Rutgers University
School of Engineering

Fall 2011

14:440:127 - Introduction to Computers for Engineers

Sophocles J. Orfanidis
ECE Department
orfanidi@ece.rutgers.edu

week 11

Weekly Topics

Week 1 - Basics – variables, arrays, matrices, plotting (ch. 2 & 3)
Week 2 - Basics – operators, functions, program flow (ch. 2 & 3)
Week 3 - Matrices (ch. 4)
Week 4 - Plotting – 2D and 3D plots (ch. 5)
Week 5 - User-defined functions (ch. 6)
Week 6 - Input-output processing (ch. 7)
Week 7 - Program flow control & relational operators (ch. 8)
Week 8 - Matrix algebra – solving linear equations (ch. 9)
Week 9 - Strings, structures, cell arrays (ch. 10)
Week 10 - Symbolic math (ch. 11)
Week 11 - Numerical methods – data fitting (ch. 12)
Week 12 – Selected topics

Textbook: H. Moore, *MATLAB for Engineers*, 2nd ed., Prentice Hall, 2009

Numerical Methods
Data Fitting, Smoothing, Filtering

- data fitting with polynomials – **`polyfit, polyval`**
- examples: Moore’s law, Hank Aaaron, US census data
- data interpolation – **`interp1, spline, pchip`**
- least-squares polynomial regression
- least-squares with other basis functions
- example: trigonometric fits
- multivariate regression – NFL data
- smoothing – **`smooth`**
- example: global warming
- digital filtering – **`filter`**
- examples: bandpass filter, filtering ECG signals

Polynomial data fitting

polyfit, polyval

$$P(x) = p_1 x^M + p_2 x^{M-1} + \cdots + p_M x + p_{M+1}$$

$$\mathbf{p} = [p_1,\ p_2, \ldots,\ p_M,\ p_{M+1}]$$

$$P(x) = 5x^4 - 2x^3 + x^2 + 4x + 3$$

$$\mathbf{p} = [5,\ -2,\ 1,\ 4,\ 3]$$

```
>> doc polyfit
>> doc polyval
>> doc roots
>> doc poly
```

Given coefficients **p**, evaluate $P(x)$ at a vector of x's – **(polyval)**

Given **p**, find the roots of $P(x)$ – **(roots)**

Given the roots, reconstruct the coefficient vector **p** – **(poly)**

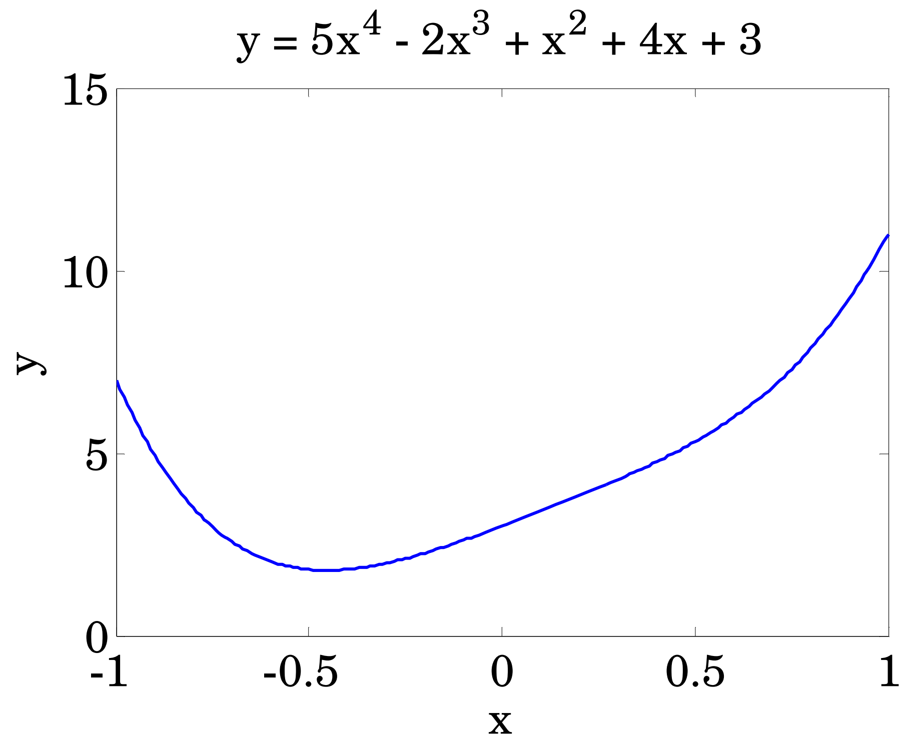
Given N data points $\{x_i, y_i\}$, i=1,2,…,N, find an M-th degree polynomial that best fits the data – **(polyfit)**

polyfit, polyval

$$P(x) = 5x^4 - 2x^3 + x^2 + 4x + 3$$

$$\mathbf{p} = [5,\ -2,\ 1,\ 4,\ 3]$$

```
>> p = [5, -2, 1, 4, 3];
>> x = linspace(-1,1,201);
>> y = polyval(p,x);
>> plot(x,y,'b');
```



Given N data points $\{x_i, y_i\}$, $i=1,2,\ldots,N$, find an M-th degree polynomial that best fits the data – **(polyfit)**

```
% design procedure:

xi = [x1,x2,...,xN];
yi = [y1,y2,...,yN];

p = polyfit(xi,yi,M);

y = polyval(p,x);
```

evaluate $P(x)$ at a given vector x

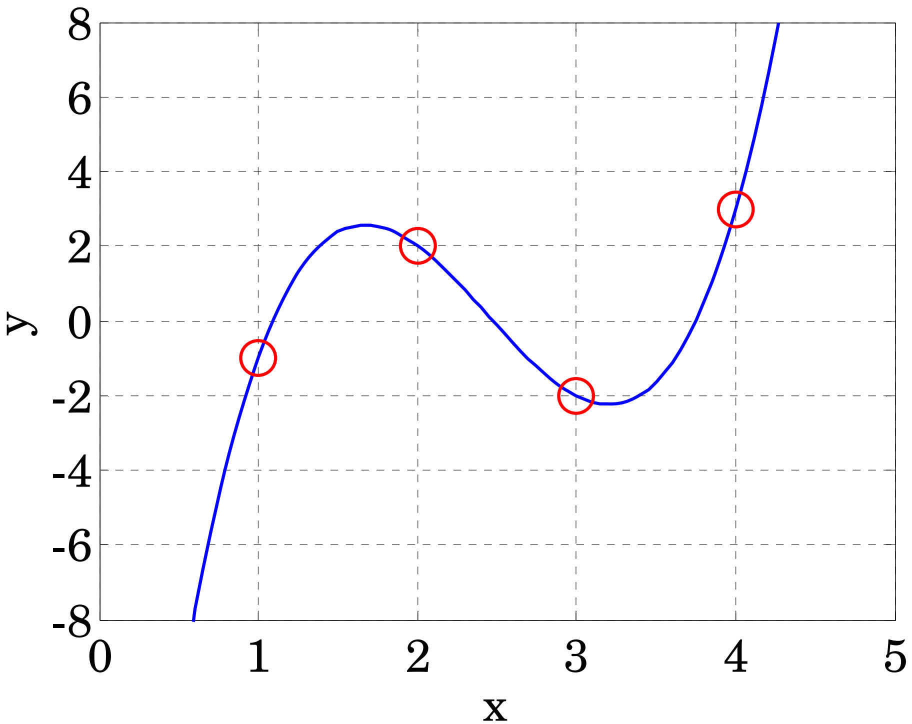
M = polynomial order

if $N = M+1$, the polynomial interpolates the data

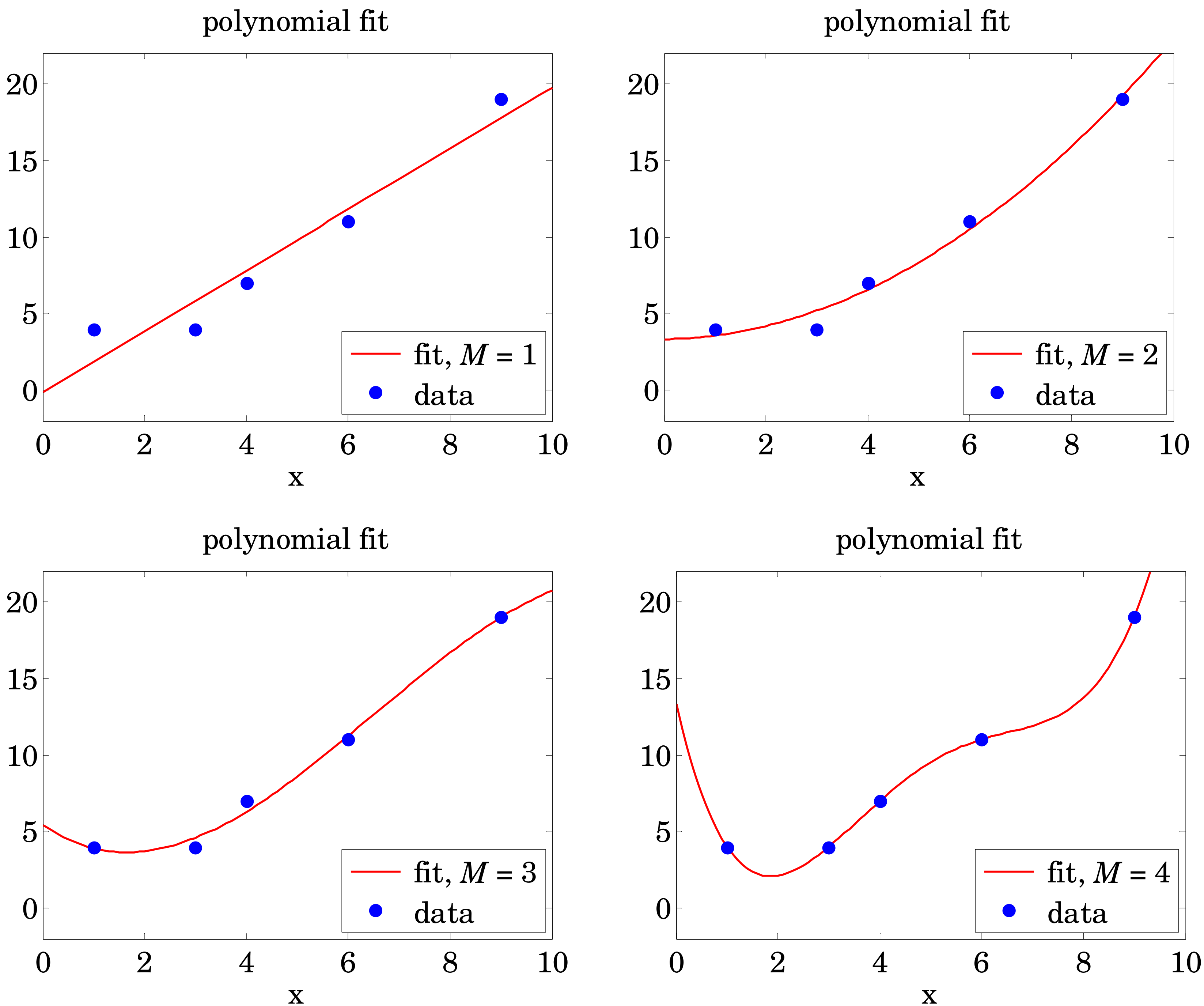
if $N > M+1$, the polynomial provides the best fit in a least-squares sense

$$J = \sum_{i=1}^{N} \left(P(x_i) - y_i\right)^2 = \min$$

```
>> xi = [1,2,3,4];     % from homework set-8
>> yi = [-1,2,-2,3];
>> p = polyfit(xi,yi,3);
>> x = linspace(0,5,101);
>> y = polyval(p,x);
>> plot(x,y,'b', xi,yi,'ro');
```



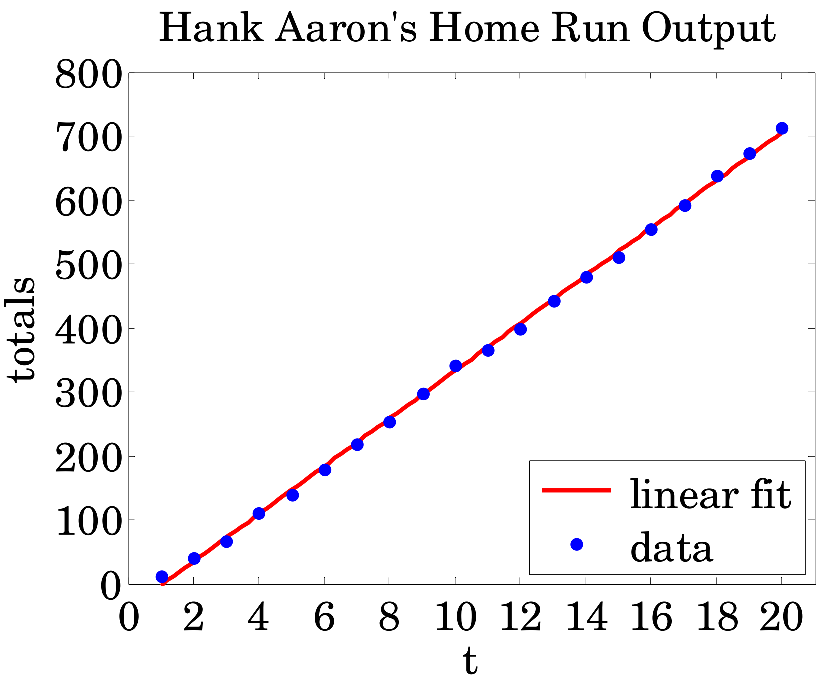
```
xi = [1, 3, 4, 6, 9];
yi = [4, 4, 7, 11, 19];

x = linspace(0,10,101);

for M = [1,2,3,4]
  p = polyfit(xi,yi,M);

  y = polyval(p,x);

  figure;
  plot(x,y,'r-', xi,yi,'b.', 'markersize',25);
  yaxis(-2,22,0:5:20); xaxis(0,10,0:2:10);
  xlabel('x'); title('polynomial fit');
  legend([' fit, {\itM} = ',num2str(M)],...
         ' data', 'location','se');
end
```

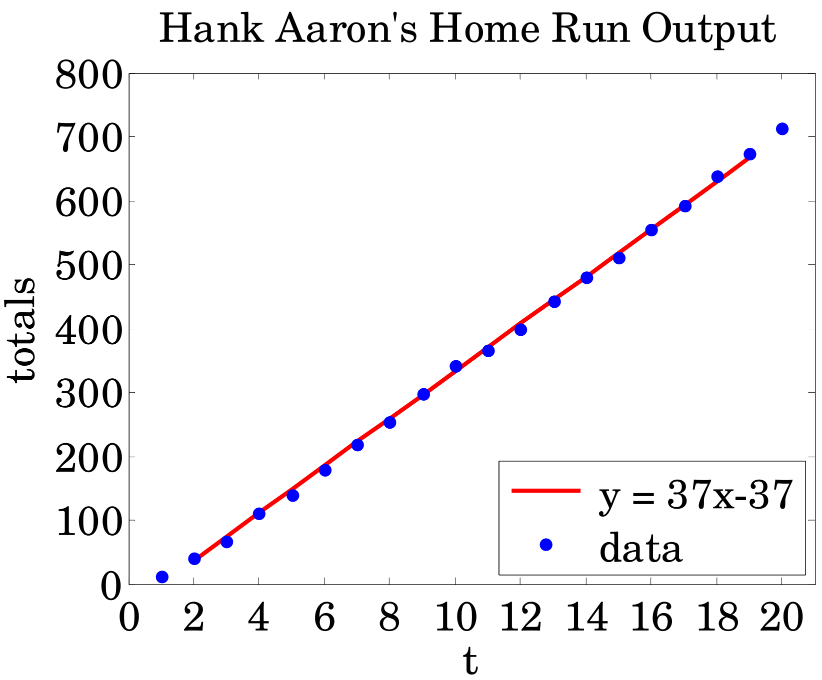


polynomial fit
20
15
10
5
0
0
2
4
6
8
10
x
fit, M = 1
data
polynomial fit
20
15
10
5
0
0
2
4
6
8
10
x
fit, M = 2
data
polynomial fit
20
15
10
5
0
0
2
4
6
8
10
x
fit, M = 3
data
polynomial fit
20
15
10
5
0
0
2
4
6
8
10
x
fit, M = 4
data

```
% year    ti   H
% ---------------
  1954     1   13
  1955     2   27
  1956     3   26
  1957     4   44
  1958     5   30
  1959     6   39
  1960     7   40
  1961     8   34
  1962     9   45
  1963    10   44
  1964    11   24
  1965    12   32
  1966    13   44
  1967    14   39
  1968    15   29
  1969    16   44
  1970    17   38
  1971    18   47
  1972    19   34
  1973    20   40

A = load('aaron.dat');

ti = A(:,2); H = A(:,3);
yi = cumsum(H);
```

from set-6





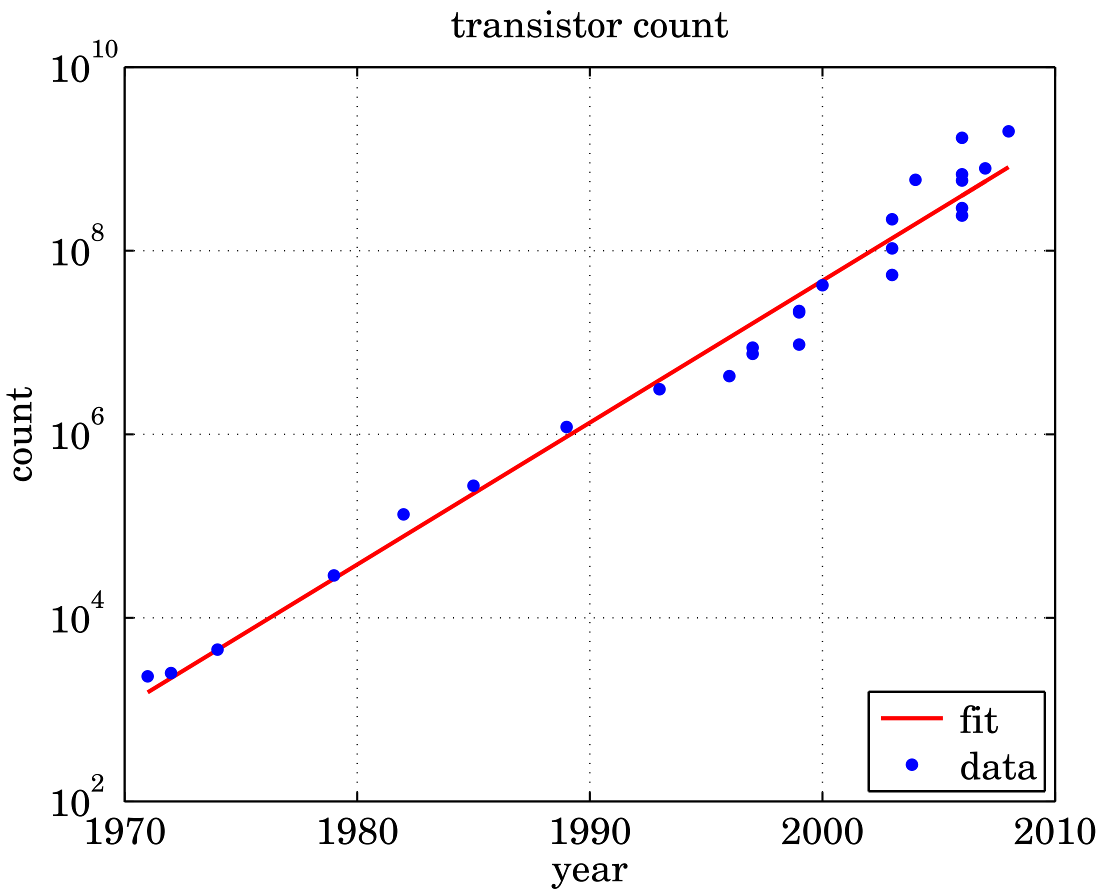
Given N data points $\{x_i, y_i\}$, $i=1,2,\ldots,N$, the following data models can be reduced to linear fits using an appropriate transformation of the data:

linear:	$y = a\,x + b$		
exponential:	$y = b\,e^{ax}$	$\Rightarrow$	$\log(y) = ax + \log(b)$
exponential:	$y = b\,2^{ax}$	$\Rightarrow$	$\log_2(y) = ax + \log_2(b)$
exponential:	$y = b\,x\,e^{ax}$	$\Rightarrow$	$\log(y/x) = ax + \log(b)$
power:	$y = b\,x^{a}$	$\Rightarrow$	$\log(y) = a\,\log(x) + \log(b)$

```
>> p = polyfit(xi,log(yi),1);  % exponential
>> y = exp(polyval(p,x));   % y=exp(a*x+log(b))
>> a = p(1);
>> b = exp(p(2));    % so that y = b*exp(a*x)
```

yi	ti
2.300e+003	1971
2.500e+003	1972
4.500e+003	1974
2.900e+004	1979
1.340e+005	1982
2.750e+005	1985
1.200e+006	1989
3.100e+006	1993
4.300e+006	1996
7.500e+006	1997
8.800e+006	1997
9.500e+006	1999
2.130e+007	1999
2.200e+007	1999
4.200e+007	2000
5.430e+007	2003
1.059e+008	2003
2.200e+008	2003
5.920e+008	2004
2.410e+008	2006
2.910e+008	2006
5.820e+008	2006
6.810e+008	2006
7.890e+008	2007
1.700e+009	2006
2.000e+009	2008

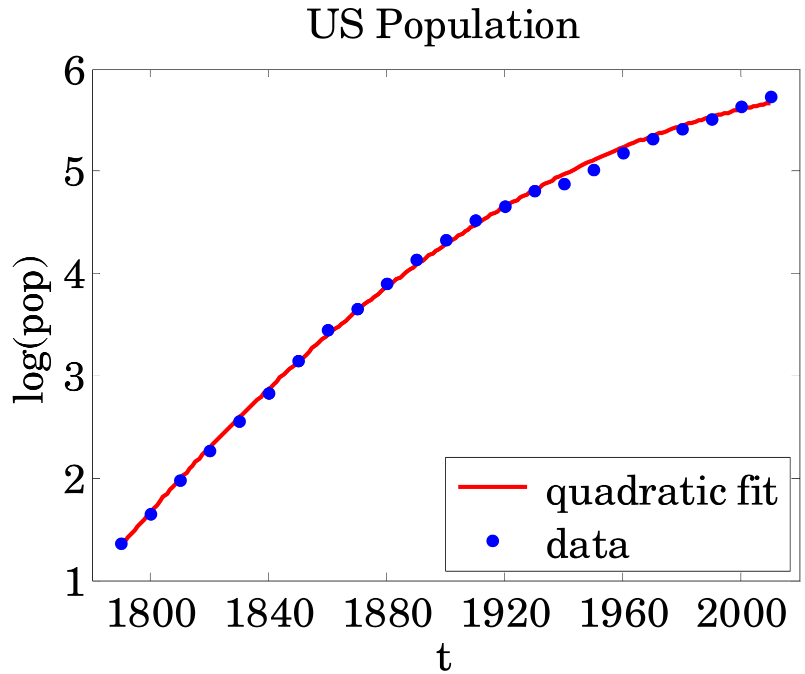
Moore's law from set-4

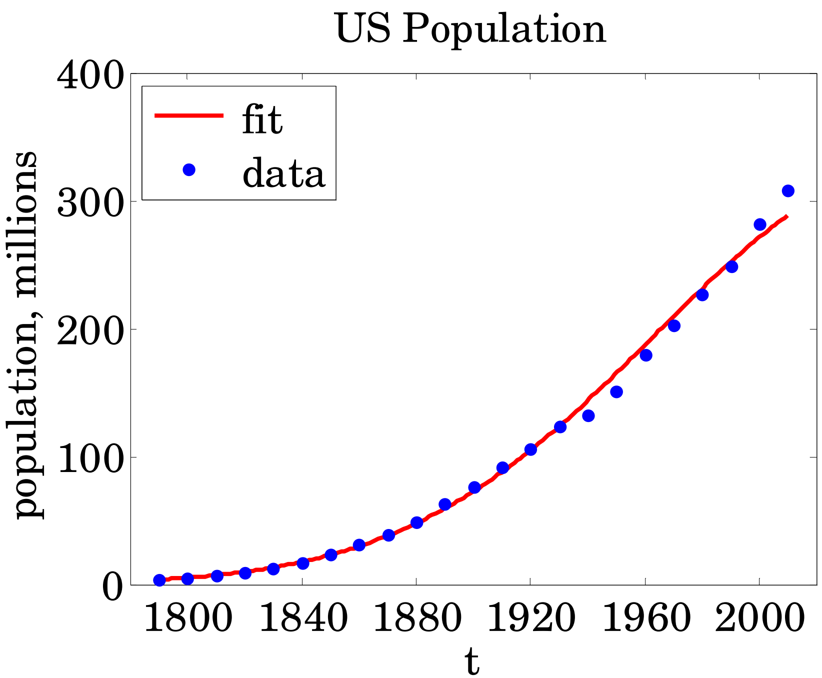


fitted model:

```
f(t) = b*2.^(a*(t-t1));
```

```
% source: Wikipedia
% US population in millions
%
%  ti        yi
% -------------
  1790     3.929
  1800     5.237
  1810     7.240
  1820     9.638
  1830    12.866
  1840    17.069
  1850    23.192
  1860    31.443
  1870    38.558
  1880    49.371
  1890    62.980
  1900    76.212
  1910    92.229
  1920   106.022
  1930   123.202
  1940   132.165
  1950   151.326
  1960   179.323
  1970   203.212
  1980   226.546
  1990   248.710
  2000   281.422
  2010   308.746
```





```
A = load('uspop.dat');   % file on sakai

ti = A(:,1);    % read data
yi = A(:,2);

p = polyfit(ti,log(yi),2)    % quadratic

p =
    -6.4657e-005  0.2653  -266.4672

population model:
y = exp(p(1)*t.^2 + p(2)*t + p(3));
```

The curve fitting toolbox allows more complicated nonlinear data fits.

```
>> doc curvefit
```

Given N data points $\{xi, yi\}$, i=1,2,…,N, find interpolated values, $y = f(x)$, at points x

interpolation

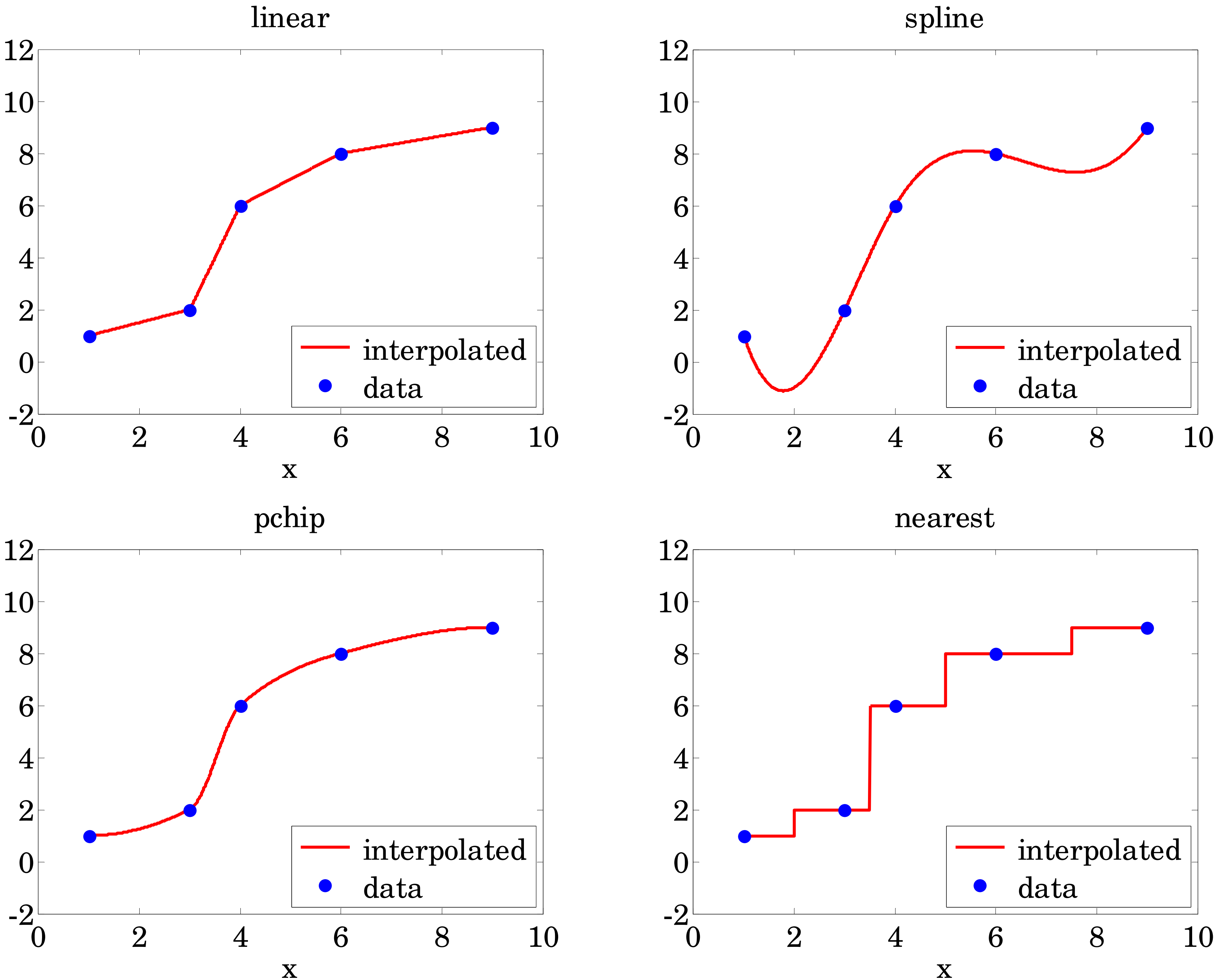
```
% procedure:

xi = [x1,x2,...,xN];
yi = [y1,y2,...,yN];

y = interp1(xi,yi,x,method);
y = spline(xi,yi,x);
y = pchip(xi,yi,x);
```

```
>> doc interp1
>> doc spline
>> doc pchip
```

```
method = 'linear'
         'spline'
         'pchip'
         'nearest'
```

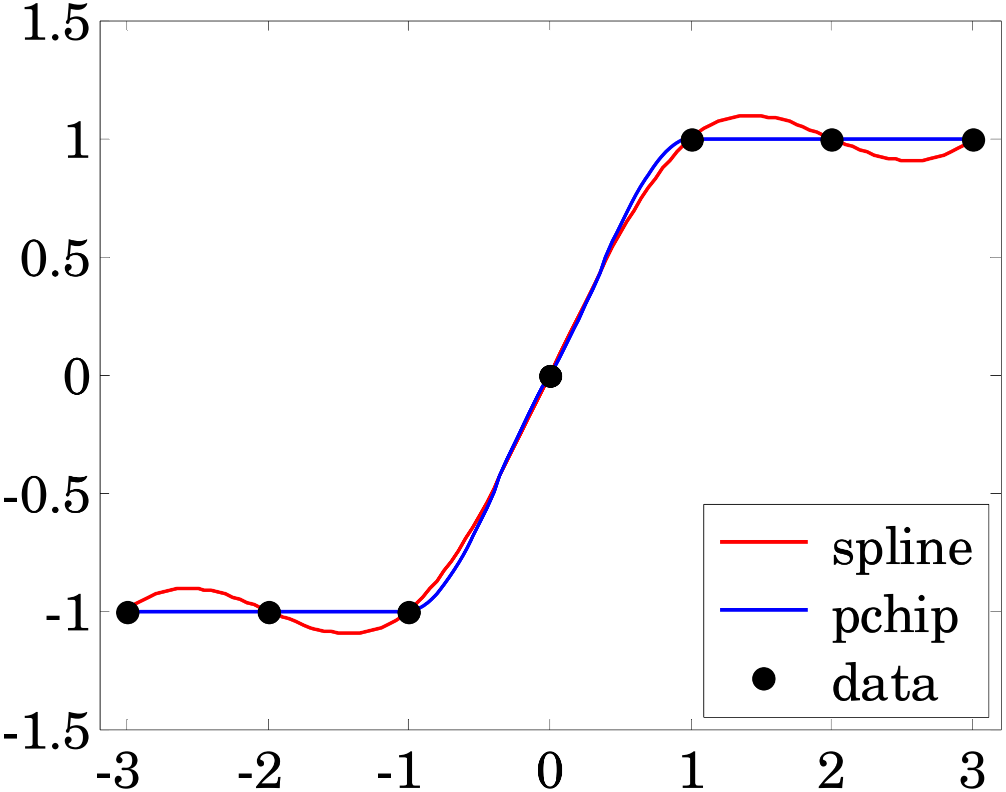
default

piecewise cubic Hermite interpolation polynomial


linear
spline
pchip
nearest
12
10
8
6
4
2
0
-2
0
2
4
6
8
10
x
interpolated
data

interpolation

```
xi = -3:3
yi = sign(xi);

x = linspace(-3,3,121);

ys = spline(xi,yi,x);
yp = pchip(xi,yi,x);

plot(x,ys,'r-',...
     x,yp,'b-',...
     xi,yi,'k.');
```



How does **polyfit** work? Consider a straight-line fit, $y = ax+b$, to N data points $\{x_i, y_i\}$, $i=1,2,\ldots,N$

polynomial regression

overdetermined & inconsistent linear system of 5 equations in 2 unknowns

least-squares solution

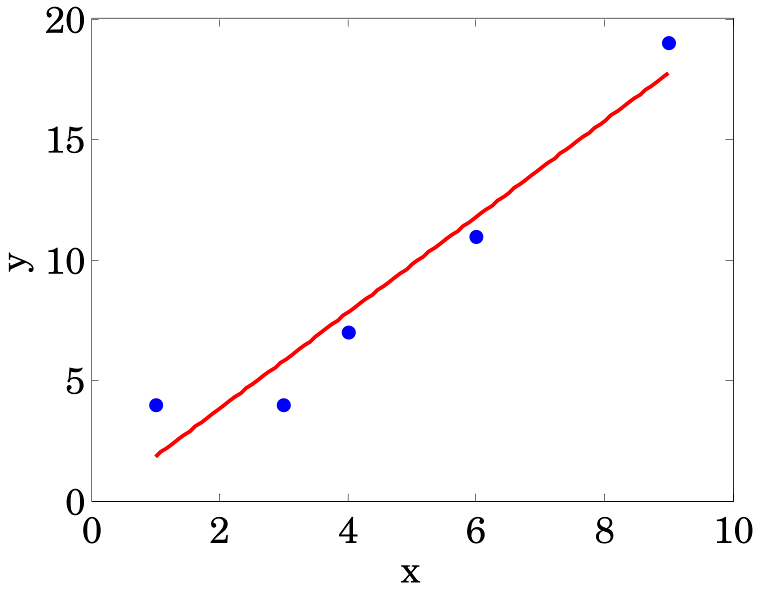
$$\begin{aligned} a\,x_1 + b &= y_1 \\ a\,x_2 + b &= y_2 \\ a\,x_3 + b &= y_3 \\ a\,x_4 + b &= y_4 \\ a\,x_5 + b &= y_5 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} a \\ b \end{bmatrix}$$

A is the design matrix

$$\mathbf{A}\,\mathbf{p} = \mathbf{y}$$

$$\mathbf{p} = \mathbf{A} \setminus \mathbf{y}$$

```
xi = [1, 3, 4, 6, 9]';        % column vectors
yi = [4, 4, 7, 11, 19]';

A = [xi, ones(5,1)];          % design matrix
p = A\yi
p =
    1.9892
   -0.1505

p = polyfit(xi,yi,1)
p =
    1.9892   -0.1505

x = linspace(1,9,91);
y = polyval(p,x);

plot(x,y,'r', xi,yi,'b.','markersize',20);
```



Quadratic fit, $y = ax^2 + bx + c$
to N data points $\{x_i, y_i\}$, $i=1, 2, \ldots, N$

polynomial regression

overdetermined & inconsistent linear system of 5 equations in 3 unknowns

least-squares solution

$$
\begin{aligned}
a\,x_1^2 + b\,x_1 + c &= y_1 \\
a\,x_2^2 + b\,x_2 + c &= y_2 \\
a\,x_3^2 + b\,x_3 + c &= y_3 \\
a\,x_4^2 + b\,x_4 + c &= y_4 \\
a\,x_5^2 + b\,x_5 + c &= y_5
\end{aligned}
\quad \Rightarrow \quad
\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \\ x_5^2 & x_5 & 1 \end{bmatrix}
\begin{bmatrix} a \\ b \\ c \end{bmatrix}
=
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}
, \quad \mathbf{p} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}
$$

$$\boxed{\mathbf{p} = \mathbf{A} \setminus \mathbf{y}}$$

$$\mathbf{A}\,\mathbf{p} = \mathbf{y}$$

least-squares solution

Euclidean L_2 norm

polynomial regression

$||\mathbf{A}\,\mathbf{p} - \mathbf{y}||^2 = \min \quad \Rightarrow$ `p = A \ y`

equivalent solutions:
```
p = (A'A)\(A'*y)
p = pinv(A)*y
```

assumes that $M+1 \leq N$ and that **A** has full rank, conditions that are typically satisfied in practice (then, **p** is unique least-squares solution)

other norms – such as L_1 – are used in practice but don't have a closed-form solution – several MATLAB toolboxes exist for such problems

For straight line fits, the equivalent solution satisfies:

```
(A'A)*p = (A'*y)
```

and leads to the following 2x2 linear system for the straight-line parameters, `p = [a,b]'` (i.e., y=a*x+b)

$$\begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & N \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix}$$

```
p = [sum(xi.^2), sum(xi); ...
     sum(xi), N ] \ [sum(xi.*yi); sum(yi)]

p = [xi,ones(N,1)] \ yi   % is much simpler
```

regression with other basis functions

The data model is assumed to be a linear combination of known basis functions, such as exponential, trigonometric, etc:

$$y = c_0 + c_1 f_1(x) + c_2 f_2(x) + \cdots + c_M f_M(x)$$

and the objective is to determine the coefficients c_i to fit N data points $\{x_i, y_i\}$, $i = 1, 2, \ldots, N$, where again we must assume $M+1 \leq N$

Polynomial fitting is a special case using the monomial basis: 1, x, x^2, …, x^M

Design procedure: set up the design matrix **A** and solve the overdetermined linear system **A c = y**

Example: $M = 3, \quad N = 5$

$$y = c_0 + c_1 f_1(x) + c_2 f_2(x) + c_3 f_3(x)$$

regression with other basis functions

$$\begin{aligned}
c_0 + c_1 f_1(x_1) + c_2 f_2(x_1) + c_3 f_3(x_1) &= y_1 \\
c_0 + c_1 f_1(x_2) + c_2 f_2(x_2) + c_3 f_3(x_2) &= y_2 \\
c_0 + c_1 f_1(x_3) + c_2 f_2(x_3) + c_3 f_3(x_3) &= y_3 \\
c_0 + c_1 f_1(x_4) + c_2 f_2(x_4) + c_3 f_3(x_4) &= y_4 \\
c_0 + c_1 f_1(x_5) + c_2 f_2(x_5) + c_3 f_3(x_5) &= y_5
\end{aligned}$$

$$\begin{bmatrix} 1 & f_1(x_1) & f_2(x_1) & f_3(x_1) \\ 1 & f_1(x_2) & f_2(x_2) & f_3(x_2) \\ 1 & f_1(x_3) & f_2(x_3) & f_3(x_3) \\ 1 & f_1(x_4) & f_2(x_4) & f_3(x_4) \\ 1 & f_1(x_5) & f_2(x_5) & f_3(x_5) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

$\mathbf{A}\,\mathbf{c} = \mathbf{y}$

$\mathbf{c} = \mathbf{A} \setminus \mathbf{y}$

regression with other basis functions

ti	yi
0	42.7
1	46.7
2	59.1
3	69.5
4	81.0
5	80.7
6	83.2
7	72.0
8	67.1
9	52.6
10	43.7
11	40.9
12	38.6
13	48.8
14	57.2
15	71.2
16	77.5
17	79.8
18	82.3
19	76.3
20	61.5
21	53.0
22	41.5
23	37.3

Example 1: modeling of temperature variations in a city over 24 months

$$y(t) = c_0 + c_1 \cos\left(\frac{2\pi t}{12}\right) + c_2 \sin\left(\frac{2\pi t}{12}\right)$$

basis functions

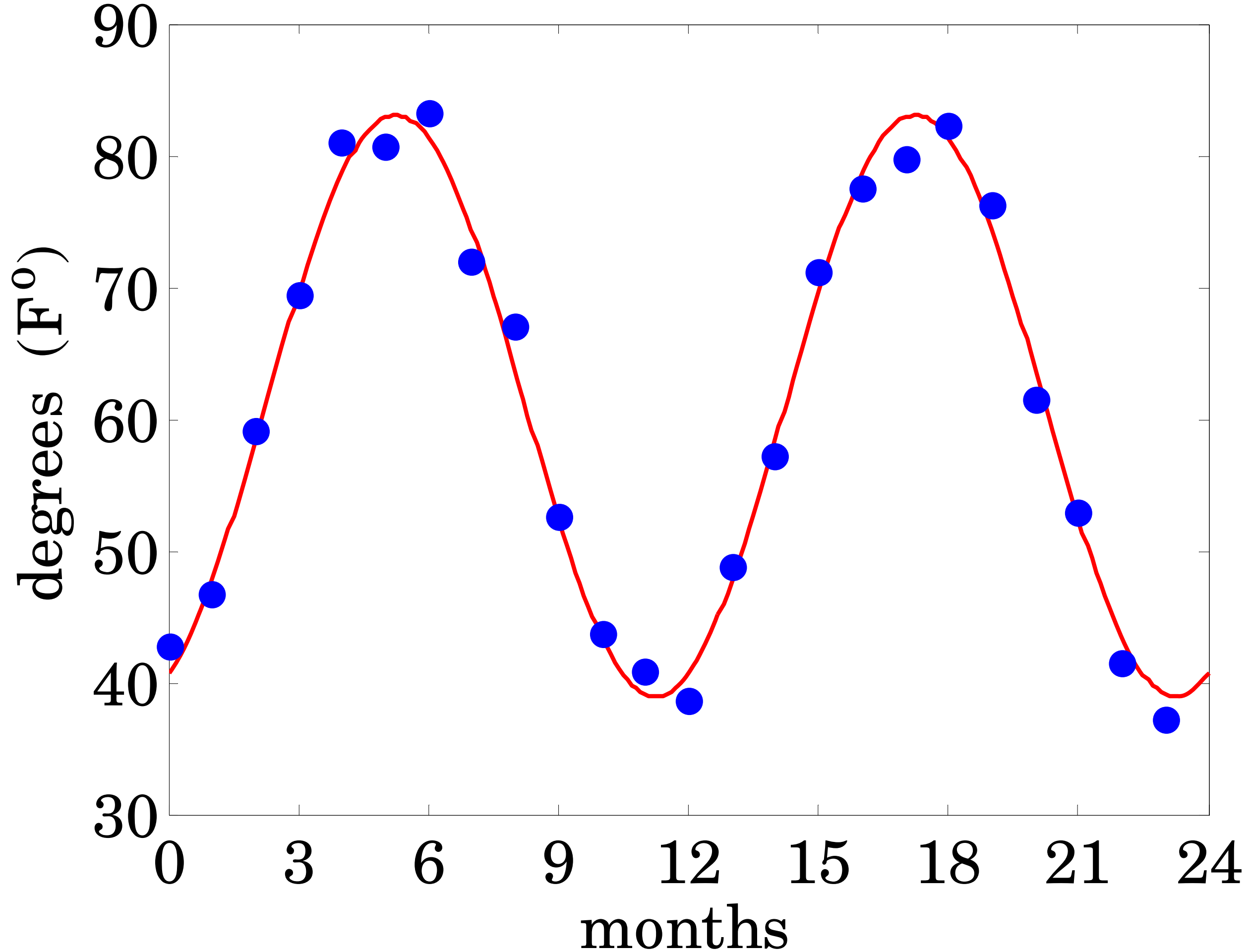
```
A = [ones(N,1), cos(2*pi*ti/12), sin(2*pi*ti/12)];
```

24x3 design matrix

```
c = A\yi

c =
   61.0083
  -20.3333
    8.5565
```

estimated model

```
f = @(t) c(1) + c(2) * cos(2*pi*t/12) + ...
         c(3) * sin(2*pi*t/12);

t = linspace(0,24,241);

plot(t,f(t),'r', ti,yi,'b.','markersize',25);
```



90
80
70
60
50
40
30
degrees (F°)
0
3
6
9
12
15
18
21
24
months

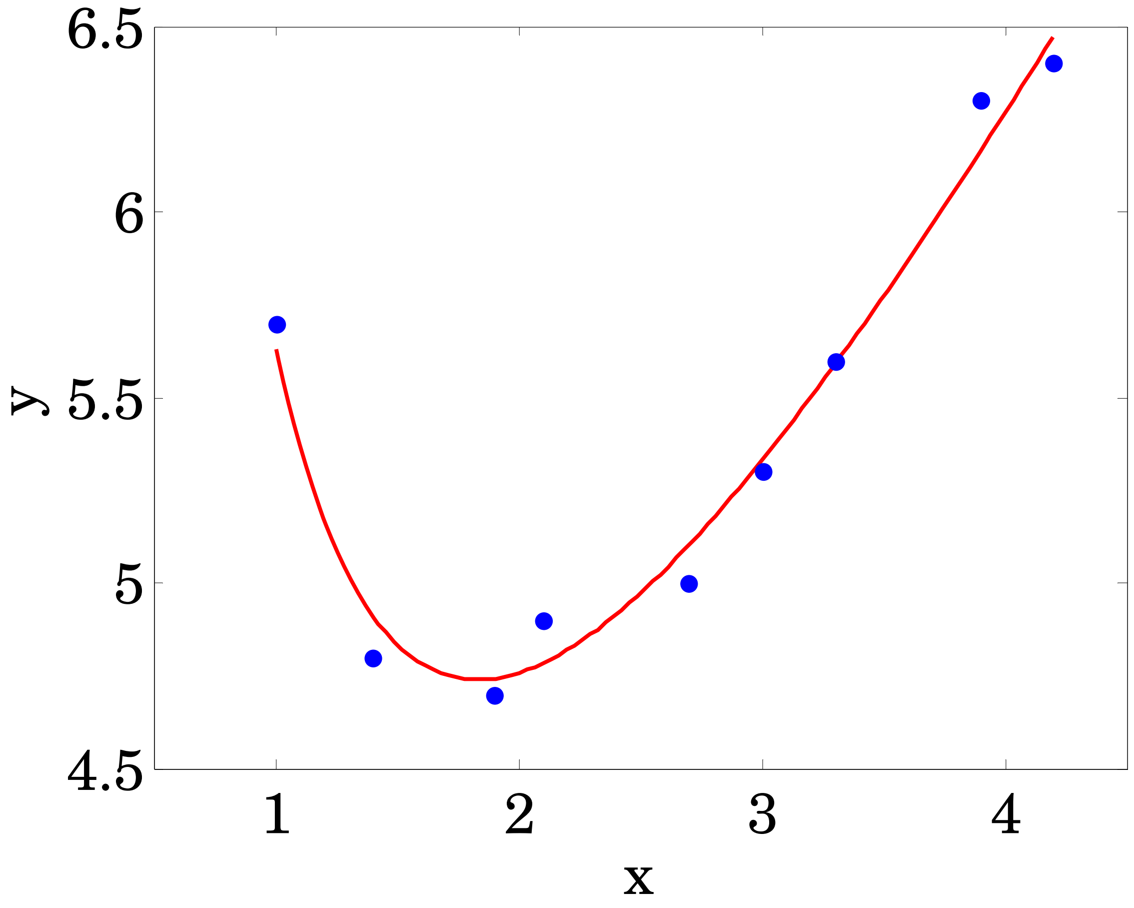
Example 2: $y = \dfrac{c_1}{x} + c_2\, x$

basis functions

xi	yi
1.0	5.7
1.4	4.8
1.9	4.7
2.1	4.9
2.7	5.0
3.0	5.3
3.3	5.6
3.9	6.3
4.2	6.4

```
A = [1./xi, xi];
c = A\yi
c =
    4.3350
    1.2950

x = linspace(1,4.2, 100);
y = c(1)./x + c(2)*x;

plot(x,y,'r-', xi,yi,'b.');
```

Example 2: $y = \dfrac{c_1}{x} + c_2\, x$



multivariate regression

Model:

$$y = c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_M x_M$$

Observations:

$$y(i) = c_0 + c_1 x_{i1} + c_2 x_{i2} + \cdots + c_M x_{iM}, \quad i = 1, 2, \ldots, N$$

Design matrix (for $M = 3$, $N = 5$):

$$\begin{array}{c} \\ i \end{array} \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \\ 1 & x_{21} & x_{22} & x_{23} \\ 1 & x_{31} & x_{32} & x_{33} \\ 1 & x_{41} & x_{42} & x_{43} \\ 1 & x_{51} & x_{52} & x_{53} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

j

observations

predictors

$\mathbf{A}\,\mathbf{c} = \mathbf{y}$

$\mathbf{c} = \mathbf{A} \setminus \mathbf{y}$

```
%   C         T         I         Y       Rating     Player
% ------------------------------------------------------------
  63.8827   5.32151   2.63612   7.65065   94.0      % Joe Montana
  59.5616   6.02740   3.42466   7.63096   89.3      % Dan Marino
  56.7177   5.51422   3.32604   8.03063   87.3      % Boomer Esiason
  57.8466   5.94652   4.08163   7.33920   83.7      % Dave Krieg
  56.9642   5.17241   3.68492   7.67410   83.416    % Roger Staubach
  58.4536   3.86598   2.42268   7.15876   83.415    % Bernie Kosar
  59.6271   3.89137   2.75638   7.12971   83.0      % Ken O'Brien
  59.2422   4.64983   3.61653   7.40586   82.74     % Jim Kelly
  57.6277   4.31335   2.85442   7.22201   82.68     % Neil Lomax
  57.0859   5.98311   4.43454   7.56077   82.6      % Sonny Jurgensen
  57.0970   6.38867   4.89174   7.67469   82.6      % Len Dawson
  59.3073   4.40223   3.57542   7.33810   81.9      % Ken Anderson
  59.6949   5.25424   4.47458   7.44373   81.7      % Danny White
  57.4151   4.82693   4.38234   7.84948   80.5      % Bart Starr
  56.9971   5.28839   4.11319   7.26813   80.4      % Fran Tarkenton
  58.4635   3.97135   3.25521   7.15299   80.3      % Tony Eason
  58.8330   4.53248   4.31834   7.68023   80.2      % Dan Fouts
  57.3457   4.20535   3.60459   7.28291   79.2      % Jim McMahon
  56.0564   4.86084   3.95923   7.13054   78.2      % Bert Jones
  54.5700   5.59198   4.87852   7.75916   78.2      % Johnny Unitas

%   x1        x2        x3        x4        y
```

see week-6 homework

reverse-engineering of the NFL ratings

multivariate regression

```
Y = load('NFL0.dat');  % on sakai

y = Y(:,5);
A = [ones(size(Y,1),1), Y(:,1:4)];

c = A\y
c =
    1.9120
    0.8389
    3.3323
   -4.1573
    4.1428

c = [46  20  80  -100  99]/24;

c ≈ [50  20  80  -100  100]/24;
```

design matrix

see week-6 homework

```
ys = smooth(y);
ys = smooth(y,span);
ys = smooth(y,method);
ys = smooth(y,span,method);
```

data smoothing

```
method = 'moving'  (default, with span=5)
         'loess',  'rloess'
         'lowess', 'rlowess'
         'sgolay'
```

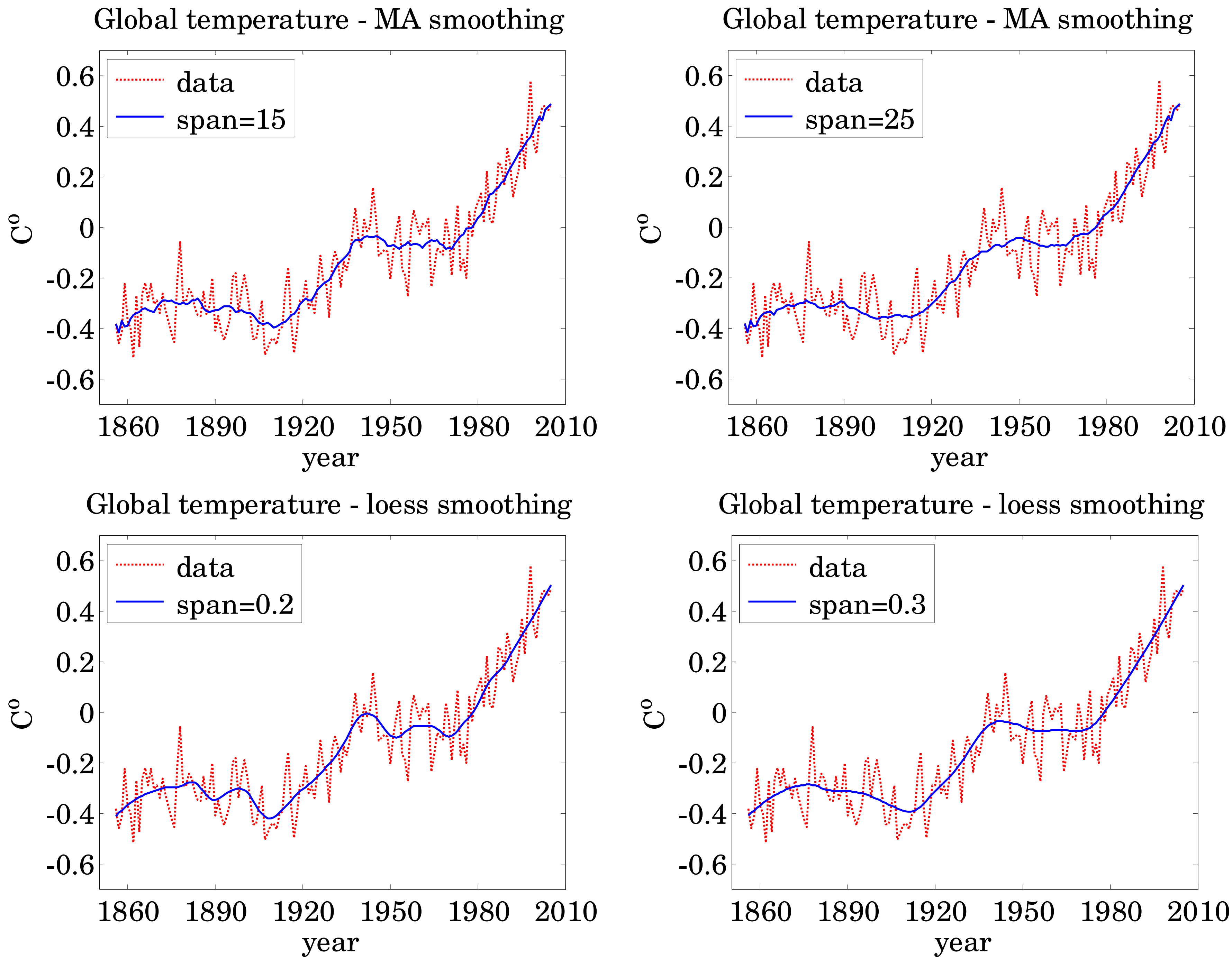
$$y(n) = \frac{x(n-2) + x(n-1) + x(n) + x(n+1) + x(n+2)}{5}$$

$$y(n) = \frac{y(n-M) + \cdots + x(n) + \cdots + x(n+M)}{2M+1}$$

global temperature data (on sakai)

data smoothing

```
A = load('taveGL2v.dat');

t = A(:,1); y = A(:,end);

ys = smooth(y,15);
figure; plot(t,y,'r:', t,ys,'b-');

ys = smooth(y,25);
figure; plot(t,y,'r:', t,ys,'b-');

ys = smooth(y,0.2,'loess');
figure; plot(t,y,'r:', t,ys,'b-');

ys = smooth(y,0.3,'loess');
figure; plot(t,y,'r:', t,ys,'b-');
```


Global temperature - MA smoothing
data
span=15
C^o
year
Global temperature - MA smoothing
data
span=25
Global temperature - loess smoothing
data
span=0.2
Global temperature - loess smoothing
data
span=0.3
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
1860
1890
1920
1950
1980
2010

```
y = filter(b,a,x);
```

output signal

b, **a**, filter coefficients

input signal

digital filtering

```
x = [x0,x1,x2,..., xN] = length-N signal
y = [y0,y1,y2,..., yN] = length-N signal

b = [b0,b1,b2,..., bM] = order-M filter
a = [1, a1,a2,..., aM] = order-M filter
```

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}}$$

transfer function

Example 1: Second-order filter, $M = 2$

digital filtering

$$\mathbf{b} = [b_0,\ b_1,\ b_2]$$
$$\mathbf{a} = [1,\ a_1,\ a_2]$$

$$\boxed{H(z) = \frac{b_0 + b_1\,z^{-1} + b_2\,z^{-2}}{1 + a_1\,z^{-1} + a_2\,z^{-2}}}$$

transfer function

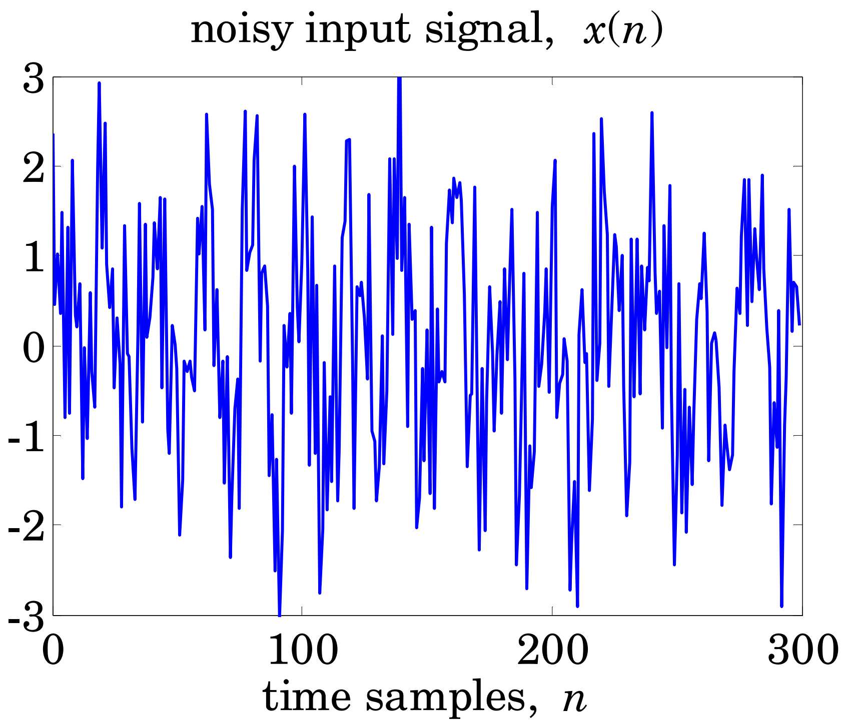
$$y_n = -a_1\,y_{n-1} - a_2\,y_{n-2} + b_0\,x_n + b_1\,x_{n-1} + b_2\,x_{n-2}$$

time-domain implementation
by input/output difference equation,

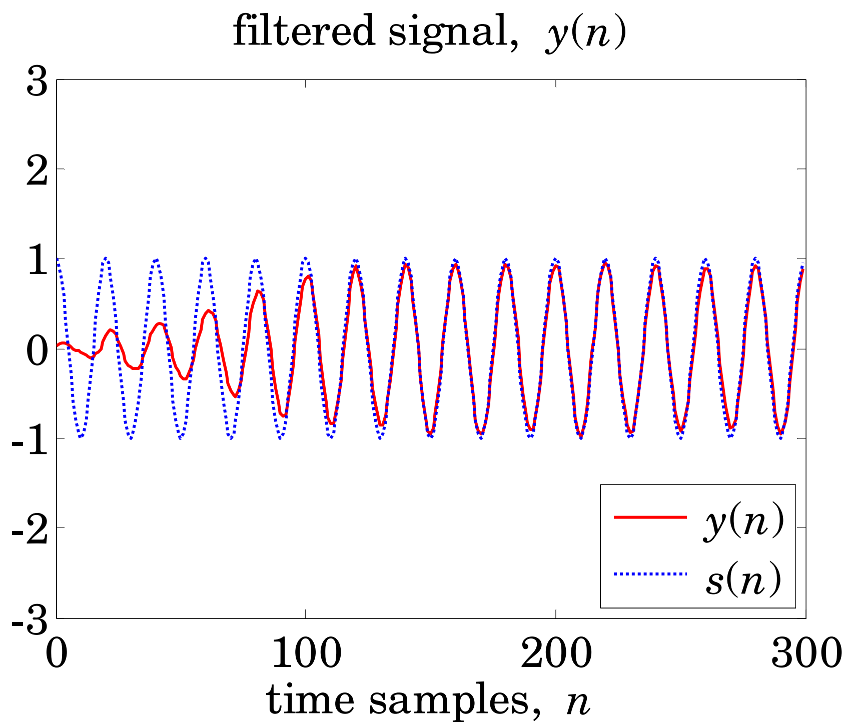
equivalent to: `y = filter(b,a,x)`

Example 2: Bandpass filter, $M = 2$

digital filtering



applications:
radio, TV, cell phone receivers



Example 2: Bandpass filter, $M = 2$

digital filtering

$$H(z) = \frac{b_0 + b_1\, z^{-1} + b_2\, z^{-2}}{1 + a_1\, z^{-1} + a_2\, z^{-2}}$$

transfer function

$$\mathbf{b} = [G,\ 0,\ -G]$$

$$\mathbf{a} = [1,\ -2R\cos\omega_0,\ R^2]$$

$$G = \frac{(1-R)\sqrt{1 - 2R\cos(2\omega_0) + R^2}}{2\sin\omega_0}$$

filter design

$$R = 0.99, \quad f_0 = 500\ \text{Hz}, \quad f_s = 10000\ \text{Hz}, \quad \omega_0 = \frac{2\pi f_0}{f_s} = 0.1\pi$$

```
R = 0.99;
f0 = 500; fs = 10000; w0 = 2*pi*f0/fs;

G = (1-R)*sqrt(1-2*R*cos(2*w0) + ...
    R^2)/2/sin(w0);

a1 = -2*R*cos(w0); a2 = R^2;

a = [1, a1, a2], b = G*[1, 0, -1]

a =
    1.0000   -1.8831    0.9801
b =
    0.0100         0   -0.0100
```

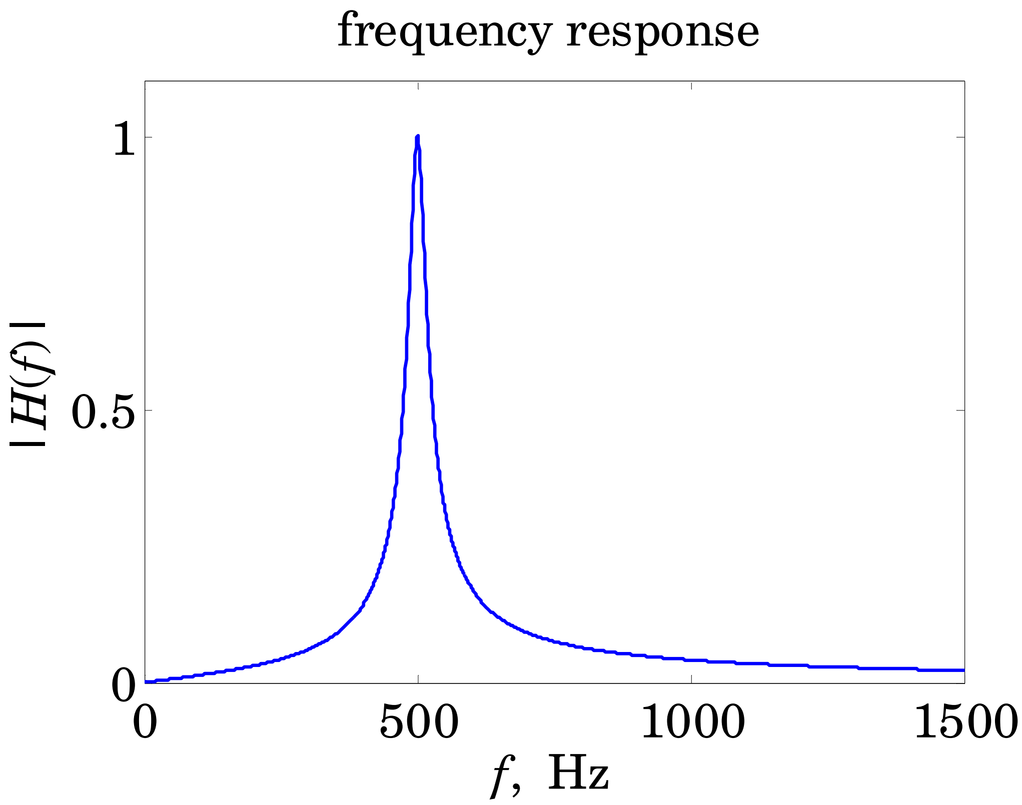
```
f = linspace(0,1500,1501); w = 2*pi*f/fs;

H = abs(freqz(b,a,w));    % frequency response
```

$$H(z) = \frac{G(1 - z^{-2})}{1 - 2R\cos\omega_0)\, z^{-1} + R^2\, z^{-2}}$$

$$z = e^{j\omega}$$

$$\omega = \frac{2\pi f}{f_s}$$

```
>> doc freqz
```



$$s(n) = \cos(\omega_0 n)$$

$$x(n) = s(n) + v(n)$$

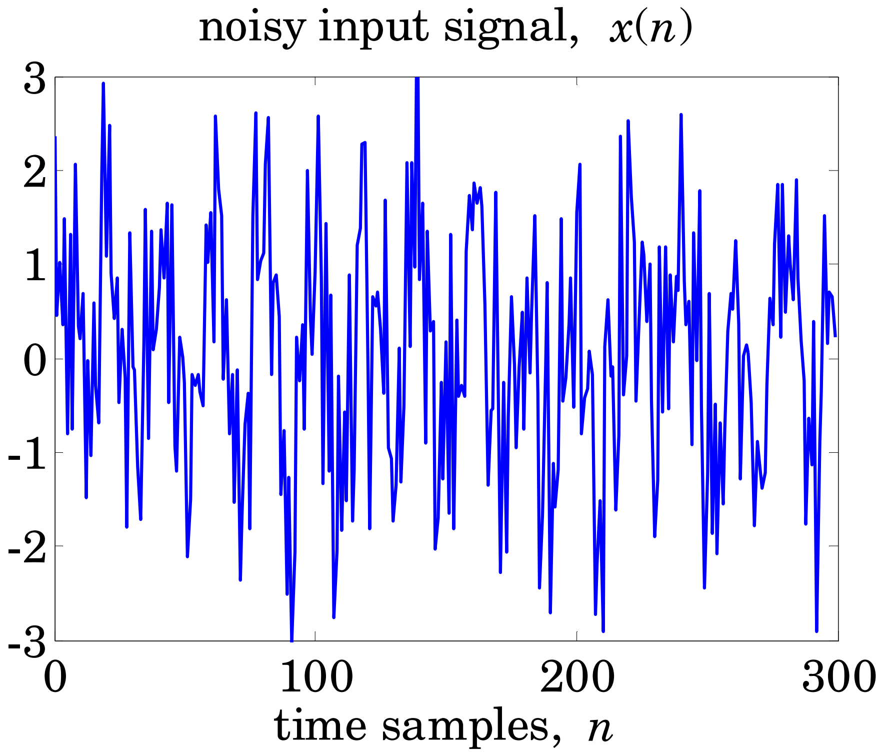
desired signal

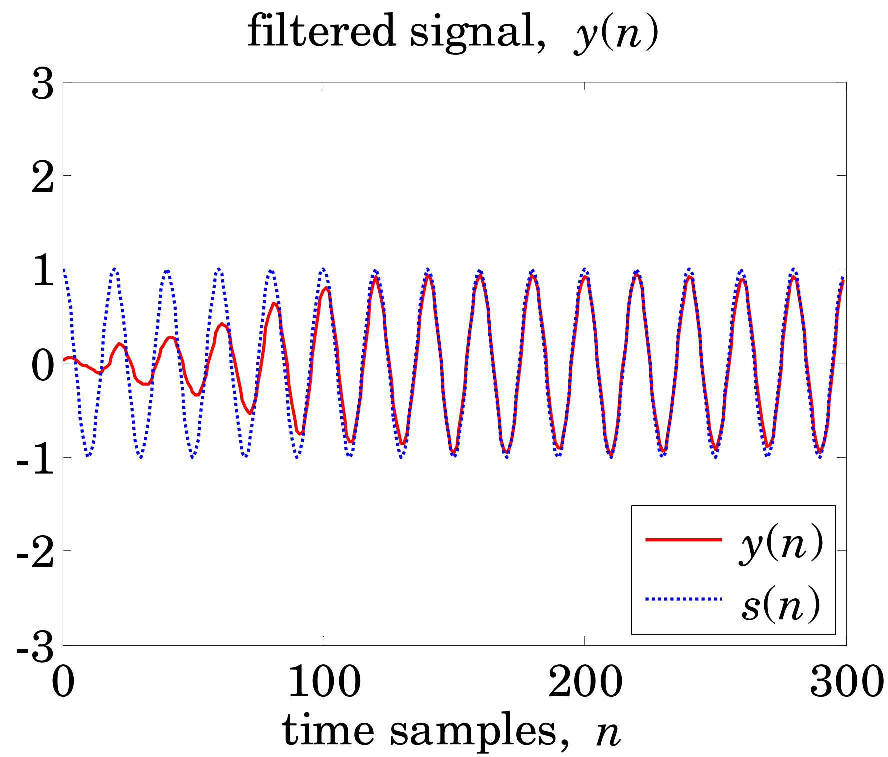
random noise

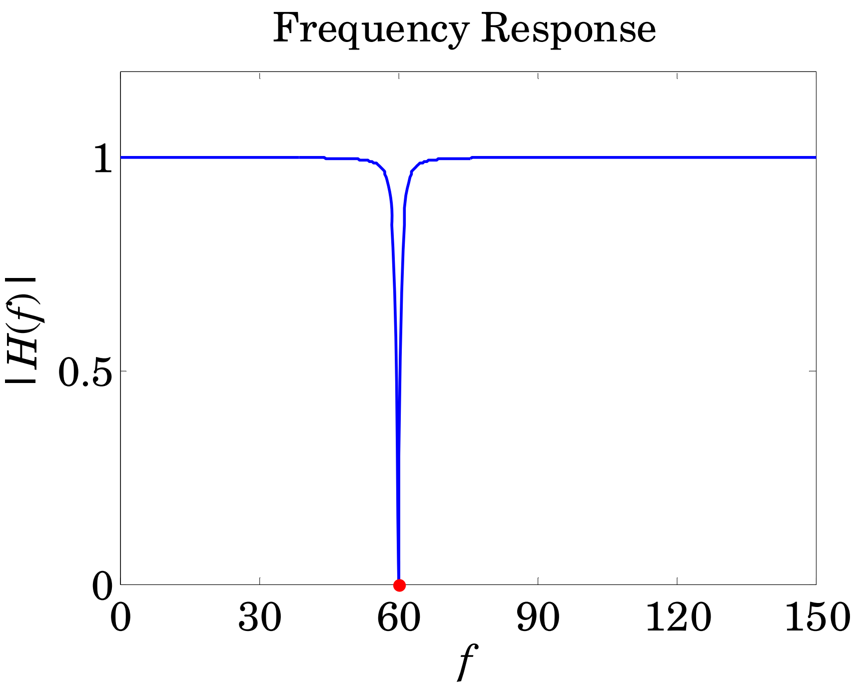
digital filtering

```
N=300; n = 0:N-1;

s = cos(w0*n);
rng(200); v = randn(1,N);

x = s + v;                  % noisy signal

y = filter(b,a,x);          % filtering

figure; plot(n,x, 'b-');

figure; plot(n,y,'r-', n,s, 'b:');
```

digital filtering

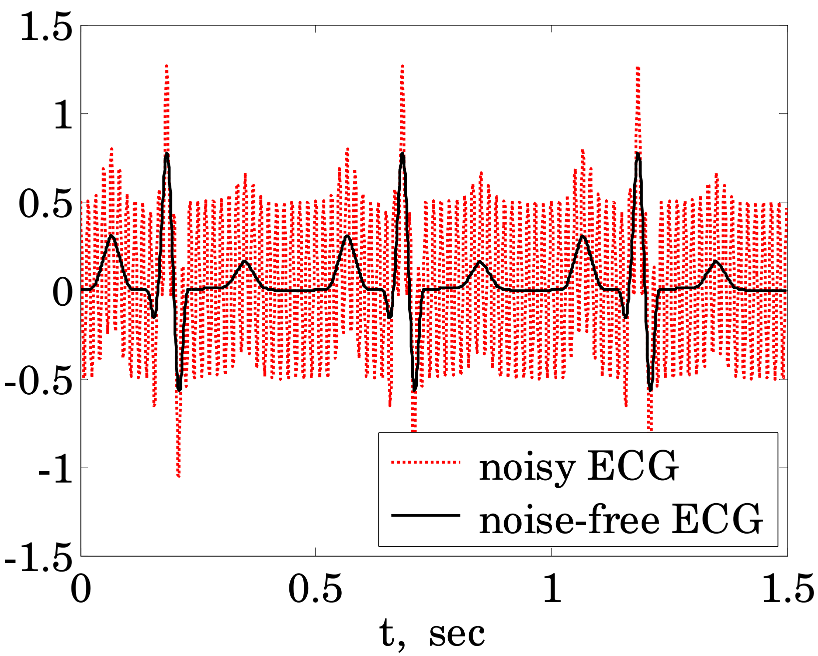


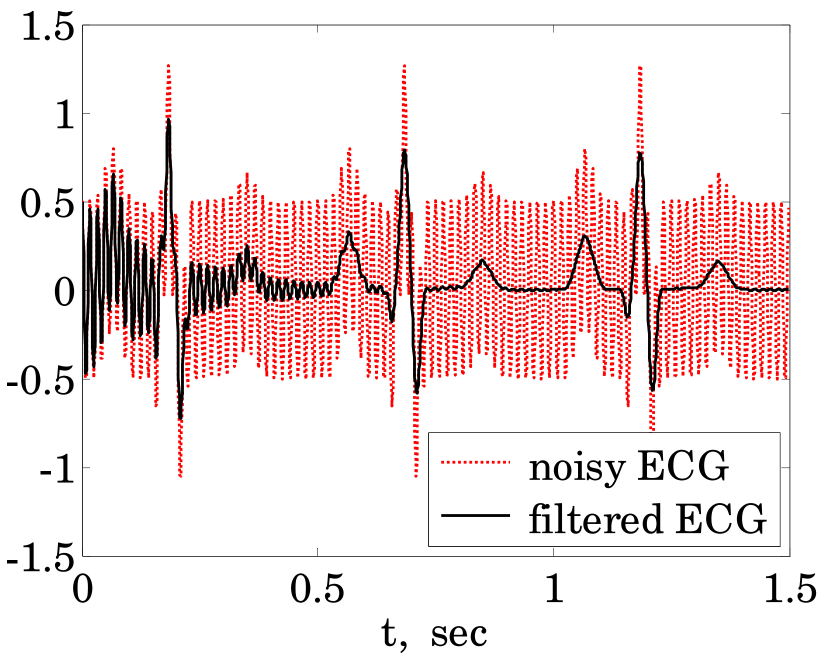




digital filtering

Example 3: Removing 60 Hz interference from an ECG using a notch filter at 60 Hz



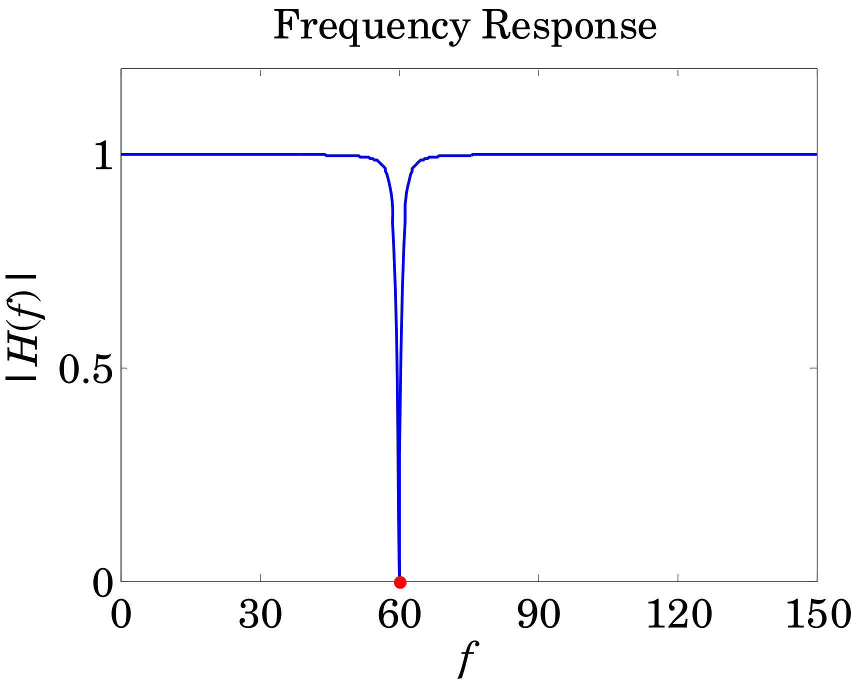


Example 3: ECG + 60 Hz interference

digital filtering

```
f0 = 60; fs = 1000; T = 1/fs;  % sampling rate

N = 500; M = 3;   % samples per beat, no. beats

s = sgolayfilt(ecg(N),0,15);   % simulated ECG
s = [s,s,s];                   % 3 beats

n = 0:length(s)-1; tn = n*T;   % sampling times

x = s + 0.5*cos(2*pi*f0*tn);
% ECG + 60 Hz interference
```

```
w0 = 2*pi*f0/fs;                    % digital frequency
R = 0.995;                          % pole radius

G = (1-2*R*cos(w0)+R^2)/(2-2*cos(w0));  % gain

a = [1, -2*R*cos(w0), R^2];         % denominator
b = G*[1, -2*cos(w0), 1];           % numerator

f = linspace(0,150,601); w = 2*pi*f/fs;

H = abs(freqz(b,a,w));              % frequency response

plot(f,H,'b', f0,0,'r.','markersize',16);

y = filter(b,a,x);                  % filter noisy ECG

plot(tn,x,'r:', tn,s,'k-');         % noisy ECG
plot(tn,x,'r:', tn,y,'k-');         % filtered ECG
```



notch filter at 60 Hz

$$H(z) = G\frac{1 - 2\cos\omega_0\, z^{-1} + z^{-2}}{1 - 2R\cos\omega_0\, z^{-1} + R^2\, z^{-2}}$$

$$z = e^{j\omega}$$

$$\omega = \frac{2\pi f}{f_s}$$