TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
ELEKTRİK VE ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELE 375 – ELEKTRİK MÜHENDİSLERİ İÇİN SAYISAL YÖNTEMLER DERSİ

YAZ 2011

ÖDEV 3

Seçkin Burak CENGİZ

Soru 1: Tabloda verilen yüksekliğe bağlı hava yoğunluğu değişim bilgilerini ve kübik interpolasyon yöntemini kullanarak h=5km ve 8km için hava yoğunluklarını hesaplayalım.

Kübik interpolasyon formülleri

$$f''_{i-1,i}(x_i) = f''_{i,i+1}(x_i) = k_i$$

$$k_1 = k_n = 0$$

$$f''_{i,i+1}(x) = k_i \ell_i(x) + k_{i+1} \ell_{i+1}(x)$$

$$f''_{i,i+1}(x) = k_i \ell_i(x) + k_{i+1} \ell_{i+1}(x)$$

$$\ell_i(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} \qquad \ell_{i+1}(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

$$f''_{i,i+1}(x) = \frac{k_i(x - x_{i+1}) - k_{i+1}(x - x_i)}{x_i - x_{i+1}}$$

$$f_{i,i+1}(x) = \frac{k_i(x - x_{i+1})^3 - k_{i+1}(x - x_i)^3}{6(x_i - x_{i+1})} + A(x - x_{i+1}) - B(x - x_i)$$

$$\frac{k_i(x_i - x_{i+1})^3}{6(x_i - x_{i+1})} + A(x_i - x_{i+1}) = y_i$$

$$A = \frac{y_i}{x_i - x_{i+1}} - \frac{k_i}{6}(x_i - x_{i+1}) \quad B = \frac{y_{i+1}}{x_i - x_{i+1}} - \frac{k_{i+1}}{6}(x_i - x_{i+1})$$

$$\begin{aligned} f_{i,i+1}(x) = {} & \frac{k_i}{6}\left[\frac{(x - x_{i+1})^3}{x_i - x_{i+1}} - (x - x_{i+1})(x_i - x_{i+1})\right] \\ & - \frac{k_{i+1}}{6}\left[\frac{(x - x_i)^3}{x_i - x_{i+1}} - (x - x_i)(x_i - x_{i+1})\right] \\ & + \frac{y_i(x - x_{i+1}) - y_{i+1}(x - x_i)}{x_i - x_{i+1}} \end{aligned}$$

$$\begin{aligned} & k_{i-1}(x_{i-1} - x_i) + 2k_i(x_{i-1} - x_{i+1}) + k_{i+1}(x_i - x_{i+1}) \\ & = 6\left(\frac{y_{i-1} - y_i}{x_{i-1} - x_i} - \frac{y_i - y_{i+1}}{x_i - x_{i+1}}\right), \quad i = 2, 3, \ldots, n-1 \end{aligned}$$

$$k_{i-1} + 4k_i + k_{i+1} = \frac{6}{h^2}(y_{i-1} - 2y_i + y_{i+1}), \quad i = 2, 3, \ldots, n-1$$

$$k_{i-1}(x_{i-1}-x_i)+2k_i(x_{i-1}-x_{i+1})+k_{i+1}(x_i-x_{i+1})$$
$$=6\left(\frac{y_{i-1}-y_i}{x_{i-1}-x_i}-\frac{y_i-y_{i+1}}{x_i-x_{i+1}}\right),\quad i=2,3,\ldots,n-1$$

Eğer datalar eşit aralıklarla bölünmüş ise yukardaki formül aşağıdaki gibi yazılabilir.

$$k_{i-1}+4k_i+k_{i+1}=\frac{6}{h^2}(y_{i-1}-2y_i+y_{i+1}),\quad i=2,3,\ldots,n-1$$

Ludec3 fonksiyonu

```
function [c,d,e] = LUdec3(c,d,e)
% LU decomposition of tridiagonal matrix A = [c\d\e].
% USAGE: [c,d,e] = LUdec3(c,d,e)
n = length(d);
for k = 2:n
lambda = c(k-1)/d(k-1);
d(k) = d(k) - lambda*e(k-1);
c(k-1) = lambda;
end
```

Lusol3 fonksiyonu

```
function x = LUsol3(c,d,e,b)
% Solves A*x = b where A = [c\d\e] is the LU
% decomposition of the original tridiagonal A.
% USAGE: x = LUsol3(c,d,e,b)
n = length(d);
for k = 2:n % Forward substitution
b(k) = b(k) - c(k-1)*b(k-1);
end
b(n) = b(n)/d(n); % Back substitution
for k = n-1:-1:1
b(k) = (b(k) -e(k)*b(k+1))/d(k);
end
x = b;
```

$$k_1 = k_n = 0$$

splineCurv fonksiyonu

```
function k = splineCurv(xData,yData)
% Returns curvatures of a cubic spline at the knots.
% USAGE: k = splineCurv(xData,yData)
% xData = x-coordinates of data points.
% yData = y-coordinates of data points.
n = length(xData);
c = zeros(n-1,1); d = ones(n,1);
e = zeros(n-1,1); k = zeros(n,1);
c(1:n-2) = xData(1:n-2) - xData(2:n-1);
d(2:n-1) = 2*(xData(1:n-2) - xData(3:n));
e(2:n-1) = xData(2:n-1) - xData(3:n);
k(2:n-1) = 6*(yData(1:n-2) - yData(2:n-1))./(xData(1:n-2) - xData(2:n-1))- 6*(yData(2:n-1) -
yData(3:n))./(xData(2:n-1) - xData(3:n));
[c,d,e] = LUdec3(c,d,e);
k = LUsol3(c,d,e,k);
```

$$f_{i,i+1}(x) = \frac{k_i}{6}\left[\frac{(x-x_{i+1})^3}{x_i - x_{i+1}} - (x-x_{i+1})(x_i - x_{i+1})\right]$$
$$- \frac{k_{i+1}}{6}\left[\frac{(x-x_i)^3}{x_i - x_{i+1}} - (x-x_i)(x_i - x_{i+1})\right]$$
$$+ \frac{y_i(x-x_{i+1}) - y_{i+1}(x-x_i)}{x_i - x_{i+1}}$$

splineEval fonksiyonu

```
function y = splineEval(xData,yData,k,x)
% Returns value of cubic spline interpolant at x.
% USAGE: y = splineEval(xData,yData,k,x)
% xData = x-coordinates of data points.
% yData = y-coordinates of data points.
% k = curvatures of spline at the knots;
% returned by function splineCurv.
i = findSeg(xData,x);
h = xData(i) - xData(i+1);
y = ((x - xData(i+1))^3/h - (x - xData(i+1))*h)*k(i)/6.0- ((x - xData(i))^3/h - (x -
xData(i))*h)*k(i+1)/6.0+ yData(i)*(x - xData(i+1))/h- yData(i+1)*(x - xData(i))/h;
function i = findSeg(xData,x)
% Returns index of segment containing x.
iLeft = 1; iRight = length(xData);
while 1
if(iRight - iLeft) <= 1
i = iLeft; return
end
i = fix((iLeft + iRight)/2);
if x < xData(i)
iRight = i;
else
iLeft = i;
end
end
```

Yukarda tanımlanan fonksiyonlar aşağıdaki Script dosyası ile çağırılarak burada girilen data değerlerine karşılık kübik interpolasyon uygulanır.

Solve scripti

```
%Soru 1(Cubic spline)
xData = [0; 1.525; 3.050; 4.575; 6.10; 7.625; 9.150];
yData = [1; 0.8617; 0.7385; 0.6292; 0.5328; 0.4481; 0.3741];
k = splineCurv(xData,yData);

disp('yükseklik değerini aşağıda giriniz.')
 h = input('h =  ');

if isempty(h)
disp('yükseklik değerini aşağıda giriniz.')
 h= input('h =  ');
else
   disp('Girmiş olduğunuz yükseklik değeri için hava yoğunluğu aşağıdaki gibidir.')
   y = splineEval(xData,yData,k,h);
   disp('y =')
   disp(y)

end
```

Solve scripti çalıştırıldıktan sonra oluşan çıktı aşağıdaki gibidir.

```
>> solve
yükseklik değerini aşağıda giriniz.
h =  5
Girmiş olduğunuz yükseklik değeri için hava yoğunluğu aşağıdaki gibidir.
y =
   0.6011

>> solve
yükseklik değerini aşağıda giriniz.
h =  8
Girmiş olduğunuz yükseklik değeri için hava yoğunluğu aşağıdaki gibidir.
y =
   0.4292
```

h (yükseklik)	y (hava yoğunluğu)
5	0.6011
8	0.4292

Soru2: Tablodaki verilere doğrusal ve ikinci dereceden grafik uydurma yapalım.

Doğrusal yakınsama(Fitting a Straight Line):

Verilen datalara yakınsayacağımız polinom aşağıdaki genel form gibi doğrusal olacaktır.

$$f(x) = a + bx$$

Yukardaki fonksiyon aşağıdaki gibi, farkların karesi şeklinde ifade edilebilir.

$$S(a,b) = \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

$$\frac{\partial S}{\partial a} = \sum_{i=1}^{n} -2(y_i - a - bx_i) = 2\left(-\sum_{i=1}^{n} y_i + na + b\sum_{i=1}^{n} x_i\right) = 0$$

$$\frac{\partial S}{\partial b} = \sum_{i=1}^{n} -2(y_i - a - bx_i)x_i = 2\left(-\sum_{i=1}^{n} x_i y_i + a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2\right) = 0$$

Her iki eşitlikte 2n ile bölünürse;

$$a + \bar{x}b = \bar{y} \qquad a\bar{x} + \left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) b = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$$

Ortalama değerler aşağıdaki gibi hesaplanır;

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Sonuç olarak çözüm aşağıdaki gibidir.

$$a = \frac{\bar{y}\sum x_i^2 - \bar{x}\sum x_i y_i}{\sum x_i^2 - n\bar{x}^2} \qquad b = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2}$$

En basit haliyle a ve b katsayılarının hesaplanması için kullanılan formül aşağıdaki gibidir;

$$b = \frac{\sum y_i(x_i - \bar{x})}{\sum x_i(x_i - \bar{x})} \qquad a = \bar{y} - \bar{x}b$$

Standart sapma: Verilen eğri etrafındaki saçılımı belirlemektedir. n=m ise interpolasyon demektir. Bu durumda standart sapma tanımlamak anlamsızdır.

$$\sigma = \sqrt{\frac{S}{n-m}}$$

m-1. dereceden polinom ile yakınsama (Polynomial Fit):

m-1. dereceden bir polinom için genel form aşağıdaki gibidir.

$$f(x) = \sum_{j=1}^{m} a_j x^{j-1}$$

$$f_j(x) = x^{j-1}, \quad j = 1, 2, \ldots, m$$

Lineer yakınsama formu aşağıdaki gibi değişir.

$$A_{kj} = \sum_{i=1}^{n} x_i^{j+k-2} \qquad b_k = \sum_{i=1}^{n} x_i^{k-1} y_i$$

$$\mathbf{A} = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{m+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_i^{m-1} & \sum x_i^m & \sum x_i^{m+1} & \cdots & \sum x_i^{2m-2} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^{m-1} y_i \end{bmatrix}$$

gaussPiv fonksiyonu

```
function x = gaussPiv(A,b)
% Solves A*x = b by Gauss elimination with row pivoting.
% USAGE: x = gaussPiv(A,b)
if size(b,2) > 1; b = b'; end
n = length(b); s = zeros(n,1);
%----------Set up scale factor array----------
for i = 1:n; s(i) = max(abs(A(i,1:n))); end
%---------Exchange rows if necessary----------
for k = 1:n-1
[Amax,p] = max(abs(A(k:n,k))./s(k:n));
p = p + k - 1;
if Amax < eps; error('Matrix is singular'); end
if p ~= k
b = swapRows(b,k,p);
s = swapRows(s,k,p);
A = swapRows(A,k,p);
end
%--------------Elimination pass---------------
for i = k+1:n
if A(i,k) ~= 0
lambda = A(i,k)/A(k,k);
A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
b(i) = b(i) - lambda*b(k);
end
end
end
%------------Back substitution phase----------
for k = n:-1:1
b(k) = (b(k) - A(k,k+1:n)*b(k+1:n))/A(k,k);
end
x = b;
```

SwapRov fonksiyonu

```
function v = swapRows(v,i,j)
% Swap rows i and j of vector or matrix v.
% USAGE: v = swapRows(v,i,j)
temp = v(i,:);
v(i,:) = v(j,:);
v(j,:) = temp;
```

polyFit fonksiyonu

```
function coeff = polynFit(xData,yData,m)
% Returns the coefficients of the polynomial
% a(1)*x^(m-1) + a(2)*x^(m-2) + ... + a(m)
% that fits the data points in the least squares sense.
% USAGE: coeff = polynFit(xData,yData,m)
% xData = x-coordinates of data points.
% yData = y-coordinates of data points.

A = zeros(m); b = zeros(m,1); s = zeros(2*m-1,1);
for i = 1:length(xData)
temp = yData(i);
for j = 1:m
b(j) = b(j) + temp;
temp = temp*xData(i);
end
temp = 1;
for j = 1:2*m-1
s(j) = s(j) + temp;
temp = temp*xData(i);
end
end
for i = 1:m
for j = 1:m
A(i,j) = s(i+j-1);
end
end
% Rearrange coefficients so that coefficient
% of x^(m-1) is first
coeff = flipdim(gaussPiv(A,b),1);
```

sigma fonksiyonu

```
function sigma = stdDev(coeff,xData,yData)
% Returns the standard deviation between data
% points and the polynomial
% a(1)*x^(m-1) + a(2)*x^(m-2) + ... + a(m)
% USAGE: sigma = stdDev(coeff,xData,yData)
% coeff = coefficients of the polynomial.
% xData = x-coordinates of data points.
% yData = y-coordinates of data points.
m = length(coeff); n = length(xData);
sigma = 0;
for i =1:n
y = polyEval(coeff,xData(i));
sigma = sigma + (yData(i) - y)^2;
end
sigma =sqrt(sigma/(n - m));
function y = polyEval(coeff,x)
% Returns the value of the polynomial at x.
m = length(coeff);
y = coeff(1);
for j = 1:m-1
y = y*x + coeff(j+1);
end
```

Yukardaki fonksiyonlar grafik uydurma (Curve Fitting) formüllerini içerirler. Bu fonksiyonları bir script dosyası içine yazacağımız kodlarla çağırarak soruda verilen datalara uygun polinomun katsayılarını dolayısıyla polinomu buldurabiliriz. Hatırlanacağı gibi doğru yakınsaması için yakınsama yapacağımız polinom birinci dereceden bir polinomdur yani bir doğru belirtir. Bu doğru aşağıdaki formdadır.

$$f(x) = a + bx$$

Birinci dereceden polinom yerine m-1 dereceli bir polinom ile yakınsama yapılırsa bulunacak polinomun formu aşağıdaki gibidir.

$$f(x) = \sum_{j=1}^{m} a_j x^{j-1}$$

$$f_j(x) = x^{j-1}, \quad j = 1, 2, \ldots, m$$

Aşağıdaki script dosyası yukarıda tanımladığımız fonksiyonları çağıracak ve verilen datalara uygun polinomlar hesaplayacaktır.

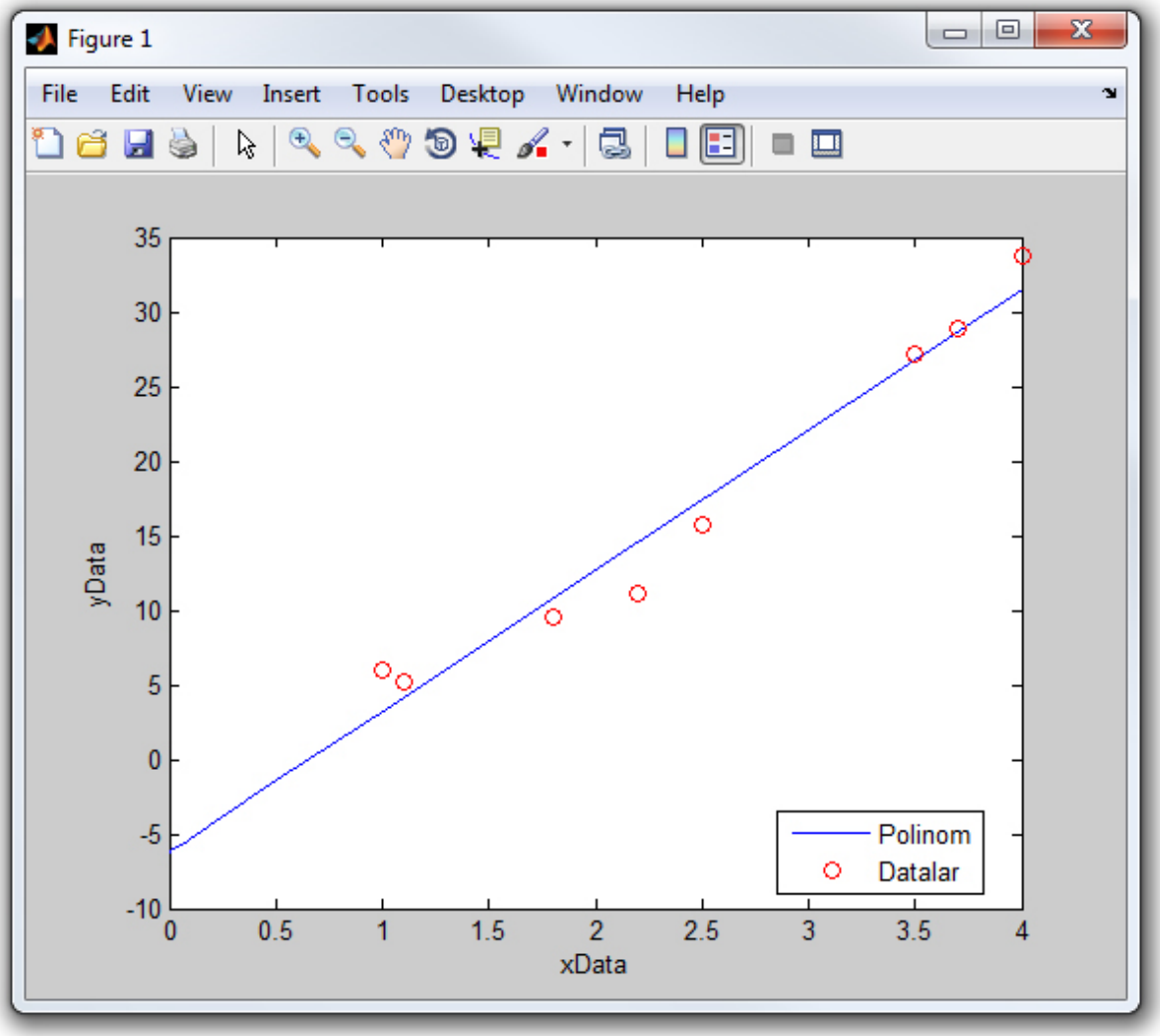
Solve sctipti

```
% Soru2 Curve Fitting
xData = [1; 2.5; 3.5; 4; 1.1; 1.8; 2.2; 3.7]';
yData = [6.008; 15.722; 27.130; 33.772; 5.257; 9.549; 11.098; 28.828]';
format short e

k = input('Yakınsama yapılacak polinomun derecesi = ');
if isempty(k) % Loop is terminated
k = input('Yakınsama yapılacak polinomun derecesi = ');
elseif k==1
      disp('Doğrusal yaklaşım seçilmiştir.');
      d=inline('9.4385e+000*xData-6.1899e+000','xData');
elseif k==2
      disp('2. dereceden polinom yaklaşımı seçilmiştir.');
      d=inline('2.1081e+000*xData^2+(-1.0689e+000)*xData+4.4057e+000','xData');
elseif k>2
      disp('sadece 1 ve 2. dereceden polinom yakınsamaları için grafik kodları vardır')
end

coeff = polynFit(xData,yData,k+1);
sigma = stdDev(coeff,xData,yData);

fplot(d, [0,4]);
hold on
plot(xData,yData,'ro')
legend('Polinom','Datalar',0)
xlabel('xData')
ylabel('yData')
hold off
disp('Katsayılar aşağıdaki gibidir: ')
disp(coeff)
disp('Standart sapma = ')
disp(sigma)
```

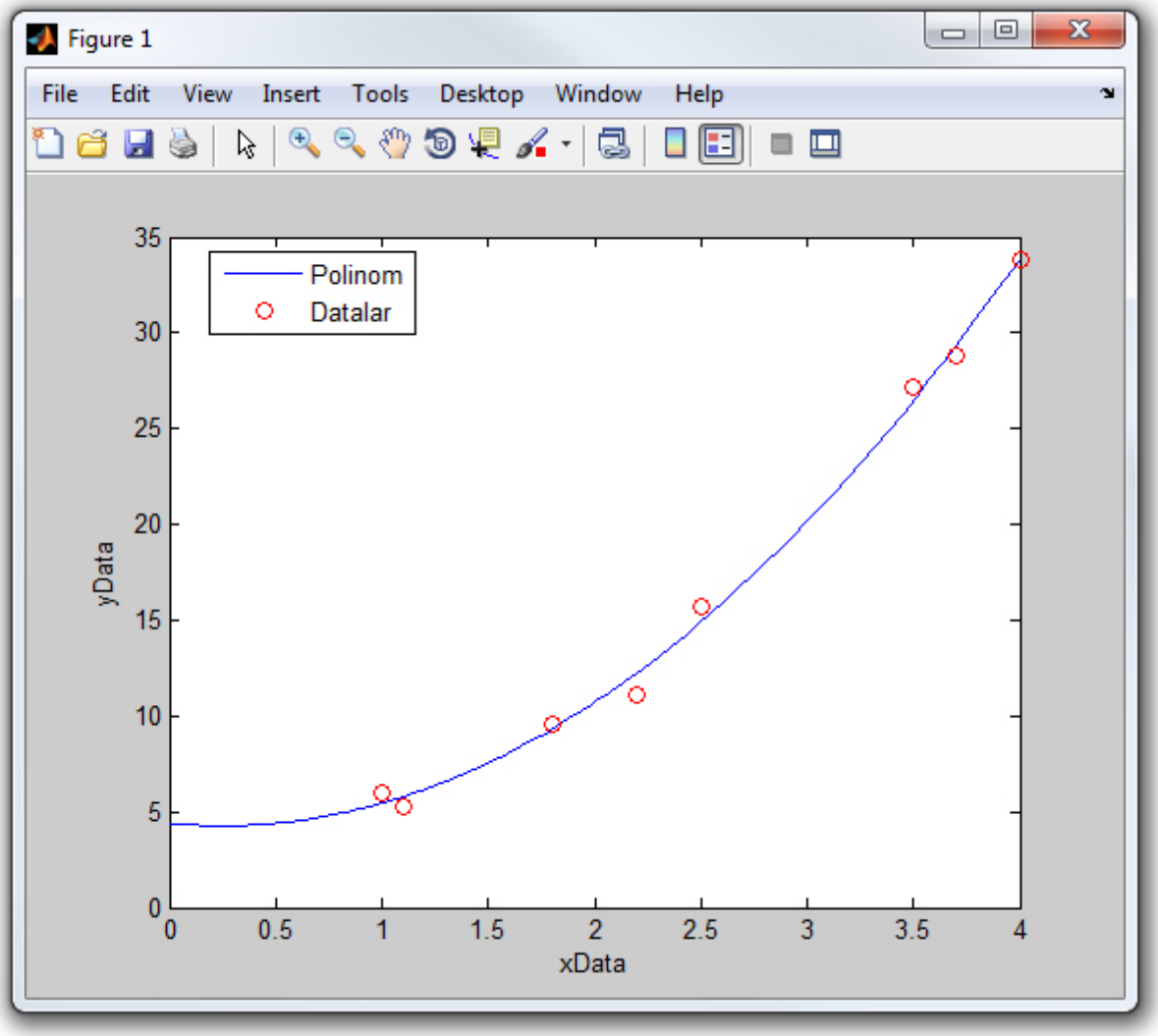
Yazdığımız fonksiyonlar sağyesinde verilen datalar ve yakınsamak istediğimiz polinomun derecesi ne olursa olsun hesaplama yapmak mümkündür. Solve scripti amaca yönelik programlanarak bu fonksiyonlar çağırılıp hesaplamalar yaptırılabilir, buda bize daha esnek bir program sağlamış olur.

Solve scripti çağrıldıktan sonra oluşan çıktı aşağıdaki gibidir.

```
>> solve
Yakınsama yapılacak polinomun derecesi = 1
Doğrusal yaklaşım seçilmiştir.
Katsayılar aşağıdaki gibidir:
  9.4385e+000
 -6.1899e+000

Standart sapma =
  2.2436e+000

fx >>
```



```
>> solve
Yakınsama yapılacak polinomun derecesi = 2
2. dereceden polinom yaklaşımı seçilmiştir.
Katsayılar aşağıdaki gibidir:
  2.1081e+000
 -1.0689e+000
  4.4057e+000

Standart sapma =
  8.1293e-001

fx >>
```



SONUÇ:

Birinci soruda kübik interpolasyon (Cubic Interpolation) yapılmıştır. Yazılan kodlar sağyesinde verilen değişik datalara göre hesaplama yapabilen esnek bir program oluşturulmuştur. Bu kodlar ve verilen datalardan faydalanılarak istenen ara değerler hesaplanmıştır. Kübik interpolasyon bu sorudada olduğu gibi ara değerler bizden istenirse kullanmamız uygun olan bir yöntemdir. Kübik interpolasyon yaparken dataların eşit farklarla bölünmüş olabileceğine dikkat edilmelidir ve buna uygun daha basit olan formüller tercih edilmelidir.

İkinci soruda Eğri uydurma (Curve Fitting) yapılmıştır. Yazılan kodlar sağyesinde verilen değişik datalara göre hesaplama yapabilen esnek bir program oluşturulmuştur. Birinci dereceden bir polinom ile yakınsama yaptığımızda standart sapma 2.243 çıkmıştır. İkinci dereceden bir polinom ile yakınsadığımızda ise bu değer 0.812 civarındadır. Sonuç olarak 2. Dereceden polinomla yakınsamak birinci dereceden yani doğrusal yaklaşımdan daha iyi sonuç vermiştir, datalar daha iyi bir performansla takip edilmiştir. Ancak bu durumun polinomun derecesi ile doğru orantılı olduğu söylenemez, çünkü derece daha da arttırıldığında gürültü artacak ve dataların takip edilmesi zorlaşacaktır.

KAYNAKLAR:

Numerical Methods in Engineering (Jaan Kiusalaas) -CAMBRIDGE

MATLAB 7 Getting Started Guide -MathWorks