MATLAB

CS*101* `lec24`

Basic Statistics

2019-12-11

Announcements

quiz: `quiz24` due on Thurs 12/12

lab: `lab` on Fri 13/12

hw: `hw12` due today

hw: `hw13` due Wed 18/12

final exam: 27 Dec on `Lec01 to Lec25`

Roadmap

Week	Session 1	Session 2
Week 1 (9/9)	L0 Intro	L1 PY - Intro
Week 2 (16/9)	L2 PY – Data& Bool	L3 PY – if & Loops
Week 3 (23/9)	L4 PY - Lists	L5 PY - Funcs
30/9	L6 PY - Files	X
9/10, 12/10	L7 PY - Muta (9/10)	Exam 01 (12/10)
Week 6 (14/10)	L8 PY - Dict	L9 PY - IO
Week 7 (21/10)	L10 PY - Numpy	L11 PY - Plot
Week 8 (28/10)	L12 PY - State	L13 PY - Rand
Week 9 (4/11)	L14 PY – De-bug	L15 PY - Eqn
Week 10 (11/11)	L16 Brute	Exam 02
Week 11 (18/11)	L17 PY - Opt II	L18 PY - Mod
Week 12 (25/11)	L19 PY – Sym I	L20 PY – Sym II
Week 13 (2/12)	L21 ML - Intro	L22 ML - IO
Week 14 (9/12)	L23 ML - Poly	L24 ML - Stats
Week 15 (16/12)	L25 ML - Fit	Re-view
Week 16 (23/12)	Study Week	
Week 17 (30/12)	Finals	

Objectives

A. Calculate basic statistics of arrays using MATLAB
B. Generate random numbers in arrays
C. Use interpolation to find function values
D. Use left-division to solve matrix equations efficiently

MATLAB Review

Error in lec 23

To define an inline function, use:

```
f = @(x) cos(x)
```

not

```
f = (@x) cos(x)
```

Question

```
A = [ 1 0 ; 4 5 ];
A( A > 0 )
```

What is the value of `ans`?

A `[ 1 0 ; 1 1 ]`

B `[ 1 0 , 1 1 ]`

C `[ 1 4 5 ]'` ★★★

D `1` (true)

Question

```
x = 10;
if ( x / 2 ) <= 5 | ( x == 1 )
  x = x + 1;
end
if x ~= 10 & x <= x
  x = x * 2;
end
```

What is the final value of `x`?

A `10`

B `11`

C `20`

D `22`

Question

```
x = 10;
if ( x / 2 ) <= 5 | ( x == 1 )
  x = x + 1;
end
if x ~= 10 & x <= x
  x = x * 2;
end
```

What is the final value of `x`?

- A 10
- B 11
- C 20
- D 22 ★★★

Statistics

Example: Seeding RNGs

```
rng( 101 );  % seed the random number generator
x = linspace( 0,2*pi,101 )';
y = x/50 + 0.002 * randn( 101,1 );

figure
plot( x,y,'.' );
```

Statistical quantities

Many operations are available:

A. `mean`, `median`, `std` deviation

Statistical quantities

Many operations are available:

A. `mean`, `median`, `std` deviation
B. `min`, `max`,
C. `range` - difference between `max` and `min`

Statistical quantities

Many operations are available:

A. `mean`, `median`, `std` deviation
B. `min`, `max`,
C. `range` - difference between `max` and `min`
D. `sort`
E. `sum`, `cumsum`
F. `prod`, `cumprod`

Statistical quantities

Many operations are available:

A. `mean`, `median`, `std` deviation
B. `min`, `max`,
C. `range` - difference between `max` and `min`
D. `sort`
E. `sum`, `cumsum`
F. `prod`, `cumprod`
G. `boxplot`, `hist`
H. more...

Statistical quantities

```
x = [ 1 2 3 4 5 ];
A = [ -5 0 10 ; -4 1 9 ; -3 2 8 ; -2 3 7 ; -1 4 6 ]
```

Statistical quantities

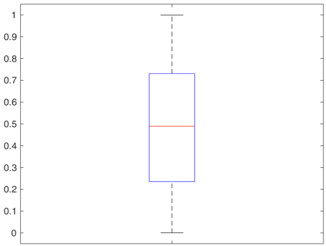
```
x = [ 1 2 3 4 5 ];
A = [ -5 0 10 ; -4 1 9 ; -3 2 8 ; -2 3 7 ; -1 4 6 ]

sort( x )
sort( x,'descend' )
sort( A )   % sort elements within a column
                 by ascending order
sort( A, 1 )   % sort elements within a column
                   by ascending order
sort( A, 2 )   % sort elements within a row
                    by ascending order

sortrows( A )    % change the position of whole row
             based on ascending order in column 1
sortrows( A,3 )   % change the position of whole row
             based on ascending order in column 3
```

Statistical quantities

```
x = [ 1 2 3 4 5 ];
A = [ -5 0 10 ; -4 1 9 ; -3 2 8 ; -2 3 7 ; -1 4 6 ]

cumsum( x )
ans =    1     3     6    10    15
```

Statistical quantities

```
x = [ 1 2 3 4 5 ];
A = [ -5 0 10 ; -4 1 9 ; -3 2 8 ; -2 3 7 ; -1 4 6 ]

cumsum( x )
ans =     1      3      6     10     15

y = rand( 1000,1 );
boxplot( y )
```



Example: Brexit polling

```
poll = importdata('brexit.csv')

poll =
  struct with fields:
          data: [179×5 double]
      textdata: {'"Date"'   '"Remain"'   '"Leave"'
                 '"Undecided"'   '"Sample"'}
    colheaders: {'"Date"'   '"Remain"'   '"Leave"'
                 '"Undecided"'   '"Sample"'}

plot( poll.data(:,2) );
plot( poll.data(:,3) );
```

Example: Brexit polling

```
poll = importdata('brexit.csv')

poll =
  struct with fields:
          data: [179×5 double]
      textdata: {'"Date"'   '"Remain"'   '"Leave"'
                 '"Undecided"'   '"Sample"'}
    colheaders: {'"Date"'   '"Remain"'   '"Leave"'
                 '"Undecided"'   '"Sample"'}

plot( poll.data(:,2) );
plot( poll.data(:,3) );
```

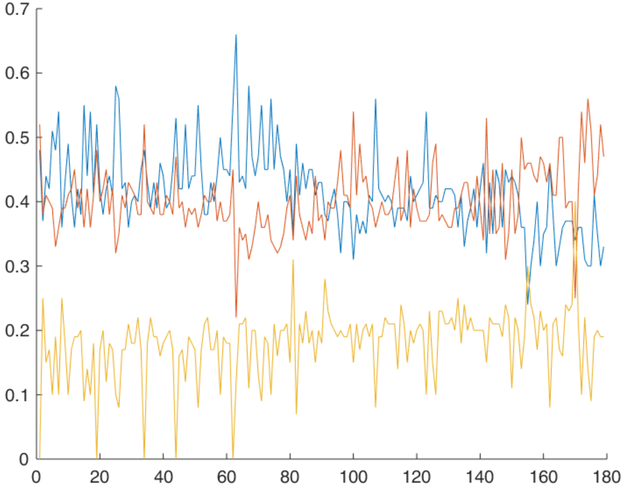
oh no! our plotted data disappeared!

Example: Brexit polling

Raw Data

```
poll = importdata('brexit.csv');
hold on;  % make plots persistent until closed
plot( poll.data(:,2) );  %Remain - red
plot( poll.data(:,3) );  %Leave - blue
plot( poll.data(:,4) );  %Undecided - yellow
```

Example: Brexit polling



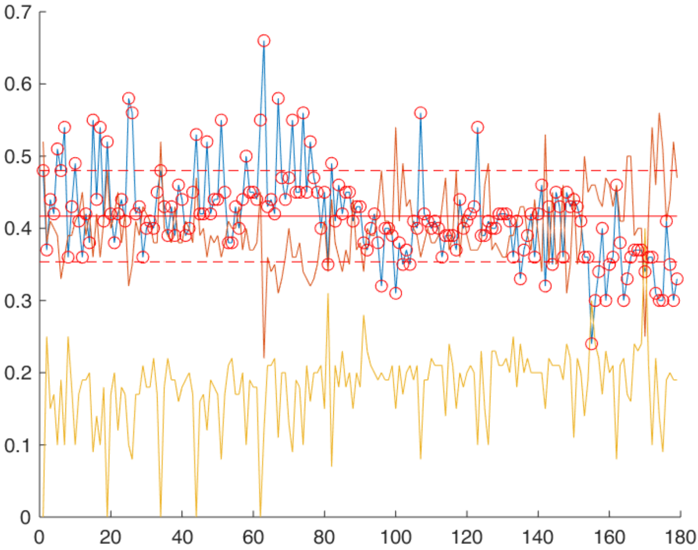
Example: Brexit polling

How the ”Remain” change through time?
Looking at the average:

```
n = numel(poll.data(:,2)); %==prod(size(A))

mean_r = mean( poll.data(:,2) ) * ones( n+1,1 );
stdev_r = std( poll.data(:,2) );
std_rp = mean_r+stdev_r;
std_rm = mean_r-stdev_r;
hold on
plot( poll.data(:,2), 'ro' ); %actual data
plot( 0:n,mean_r, 'r-' ); %average
plot( 0:n,std_rp, 'r--' ); %+ std dev
plot( 0:n,std_rm, 'r--' ); %- std dev
```

Example: Brexit polling

Looking at the average:



Example: Brexit polling

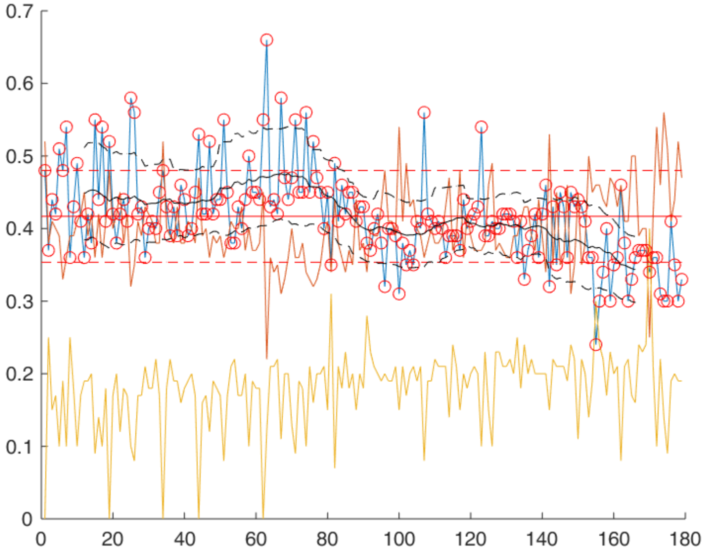
How the ”Remain” change through time?

Looking at the 25-day moving average:

```
n = numel(poll.data(:,2));

%rolling_mean() and rolling_std from your lab
mean_r = rolling_mean( poll.data(:,2)', 25 );
stdev_r = rolling_std( poll.data(:,2)', 25 );
std_rp = mean_r+stdev_r;
std_rm = mean_r-stdev_r;
hold on
plot( poll.data(:,2), 'ro' );
plot( 0:n-1,mean_r, 'k-' );
plot( 0:n-1,std_rp, 'k--' );
plot( 0:n-1,std_rm, 'k--' );
```

Example: Brexit polling

Looking at the 25-day moving average:



Interpolation

Interpolation

Generally, means drawing a line between data values to approximate data at other points.

Interpolation

Generally, means drawing a line between data values to approximate data at other points.

"Inter" means between. Distinguish interpolation from other kinds of estimation!

`interp1( x,y,x0 )` # `1` in `interp1` is number `1` not small L;

Interpolation

Generally, means drawing a line between data values to approximate data at other points.

“Inter” means between. Distinguish interpolation from other kinds of estimation!

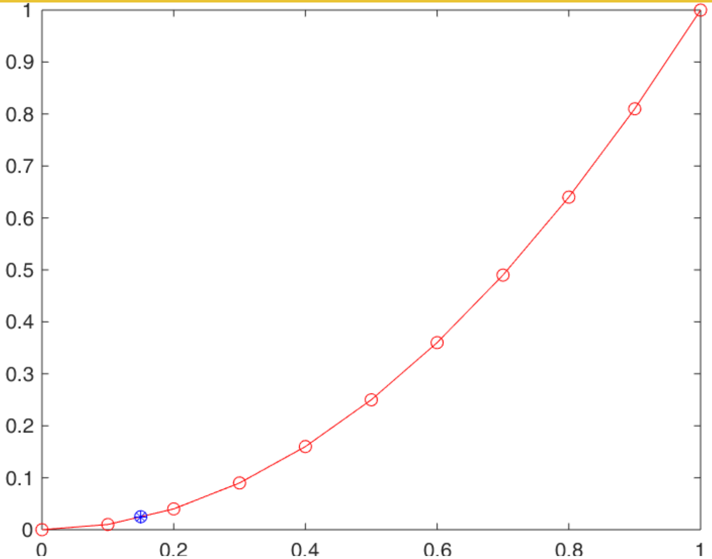
`interp1( x,y,x0 )` # `1` in `interp1` is number `1` not small L;

```
x = linspace( 0,1,11 );
y = x .^ 2;
plot( x,y,'ro-' );

x_est = 0.15;
y_est = interp1( x,y,x_est );
```

Interpolation

```
hold on
plot( x,y,'ro-' )
plot( x_est,y_est,'bo' )
```

Interpolation - error



Interpolation - error

Default: ’linear’

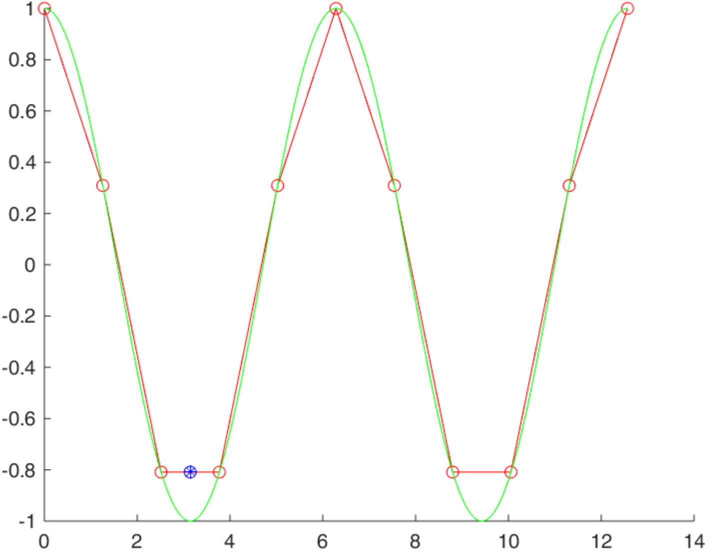
This works well if points are close, but can have problems:

```
x = linspace( 0,4*pi,11 );
y = cos( x );

x_real = linspace( 0,4*pi,501 );
y_real = cos( x_real );

x_est = pi;
y_est = interp1( x,y,x_est );
% or interp1( x,y,x_est, 'linear' );
```

Interpolation - error

```
hold on
plot( x,y,'ro-' )
plot( x_est,y_est,'bo' )
plot( x_real,y_real,'g-' )
```

Interpolation - error



Interpolation - Others

Other options include `'nearest'` and `'pchip'` and more...

`'nearest'` = nearest y-value in the actual data at the required x-point

Interpolation - Others

Other options include `'nearest'` and `'pchip'` and more...

`'nearest'` = nearest y-value in the actual data at the required x-point

`'pchip'` = y-value at the required x => interpolation from a cubic equation using at least 4 nearest x-points

```
x = linspace( 0,4*pi,11 );
y = cos( x );

x_real = linspace( 0,4*pi,501 );
y_real = cos( x_real );
```

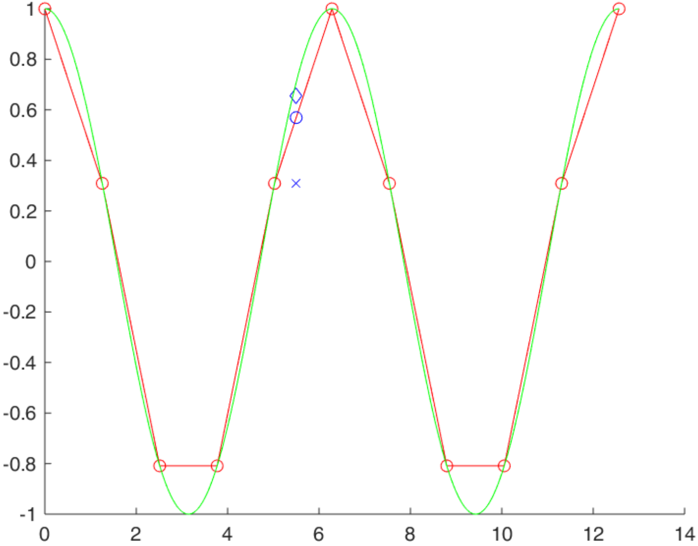
Interpolation - Others

```
x_est_linear = 5.5;
y_est_linear = interp1( x,y,x_est_linear );

x_est_nearest = 5.5;
y_est_nearest = interp1( x,y,x_est_nearest,
                    'nearest' );

x_est_cubic = 5.5;
y_est_cubic = interp1( x,y,x_est_cubic,'pchip');
```

Interpolation - Others

```
hold on
plot( x,y,'ro-' )
plot( x_est_linear,y_est_linear,'bo' )
plot( x_est_nearest,y_est_nearest,'bx' )
plot( x_est_cubic,y_est_cubic,'bd' )
plot( x_real,y_real,'g-' )
```

Interpolation - Others



Matrix Equations

Matrix Equations

$$\underline{\underline{A}}\,\underline{x} = \underline{y}$$

This is the canonical equation of engineering.

Matrix Equations

$$\underline{\underline{A}}\,\underline{x} = \underline{y}$$

This is the canonical equation of engineering.

No matter what your kind of equation, when it comes time to solve a problem numerically, this is the general equation you will probably use.

Normally, we want to know $\underline{x}$. How to solve for $\underline{x}$?

Solve for x

Matrix:

$$\underline{\underline{A}}\,\underline{x} = \underline{y}$$

Solve for x

Matrix:

$$\underline{\underline{A}}\underline{x} = \underline{y}$$

Scalar variable:

$$3x = y$$

Solve for x

Matrix:

$$\underline{\underline{A}}\underline{x} = \underline{y}$$

Scalar variable:

$$3x = y$$

$$(1/3) * 3x = (1/3)y$$

$$x = (1/3)y$$

Here: $1/3 = 3^{-1}$

Matrix Equations

$$\underline{\underline{A}}\,\underline{x} = \underline{y}$$

Formally, the solution is:

$$\underline{\underline{A}}^{-1}\underline{\underline{A}}\,\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

Matrix Equations

$$\underline{\underline{A}}^{-1}\underline{\underline{A}}\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

$$\underline{\underline{I}}\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

$$\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

Matrix Equations

```
A = [ 2 -1 0 ; -1 2 -1 ; 0 -1 2 ];
y = [ 1 2 3 ]';
x = inv( A ) * y;
```

$$\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

$$\underline{x} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Matrix Equations

```
A = [ 2 -1 0 ; -1 2 -1 ; 0 -1 2 ];
y = [ 1 2 3 ]';
x = inv( A ) * y;
```

$$\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

$$\underline{x} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\underline{x} = \begin{pmatrix} 3/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 3/4 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Matrix Equations

```
A = [ 2 -1 0 ; -1 2 -1 ; 0 -1 2 ];
y = [ 1 2 3 ]';
x = inv( A ) * y;
```

$$\underline{x} = \underline{\underline{A}}^{-1}\underline{y}$$

$$\underline{x} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\underline{x} = \begin{pmatrix} 3/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 3/4 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2.5 \\ 4 \\ 3.5 \end{pmatrix}$$

Matrix Equations

$$\begin{pmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{pmatrix}$$

Most engineering equations have most nonzero values near the diagonal.

This means most of the matrix is zero, and efficient to store and calculate with.

Matrix Equations

$$\begin{pmatrix} 0.268 & 0.072 & 0.019 & 0.005 & 0.001 \\ 0.072 & 0.287 & 0.077 & 0.021 & 0.005 \\ 0.019 & 0.077 & 0.288 & 0.077 & 0.019 \\ 0.005 & 0.021 & 0.077 & 0.287 & 0.072 \\ 0.001 & 0.005 & 0.019 & 0.072 & 0.268 \end{pmatrix}$$

The inverse of a matrix does not have the same properties!

This means inverse of a matrix is NOT efficient to store and calculate with.

Matrix Equations

MATLAB's solution: left-division uses more sophisticated techniques:

```
A = [  4 -1 0 0 0 ; ...  %... tells matlab this
      -1 4 -1 0 0 ; ...  %  sentence has not
      0 -1 4 -1 0 ; ...  %  ended and continues
      0 0 -1 4 -1 ; ...  %  on next line
      0 0 0 -1 4 ];

f = [ 1 2 3 4 5 ]';
x = A \ y;
```

Question

$$\begin{pmatrix} 3 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 4 \end{pmatrix} \underline{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Assuming $\underline{\underline{A}}\underline{x} = \underline{y}$, how can we correctly solve for `x`?

- A `x = inv( A ) * y;`
- B `A * x == y;`
- C `x = inv(A) .* y;`
- D `x = A \y;`
- E Both A and D are correct

Question

$$\begin{pmatrix} 3 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 4 \end{pmatrix} \underline{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Assuming $\underline{\underline{A}}\underline{x} = \underline{y}$, how can we correctly solve for `x`?

A `x = inv( A ) * y;`

B `A * x == y;`

C `x = inv(A) .* y;`

D `x = A \y;`

E Both A and D are correct ⋆⋆⋆

Timing Code

We can compare solution speed with `tic` and `toc`:

```
A = [ 1 -2 0 0 0 ; -2 1 -2 0 0 ; ...
      0 -2 1 -2 0 ; 0 0 -2 1 -2 ; ...
      0 0 0 -2 1 ];
b = [ 1 2 3 4 5 ]';

tic; x1 = inv(A) * b; toc
tic; x2 = A \ b; toc
```

Summary

A. Statistics in MATLAB using `Mean`, `Median`, `Std`
B. Interpolation with different methods, `interp1`
C. left division operator (\) for finding inverse
D. Timing with `tic` then `toc`.