinant: A must be square

$$\det\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

Example: $\det\begin{bmatrix} 2 & 5 \\ 3 & 1 \end{bmatrix} = 2 - 15 = -13$

$$\det\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Inverse of a Matrix

```
>> a=[1 2 3;4 5 6; 7 8 9]
a =
     1     2     3
     4     5     6
     7     8     9
>> ainv=inv(a)
ainv =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
>> ainv*a
ans =
    1.0000         0   -0.0000
         0    1.0000         0
         0    0.0000    1.0000
>> a*ainv
ans =
    1.0000         0   -0.0000
   -0.0000    1.0000         0
    0.0000         0    1.0000
>>
```

If A is a square matrix, the **inverse** of A, called A^{-1}, satisfies

$$AA^{-1} = I \quad \text{and} \quad A^{-1}A = I,$$

Where I, the **identity matrix**, is a diagonal matrix with all 1's on the diagonal.

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If square matrix invertible, has same right and left inverse.

For a 2-D matrix

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

the principal diagonal elements switch positions

$$A^{-1} = \frac{\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}}{|A|}$$

the off diagonal elements change sign

the determinant

Example: $$\begin{bmatrix} 6 & 2 \\ 1 & 5 \end{bmatrix}^{-1} = \frac{1}{28}\begin{bmatrix} 5 & -2 \\ -1 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 2 \\ 1 & 5 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 6 & 2 \\ 1 & 5 \end{bmatrix} = \frac{1}{28}\begin{bmatrix} 5 & -2 \\ -1 & 6 \end{bmatrix} \cdot \begin{bmatrix} 6 & 2 \\ 1 & 5 \end{bmatrix} = \frac{1}{28}\begin{bmatrix} 28 & 0 \\ 0 & 28 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Square matrices with inverses are said to be nonsingular

Not all square matrices have an inverse. These are said to be singular.

Square matrices with determinants = 0 are singular. (determinant in denominator)

Rectangular matrices are always singular.

Right- and Left- Inverse

If a matrix G exists such that GA = I, than G is a left-inverse of A

If a matrix H exists such that AH = I, than H is a right-inverse of A

Rectangular matrices may have right- or left- inverses, but they are still singular.

For

$$A = m\,x\,n,\; m > n: \quad \text{we have a left inverse } \left(A^T A\right)^{-1} A^T A = I_n,$$

$$A_{left}^{-1} = \left(A^T A\right)^{-1} A^T$$

$$A = m\,x\,n,\; n > m: \quad \text{we have a left inverse } AA^T \left(AA^T\right)^{-1} = I_m,$$

$$A_{right}^{-1} = A^T \left(AA^T\right)^{-1}$$

```
>> a=[1 2 3;4 5 6]
a =
     1     2     3
     4     5     6
>> ainv=a'*inv(a*a')
ainv =
   -0.9444    0.4444
   -0.1111    0.1111
    0.7222   -0.2222
>> a*ainv
ans =
    1.0000   -0.0000
   -0.0000    1.0000
>> >> det(a'*a)
ans =
     0
```

Right inverse exists, but left doesn't.

Matrix Division in Matlab

In ordinary math, division (a/b) can be thought of as a*(1/b) or a*b^{-1}.

There really is no such thing as matrix division in any simple sense.

A unique inverse matrix of b, b^{-1}, only potentially exists if b is square.

Also, matrix multiplication is not communicative, unlike ordinary multiplication.

Matrix Division in Matlab

/ : B/A (right division) is roughly the same as B*inv(A).

A and B must have the same number of columns for right division.

More precisely, B/A = (A'\B')'.

Matrix Division in Matlab

\ : If A is a square matrix, A\B (left division) is roughly the same as inv(A)*B, except it is computed in a different way. A and B must have the same number of rows for left division.

```
a =
    1     2
    3     4
>> b
b =
    1
    2
>> c=a\b
c =
        0
   0.5000
>> ainv=inv(a)
```

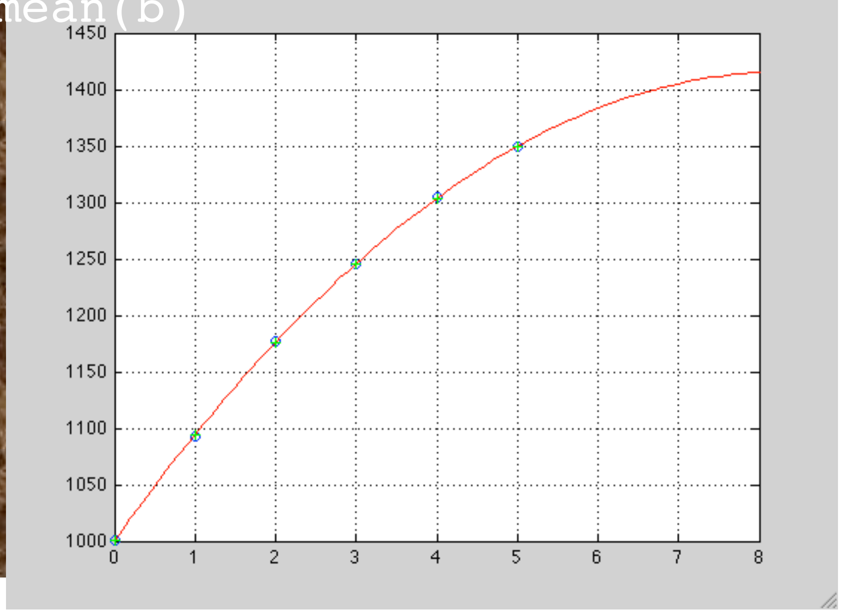
```
ainv =
   -2.0000    1.0000
    1.5000   -0.5000
>> ainv*b
ans =
         0
    0.5000
>>>> a*c
ans =
     1
     2
```

Matrix Division in Matlab

\ : If A is an m-by-n matrix (not square) and b is a matrix of m rows, Ax=b is solved by least squares using A\b (left division).

```
>> A
A =
1.0000       0         0
1.0000  1.0000    0.5000
1.0000  2.0000    2.0000
1.0000  3.0000    4.5000
1.0000  4.0000    8.0000
1.0000  5.0000   12.5000
>> b
b =
        1001
        1093
        1177
        1245
        1305
        1349
```

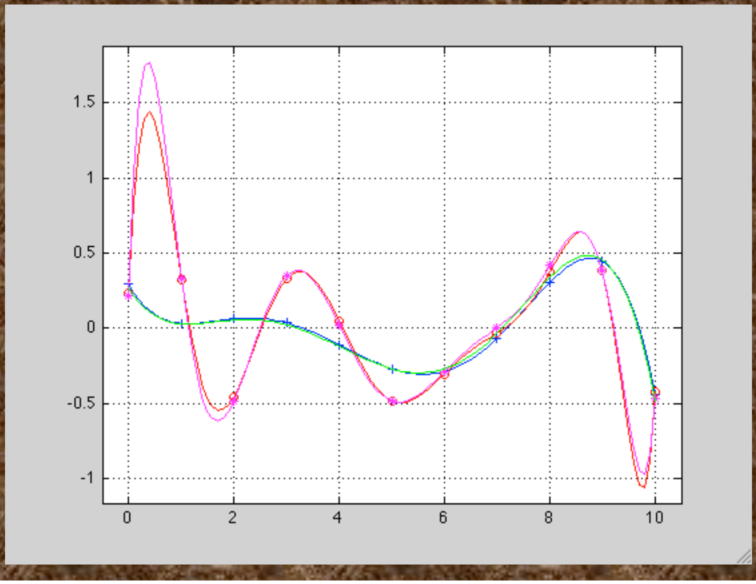
```
>> x_=A\b
x_ =
   1.0e+03 *
    1.0004
    0.0998
   -0.0120
>> (A*x_-b)/mean(b)
ans =
   -0.0005
    0.0011
   -0.0008
    0.0008
   -0.0011
    0.0005
```



Matlab also has routines to do polynomial fits (positive powers only).

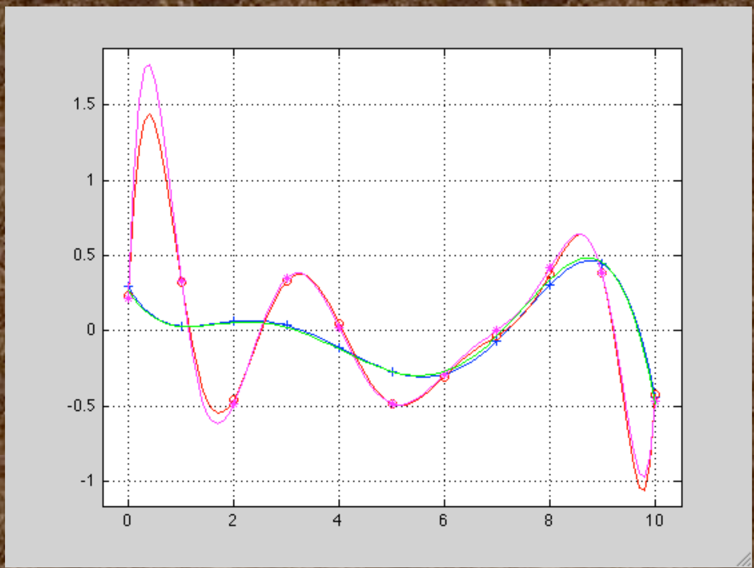
Data – 11 points (red circles)

5^{th} order polynomial fit to 11 points – fewer parameters than data – get LS fit (blue line, blue '+') – does not go through data points (but misfit "minimized").



Data – 11 points (red circles)

10th order polynomial fit to 11 points – same number parameters as data – get exact solution (red line). Goes though each point exactly.

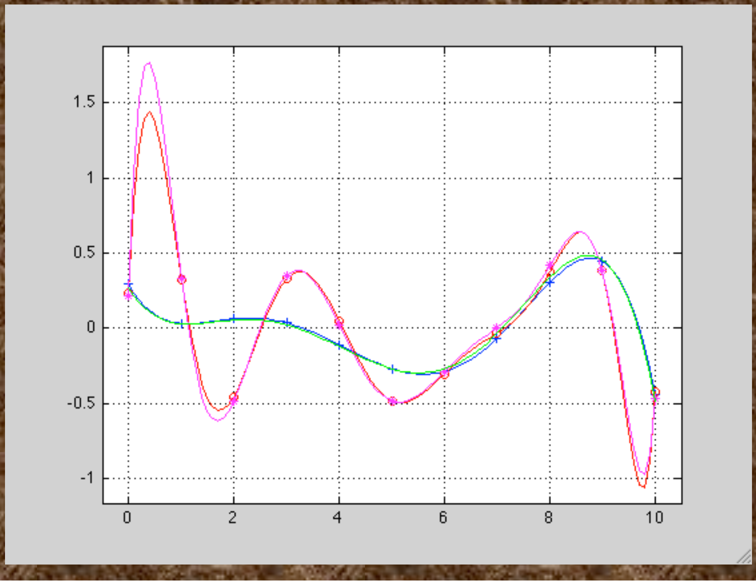


Now add some noise.

Data – 11 points (red circles)

Magenta and green – 5^{th} and 10^{th} order polynomials fit to data with 10% noise.

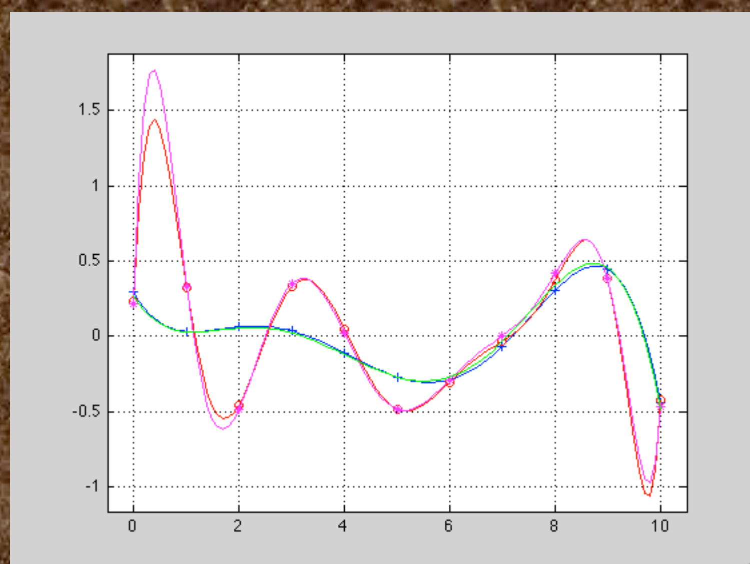
The fits to the noisy and perfect data look pretty much the same when plotted.



```
>> poly_demo
p5 =
    -0.0010    0.0215   -0.1629    0.4980   -0.6166    0.2926
pn5 =
    -0.0009    0.0209   -0.1559    0.4696   -0.5767    0.2733
p10 =
  Columns 1 through 11
    0.0000   -0.0003    0.0051   -0.0474    0.2251   -0.3195   -1.7232    8.6334  -14.0467    7.3647    0.2333
pn10 =
  Columns 1 through 11
    0.0000   -0.0002    0.0024   -0.0145   -0.0298    0.9338   -5.6002   15.8159  -21.1548   10.1628    0.2137
```

The models (the values for polynomial coefficients), however, are quite different.

Compare "stability" of the solutions.



Array Operators (review)

+ Addition

- Subtraction

.* Element-by-element multiplication

./ Element-by-element division.
(A./B: divides A by B by element)

.\ Element-by-element left division
(A.\B   divides B by A by element)

.^ Element-by-element power

.' Unconjugated array transpose
(does not take complex conjugate, unlike a regular [no dot] matrix transpose)

Some Special Matrices

Square matrix: m (# rows) = n (# columns)

Symmetric matrix: subset of square matrices where $A^T = A$

Diagonal matrix: subset of square matrices where elements off the principal diagonal are zero, $a_{ij} = 0$ if $i \neq j$

Identity or unit matrix: special diagonal matrix where all principal diagonal elements are 1

```
>> a=[1 2 3;4 5 6;7 8 9]
a =
   1   2   3
   4   5   6
   7   8   9
>> c=trace(a)
c =
  15


>> a=[.96 -.28; .28 .96]
a =
  0.9600  -0.2800
  0.2800   0.9600
>> inv(a)
ans =
  0.9600   0.2800
 -0.2800   0.9600
>> a'*a
ans =
   1   0
   0   1
>>
```

trace of a Matrix is Tr(A) =

$$\sum_{i=1}^{N} a_{ii}$$

(the sum of the diagonal entries)

In matlab use trace(A)

a matrix A is **orthonormal** if

$$A^T = A^{-1}$$

and in this case

$$AA^T = I$$