MATLAB

Fitting Data

Announcements

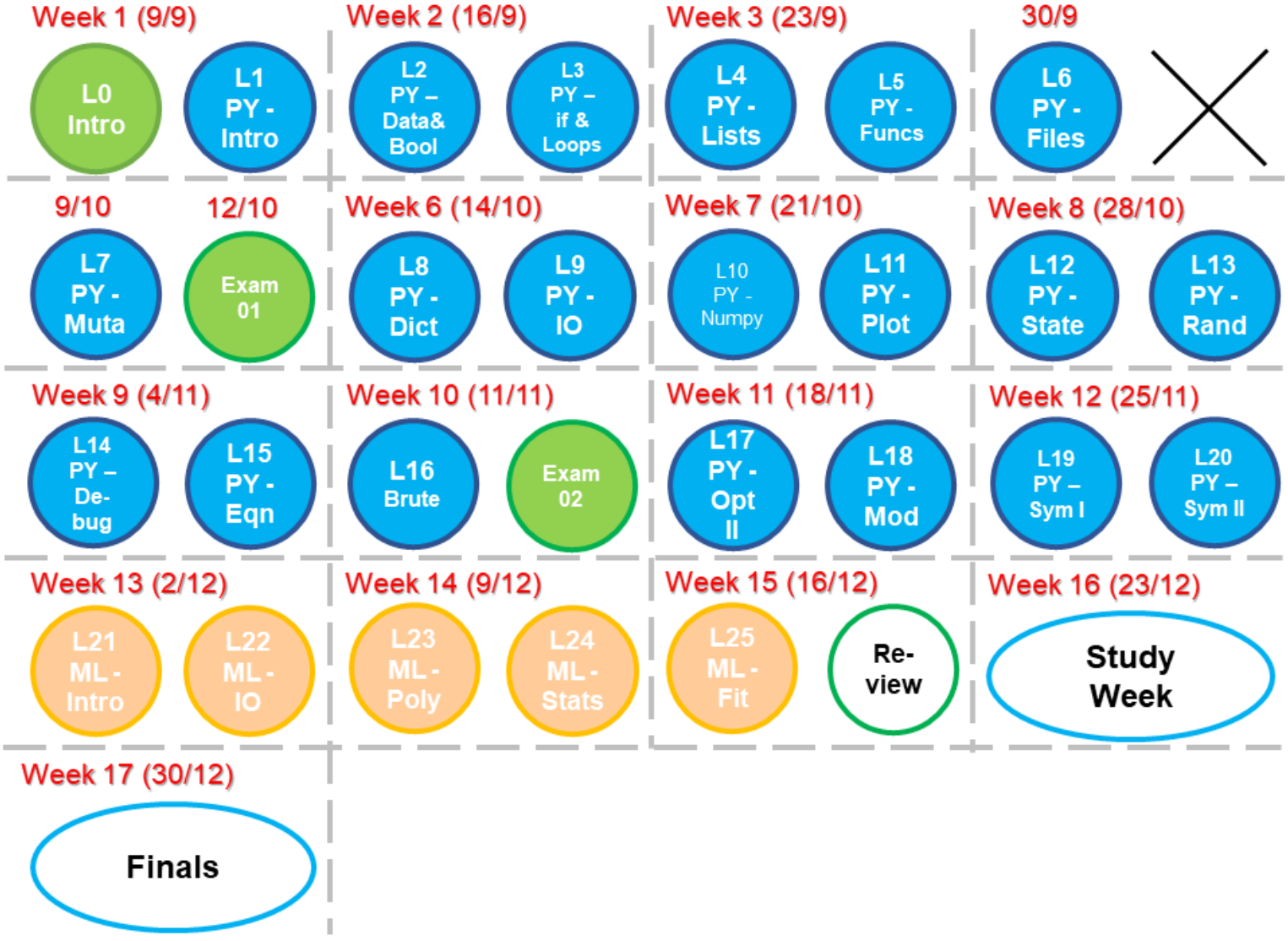
quiz: `quiz25` due on Tues 17/12

lab: One last one.....

hw: `hw12` due Fri 18/12

Paper Exam - MCQs + Short Questions + Programming question

Roadmap


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
9/10
L7 PY - Muta
12/10
Exam 01
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. Fit polynomials to data, from linear to high-order.
B. Understand limitations of high-order fit.
C. Distinguish interpolation and fitting (regression).
D. Employ spline-based or piecewise fitting.

MATLAB Review

Question

```
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 $4x + 3x^2 + 2x^3 + x^4$

Question

```
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 $4x + 3x^2 + 2x^3 + x^4$

Question

```
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

Question

```
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 or Curve Fitting

Interpolate

Given $n+1$ (x,y) points,

find y-values at x-values which are not in your given (x, y) points

Interpolate

Given $n+1$ (x,y) points,

find y-values at x-values which are not in your given (x, y) points

Many methods inside `interp1`

- 'linear'
- 'nearest'
- 'pchip'

Fitting Polynomials

Given $n+1$ (x,y) points (or more), we can fit an nth order polynomial as a model. `polyfit(x,y,n)`

Why nth order is the highest order with $n+1$ points?

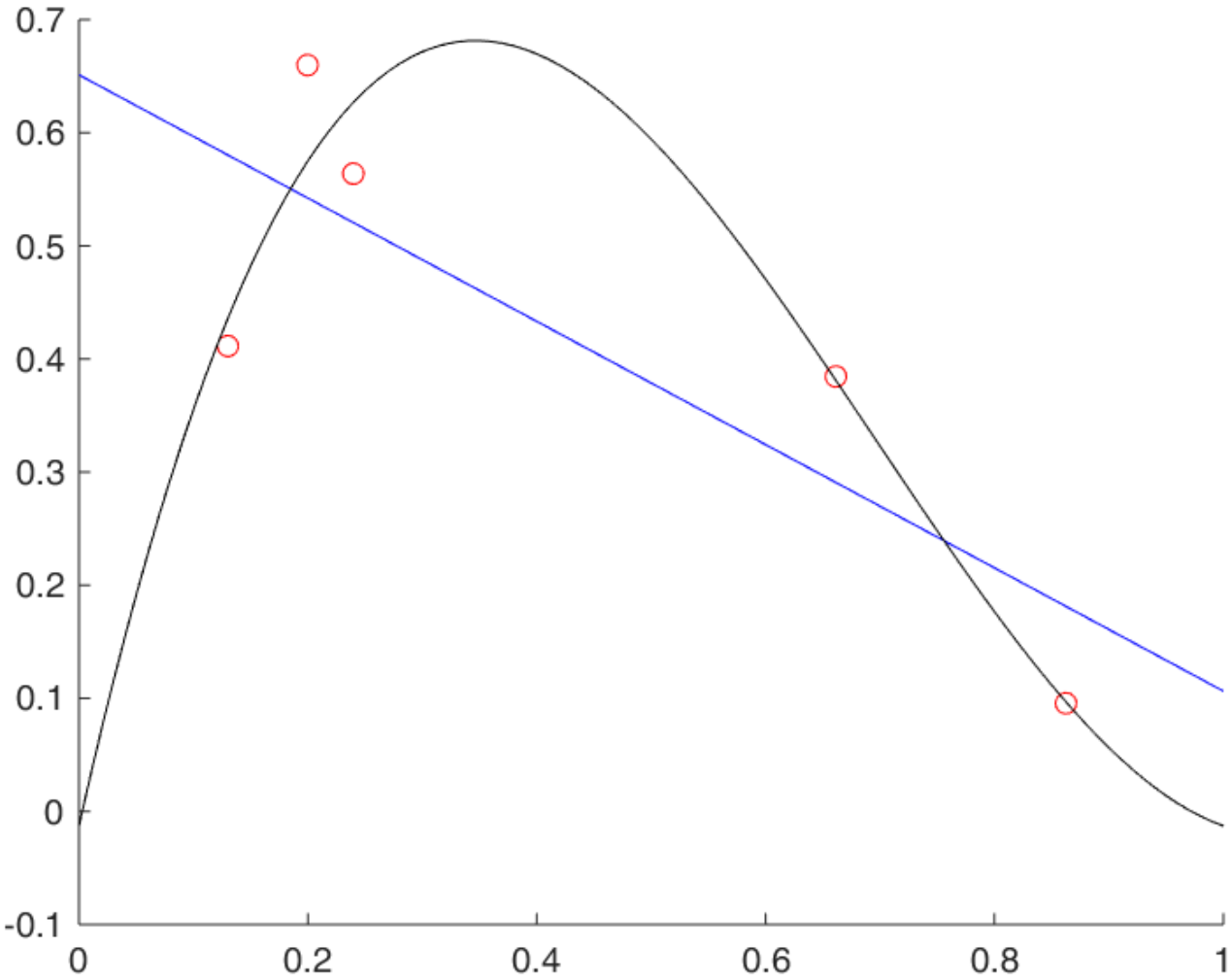
Fitting Polynomials

Given $n+1$ (x,y) points (or more), we can fit an nth order polynomial as a model. `polyfit(x,y,n)`

Why nth order is the highest order with $n+1$ points?

```
%generate our data
x = rand( 5,1 ); y = rand( 5,1 );
%generate our fit
xf = 0:0.01:1;
pf1 = polyfit( x,y,1 ); %linear
pf3 = polyfit( x,y,3 ); %cubic

%Plot our fit
yf1 = polyval( pf1,xf );
yf3 = polyval( pf3,xf );
hold on
plot( x,y,'ro',  xf,yf1,'b-', xf,yf3,'k-' );
```

Fitting Polynomials



Fitting Polynomials

```
n = input( 'Give a polynomial order' );
x = rand( n+1,1 );
y = rand( n+1,1 );


xf = 0:0.01:1;
n1 = n % what polynominal order you want to fit
pf = polyfit( x,y,n1 );
yf = polyval( pf,xf );
plot( x,y,'rx',  xf,yf,'r-' );
```

Fitting Polynomials

```
n = input( 'Give a polynomial order' );
x = rand( n+1,1 );
y = rand( n+1,1 );


xf = 0:0.01:1;
n1 = n % what polynominal order you want to fit
pf = polyfit( x,y,n1 );
yf = polyval( pf,xf );
plot( x,y,'rx',  xf,yf,'r-' );
```

If `n1` > `n`, MATLAB gives *Warning: Polynomial is not unique; degree >= number of data points.* Why?

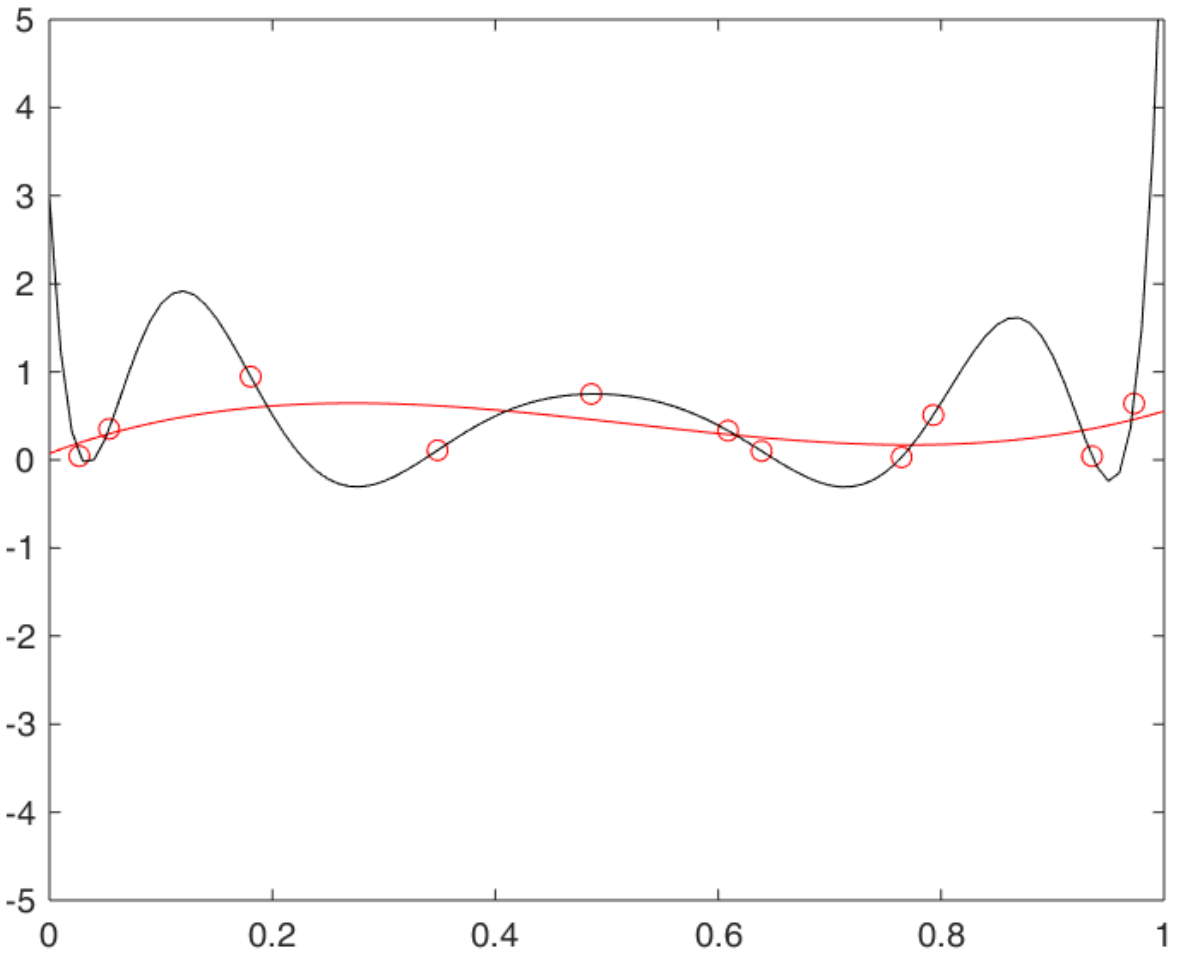
Fitting Polynomials

High-order polynomials tend to wobble a lot, and overshoot at the ends.

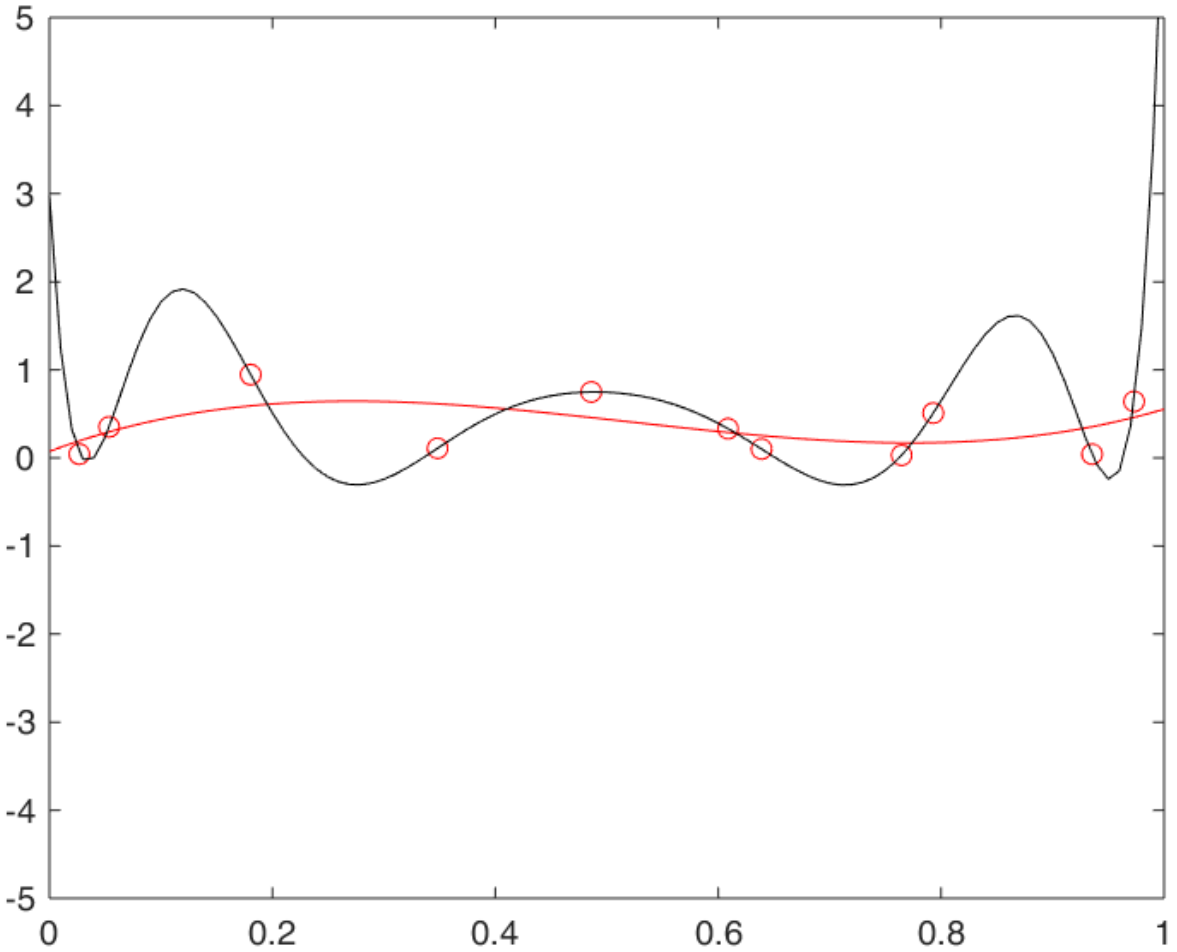
A simpler fit is often better!

```
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

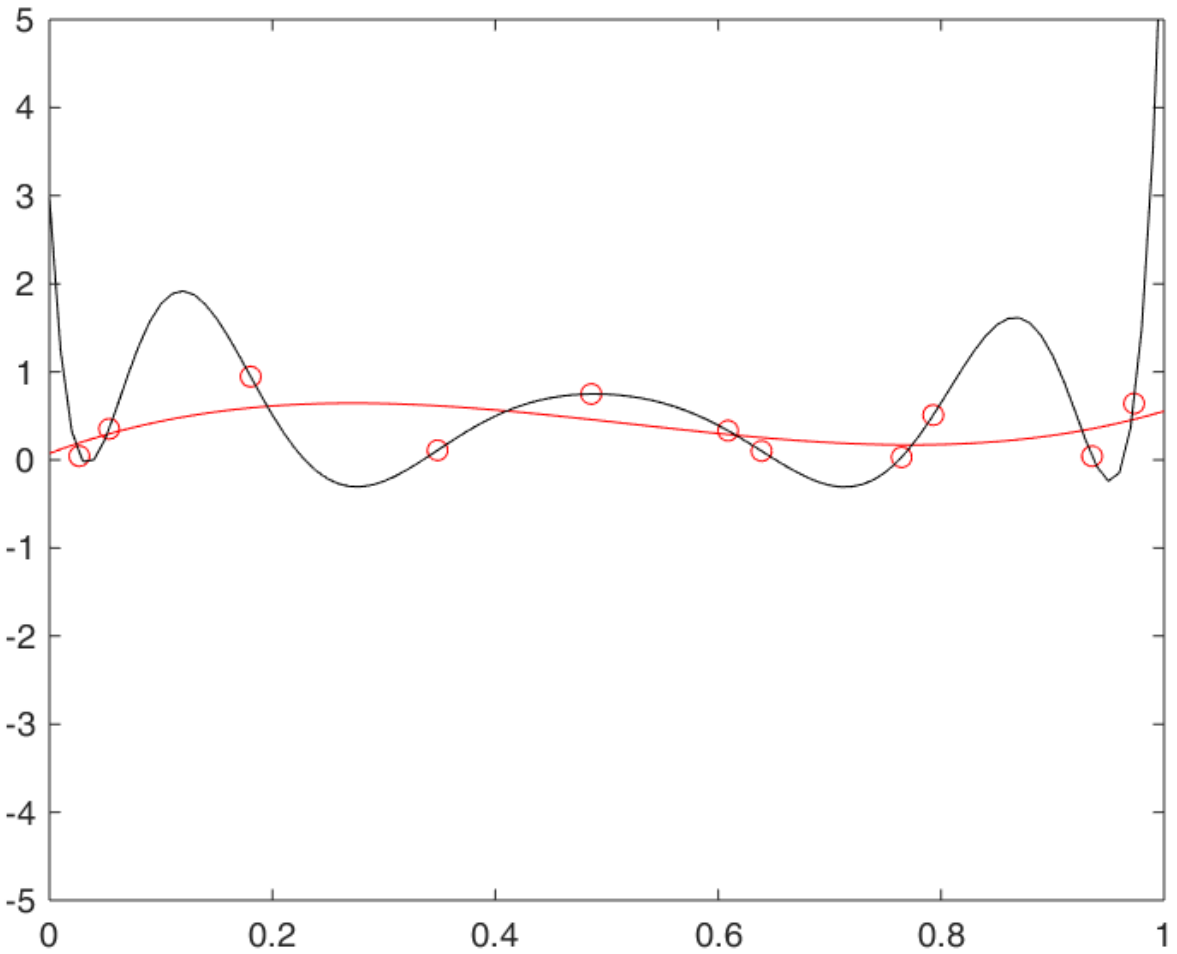


Fitting Polynomials



Which one looks better?

Fitting Polynomials

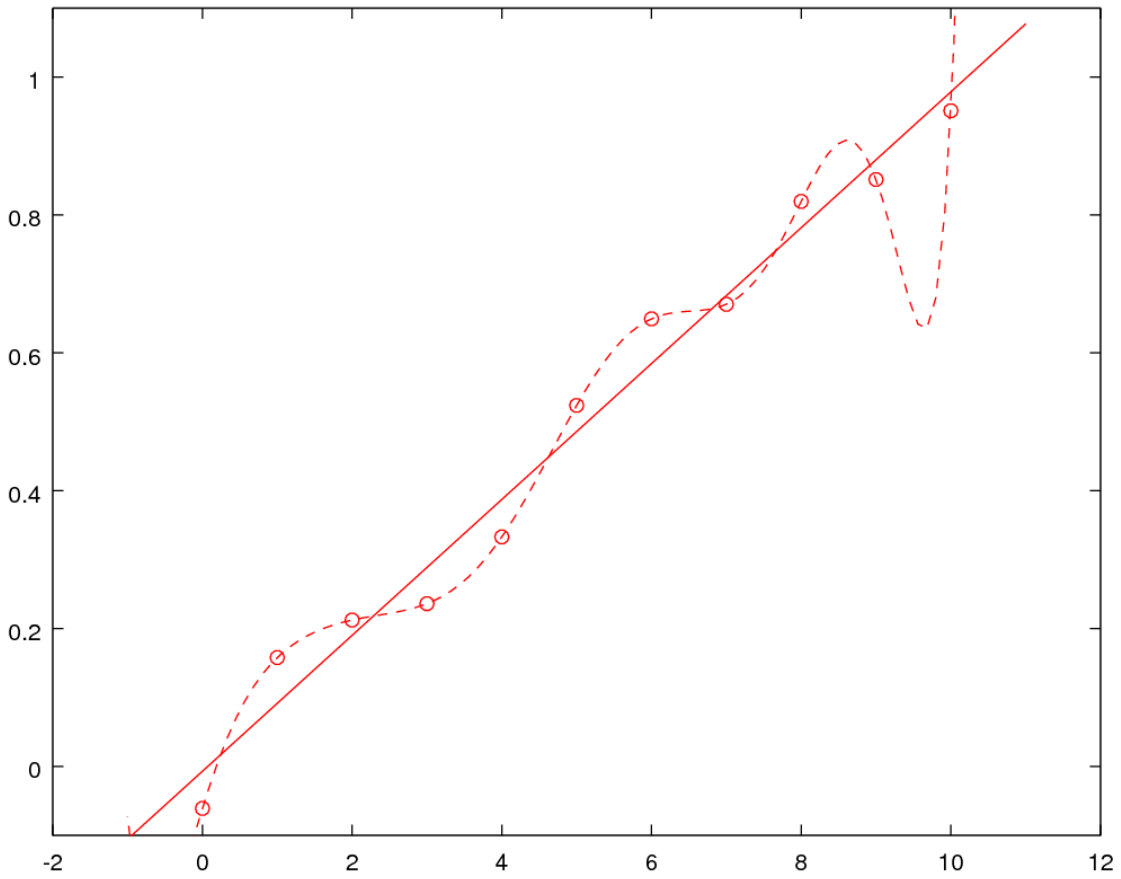


Which one looks better? Which one do you think is more real?

Overfitting

Insisting on too much precision can cause major problems!

Lower-order polynomial fits are best.



Error from Fitting

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.

Error from Fitting

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.

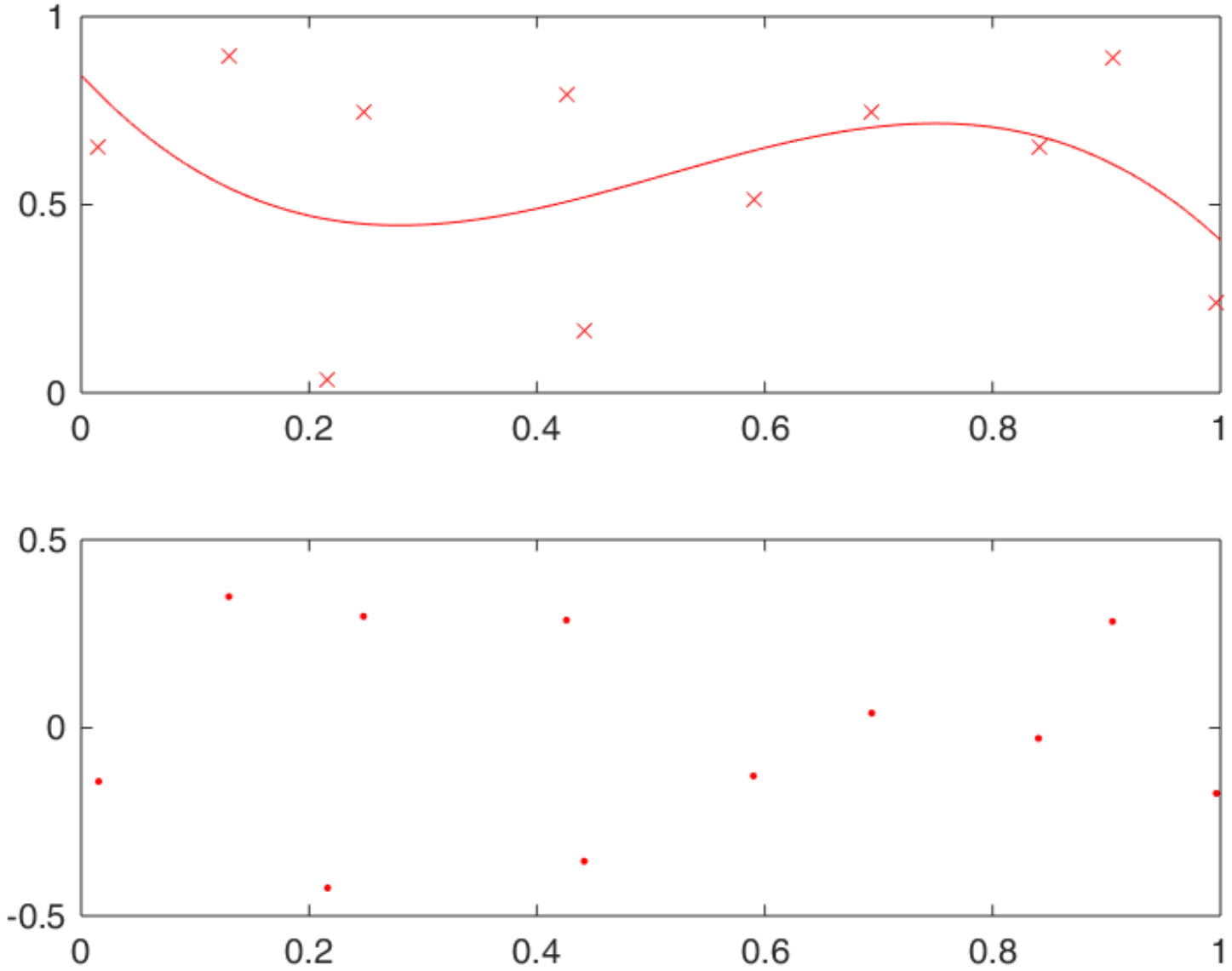
The quantity $y_{\text{known}} - y_{\text{fitted}}$ is the *residual*.

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

Error from Fitting

```
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( 2,1,2 );
plot( x,resids,'r.' );
```

Fitting Polynomials



Error from Fitting

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*.)

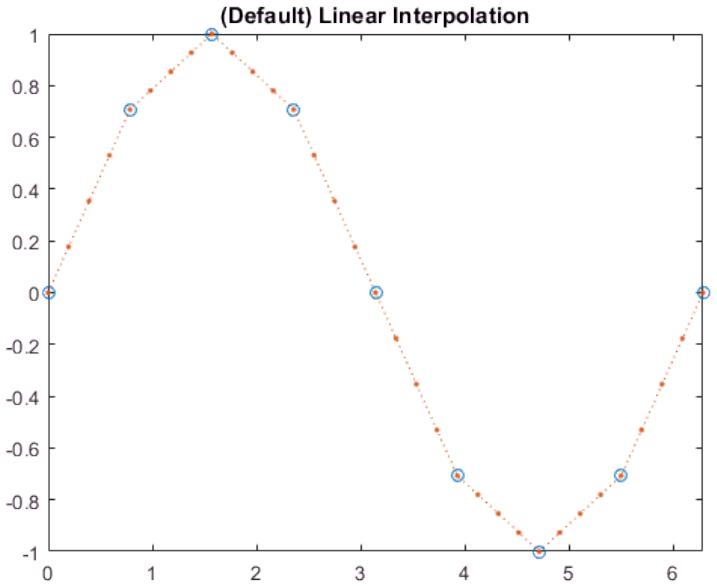
We may also want to know estimated error at any point (along the entire curve).

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).

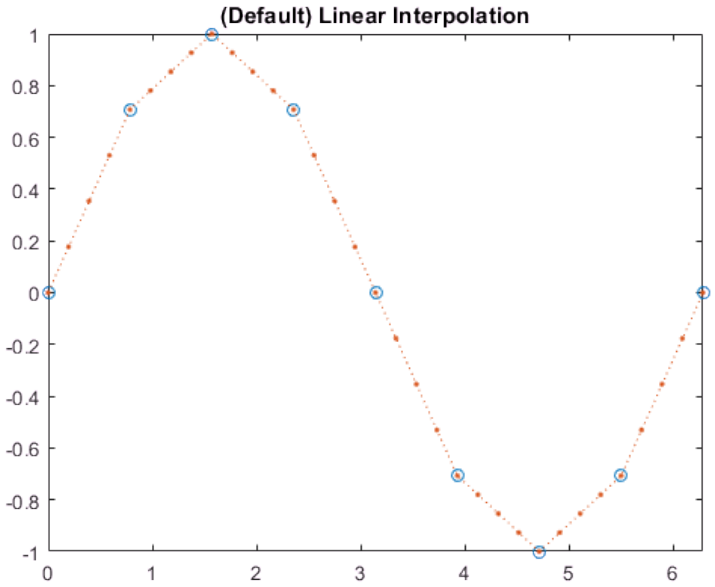
Be careful to distinguish this from a linear fit (regression).



Linear interpolation

A related process, *interpolation* means estimating intermediate values (between two known points).

Be careful to distinguish this from a linear fit (regression).



$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0}$$

Linear interpolation vs Fitting 1

Plot to see to see the difference

```
% Sample
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' );
```

Linear interpolation vs Fitting 2

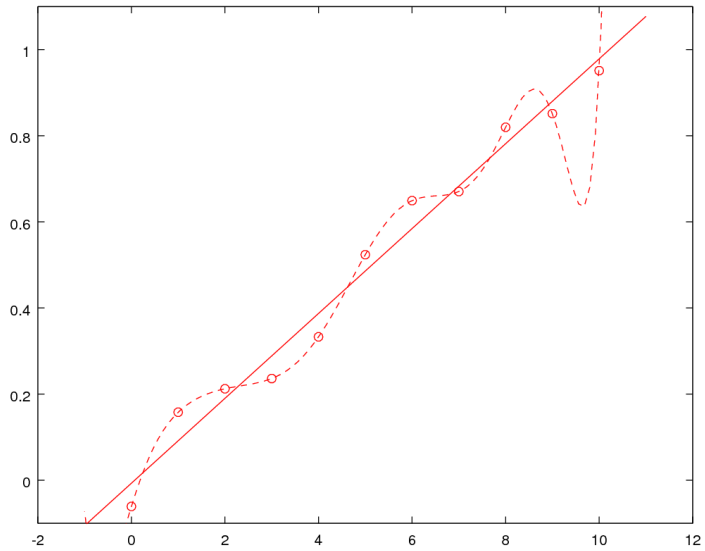
```
% interpolation
x_span = linspace( 0,1,111 );
y_interp = interp1( x_samples,y_samples,x_span );
plot( x_span,y_interp,'b-' )

% polyfit
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-' )
```

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 too much overfitting.

Splines are typically constructed from cubic polynomials.

Similar to `'pchip'` in `interp1` but the results are different!

```
y_est = spline(x,y,x_est)
```

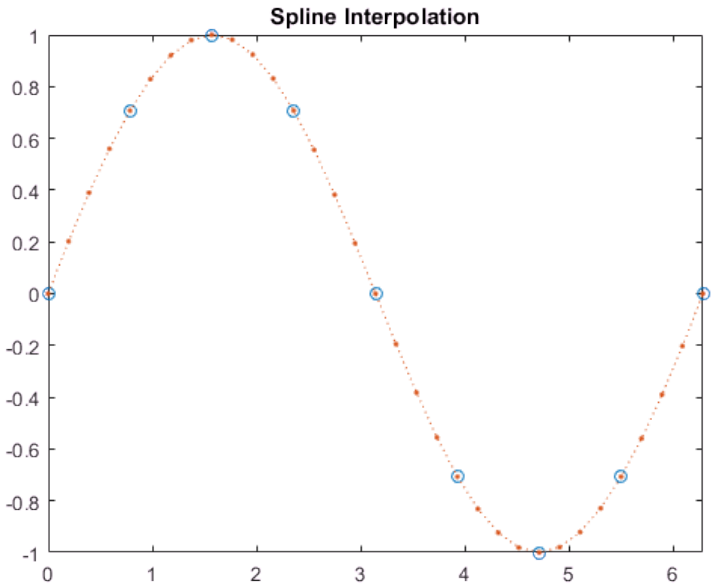
Spline interpolation

Splines are piecewise polynomials designed to smoothly fit long segments without too much overfitting.

Splines are typically constructed from cubic polynomials.

Similar to `'pchip'` in `interp1` but the results are different!

```
y_est = spline(x,y,x_est)
```



https://blogs.mathworks.com/cleve/2012/07/16/splines-and-pchips/

2D Interpolation - Optional

In 2D, use `interp2` to get a single point, or `griddata` to reconstruct a whole grid:

Ex 1

```
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 - Optional

Ex 2

```
f = @( x,y ) x.*( 1-x ).*cos( 4*pi*x ) ...
             .* 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) );
```

2D Interpolation - Optional

Ex 2

```
% Generate a grid.  'nearest','linear','cubic'
[ X Y ] = meshgrid( grid_x,grid_y );
Z = griddata( pts( :,1 ),pts( :,2 ),vals, ...
              X,Y,'linear' );
hold on
mesh( X,Y,Z )
plot3( pts( :,1 ),pts( :,2 ),f(pts(:,1),...
          pts(:,2)),'r.' )
```

Demo?

Summary

A. `polyfit(x,y,n)`
B. Interpolation vs Regression or Fitting
C. Problems of fitting to higher power polynomials
D. `spline(x,y,x-est)`