5. Hafta-2

Eğri uydurma

`polyfit, polyval` **ile en küçük kareler eğri uydurma**
Baz fonksiyonları metodu ile en küçük kareler eğri uydurma
`polyfit` **veya baz fonksiyonları uygulamadan önce datanın dönüştürülmesi**
`nlinfit` **ile en küçük kareler eğri uydurma**
`fminsearch` **ile en küçük kareler eğri uydurma**
İnterpolasyon fonksiyonları, `interp1`, `interp2`

Polinom datası uydurma

polyfit, polyval

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

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

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

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

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

Verilen p katsayıları için x vektöründe P(x)'i hesaplar (polyval)

Verilen p katsayıları için P(x)'in köklerini bulur (roots)

Verilen kökler için p katsayılar vektörünü oluşturur (poly)

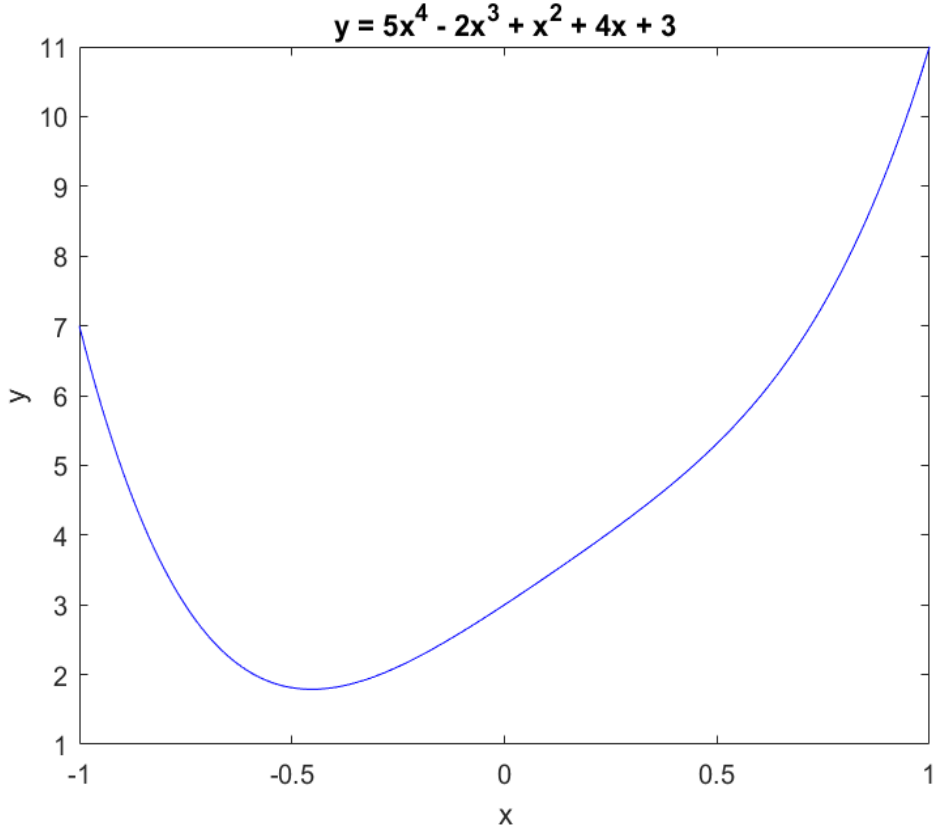
Verilen $\{x_i, y_i\}, i = 1, 2, \ldots, N$ **N data noktasına uyan M. dereceden bir polinom uydurur (polyfit)**

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

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

polyfit, polyval

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



Verilen $\{x_i, y_i\}, i = 1, 2, ..., N$ N data noktasına uyan M. dereceden bir polinom uydurur (polyfit)

```
%Tasarım işlemi:

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

p = polyfit(xi,yi,M)

y = polyval(p,x)
```

M = polinomun derecesi

Eğer N = M+1 ise verilen datalara uyan polinom interpolasyonu yapar

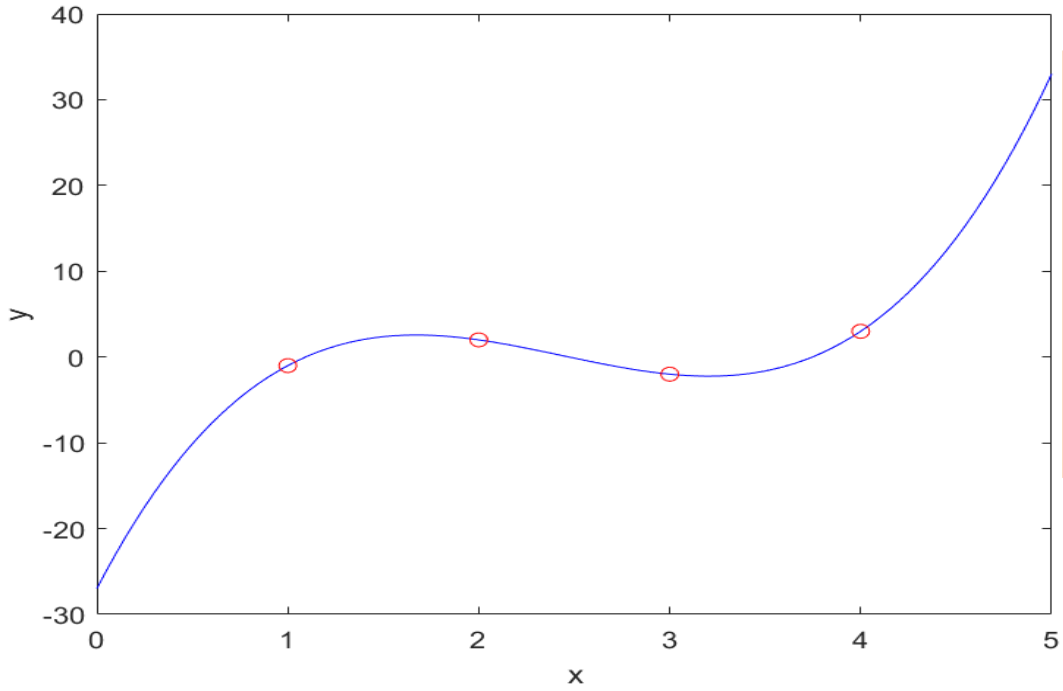
Eğer N > M+1 ise en küçük kareler yöntemi ile en iyi polinom uydurma (best fit) yapar

Verilen x vektörü için P(x)'i hesaplar

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

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');
```



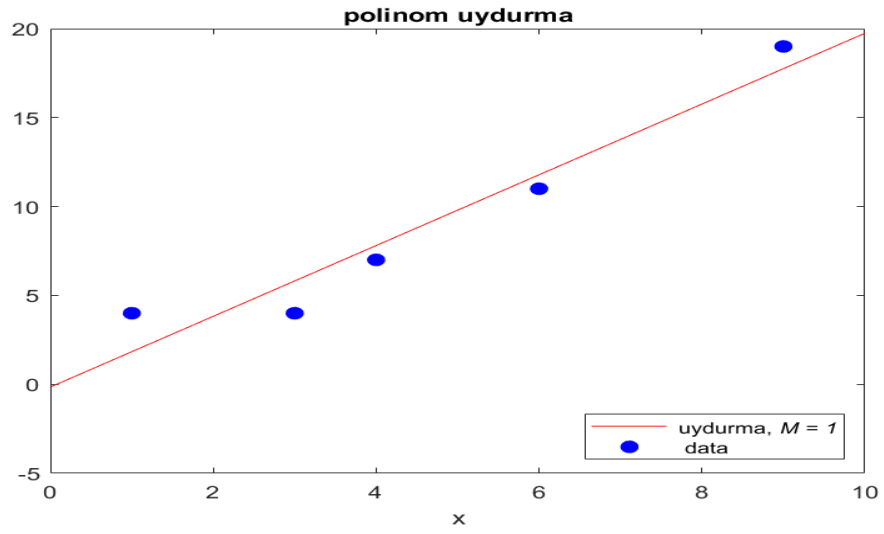
N = data noktası sayısı
M = polinom derecesi

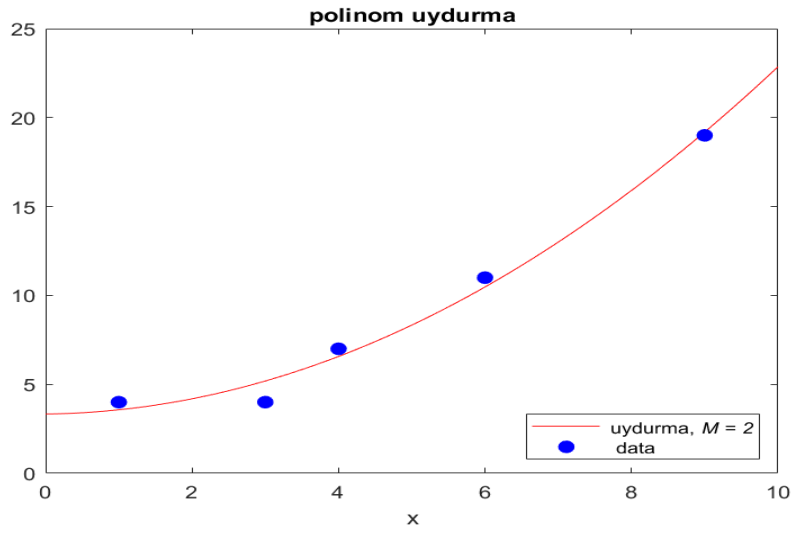
Burada, N = M+1 = 3+1 = 4

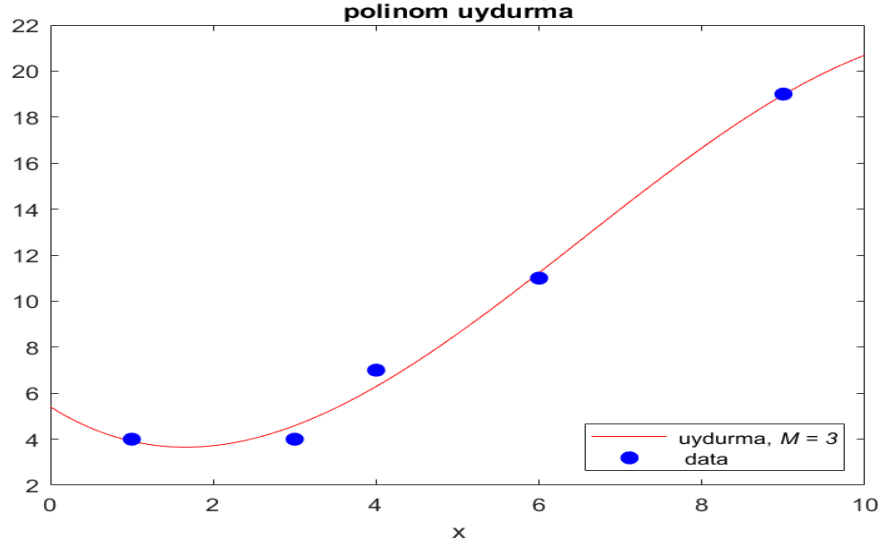
Bu yüzden polinom interpolasyonu yaptı.

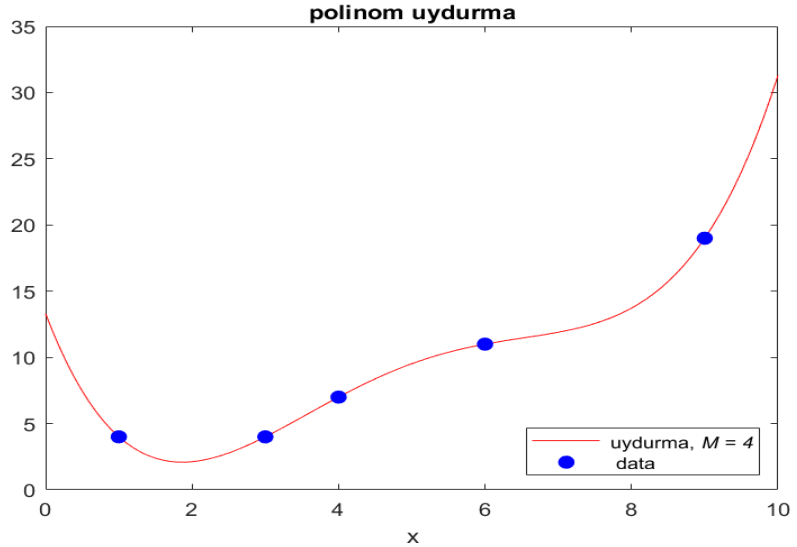
polyfit, polyval

```
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);
    xlabel('x');
    title('polinom uydurma');
    legend([' uydurma, \it{M} = ', num2str(M)],...
           '  data', 'location', 'se');
end
```


polinom uydurma
uydurma, M = 1
data
x

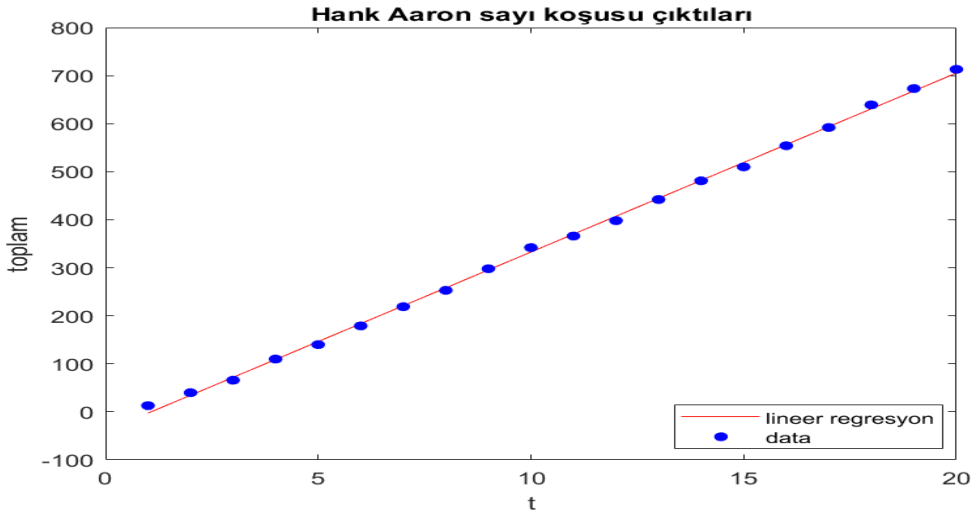

polinom uydurma
uydurma, M = 2
data
x

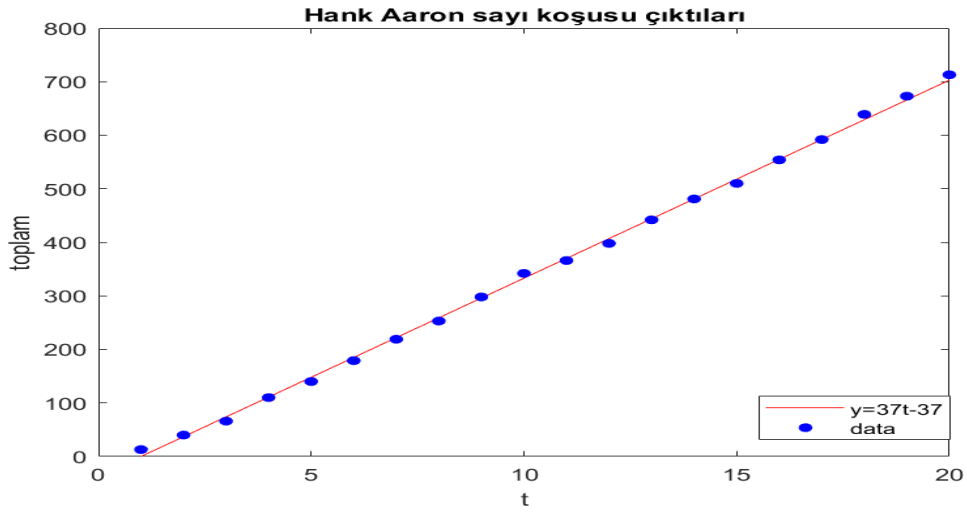

polinom uydurma
uydurma, M = 3
data
x


polinom uydurma
uydurma, M = 4
data
x

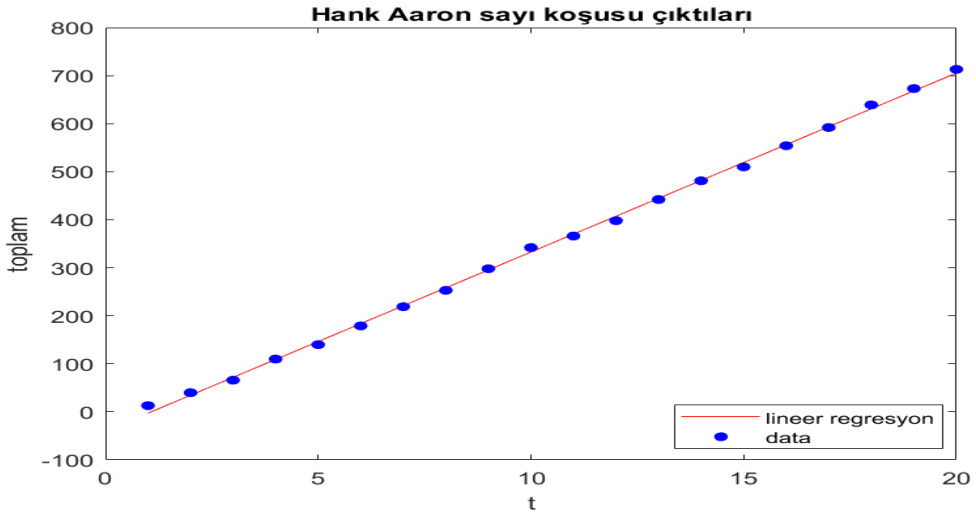
% aaron.dat

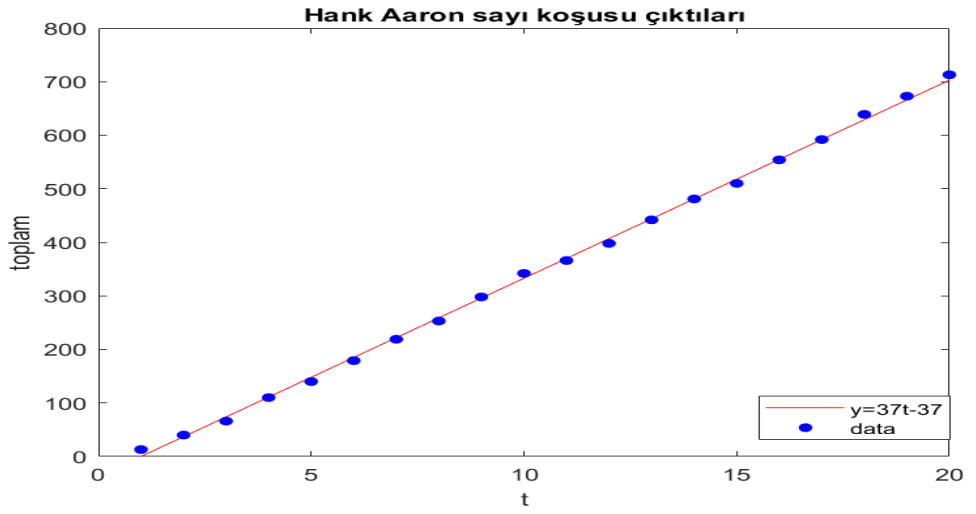
Yıl	ti	H	Yıl	ti	H
1954	1	13	1964	11	24
1955	2	27	1965	12	32
1956	3	26	1966	13	44
1957	4	44	1967	14	39
1958	5	30	1968	15	29
1959	6	39	1969	16	44
1960	7	40	1970	17	38
1961	8	34	1971	18	47
1962	9	45	1972	19	34
1963	10	44	1973	20	40





```
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);
```





Verilen $\{x_i, y_i\}, i = 1, 2, \ldots, N$ **N data noktası için aşağıdaki data modelleri dataların uygun dönüşümleri ile lineer regresyona indirgenebilir.**

lineer: $y = ax + b$

üstel: $y = be^{ax}$ $\log(y) = ax + \log(b)$

üstel: $y = b2^{ax}$ $\log_2(y) = ax + \log_2(b)$

üstel: $y = bxe^{ax}$ $\log(y/x) = ax + \log(b)$

kuvvet: $y = bx^a$ $\log(y) = a\log(x) + \log(b)$

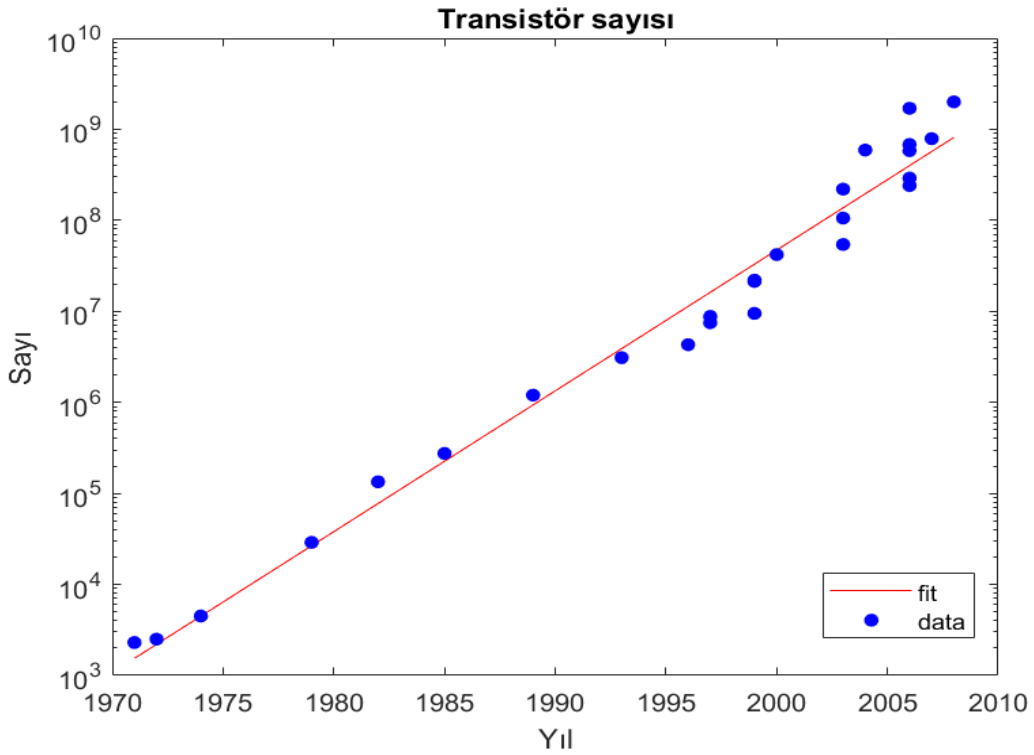
```
p = polyfit(xi,log(yi),1);      % üstel
y = exp(polyval(p,x));          % y = exp(a*x + log(b))
a = p(1);                       % y = exp(p(1)*x + p(2))
b = exp(p(2));                  % y = b*exp(a*x)
```

yi	ti
2,30E+03	1971
2,50E+03	1972
4,50E+03	1974
2,90E+04	1979
1,34E+05	1982
2,75E+05	1985
1,20E+06	1989
3,10E+06	1993
4,30E+06	1996
7,50E+06	1997
8,80E+06	1997
9,50E+06	1999
2,13E+07	1999
2,20E+07	1999
4,20E+07	2000
5,43E+07	2003
1,06E+08	2003
2,20E+08	2003
5,92E+08	2004
2,41E+08	2006
2,91E+08	2006
5,82E+08	2006
6,81E+08	2006
7,89E+08	2007
1,70E+09	2006
2,00E+09	2008

Moore yasası

fitted model: $f(t) = b2^{a(t-t_1)}$

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



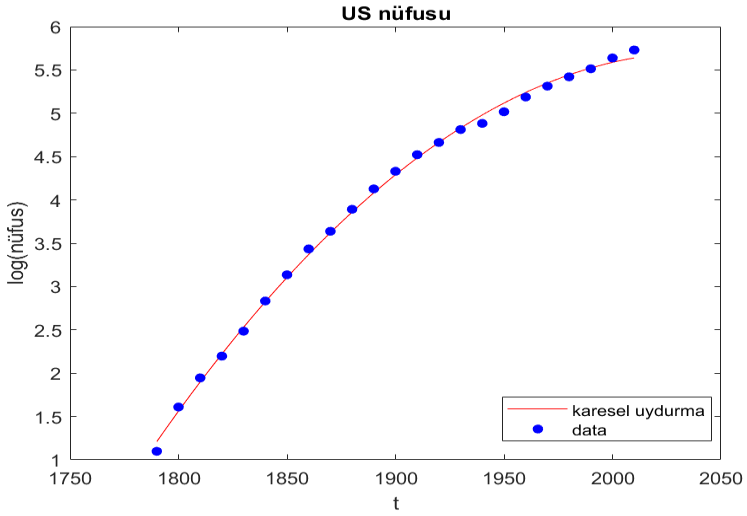
```
Y = load('transistor_count.dat');

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)
```

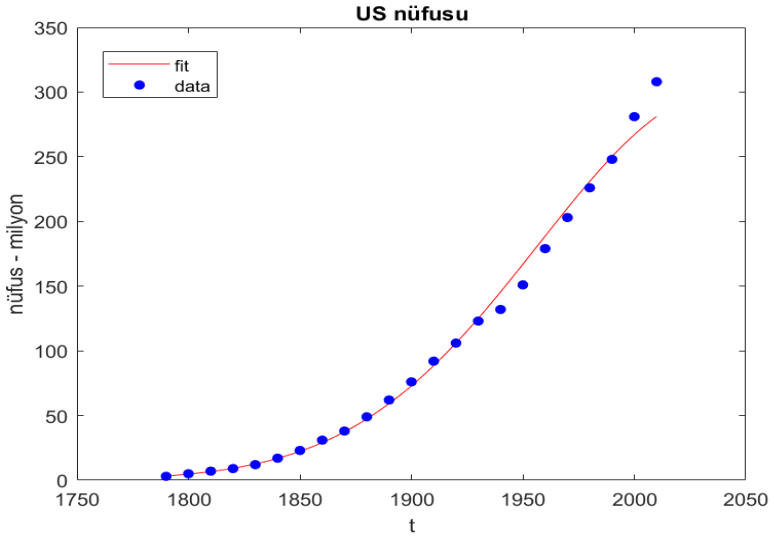
Uydurulmuş model:

$f(t) = b*2.^{(a*(t-t1))} = 2.^{(a*(t-t1) + \log2(b))};$

% a = p(1), log2(b) = p(2) → b = 2^(p(2))

ti	yi
1790	3,929
1800	5,237
1810	7,24
1820	9,638
1830	12,866
1840	17,069
1850	23,192
1860	31,443
1870	38,558
1880	49,371
1890	62,98
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	248,71
2000	281,422
2010	308,746





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

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

p = polyfit(ti,log(yi),2)   % karesel uydurma

% 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);
```

Gezegensel sıcaklık

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

T_e = **Dünya sıcaklığı**

r = **Güneşten uzaklık**

Dönüştürülen model

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

Varsayılan model

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

gezegen	ri	Ti	T_est
Merkur	0.390	452	468
Dunya	1.000	285	283
Mars	1.520	230	227
Jupiter	5.200	120	118
Saturn	9.539	88	85
Uranus	19.180	59	59
Neptun	30.060	48	46
Pluton	39.530	37	40

```
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]';  % sütunlar
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. ',...
            'markersize', 18);
```

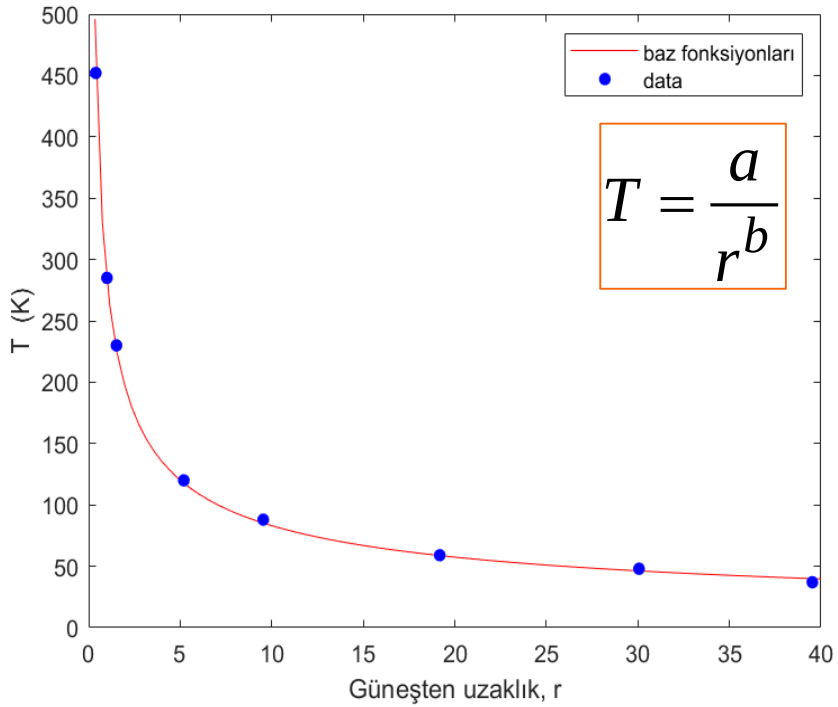
Baz fonksiyonları metodu:

```
c = [r,.^0, -log(ri)] \ log(Ti)
c(1) = exp(c(1));
```

Tahmin:

```
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 versiyonu
```



Eğri uydurma araç kutusu daha karmaşık doğrusal olmayan datalara uygun eğriler uydurmaya izin verir. Ayrıca istatistik ve optimizasyon araç kutularına da bakın.

```
>> doc curvefit      % eğri uydurma araç kutusu
>> doc nlinfit       % istatistik araç kutusunda
>> doc lsqcurvefit   % optimizasyon araç kutusunda
```

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

% c = tahmini parametre vektörü
% xi,yi = data noktaları vektörü
% f = uydurma fonksiyonu
% c0 = başlangıç parametre vektörü

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

→ y=f(c,x)

c ilk olmalı

vektörleştirilmiş olmalı

Eşdeğer olarak en küçük kareler versiyonu fminsearch hazır fonksiyonunu kullanır. En küçük kareler hata kriterini minimize eder:

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

% bir wi ağırlık vektörü içerebilir
% J = @(c) sum(wi.*(yi-f(c,xi)).^2)

% datalar aykırı değerler varsa daha uygun olan
% L1-norm kriteri kullanılabilir

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

L_2 **kriteri:** $J(c) = \sum_{i=1}^{N} |y_i - f(c, x_i)|^2 = \min$

L_1 **kriteri:** $J(c) = \sum_{i=1}^{N} |y_i - f(c, x_i)| = \min$

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

% J(c)'yi tanımlamak için norm(x), norm(x,1)
% hazır fonksiyonlarını kullanabiliriz:

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

Kullanışlı MATLAB hazır fonksiyonları

```
>> doc curvefit                 % eğri uydurma araç kutusu

% bunlara da bakın

>> doc nlinfit                  % istatistik araç kutusunda

>> doc lsqcurvefit              % optimizasyon araç kutusunda
>> doc lsqnonlin                % doğrusal olmayan en küçük kareler
>> doc lsqlin                   % kısıtlamalı doğrusal
>> doc linprog                  % lineer programlama
>> doc lsqnonneg                % kısıtlamaların pozitifliği
>> doc quadprog                 % karesel programlama
>> doc bintprog                 % ikili int programlama
```

Örnek 1:

$$f(c,t) = c_1 + c_2 \cos(c_3 t) e^{-c_4 t}$$

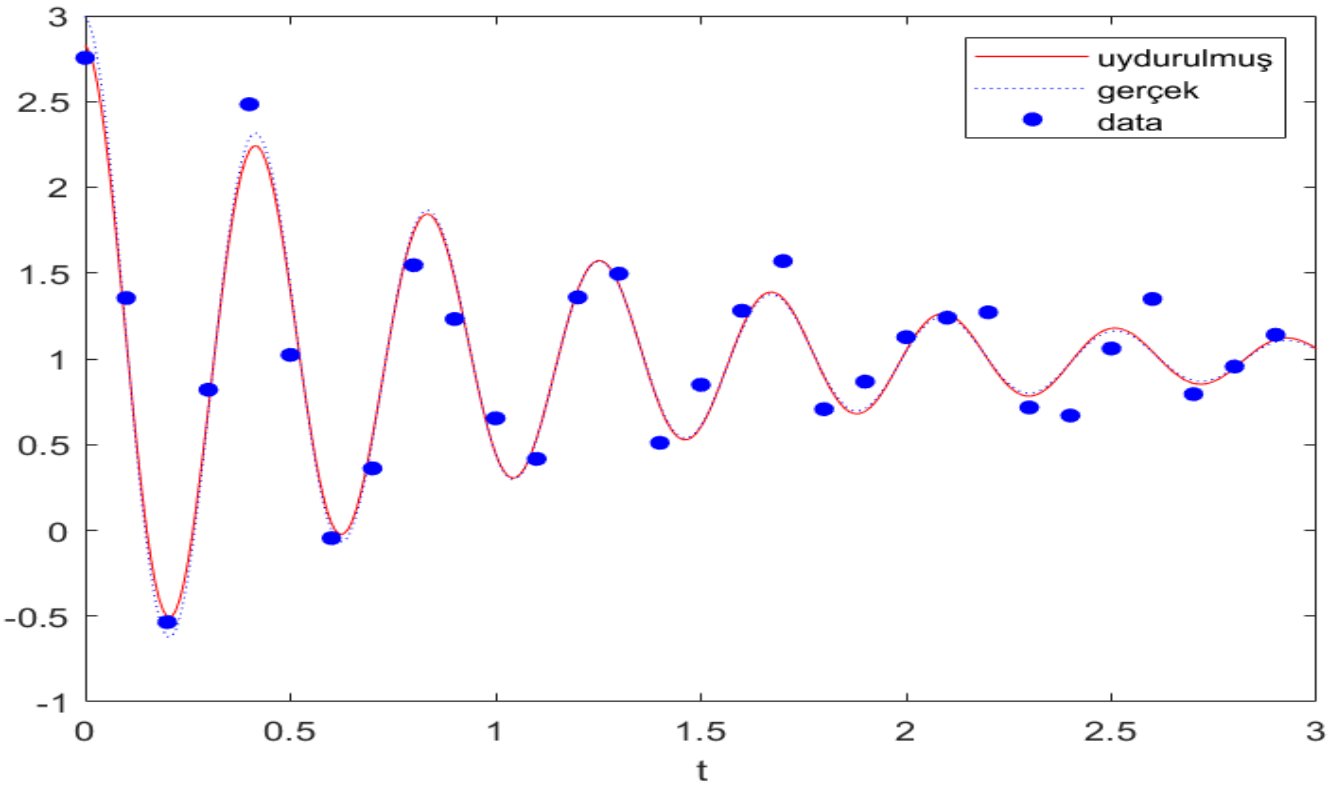
Parametre vektörü c = [c1,c2,c3,c4]'

```
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]' = sütun

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);
```

```
[c,ce] =
    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.')
```

uydurulmuş **gerçek** **data**



Örnek 2:

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

Varsayılan model

```
% temsili değerler üretir

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]';

% nlinfit model fonksiyonunu tanımlar
f = @(c,t) (c(1)*t+c(2)).*exp(-c(3)*t);
```

gerçek değerler

gerçek parametreler

başlangıç parametreleri

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)
```

uydurulmuş parametreler

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

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

gerçek **uydurulmuş** **data**

```
[ce,c] =

    5     4.8154
    1     1.0058
    2     1.9770
```