Fitting Data
p = [ 1 2 3 4 ];
Which polynomial does p represent?

A  \( x + 2x^2 + 3x^3 + 4x^4 \)
B  \( 4 + 3x + 2x^2 + x^3 \)
C  \( 1 + 2x + 3x^2 + 4x^3 \)
D  \( 4x + 3x^2 + 2x^3 + x^4 \)
p = [ 1 2 3 4 ];

Which polynomial does p represent?

A  \( x + 2x^2 + 3x^3 + 4x^4 \)
B  \( 4 + 3x + 2x^2 + x^3 \)
C  \( 1 + 2x + 3x^2 + 4x^3 \)
D  \( 4x + 3x^2 + 2x^3 + x^4 \)
p = [ 1 0 0 0 -1 ];
x = polyval( p,5 );

What is the final value of x? (No calculator necessary!)

A 624
B 124
C 3120
D -724
p = [ 1 0 0 0 -1 ];
x = polyval( p,5 );

What is the final value of x? (No calculator necessary!)

A 624 ★
B 124
C 3120
D -724
Regression
Given \( n + 1 \) points (or more), we can fit an \( n \)th order polynomial as a model.

\[
x = \text{rand}(3,1);
y = \text{rand}(3,1);
xf = 0:0.01:1;
pf = \text{polyfit}(x,y,2);
yf = \text{polyval}(pf,xf);
\text{plot}(x,y,'rx', \ xf,yf,'r-');
\]
Given \( n + 1 \) points (or more), we can fit an \( n \)th order polynomial as a model.

```matlab
x = rand( 5,1 );
y = rand( 5,1 );
xf = 0:0.01:1;
pf = polyfit( x,y,3 );
yf = polyval( pf,xf );
plot( x,y,'rx', xf,yf,'r-' );
```
n = input( 'Give an order' );
x = rand( n+1,1 );
y = rand( n+1,1 );
xf = 0:0.01:1;
pf = polyfit( x,y,n );
yf = polyval( pf,xf );
plot( x,y,'rx', xf,yf,'r-' );
High-order polynomials tend to wobble a lot, and overshoot at the ends.

A simpler fit is often better!

```matlab
x = rand( 11,1 );
y = rand( 11,1 );
xf = 0:0.01:1;
pf3 = polyfit( x,y,3 );
pf10 = polyfit( x,y,10 );
figure;
plot( x,y,'rx',xf,polyval( pf3,xf ),'r-',xf,polyval( pf10,xf ),'g-');
ylim( [ -5 5 ] );
```
Fitting Polynomials

- We would like to estimate the error of a fit.
- Traditionally, this is the $L^2$ metric:

$$L^2 = |y_{\text{known}} - y_{\text{fitted}}|^2$$

- This is always positive and should be close to zero.

```matlab
x = rand( 11,1 );
y = rand( 11,1 );
xf = 0:0.01:1;
pf = polyfit( x,y,3 );
err = sum( abs( y - polyval( pf,x ) ) .^ 2 );
figure;
ylim( [ 0 1 ] );
plot( x,y,'rx',xf,polyval( pf3,xf ),'r-' );
disp( err );
```
We would like to estimate the error of a fit.

Traditionally, this is the $L^2$ metric:

$$L^2 = |y_{\text{known}} - y_{\text{fitted}}|^2$$

This is always positive and should be close to zero.

(The quantity $y_{\text{known}} - y_{\text{fitted}}$ is the residual.)

We may also want to know estimated error at any point (along the entire curve).

(This is easier if we have an analytical solution to compare to.)
x = rand( 11,1 );
y = rand( 11,1 );
xf = 0:0.01:1;
pf = polyfit( x,y,3 );
resids = y - polyval( pf,x )
figure;
subplot( 2,1,1 );
ylim( [ 0 1 ] );
plot( x,y,'rx',xf,polyval( pf3,xf ),'r-' );
subplot( 2,2,2 );
plot( x,resids,'r.' );
We would like to estimate the error of a fit.

Traditionally, this is the $L^2$ metric:

$$L^2 = |y_{\text{known}} - y_{\text{fitted}}|^2$$

This is always positive and should be close to zero.

(The quantity $y_{\text{known}} - y_{\text{fitted}}$ is the residual.)

We may also want to know estimated error at any point (along the entire curve).

(This is easier if we have an analytical solution to compare to.)

A good fit has a small residual (but the absolute size is contextual).
Linear interpolation

- A related process, *interpolation* means estimating intermediate values (between two known points).
- Be careful to distinguish this from a linear fit (regression).
Linear interpolation

- A related process, *interpolation* means estimating intermediate values (between two known points).
- Be careful to distinguish this from a linear fit (regression).

\[ y = y_0 + (x - x_0) \frac{y_1 - y_0}{x_1 - x_0} = y_0 \frac{x_1 - x}{x_1 - x_0} + y_1 \frac{x - x_0}{x_1 - x_0} \]
x_samples = linspace( 0,1,11 );
y_samples = ( x_samples - 0.5 ) .^ 2 + 0.01 * randn( size( x_samples ) );

hold on
plot( x_samples,y_samples,'ro' );
x_span = linspace( 0,1,111 );
y_interp = interp1( x_samples,y_samples,x_span );
plot( x_span,y_interp,'b-' )

coefs = polyfit( x_samples,y_samples,2 );
y_regression = coefs(1)*x_span.^2 + coefs(2)*x_span + coefs(3);
plot( x_span,y_regression,'g-' )
Insisting on too much precision can cause major problems!

Lower-order polynomial fits are best.
Extrapolation

- Modeling outside of the domain of reliable data is fraught with difficulty.
- Consider the behavior outside of the domain here for either curve.
- Unless you have a very good theoretical reason, *don’t!*
Spline interpolation

- Splines are piecewise polynomials designed to smoothly fit long segments without overfitting.
- They are first-order continuous and thus differentiable.

- Splines are typically constructed from cubic polynomials.
Fit the Brexit data with the following methods:
- Linear interpolation
- Linear fit
- Quadratic fit
- Cubic fit
- Quartic fit
- Tenth-order fit
- Spline fit
2D Interpolation

- Not on exam6.
- In 2D, use interp2 to get a single point, or griddata to reconstruct a whole grid:

```matlab
x = 0:0.05:1;
y = 0:0.05:1;
z = rand( 21 );
xstar = 0.33;
ystar = 0.33;
zstar = interp2( x,y,z,xstar,ystar );

figure
hold on
mesh( x,y,z )
plot3( xstar,ystar,zstar,'o' )
```
2D Interpolation

\[
f = @(x,y) x \cdot (1-x) \cdot \cos(4\pi x) \cdot \sin(2\pi \sqrt{y});
\]

% Define the basic grid coordinates.
grid_x = 0:0.01:1;
grid_y = grid_x(1,:);

% Define a random subset of the grid for which
% we will generate data.
pts = rand(800,2); % x,y pairs
vals = f(pts(:,1),pts(:,2));

% Generate a grid. 'nearest','linear','cubic'
[X Y] = meshgrid(grid_x,grid_y);
Z = griddata(pts(:,1),pts(:,2),vals,...
             X,Y,'linear');
2D Interpolation

hold on
mesh( X,Y,Z )
plot3( pts(:,1),pts(:,2),f(pts(:,1),pts(:,2)),'r.' )