Introduction to Applied Scientific Computing using MATLAB

Mohsen Jenadeleh

In this lecture, slides from Rutgers and Waterloo universities are used to form the lecture slides

Numerical Methods
Data Fitting, Smoothing, Filtering

- data fitting with polynomials – **`polyfit, polyval`**
- examples: Moore's law, Hank Aaron, US census data
- nonlinear data fitting, **`nlinfit, lsqcurvefit`**
- data interpolation – **`interp1,2, 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)**

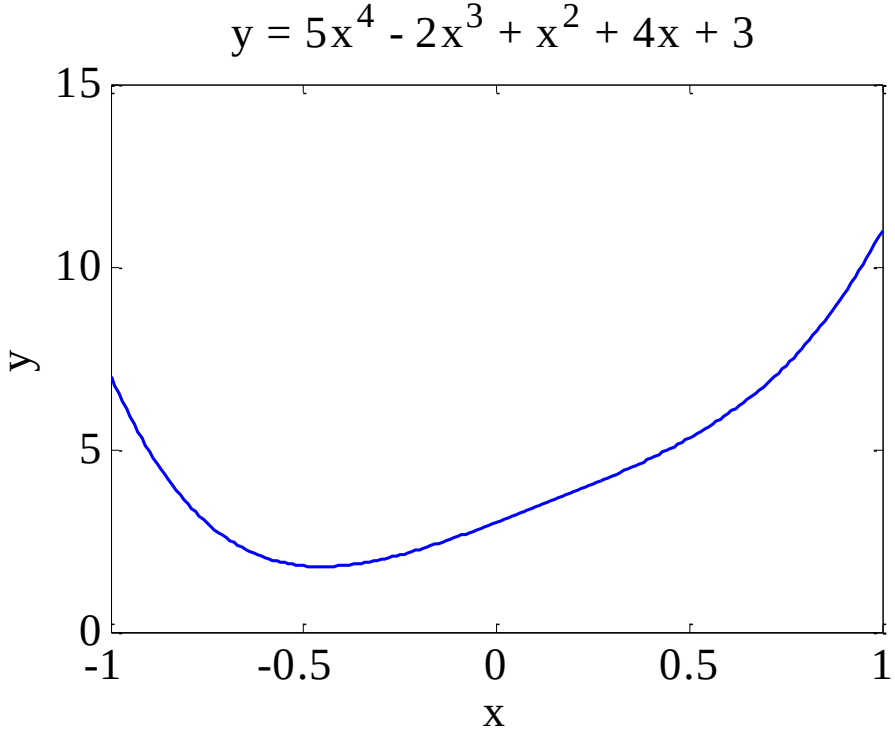
Given N data points $\{x_i, y_i\}$, i=1,2,…,N, find an M-th degree polynomial that best fits the data – **(polyfit)**

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,…,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

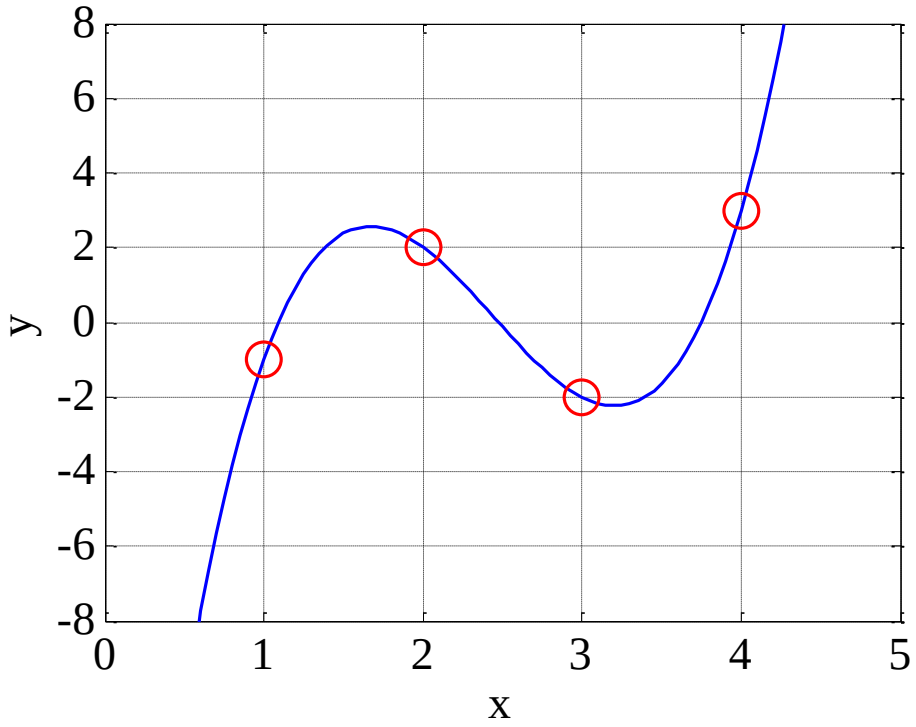
if $N = M+1$, the polynomial interpolates the data

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

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

polyfit, polyval

```
>> xi = [1,2,3,4];
>> 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');
```



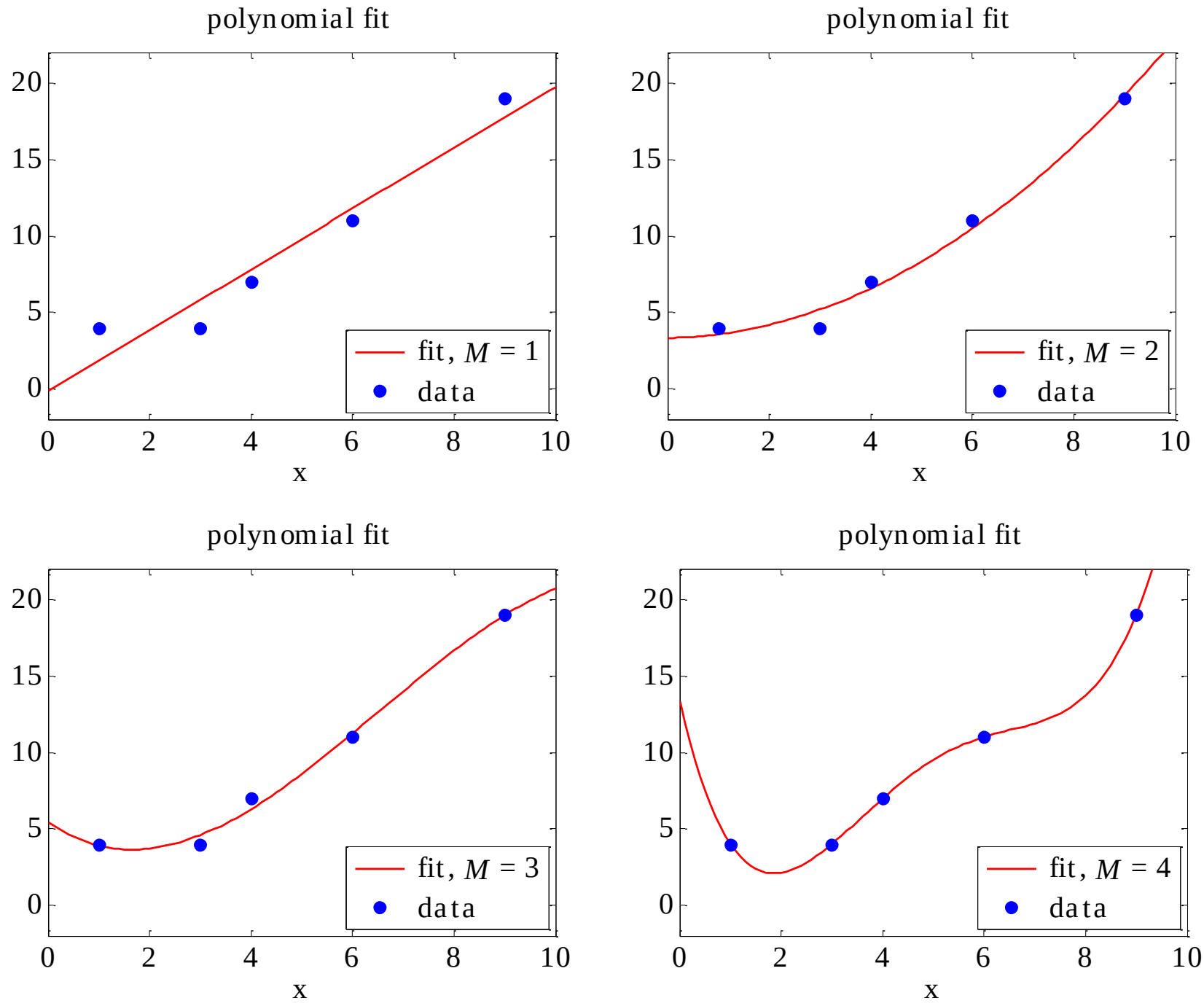
N = number of data points
M = polynomial order

here, $N = M+1 = 3+1=4$,

therefore, the polynomial interpolates the data

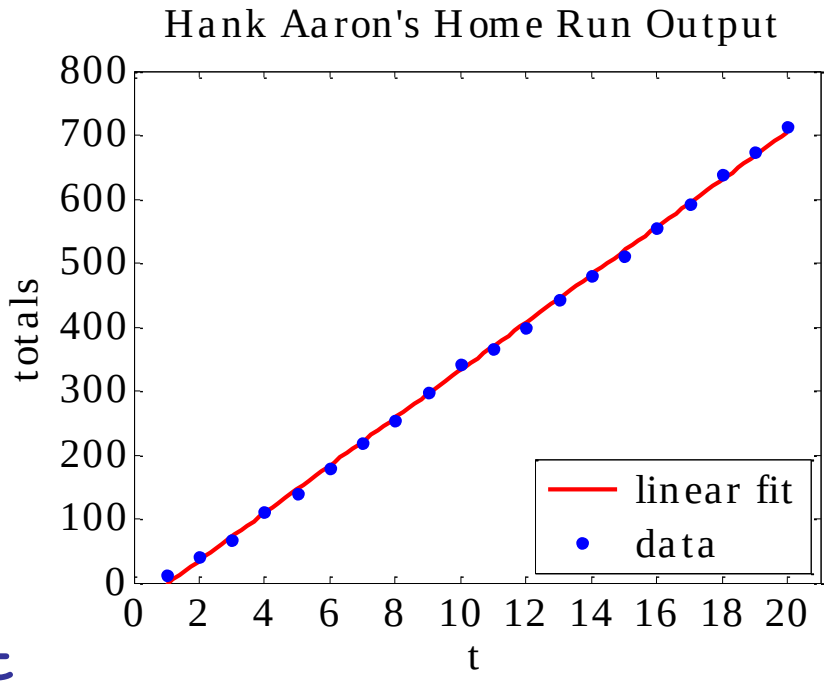
polyfit, polyval

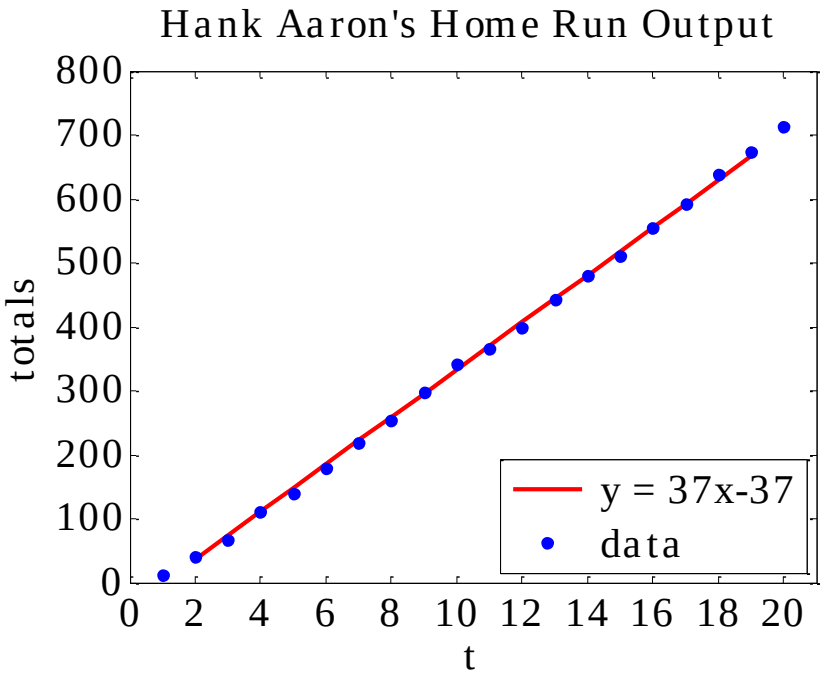
```
xi = [1, 3, 4,  6,  9];
yi = [4, 4, 7, 11, 19];

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
fit, M = 1
data
x
polynomial fit
fit, M = 2
data
x
polynomial fit
fit, M = 3
data
x
polynomial fit
fit, M = 4
data
x

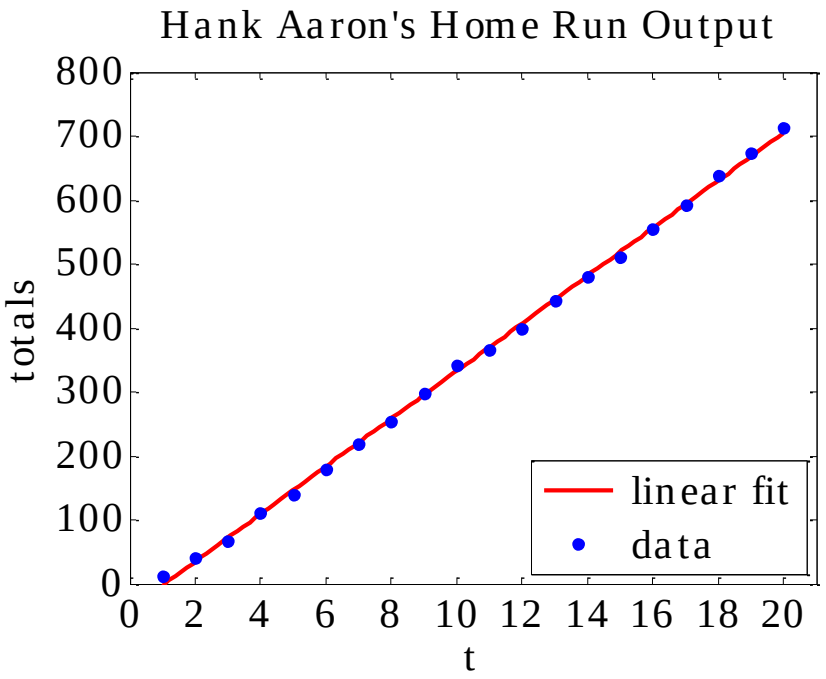
```
% year     ti    H
% ---------------
   1954     1    13
   1955     2    27
   1956     3    26
   1957     4    44
   1958     5    30
   1959     6    39
   1960     7    40
   1961     8    34
   1962     9    45
   1963    10    44
   1964    11    24
   1965    12    32
   1966    13    44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

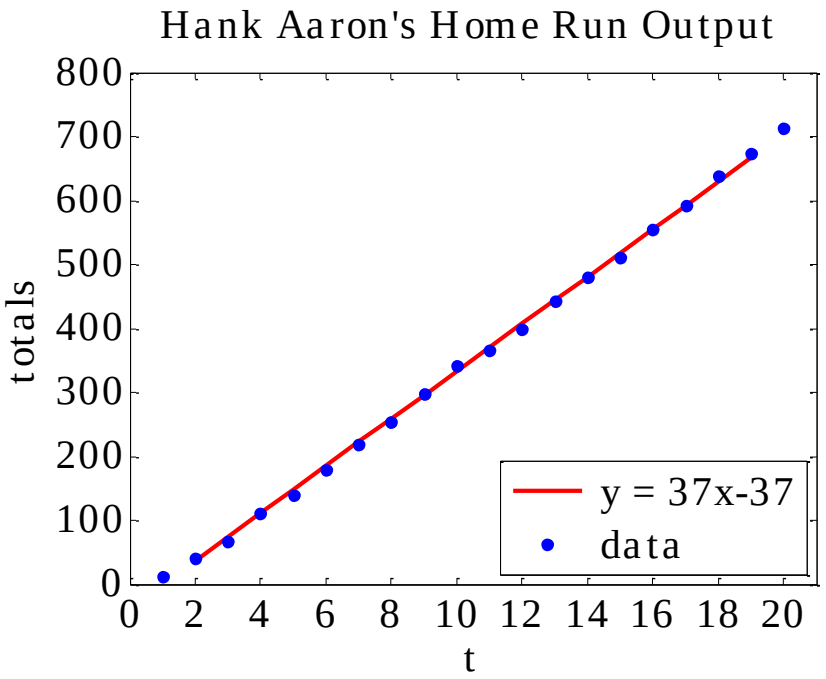
aaron.dat





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

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

p = polyfit(ti,yi,1)

% p =
%    37.2617  -39.8474

t = linspace(1,20, 101);
y = polyval(p,t);

plot(t,y,'r-', ...
     ti,yi,'b.', ...
     'markersize', 18);
```





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:	$y = b\,x^a$	$\Rightarrow$	$\log(y) = a\,\log(x) + \log(b)$

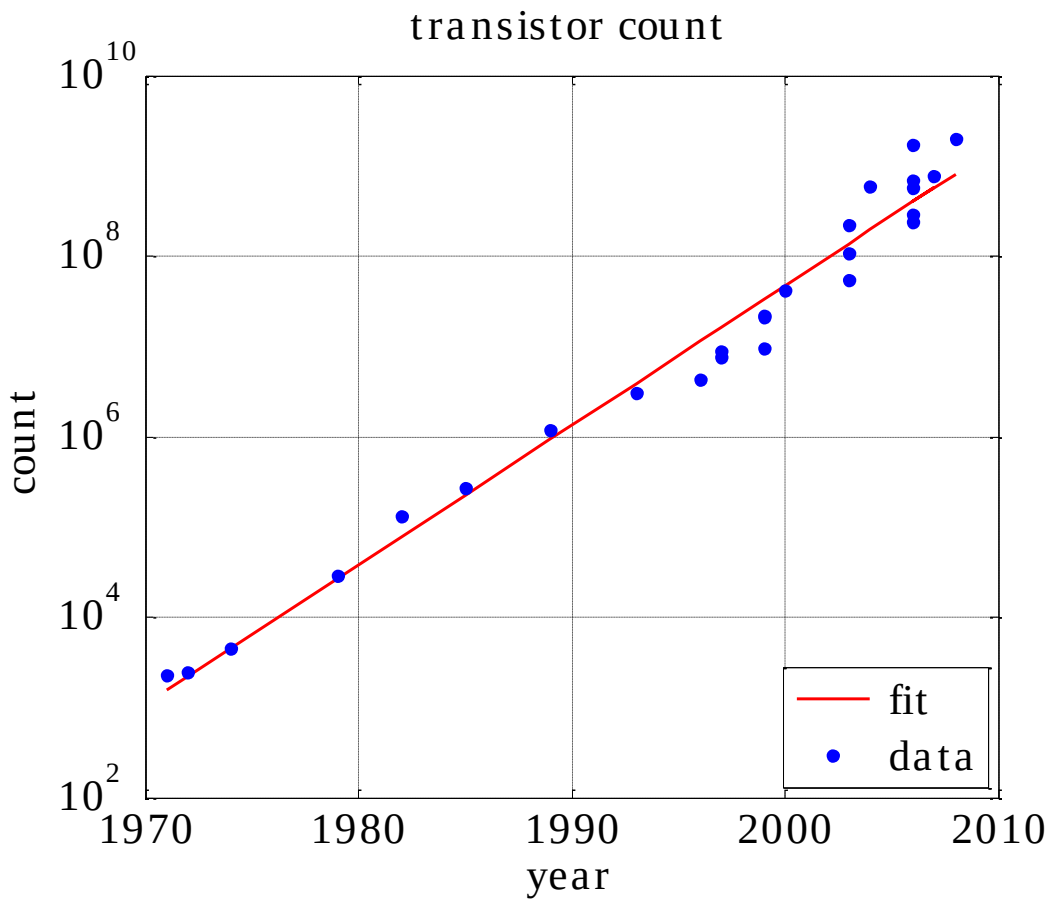
```
p = polyfit(xi,log(yi),1);   % exponential
y = exp(polyval(p,x));       % y=exp(a*x+log(b))
a = p(1);                  % y = exp(p(1)*x+p(2))
b = exp(p(2));           % so that y = b*exp(a*x)
```

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

fitted model $f(t) = b\,2^{a(t-t_1)}$

$$\log_2 f(t) = \log_2 b + a(t - t_1)$$

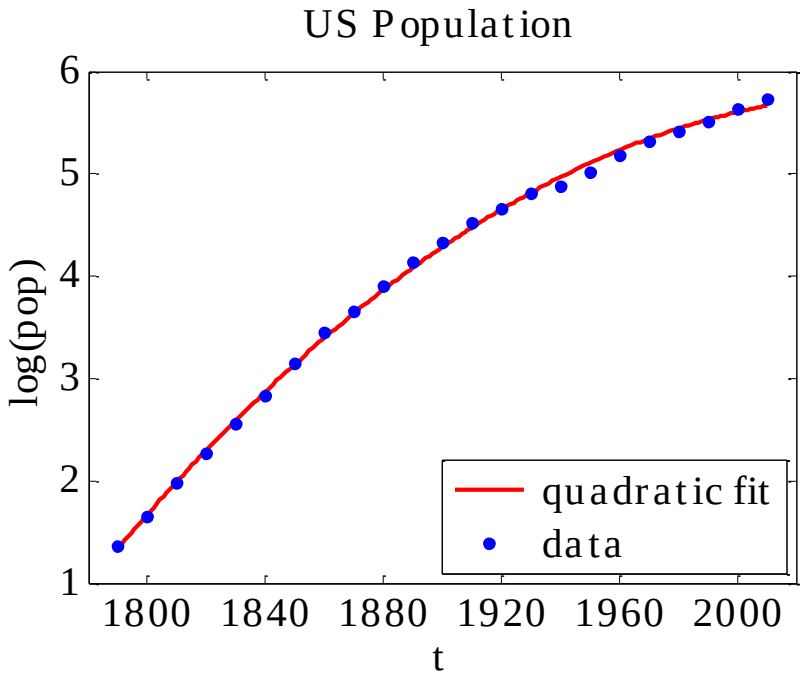


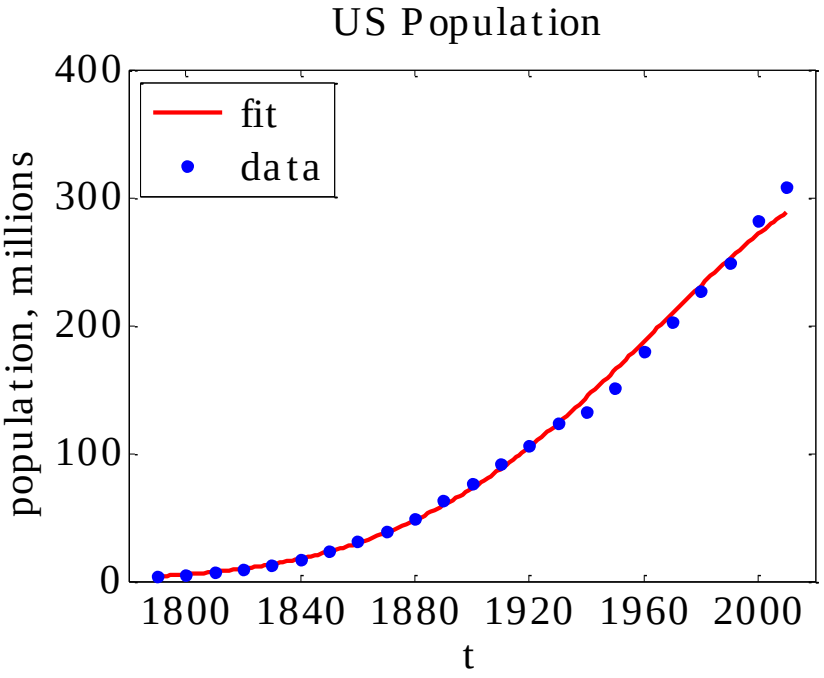
```
Y = load('transistor_count.dat');

y = Y(:,1);  t = Y(:,2);

t1 = t(1);   % t1 = 1971

p = polyfit(t-t1, log2(y), 1);

% p =
%   0.5138  10.5889  % b = 2^p(2) = 1.5402e+003

f = 2.^(polyval(p,t-t1));

semilogy(t,f,'r-', t,y,'b.', 'markersize',18)
```

fitted model:

```
f(t) = b * 2.^(a*(t-t1)) = 2.^(a*(t-t1)+log2(b));

% a = p(1), log2(b) = p(2) --> b = 2^(p(2))
```

```
% 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');

ti = A(:,1); yi = A(:,2);

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

% p =
%  -0.0001   0.2653  -266.4672

t = linspace(1790, 2010, 201);
y = exp(polyval(p,t));

figure; plot(t, log(y), 'r-', ...
             ti,log(yi),'b.','markersize',18);

figure; plot(t, y,'r-', ...
             ti,yi,'b.','markersize',18);
```

Planetary Temperatures

$$T = \frac{T_e}{r^{1/2}}$$

T_e = Earth's temperature
r = distance from sun in AU

assumed model

$$T = \frac{a}{r^b}$$

transformed model

$$\ln(T) = \ln(a) - b\,\ln(r)$$

planet	ri	Ti	T_est
Mercury	0.390	452	468
Earth	1.000	285	283
Mars	1.520	230	227
Jupiter	5.200	120	118
Saturn	9.539	88	85
Uranus	19.180	59	59
Neptune	30.060	48	46
Pluto	39.530	37	40

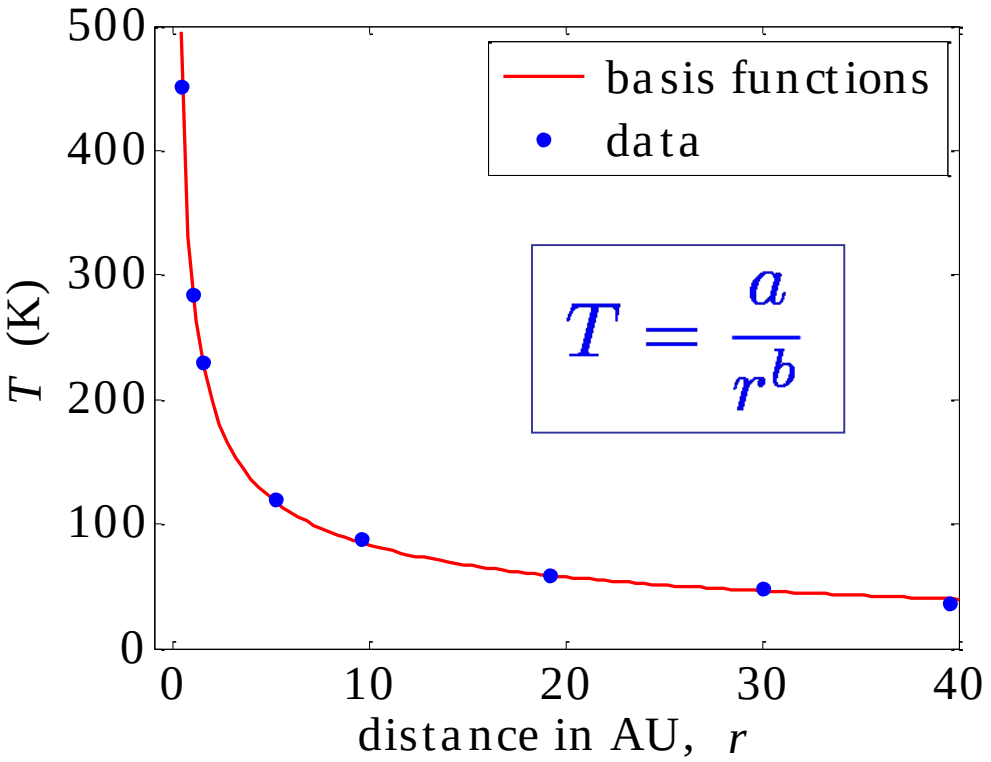
Reference: M. C. LoPresto and N. Hagoort, "Determining Planetary Temperatures with the Stefan-Boltzmann Law," *Phys. Teacher*, vol.49, 113 (2011). On sakai.

```
ri = [0.39, 1, 1.52, 5.2, 9.539, 19.18, 30.06, 39.53]';
Ti = [452, 285, 230, 120, 88, 59, 48, 37]';      % columns

p = polyfit(log(ri),log(Ti),1)
c(2)=-p(1); c(1)=exp(p(2));

f = @(c,r) c(1)./r.^c(2);

r = linspace(0.35, 40, 100);
T = f(c,r);
T_est = f(c,ri);

plot(r,T,'r-', ri,Ti,'b.')
```

```
% basis functions method
c = [ri.^0, -log(ri)] \ log(Ti);
c(1) = exp(c(1));
```

estimated:
a = c(1) = 283.48
b = c(2) = 0.5329

```
% c0 = [200;1]';
% c = nlinfit(ri,Ti,f,c0)
% c = [280.78; 0.5130]
% nlinfit version
```



The curve fitting toolbox allows more complicated nonlinear data fits. See also the statistics and optimization toolboxes.

```
>> doc curvefit        % curve fitting toolbox

>> doc nlinfit         % in statistics toolbox

>> doc lsqcurvefit     % in optimization toolbox
```

```
c = nlinfit(xi,yi,f,c0);

% c = estimated parameter vector
% xi,yi = vectors of data points
% f = handle to fitting function y = f(c,x)
% c0 = initial parameter vector


% c = lsqcurvefit(f,c0,xi,yi);
```

c must be first

must be vectorized

An equivalent, do-it-yourself, least-squares version, uses the built-in function **fminsearch**, to mininize the least-squares error criterion:

```
J = @(c) sum((yi-f(c,xi)).^2);     % L2 norm
c = fminsearch(J,c0);

% one can incorporate a weight vector wi
% J = @(c) sum(wi.*(yi-f(c,xi)).^2)

% one can also use the L1-norm criterion,
% which is generally more appropriate if
% there are outliers in the data

J = @(c) sum(abs(yi-f(c,xi)));     % L1 norm
c = fminsearch(J,c0);
```

L_2 criterion:

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

L_1 criterion:

$$J(c) = \sum_{i=1}^{N} \left| y_i - f(c, x_i) \right| = \min$$

```
J = @(c) sum((yi-f(c,xi)).^2);      % L2
J = @(c) sum(abs(yi-f(c,xi)));      % L1

% we can also use the built-in function
% norm(x), norm(x,1) to define J(c):

J = @(c) norm(yi-f(c,xi))^2;        % L2 norm
J = @(c) norm(yi-f(c,xi),1);        % L1 norm
```

useful built-in MATLAB functions

```
>> doc curvefit          % curve fitting toolbox

% see also

>> doc nlinfit           % in statistics toolbox

>> doc lsqcurvefit       % in optimization toolbox
>> doc lsqnonlin         % nonlinear least-squares
>> doc lsqlin            % linear with constraints
>> doc linprog           % linear programming
>> doc lsqnonneg         % positivity contraints
>> doc quadprog          % quadratic programming
>> doc bintprog          % binary int programming
```

nlinfit Example 1: $f(c,t) = c_1 + c_2 \cos(c_3 t) \exp(-c_4 t)$

parameter vector $c = [c_1, c_2, c_3, c_4]'$

```
f = @(c,t) c(1) + c(2)*cos(c(3)*t).*exp(-c(4)*t);

ce = [1 2 15 1]';    % c = [c1,c2,c3,c4]' = column

rng(201);
ti = 0:0.1:2.9;
yi = f(ce,ti) + 0.2*randn(size(ti));

c0 = [10 10 10 10]';

c = nlinfit(ti,yi,f,c0);
% c = lsqcurvefit(f,c0,ti,yi);
```

exact model

noisy data

initial guess

estimated parameter vector

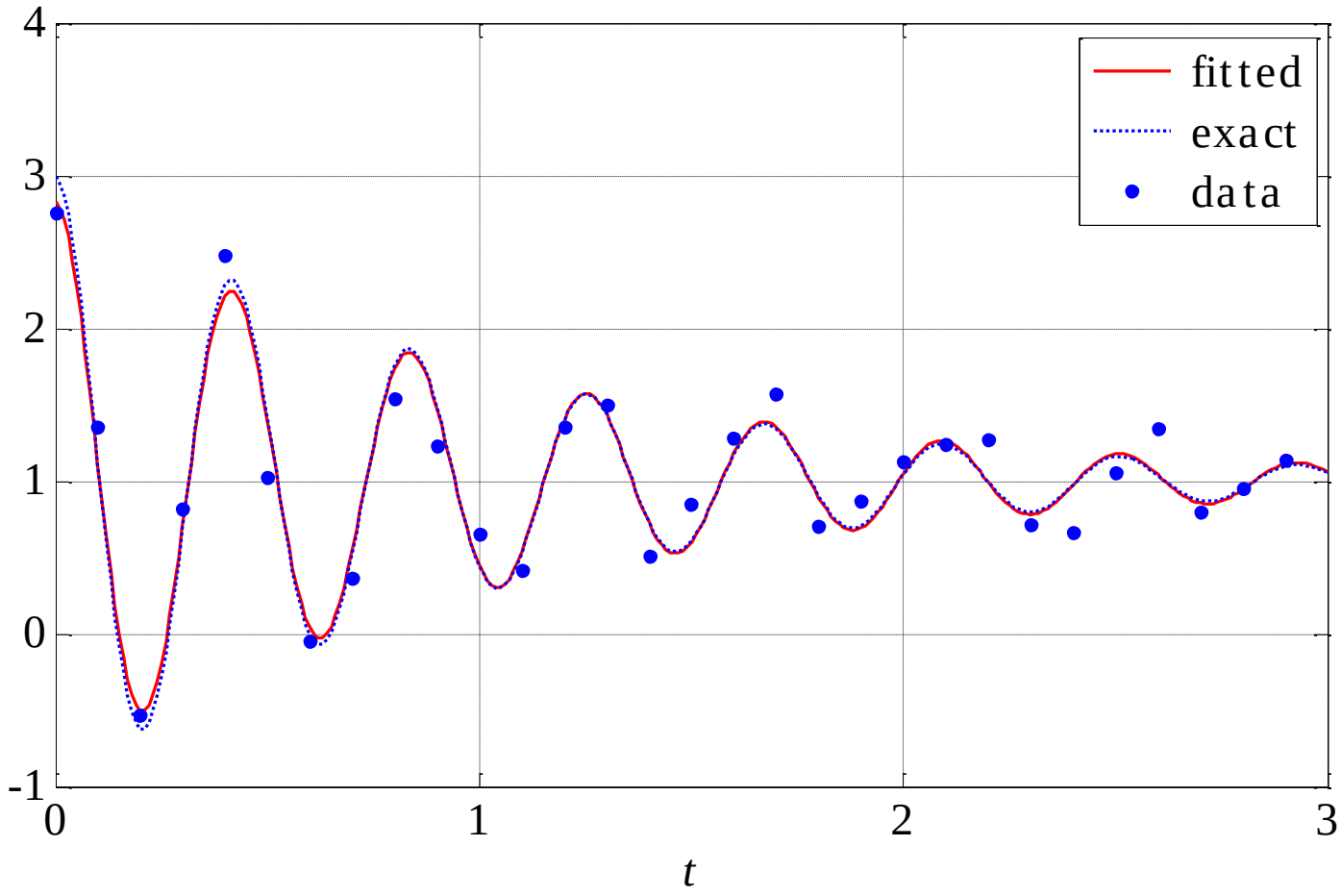
```
[c,be] =
    0.9999    1
    1.8271    2
   15.0056   15
    0.9238    1
```

```
t = linspace(0,3,301);

plot(t,f(c,t),'r-', t,f(ce,t),'b:', ti,yi,'b.')
```

fitted

exact

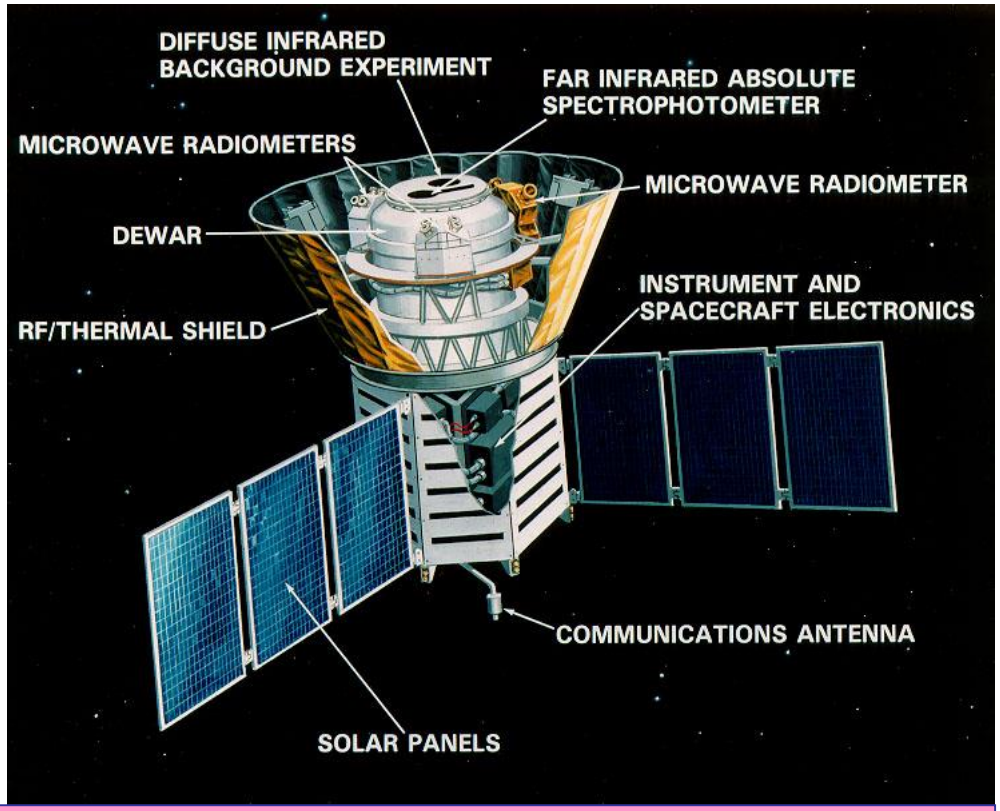
data



`nlinfit` Example 2: Cosmic Microwave Background (CMB)

In 1964 Penzias and Wilson discovered the cosmic blackbody thermal radiation background, a remnant of the Big Bang, that almost uniformly fills the entire universe, and predicted theoretically in 1948 by Gamow, Alpher and Herman.

NASA's Cosmic Background Explorer (COBE) satellite, launched in 1989, carried out very accurate measurements of the CMB frequency spectrum using a Far-Infrared Absolute Spectrophotometer (FIRAS). The measurements provided an almost perfect fit to Planck's blackbody spectrum at a temperature of $T = 2.7250$ K



source: http://aether.lbl.gov/www/projects/cobe/cobe_pics.html

Planck's blackbody radiation formulas

$$I(T,f) = \frac{2hf^3}{c^2} \frac{1}{\exp\left(\frac{hf}{kT}\right) - 1}$$

$$\left[\frac{\mathrm{W}}{\mathrm{m}^2 \cdot \mathrm{Hz} \cdot \mathrm{sr}}\right]$$

$$A = \frac{10^{47} 2h}{c^2}, \quad B = \frac{10^9 h}{k}$$

spectral radiance

$$I(T,f) = \frac{Af^3}{\exp\left(\frac{Bf}{T}\right) - 1}$$

$[f \text{ in GHz}, \ I(T,f) \text{ in MJy/sr}]$

$$1 \text{ Jy} = 10^{-26} \frac{\mathrm{W}}{\mathrm{m}^2 \cdot \mathrm{Hz}} = 1 \text{ Jansky}$$

Least-squares method

$$I(T,f) = \frac{Af^3}{\exp(Bf/T) - 1}$$

Given N measured data, $\{f_i, I_i\}$, $i = 1, 2, \ldots, N$,
the ith measurement error is:

$$e_i(T) = I_i - I(T, f_i)\,, \quad i = 1, 2, \ldots, N$$

Least-squares error criterion:

$$E(T) = \sum_{i=1}^{N} |e_i(T)|^2 = \sum_{i=1}^{N} |I_i - I(T, f_i)|^2 = \min$$

find T that minimizes $E(T)$
using **`fminbnd`** or **`nlinfit`**

FIRAS data, see file **firas.dat** and Fixsen, et al.

http://lambda.gsfc.nasa.gov/product/cobe/firas_monopole_get.cfm

fi	Ii	fi	Ii	fi	Ii
2.27	200.723	9.08	248.239	15.88	36.352
2.72	249.508	9.53	225.940	16.34	31.062
3.18	293.024	9.98	204.327	16.79	26.580
3.63	327.770	10.44	183.262	17.24	22.644
4.08	354.081	10.89	163.830	17.70	19.255
4.54	372.079	11.34	145.750	18.15	16.391
4.99	381.493	11.80	128.835	18.61	13.811
5.45	383.478	12.25	113.568	19.06	11.716
5.90	378.901	12.71	99.451	19.51	9.921
6.35	368.833	13.16	87.036	19.97	8.364
6.81	354.063	13.61	75.876	20.42	7.087
7.26	336.278	14.07	65.766	20.87	5.801
7.71	316.076	14.52	57.008	21.33	4.523
8.17	293.924	14.97	49.223		
8.62	271.432	15.43	42.267		

fi in units of cm^{-1}

Ii in units of MJy/sr

```
h = 6.62606957e-34;     % J/Hz, Planck's constant
c = 2.99792458e8;       % m/s,  speed of light
k = 1.3806488e-23;      % J/K,  Boltzmann constant
A = 10^47 * 2*h/c^2;    % A = 0.00147450
B = 10^9 * h/k;         % B = 0.04799243

Y = load('firas.dat');     % load COBE data
fi = Y(:,1) * c * 1e-7;    % convert fi to GHz
Ii = Y(:,2);               % Ii in MJy/sr

I = @(T,f) A*f.^3./(exp(B*f/T) - 1);
T = nlinfit(fi,Ii,I,3);

% implement our own least-squares method

E = @(T) sum((Ii - I(T,fi)).^2);
T = fminbnd(E,1,3);
```

vectorized in **f**

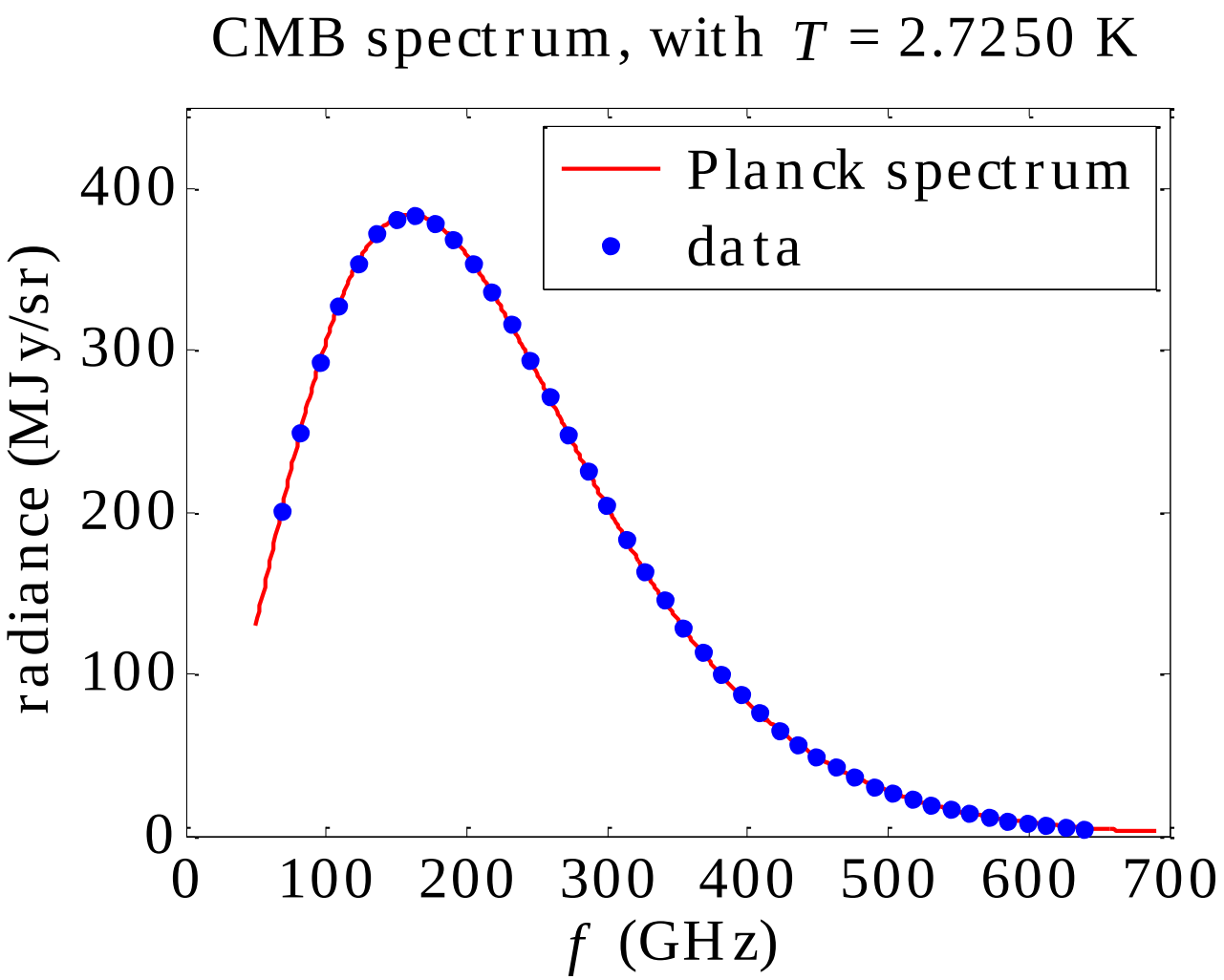
```
% T = 2.725013 K
```

sum of squared errors

```
% T = 2.725015 K
```

```
f = linspace(50,690,641);
If = I(T,f);
plot(f,If,'r-', fi,Ii,'b.');
```

CMB data provide a remarkable confirmation of quantum theory



nlinfit Example 3:

$$y = (At + B)e^{-at}$$

assumed model

```
% generate simulated data
A = 5; B = 1; a = 2;
rng(100);
ti = 0:0.3:3;
yi = (A*ti+B).*exp(-a*ti) + ...
      0.05*randn(size(ti));
yi = round(yi*100)/100;
ce = [A,B,a]';
c0 = [1 1 1]';
% define nlinfit model function
f = @(c,t) (c(1)*t+c(2)).*exp(-c(3)*t);
```

exact values

exact parameters

initial parameters

ti	yi
0.0	1.01
0.3	1.34
0.6	1.19
0.9	0.94
1.2	0.59
1.5	0.44
1.8	0.32
2.1	0.09
2.4	0.11
2.7	0.13
3.0	-0.01

```
c = nlinfit(ti,yi,f,c0)
t = linspace(0,3,301);
plot(t,f(ce,t),'k:', t,f(c,t),'r-',ti,yi,'b.')
```

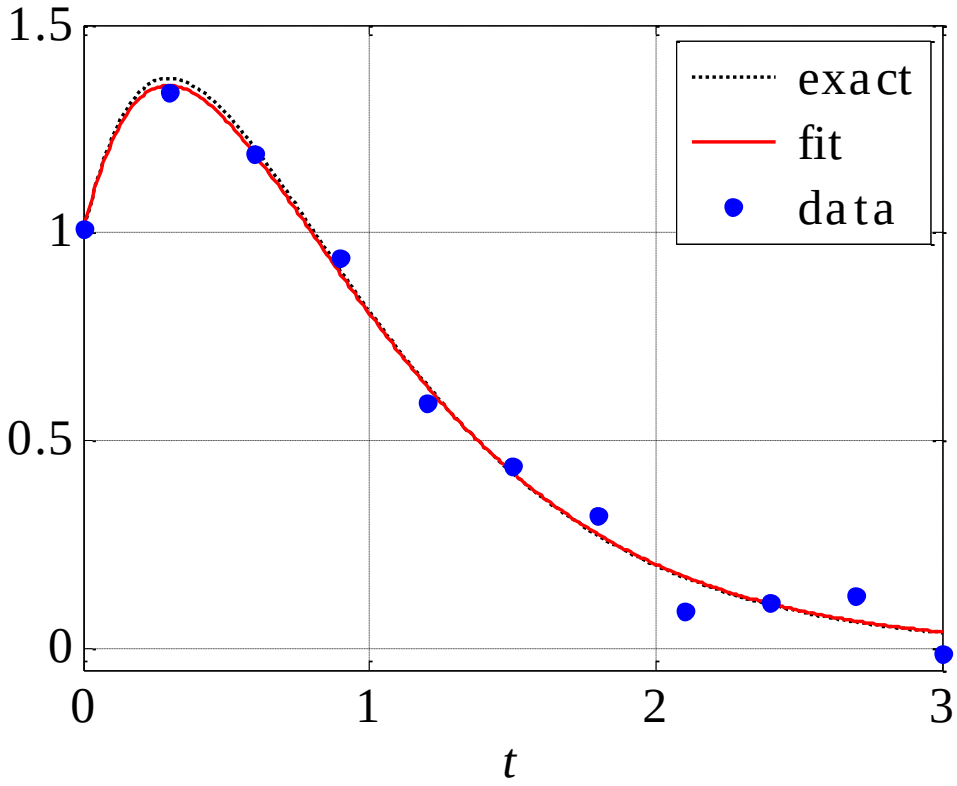
fitted parameters

exact

fitted

data

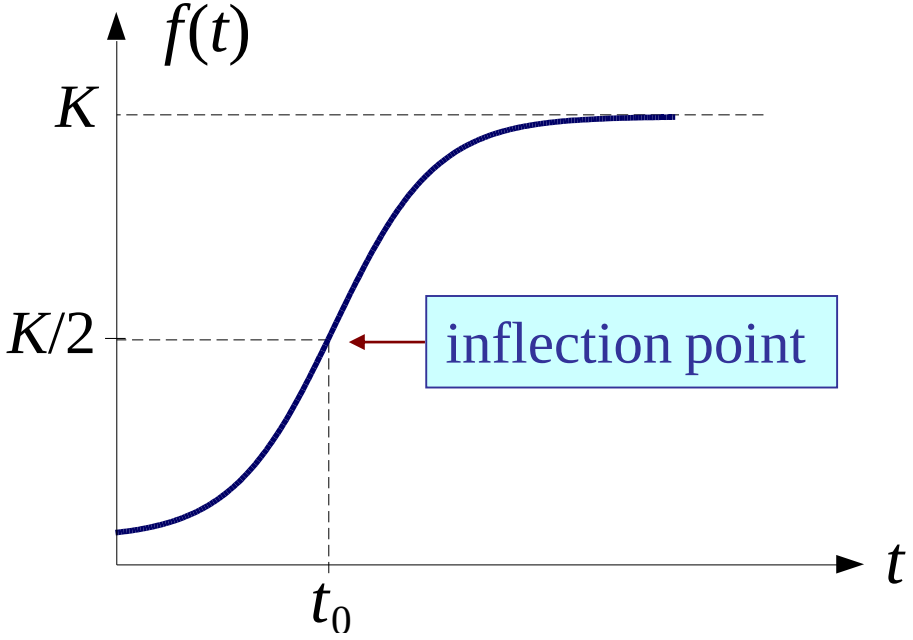
```
[ce,c] =

    5    4.8154
    1    1.0058
    2    1.9770
```

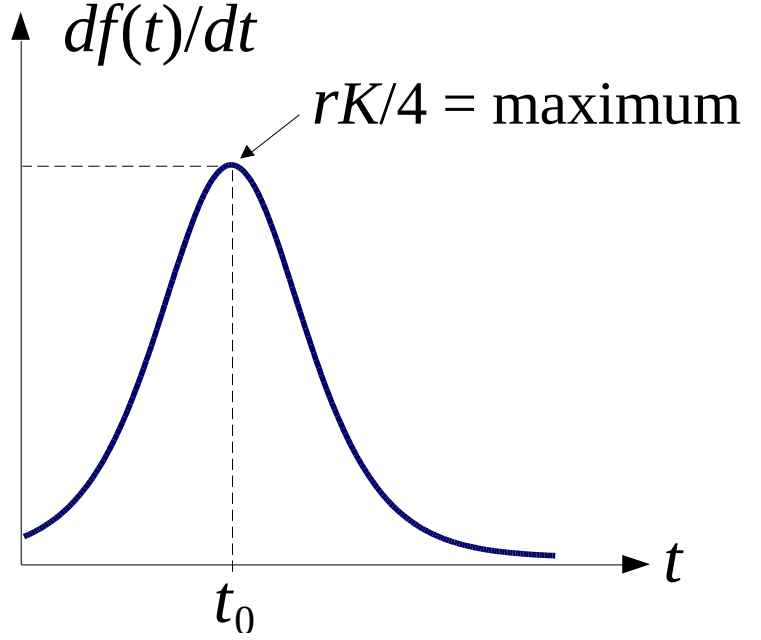


`nlinfit` Example 4: Logistic function

$$f(t) = \frac{K}{1 + e^{-r(t-t_0)}}$$

$$\dot{f}(t) = \frac{df}{dt} = \frac{rKe^{-r(t-t_0)}}{\left[1 + e^{-r(t-t_0)}\right]^2}$$





$$\dot{f}(t_0) = \frac{rK}{4} \quad \Rightarrow \quad r = \frac{4\,\dot{f}(t_0)}{K}$$

define logistic function of the three parameters K, r, t_0

$$f(t) = \frac{K}{1 + e^{-r(t-t_0)}}$$

```
f = @(c,t) c(1)./ (1 + exp(-c(2)*(t-c(3))));

[df0,i0] = max(diff(yi)./diff(ti));
K0 = max(yi);
r0 = 4*df0/K0;
t0 = ti(i0);

c0 = [K0, r0, t0]';

c = nlinfit(ti,yi,f,c0);

K = c(1); r = c(2); t0 = c(3);

% data inluded in file logsim.m
```

initial estimates

initial parameters

fitted

data

ti	yi
1.1	0.04
2.9	1.13
4.5	1.22
7.5	3.87
9.0	5.33
10.5	6.51
12.9	8.37
15.3	8.67
17.4	9.03

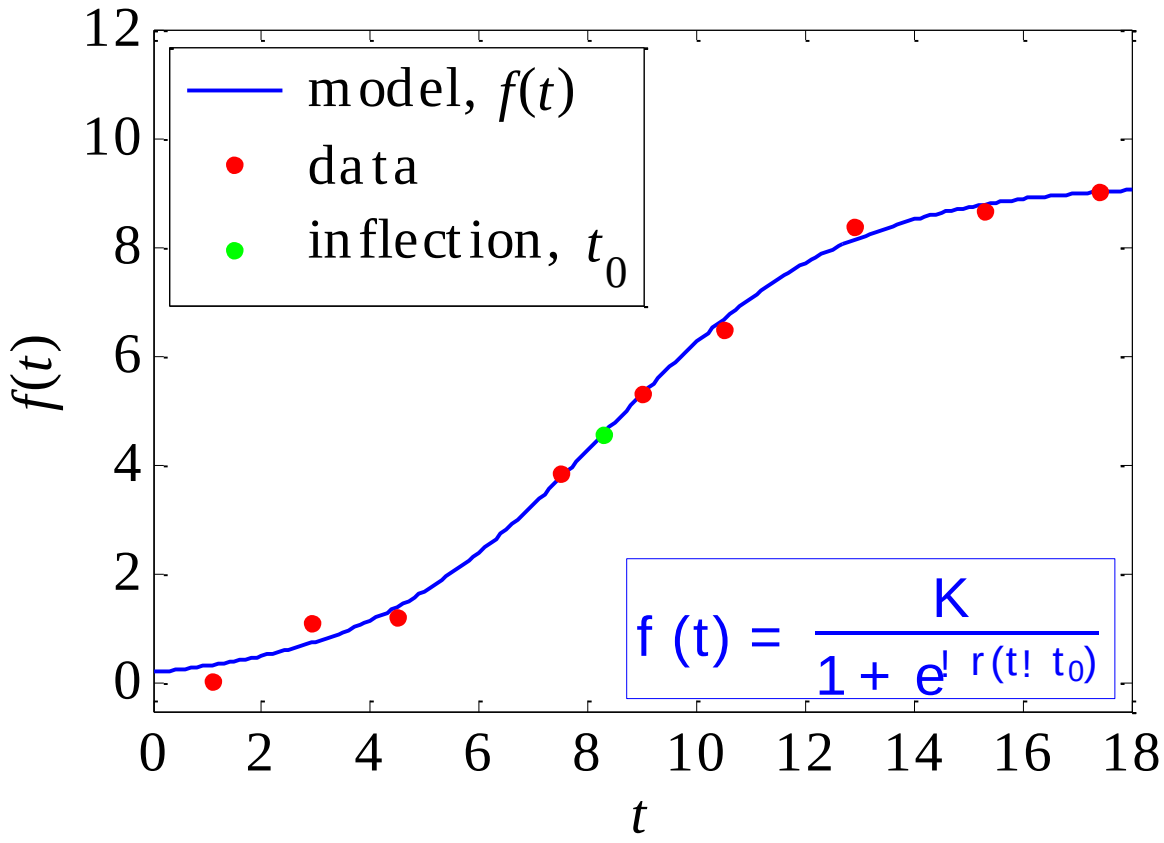
```
t = linspace(0,18,181);
plot(t,f(c,t), 'b-', ti,yi,'r.', t0,K/2,'g.');
```

fitted

data

inflection point

$K = 9.1672,\ r = 0.4496,\ t_0 = 8.2870$



see M-file

logsim.m

on sakai, week-11

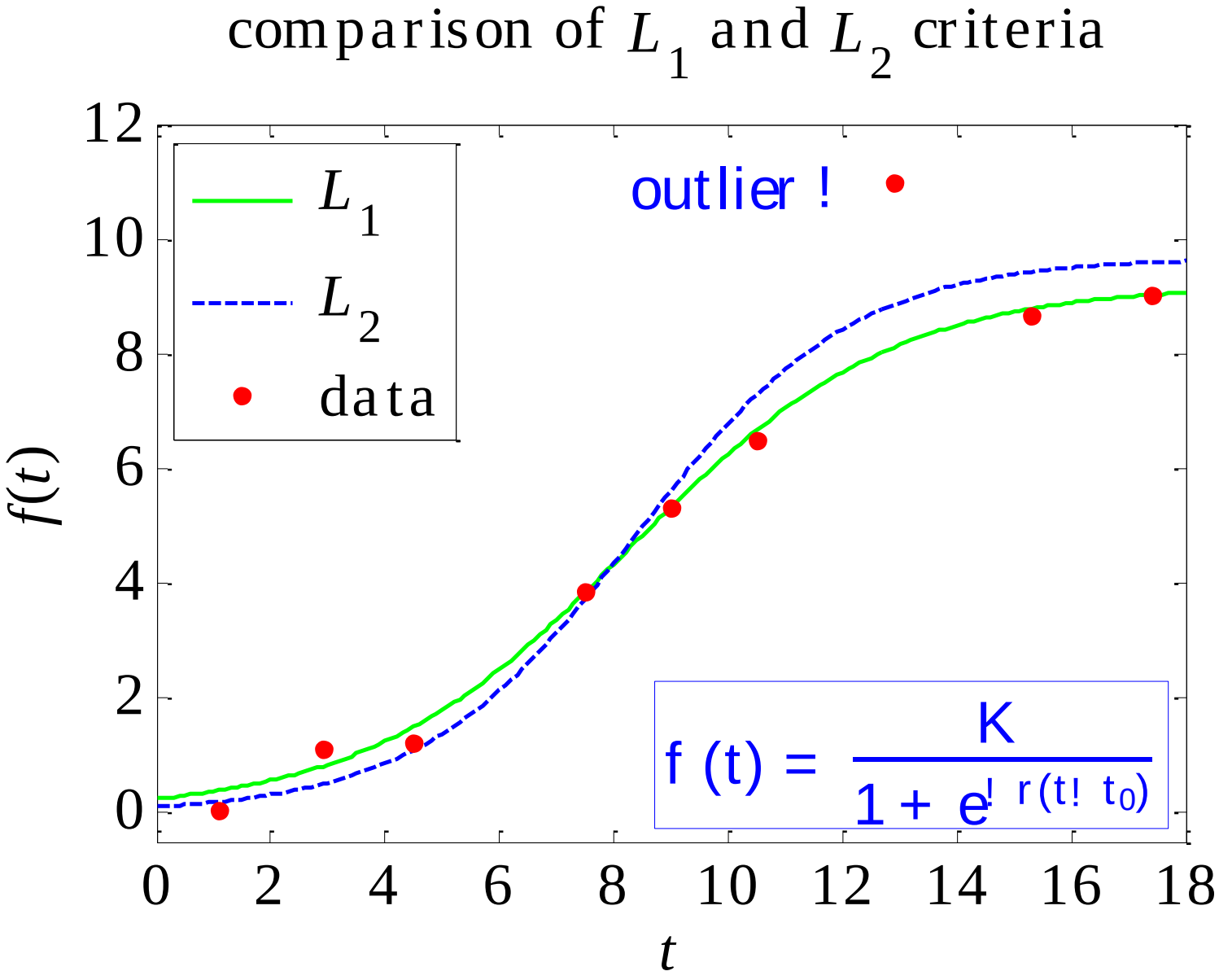
compare the L_1 and L_2 criteria when there are outliers in the data

```
% use the same definitions of
% f(c,t), c0, and t

J = @(c) sum(abs(yi - f(c,ti)));
c1 = fminsearch(J,c0);    % L1

J = @(c) sum((yi - f(c,ti)).^2);
c2 = fminsearch(J,c0);    % L2

plot(t,f(c1,t), 'b-',...
     t,f(c2,t), 'b--',...
     ti,yi,'r.');
```

data

ti	yi
1.1	0.04
2.9	1.13
4.5	1.22
7.5	3.87
9.0	5.33
10.5	6.51
12.9	11.00
15.3	8.67
17.4	9.03

outlier


comparison of L_1 and L_2 criteria
L_1
L_2
data
outlier !
$f(t) = \frac{K}{1 + e^{-r(t - t_0)}}$
$f(t)$
t
0
2
4
6
8
10
12
14
16
18

Variety of Growth Models

Model	Function	Differential Equation
Logistic	$y(t) = \dfrac{K}{1 + e^{-r(t-t_0)}}$	$\dfrac{dy}{dt} = r\,y\left[1 - \dfrac{y}{K}\right]$
Richards	$y(t) = \dfrac{K}{\left[1 + \alpha\, e^{-r(t-t_0)}\right]^{\frac{1}{\alpha}}}$	$\dfrac{dy}{dt} = \dfrac{r}{\alpha}\,y\left[1 - \left(\dfrac{y}{K}\right)^{\alpha}\right]$
Pearl–Reed	$y(t) = \dfrac{K}{1 + A\, e^{-r(t)}}$	$\dfrac{dy}{dt} = \dot{r}(t)\,y\left[1 - \dfrac{y}{K}\right]$
	$r(t) = r_1 t + r_2 t^2 + r_3 t^3$	$\dot{r}(t) = \dfrac{dr}{dt} = r_1 + 2r_2 t + 3r_3 t^2$
Gompertz	$y(t) = K \exp\left[-e^{-r(t-t_0)}\right]$	$\dfrac{dy}{dt} = -r\,y \ln\left(\dfrac{y}{K}\right)$
Bass	$y(t) = p\,K\,\dfrac{1 - e^{-(p+q)t}}{p + q\, e^{-(p+q)t}}$	$\dfrac{dy}{dt} = [p\,K + q\,y]\left[1 - \dfrac{y}{K}\right]$