Numerical Methods HW-2
REPORT

Nilay Altun 150110224
Okan Gül 150110214

Q1.

Mathlab Code for Q1.a.:

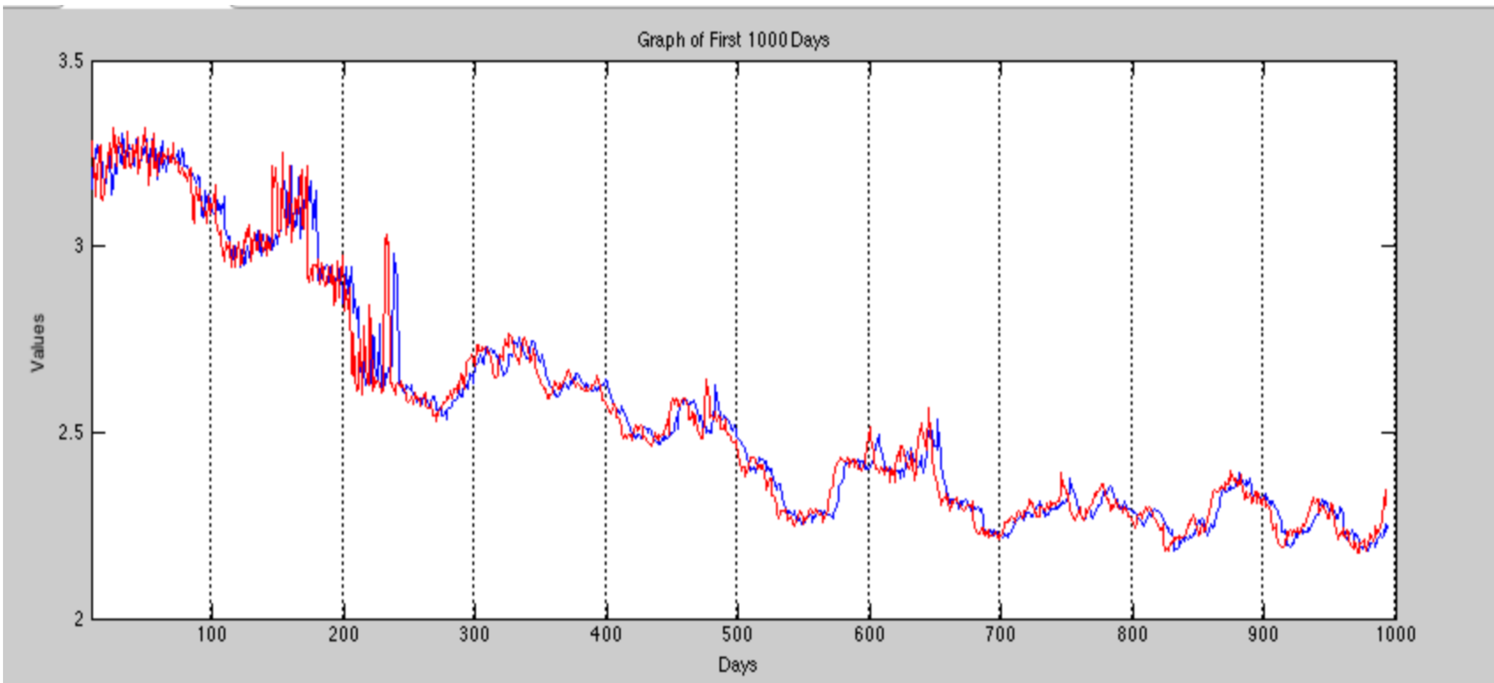
```
y= load ('dataset.txt');
% ‘y’ keeps the values in dataset.txt
i=1;
%  in for loop initializes the ‘py’ and ‘XM’.
% ’py’ keeps dataset values but from 8 to 1000.
% ‘XM’ keeps 993 row which is the values of 1000 day
% ’XM’ keeps 7 columns which is the values of day to day
for j= 8:1000
  py(i) = y(j);
  XM(i,1:7) = y(j-7:j-1,1)';
  i = i+1;
end
% alpha keeps the result of Least Square Method.
alpha = inv(XM'*XM)*XM'*py';

func=[];
%’func’ array keeps the coefficients.
% in for loop we calculated estimated values by using coefficients.
for n= 8:1000
  func(n)=alpha(1)*p(n-1)+alpha(2)*p(n-2)+alpha(3)*p(n-3)+alpha(4)*p(n-
  4)+alpha(5)*p(n-5)+alpha(6)*p(n-6)+alpha(7)*p(n-7);
end

result=[];
%’result’ array keeps the errors
% in for loop we calculated the errors.
for c= 8:1000
  result(c)=(abs(y(c)-func(c))/y(c))*100;
end

%’sum’ keeps the MSE
sum=0;
for d= 8:1000
```

```
  sum=sum+result(d);
end

sum=sum/993;
plot(func);
hold on;
plot(py,'r');
title('Graph of First 1000 Days');
xlabel('Days');
ylabel('Values')
```

Results for Q1.a.:

We found the coefficients: a1=0.0570
a2=-0.0185
a3=0.0722
a4=0.0062
a5=0.1133
a6=-0.0184
a7=0.7876

MSE : 0.0037



Mathlab Code for Q1.b.:

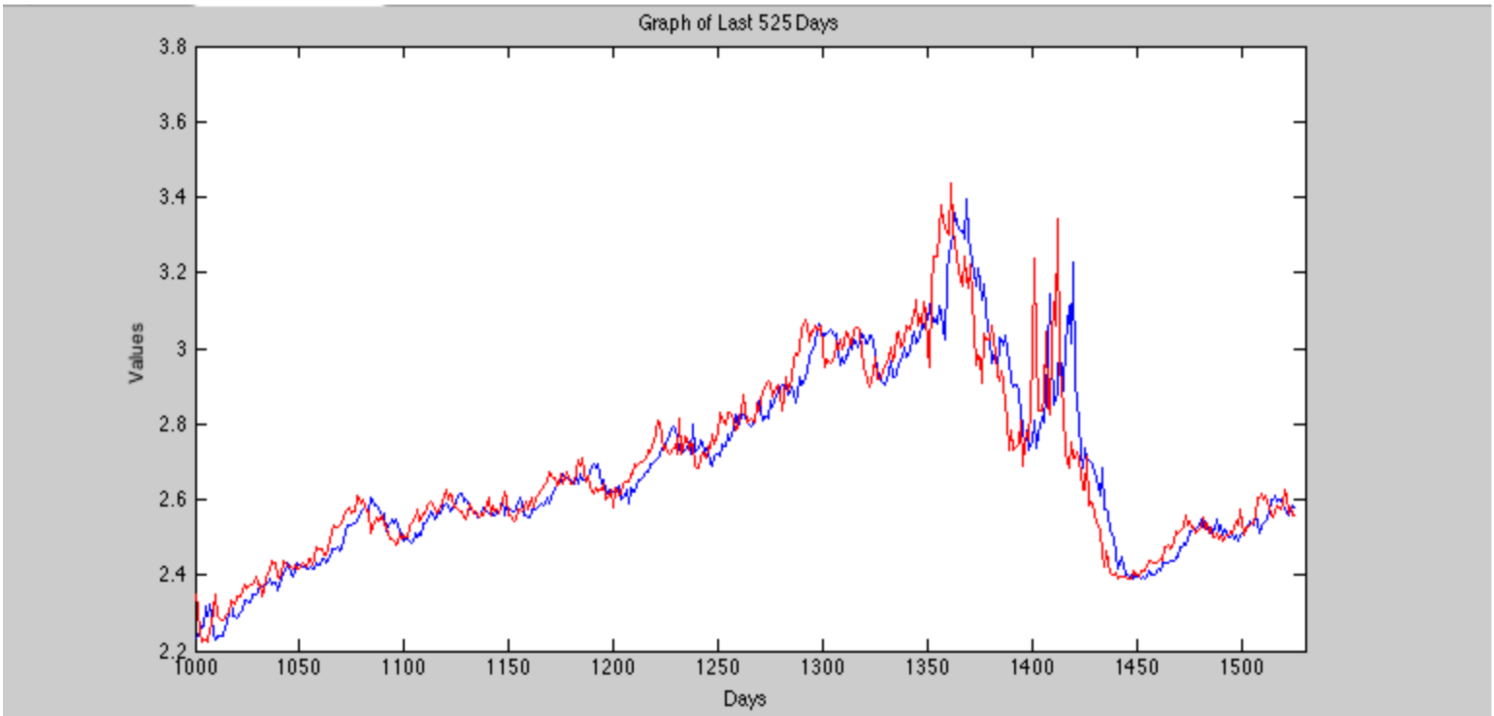
```
y= load ('dataset.txt');
% 'y' keeps the values in dataset.txt
i=1;
```

```
%  in for loop initializes the 'py' and 'XM'.
% 'py' keeps dataset values but from 8 to 1000.
% 'XM' keeps 993 row which is the values of 1000 day
% 'XM' keeps 7 columns which is the values of day to day
for j= 8:1000
  py(i) = y(j);
  XM(i,1:7) = y(j-7:j-1,1)';
  i = i+1;
end
% alpha keeps the result of Least Square Method.
alpha = inv(XM'*XM)*XM'*py'; % solving for m and c

func2=[];
%'func2' array keeps the coefficients.
% in for loop we calculated estimated values by using coefficients.
for k= 1000:1525
  func2(k)=alpha(1)*y(k-1)+alpha(2)*y(k-2)+alpha(3)*y(k-3)+alpha(4)*y(k-
  4)+alpha(5)*y(k-5)+alpha(6)*y(k-6)+alpha(7)*y(k-7);
end

result2=[];
%'result' array keeps the errors
% in for loop we calculated the errors.
for e= 1000:1525
  result2(e)=(abs(y(e)-func2(e))/y(e))*100;
end
%'sum2' keeps the MSE
sum2=0;

for f= 1000:1525
  sum2=sum2+result2(f);
end

sum2=sum2/526;

plot(func2);
hold on;
plot(y,'r');
title('Graph of Last 525 Days');
xlabel('Days');
  ylabel('Values');
```

Results for Q1.b.:

We used same the coefficients: a1=0.0570
a2=-0.0185
a3=0.0722
a4=0.0062
a5=0.1133
a6=-0.0184
a7=0.7876

MSE : 0.0069



Q1.c.:

We found MSE 0.0037 in first part and 0.0069 in the second part. Error is bigger in the second part because we are using 525 values to find the value. But in the second part we are using 1000 value for find it. So it is normal to find the big error in second part of the question.

Q2. We used Simpson's 1/3rd rule for calculating 2 h values and their errors.

Mathlab Code for Q2:

```
func= @(x) 1./(1+ x.^2);
 % We wrote the given function in 'func' variable
```

```
a=0;
b=4;
% We wrote upper limit 'b' and lower limit 'a'
h1=1;
h2=0.5;
% We wrote the 2 h values 'h1' and 'h2'
otherFunc=(h1/3)*(func(a)+4*func((a+b)/2)+func(b));
% 'otherFunc' keeps the value for h=1
otherFunc2=(h2/3)*(func(a)+4*func((a+b)/2)+func(b));
% 'otherFunc2' keeps the value for h=0.5
exactValue=integral(func,a,b);
% exactValue keeps the value of integral which is real value
error=((exactValue-otherFunc)/exactValue)*100;
% 'error'  keeps the error for h=1;
error2=((exactValue-otherFunc2)/exactValue)*100;
% 'error2' keeps the error for h=0.5
disp(error);
disp(error2);
```

Results for Q2:

We found the value for h=1 by using Simpson's $1/3^{rd}$ rule : 0.6196

We found the value for h=0.5 by using Simpson's $1/3^{rd}$ rule: 0.3098

Exact value of the integral : 1.3258

Error for h=1 is %53.2660
Error for h=0.5 is %76.6330

Q3. We used Newton's divided difference formula in Quadratic for calculating f(5).

Mathlab Code for Q3:

```
x= [1 3 4 6 7];

y= [1 27 81 729 2187];

% We wrote the given x and y values in array.
```

```
x0=x(2);

x1=x(3);

x2=x(4);

% We chose 3 values for x and values are: 4, 6, 7. Because the question ask for 5.

y0=y(2);

y1=y(3);

y2=y(4);

% Y values are the values of corresponding the x values.

b0=y0;

b1=(y1-y0)/(x1-x0);

b2=(((y2-y1)/(x2-x1))-((y1-y0)/(x1-x0)))/(x2-x0);

% We calculated the b0, b1 and b2 values for using in Newton's divided difference formula.

func = @(a) b0+b1*(a-x0)+b2*(a-x0)*(a-x1);

% 'func' keeps the value of the result.

disp(func(5));
```

Results for Q3:

We found the value for f(5) by using Newton's divided difference formula: 315.
Exact value by using f(x)=3^x is: 243.

Briefly, estimated value differs from the exact value. Because we are finding a function by using 3 point in given results. So, the function tries to generalize by using these 3 points and using Newton's divided difference formula. If the point number increase than the result will converge the exact value.