Assignment 4

Group Number 32

Software used: **Matlab**

Sanit Prashant Bhatkar (173109003)
Omkar Anil Pawar (173106001)

a. Lagrange Method

Code	Output

clearvars
clc
fprintf('\nLagrange Polynomial Interpolation')
 	
%Assuming 'n' stands for number of divisions, thus n+1 points
n=input('\n\nNumber of divisions: ');
n=n+1;
x =-1:2/(n-1):1;
y=cos(pi*x/2);
 	
%Initial polynomial coefficient matrix
ply=zeros(n);
 	
for i=1:n
dummy=x;
%dummy =[] deletes x(i) and lagrange formula numerator is obtained
%that is for L1=(x-x2)(x-x3)....(x-xn)
dummy(i)=[];
 	
%poly(dummy) forms coeff. matrix (x-x(2))*(x-x(3))*...(x-x(n)) @i=1
ply(i,:) = poly(dummy);
 	
%Denominator is L1 @ x=x1 or Li @ x=xi as per the formula
denom=polyval(ply(i,:),x(i));
 	
%Final form should have values L1y1,L2y2,....,Lnyn
ply(i,:)=ply(i,:)*y(i)/denom;
 	
end
 	
for i=1:n
%Adding L1y1,L2y2,....,Lnyn to get coefficients of interpolation poly.
    lan(i) = sum(ply(:,i));
end
	a. For N = 4
	
	Lagrange Polynomial Interpolation
	
	Number of divisions: 4
	
	lan =
	
	    0.2288 0.0000 -1.2288 -0.0000 1.0000
	
	max_error =
	
	    0.0017
	
	rms_error =
	
	    0.0017
	
	pn =
	
	(8242085728639369*x^4)/36028797018963968 + (6623676800028681*x^3)/16225927682921336339 1578010288128 - (5533860343450417*x^2)/4503599627370496 - (6623676800028681*x)/649037107316853453566312041152512 + 1

```
%Final lagrange polynomial
%for n divisions, n+1 points and n order polynomial
%Lagrange polynomial form: c1*x^n + c2*x^(n-1)+...+ cn-1*x + cn
%lan represents coeff. of lagrange poly. in [c1 c2 ... cn]

lan

%RMS error between [-1,1]
xx=-1:0.1:1;
yy=polyval(lan(1,:),xx);
y=cos(pi*xx/2);
error=abs(y-yy);
rms_error=error.^2;
max_error=max(error)
rms_error=sqrt((sum(rms_error(:))/n))


xx=-2:0.1:2;
yy=polyval(lan(1,:),xx);
y=cos(pi*xx/2);
error=abs(y-yy);
pn=poly2sym(lan)


figure
plot(xx,y,'-y','LineWidth',3)
ylim([-1.2,1.2])
title(['Lagrange Polynomial Interpolation for ' num2str(n-1) ' divisions'])
hold on
plot(xx,yy,'*')
hold on
plot(xx,error,'-r')
hold on
y=cos(pi*x/2);
plot(x,y,'ko')
hold off
legend('Actual function','Lagrange polynomial','Error','Location','south')
grid on
line([-1 -1],ylim,'LineStyle','--','Color','k')
line([1 1],ylim,'LineStyle','--','Color','k')
```

b. For N = 10

```
Lagrange Polynomial Interpolation

Number of divisions: 10

lan =

  Columns 1 through 6

   -0.0000    0.0000    0.0009   -0.0000   -0.0209    0.0000

  Columns 7 through 11

    0.2537   -0.0000   -1.2337    0.0000    1.0000


max_error =

   2.6651e-09


rms_error =

   1.1454e-09


pn =

-(1784600883220609*x^10)/73786976294838206464 + (2472518115395319*x^9)/4951760157141521099596496896 + (264742136488391*x^8)/288230376151711744 - (2930078147781109*x^7)/4951760157141521099596496896 - (6013424328065771*x^6)/288230376151711744 + (3096937580586111*x^5)/9903520314283042199192993792 + (1142425790112657*x^4)/4503599627370496 - (5502298764819965*x^3)/396140812571321687967719751 68 - (5556093334843627*x^2)/4503599627370496 + (4666635368965869*x)/1267650600228229401496703205376 + 1
```

c. For N = 20

Lagrange Polynomial Interpolation

Number of divisions: 20

lan =

Columns 1 through 8

-0.0000 0.0000 0.0000 -0.0000 -0.0001
0.0000 0.0000 -0.0000

Columns 9 through 16

-0.0000 0.0000 -0.0000 -0.0000 0.0009
0.0000 -0.0209 -0.0000

Columns 17 through 21

0.2537 0.0000 -1.2337 -0.0000 1.0000

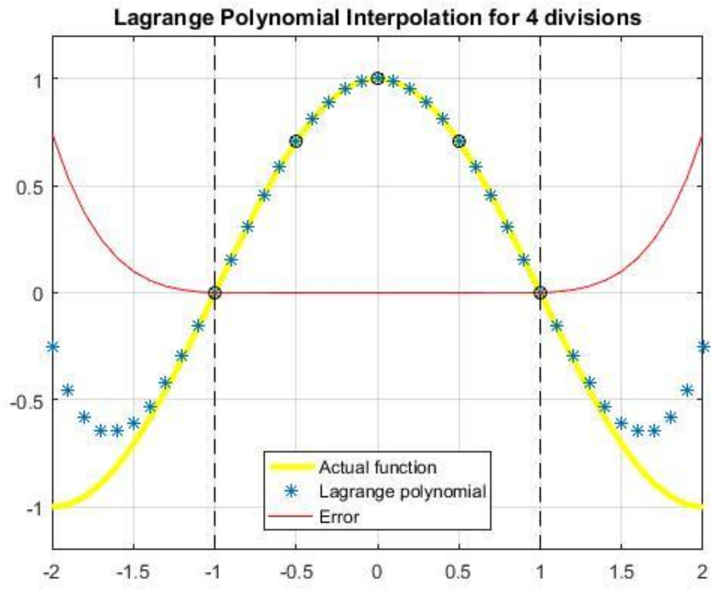
max_error =

1.7137e-07

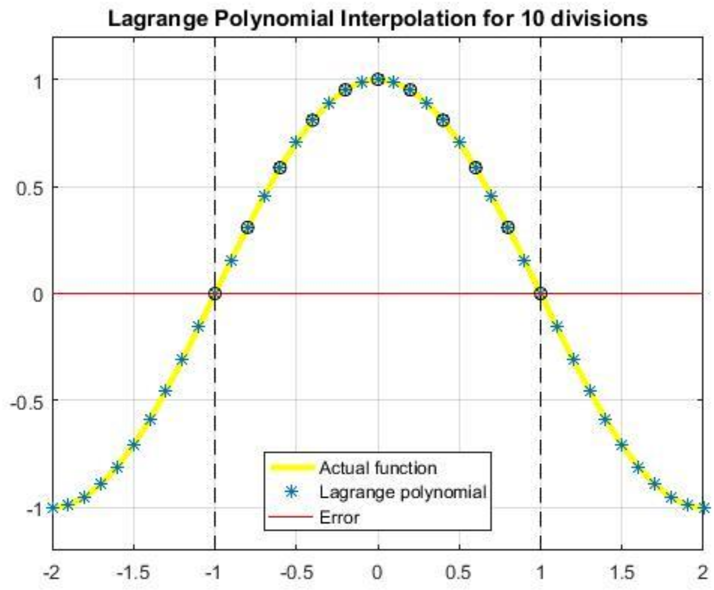
rms_error =

3.8450e-08

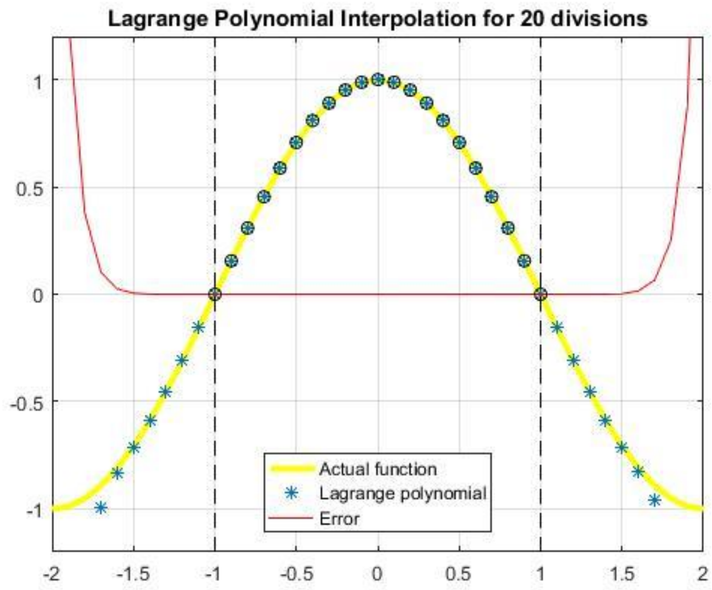
	pn = -(2645101880060487*x^20)/295147905179352825856 + (7660396971994569*x^19)/2361183241434822606848 + (4924173603297547*x^18)/147573952589676412928 - (3571874841775593*x^17)/295147905179352825856 - (3760669907659263*x^16)/73786976294838206464 + (1384008324306433*x^15)/73786976294838206464 + (6019586020649777*x^14)/147573952589676412928 - (8912674812154685*x^13)/590295810358705651712 - (1342155900455901*x^12)/73786976294838206464 + (8153469011575343*x^11)/1180591620717411303424 - (2989999570601579*x^10)/147573952589676412928 - (8864260043623783*x^9)/4722366482869645213696 + (4236267329734831*x^8)/4611686018427387904 + (4800592522965529*x^7)/18889465931478580854784 - (6013475391317521*x^6)/288230376151711744 - (2647249327025365*x^5)/151115727451828646838272 + (4569703582907389*x^4)/18014398509481984 + (8960994626703315*x^3)/19342813113834066795298816 - (2778046668920889*x^2)/2251799813685248 - (7796123293746515*x)/2475880078570760549798248448 + 1



Lagrange Interpolation at N=4



Lagrange Interpolation at N=10



Lagrange Interpolation at N=20

b. Newton Divided Difference Method

Code	Output

<pre>clc
clearvars
fprintf('\nNewton Divided Difference Interpolation')
 	
%Assuming 'n' stands for number of divisions, thus
n+1 points
n=input('\n\nNumber of divisions: ');
n=n+1;
x =-1:2/(n-1):1;
y=cos(pi*x/2);
 	
%divided difference in newton formula
divdif(:,1)=y';
%Initial polynomial coefficient matrix
ply=zeros(n);
 	
for i=2:n
 for j=i:n
 	
  %Divided difference formula.Put value of i=2 and
j=1,2,3..n
  %Formula is satisfied at all values of i and j
  divdif(j,i)=(divdif(j,i-1)-divdif(j-1,i-1))/(x(j)-x(j-i+1));
 end
end
%divdif gives divided differences. first column is
f(x0)
%second column is f[x0,x1], nth column is
f[x0,x1....xn]
divdif
divdif=diag(divdif)';
ply(1,n)=1*y(1);
for i=1:n-1
    dummy=x(1:i);
 	
%poly(dummy) forms coeff. matrix (x-x(1))*(x-
x(1))*...(x-x(n-1))@i=n
    ply(i+1,n-i:n)=poly(dummy);
 	
%Final form should have values N1b1,N2b2,....,Nnbn
    ply(i+1,:)= ply(i+1,:)*divdif(i+1);
end
 	
for i=1:n
 	
%Adding N1b1,N2b2,...,Nnbn to get coefficients of
interpolation poly.
newt(i)=sum(ply(:,i));
end</pre>
	a. For N = 4
	
	Newton Divided Difference Interpolation
	
	Number of divisions: 4
	
	newt =
	
	    0.2288 0 -1.2288 -0.0000 1.0000
	
	max_error =
	
	    0.0017
	
	rms_error =
	
	    0.0017
	
	pn =
	
	(8242085728639387*x^4)/36028797018963968 - (5533860343450419*x^2)/4503599627370496 - (17*x)/144115188075855872 + 1

```matlab
%Final Newton Divided Difference polynomial
%for n divisions, n+1 points and n order polynomial
%Newton polynomial form: c1*x^n + c2*x^(n-1)+...+
cn-1*x + cn
%newt represents coeff. of Newton poly. in [c1 c2 ...
cn]
newt

%RMS error between [-1,1]
xx=-1:0.1:1;
yy=polyval(newt(1,:),xx);
y=cos(pi*xx/2);
error=abs(y-yy);
max_error=max(error)
rms_error=error.^2;

rms_error=sqrt((sum(rms_error(:))/n))

xx=-2:0.1:2;
yy=polyval(newt(1,:),xx);
y=cos(pi*xx/2);
error=abs(y-yy);
pn=poly2sym(newt)


figure
plot(xx,y,'-y','LineWidth',3)
ylim([-1.2,1.2])
title(['Newton Divided Difference Interpolation for '
num2str(n-1) ' divisions' ])
hold on
plot(xx,yy,'*')
hold on
plot(xx,error,'-r')
hold on
y=cos(pi*x/2);
plot(x,y,'ko')
hold off
legend('Actual function','Newton
polynomial','Error','Location','south')
grid on
line([-1 -1],ylim,'LineStyle','--','Color','k')
line([1 1],ylim,'LineStyle','--','Color','k')
```

b. For N = 10

```
Newton Divided Difference Interpolation

Number of divisions: 10

newt =

  Columns 1 through 8

   -0.0000    0.0000    0.0009   -0.0000   -0.0209   -
0.0000    0.2537   -0.0000

  Columns 9 through 11

   -1.2337   -0.0000    1.0000


max_error =

   2.6648e-09


rms_error =

   1.1453e-09


pn =

-
(7138403359526329*x^10)/295147905179352825856 + x^9/147573952589676412928 +
(4235874178074747*x^8)/4611686018427387904 -
(51*x^7)/147573952589676412928 -
(6013424327923043*x^6)/288230376151711744 -
(951*x^5)/295147905179352825856 +
(4569703160443659*x^4)/18014398509481984 -
(324031*x^3)/4722366482869645213696 -
(1389023333710883*x^2)/1125899906842624 -
(683447*x)/37778931862957161709568 + 1
```

c. For N = 20

Newton Divided Difference Interpolation

Number of divisions: 20

newt =

Columns 1 through 8

-0.0000 0 0.0000 -0.0000 -0.0000 -0.0000
-0.0000 0.0000

Columns 9 through 16

0.0000 0.0000 -0.0000 -0.0000 0.0009
0.0000 -0.0209 0.0000

Columns 17 through 21

0.2537 0.0000 -1.2337 0.0000 1.0000

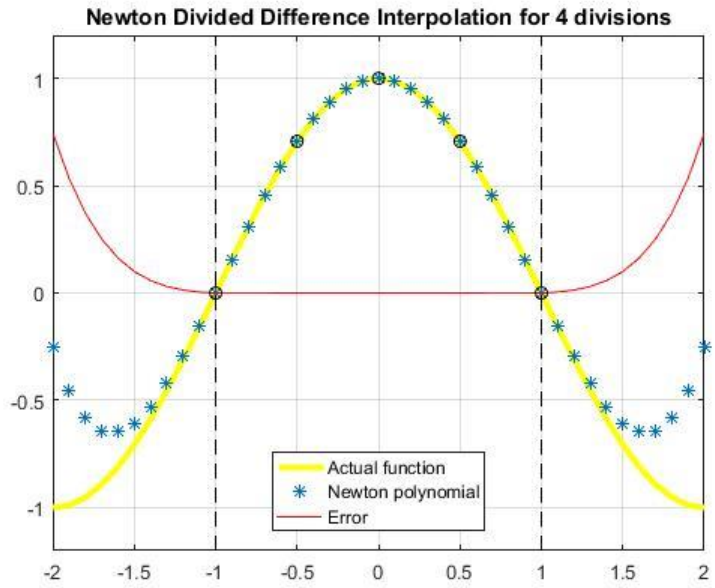
max_error =

3.3307e-16

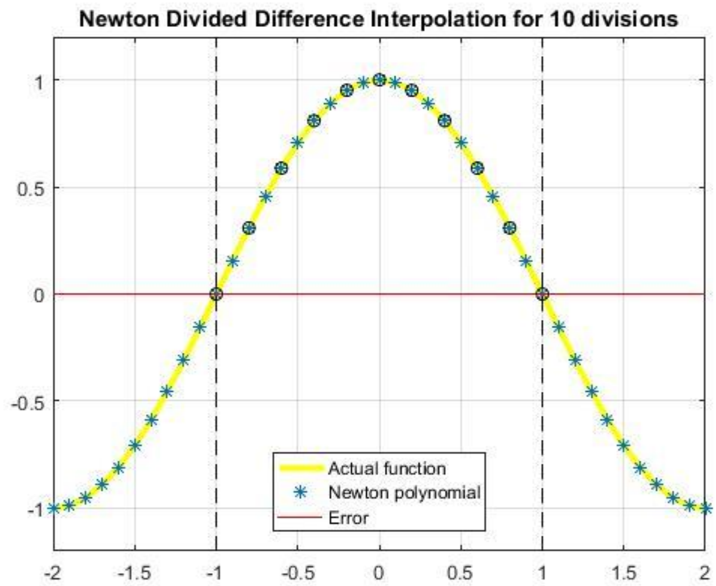
rms_error =

1.5651e-16

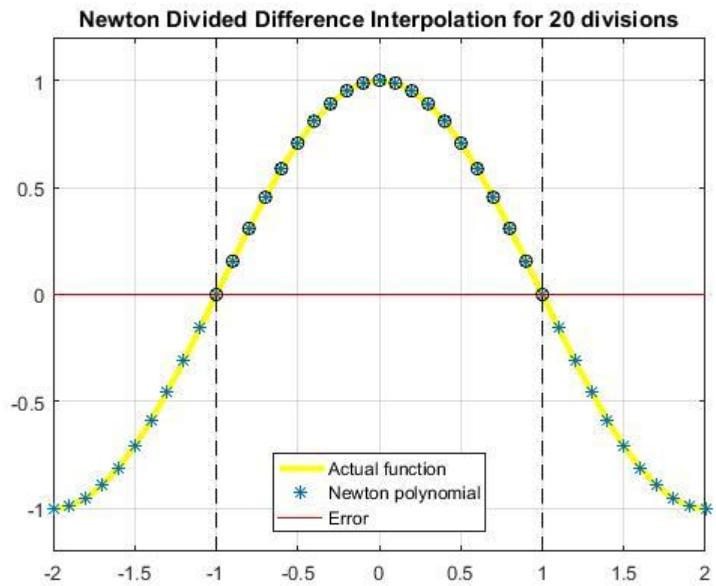
	pn = - (8456948818712247*x^20)/3868562622766813359 0597632 + (3941927195681103*x^18)/4835703278458516698 824704 - (3*x^17)/6044629098073145873530888 - (5732538901821299*x^16)/4835703278458516698 824704 - (81*x^15)/9671406556917033397649408 - (6480586690466133*x^14)/1208925819614629174 706176 + (111*x^13)/19342813113834066795298816 + (8889302123011159*x^12)/1888946593147858085 4784 + (2021*x^11)/19342813113834066795298816 - (3719143975115463*x^10)/1475739525896764129 28 - (438907*x^9)/773712524553362671811952664 + (4239339639880013*x^8)/4611686018427387904 + (31971267*x^7)/618970019642690137449562112 - (3006744453724177*x^6)/144115188075855872 + (85885198363*x^5)/3961408125713216879677197 5168 + (4569703605030317*x^4)/18014398509481984 + (5885497570921*x^3)/6338253001141147007483 51602688 - (2778046668940003*x^2)/2251799813685248 + (5338766571041131*x)/2028240960365167042394 7251286016 + 1



Newton Divided Difference at N=4



Newton Divided Difference at N=10



Newton Divided Difference at N=20

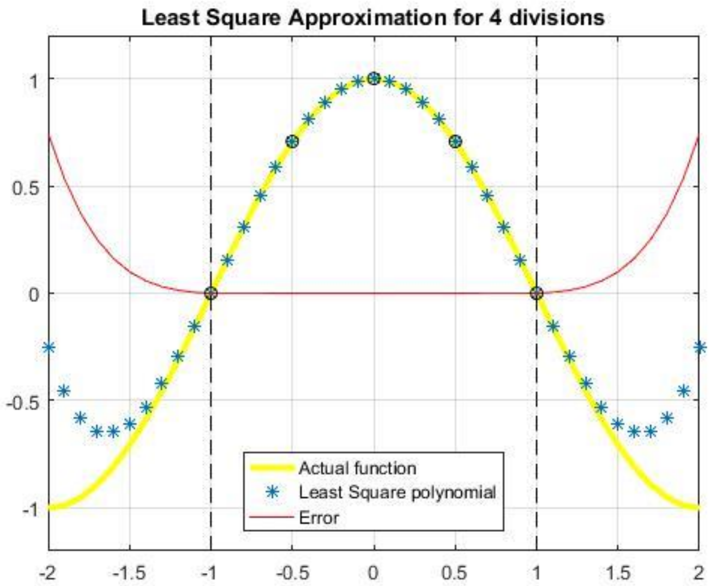
c. Least Square Method

Code	Output

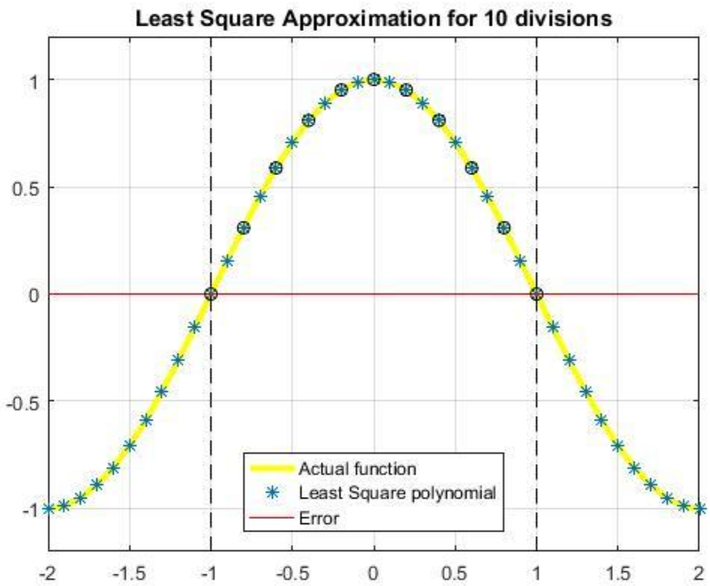
<pre>clc
clearvars
fprintf('\nLeast Square Approximation')
%Assuming 'n' stands for number of divisions, thus n+1 points
n=input('\n\nNumber of divisions: ');
n=n+1;
x =-1:2/(n-1):1;
y=cos(pi*x/2);
 	
for i=1:n
%x represents coeff of poly c1 + c2*x^2 +...+ cn*x^n
%substituting value of x in poly to obtain A matrix
  for j=1:n
  A(j,i)=(x(j))^(i-1);
  end
end
A;
%Least square formulation
%A'Ax=A'b for Ax=b
 	
y=A'*y';
M=A'*A;
x=M\y;
 	
%Final least square polynomial
%for n divisions, n+1 points and n order polynomial
%Least square polynomial form: c1*x^n + c2*x^(n-1)+...+ cn-1*x + cn
%lsqr represents coeff. of Least square poly. in [c1 c2 ... cn]
lsqr(1,n:-1:1)=x(:,1)
 	
%RMS error between [-1,1]
xx=-1:0.1:1;
yy=polyval(lsqr(1,:),xx);
y=cos(pi*xx/2);
error=abs(y-yy);
rms_error=error.^2;
max_error=max(error)
rms_error=sqrt((sum(rms_error(:))/n))
 	
xx=-2:0.1:2;
yy=polyval(lsqr(1,:),xx);
y=cos(pi*xx/2);
error=abs(y-yy);
pn=poly2sym(lsqr)</pre>
	a. For N = 4
	
	Least Square Approximation
	
	Number of divisions: 4
	
	lsqr =
	
	0.2288 0 -1.2288 0 1.0000
	
	max_error =
	
	0.0017
	
	rms_error =
	
	0.0017
	
	pn =
	
	(128782589509989*x^4)/562949953421312 - (1383465085862597*x^2)/1125899906842624 +

figure
plot(xx,y,'-y','LineWidth',3)
ylim([-1.2,1.2])
title(['Least Square Approximation for ' num2str(n-1) ' divisions'])
hold on
plot(xx,yy,'*')
hold on
plot(xx,error,'-r')
hold on
x =-1:2/(n-1):1;
y=cos(pi*x/2);
plot(x,y,'ko')
hold off
legend('Actual function','Least Square polynomial','Error','Location','south')
grid on
line([-1 -1],ylim,'LineStyle','--','Color','k')
line([1 1],ylim,'LineStyle','--','Color','k')
	b. For N = 10
	
	Least Square Approximation
	
	Number of divisions: 10
	
	lsqr =
	
	Columns 1 through 8
	
	-0.0000 0.0000 0.0009 -0.0000 -0.0209 0.0000 0.2537 -0.0000
	
	Columns 9 through 11
	
	-1.2337 0.0000 1.0000
	
	max_error =
	
	2.6952e-09
	
	rms_error =
	
	1.1583e-09
	
	pn =
	
	- (831159*x^10)/34359738368 + (120709202351*x^9)/2361183241434822606848 + (15784461*x^8)/17179869184 - (31008497017*x^7)/295147905179352825856 - (716834329*x^6)/34359738368 + (81597828279*x^5)/1180591620717411303424 + (34864086947*x^4)/137438953472 - (76293887429*x^3)/4722366482869645213696 - (2712936193291*x^2)/2199023255552 + (37004868325*x)/37778931862957161709568 + 17592186045145/17592186044416
	
	c. For N = 20
	
	Least Square Approximation
	
	Number of divisions: 20
	
	lsqr =
	
	Columns 1 through 7
	
	0.1289 0.0021 -0.4928 -0.0078 0.7792 0.0121 -0.6625

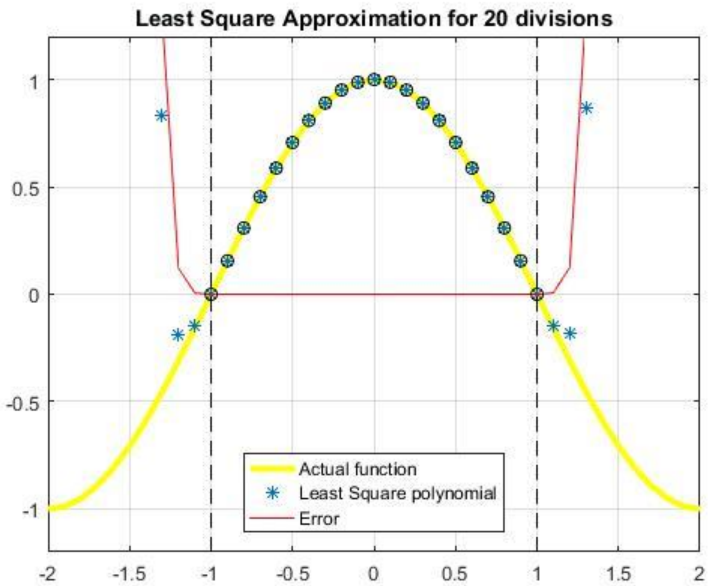
	Columns 8 through 14 -0.0100 0.3301 0.0048 -0.0986 -0.0014 0.0183 0.0002 Columns 15 through 21 -0.0226 -0.0000 0.2538 0.0000 -1.2337 -0.0000 1.0000 max_error = 4.1445e-09 rms_error = 1.8527e-09 pn = (4643591089207221*x^20)/36028797018963968 + (2392952061941535*x^19)/1152921504606846976 - (4438946944342217*x^18)/9007199254740992 - (4508558373663745*x^17)/576460752303423488 + (175453716021 6029*x^16)/2251799813685248 + (436746665663071*x^15)/36028797018963968 - (5967068792544121*x^14)/9007199254740992 - (2890025400909103*x^13)/288230376151711744 + (5946613110483717*x^12)/18014398509481984 + (2769814608130591*x^11)/576460752303423488 - (888158111970931*x^10)/9007199254740992 - (6250340312369889*x^9)/4611686018427387904 + (2630285659849161*x^8)/144115188075855872 + (8061441263561153*x^7)/36893488147419103232 - (3250430983534043*x^6)/144115188075855872 - (5474162741568005*x^5)/29514790517935282585 6 + (285697550502167*x^4)/1125899906842624 + (204998006630671*x^3)/295147905179352825856 - (1389024966613609*x^2)/1125899906842624 - (8928711415491907*x)/1208925819614629174706 176 + 4503599646035793/4503599627370496



Least Square at N=4



Least Square at N=10



Least Square at N=20

d. Cubic Spline

Code

```
clearvars
clc
fprintf('\nCubic Spline Interpolation')
%Assuming 'n' stands for number of divisions, thus n+1 points
n=input('\n\nNumber of divisions: ');
n=n+1;
x =-1:2/(n-1):1;
y=cos(pi*x/2);
%Calculation for S'(x) coefficient matrix
%ds stands for S'(x)
ds=zeros(n);
ds(1)=1;
ds(n,n)=1;
%rhs is RHS side of formula S'(x)
rhs=zeros(n,1);
for i=2:n-1
%Formula taken from Atkinson book
   ds(i,i-1)=(x(i)-x(i-1))/6;
   ds(i,i)=(x(i+1)-x(i-1))/3;
   ds(i,i+1)=(x(i+1)-x(i))/6;
   rhs(i)=((y(i+1)-y(i))/(x(i+1)-x(i))-(y(i)-y(i-1))/(x(i)-x(i-1)));

end
%m is S''(x)
m=ds\rhs;

%substitution in y formula
%It is very big term, so separate terms are calculated and then added
%a b c are those separate terms which add up to form final y=polynomial(x)
for i=2:n

%Formula taken from Atkinson book
a=(-(poly([x(i),x(i),x(i)])*m(i-1))+(poly([x(i-1),x(i-1),x(i-1)])*m(i)));
a=a/(6*(x(i)-x(i-1)));
b=(-poly(x(i))*y(i-1)+poly(x(i-1))*y(i))/(x(i)-x(i-1));
c=(poly(x(i))*m(i-1)-poly(x(i-1))*m(i))*((x(i)-x(i-1))/6);
size(a,2);
dummy=a+[zeros(1,size(a,2)-size(b,2)) b] + [zeros(1,size(a,2)-size(c,2)) c];
spline(i-1,:)=[dummy];

end
```

Output

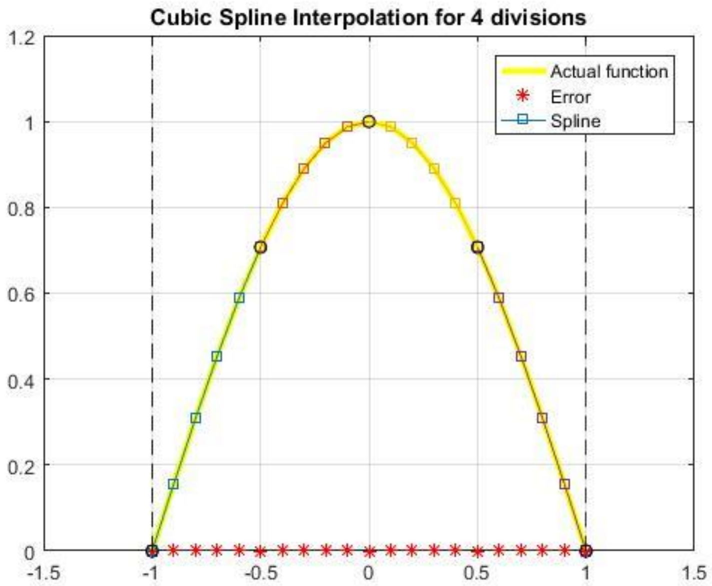
a. For N = 4

```
Cubic Spline Interpolation

Number of divisions: 4

spline =

  -0.6120  -1.8361  -0.2689   0.9552
  -0.2535  -1.2983        0   1.0000
   0.2535  -1.2983        0   1.0000
   0.6120  -1.8361   0.2689   0.9552


max_error =

  4.6865e-04


rms_error =

   0.0078
```

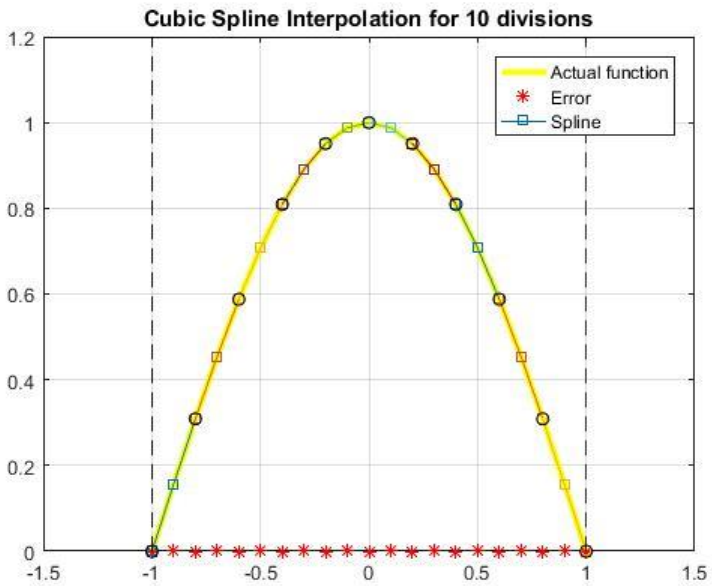
b. For N = 10

```
Cubic Spline Interpolation

Number of divisions: 10

spline =

  -0.6406  -1.9219  -0.3512   0.9301
  -0.5779  -1.7714  -0.2308   0.9622
  -0.4586  -1.5567  -0.1020   0.9879
  -0.2945  -1.3597  -0.0232   0.9985
  -0.1015  -1.2439  -0.0000   1.0000
   0.1015  -1.2439   0.0000   1.0000
   0.2945  -1.3597   0.0232   0.9985
   0.4586  -1.5567   0.1020   0.9879
   0.5779  -1.7714   0.2308   0.9622
   0.6406  -1.9219   0.3512   0.9301


max_error =

  4.0670e-06


rms_error =

  3.3388e-04
```

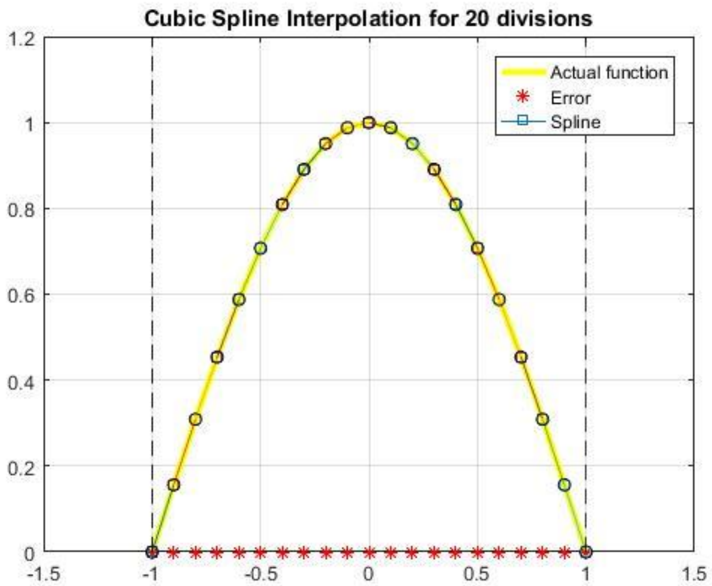
%Each row of spline matrix is polynomial fitting in that interval
%cubic spline form: c1*x^3 + c2*x^(2)+ c3*x + c4
%each row of spline represents coeff. of spline [c1 c2 c3 c4]
spline
 	
figure
xk=-1.5:0.1:1.5;
y=cos(pi*xk/2);
plot(xk,y,'-y','LineWidth',3)
ylim([0,1.2])
title(['Cubic Spline Interpolation for ' num2str(n-1) ' divisions'])
hold on
 	
%This loop is written in order to plot piecewise polynomial
%-1<=x<=-0.5 has differnt polynomial and -0.5<=x<=0 has differnt polynomial
%coeffs of first polynomial are the first row of spline matrix
%Thus between each interval differnt spline is fitted
for i=1:n-1
 	
xx=x(i):0.1:x(i+1);
y=cos(pi*xx/2);
error=abs(y-polyval(spline(i,:),xx));
rms_error =error.^2;
rms_error(i)=sqrt((sum(rms_error(:))/n));
plot(xx,error,'r*');
hold on
plot(xx,polyval(spline(i,:),xx),'-s');
hold on
end
hold on
y=cos(pi*x/2);
plot(x,y,'ko')
hold off
legend('Actual function','Error','Spline')
grid on
 	
%RMS error between [-1,1]
max_error=max(error)
rms_error=sqrt((sum(rms_error(:))/n))
line([-1 -1],ylim,'LineStyle','--','Color','k')
line([1 1],ylim,'LineStyle','--','Color','k')
	c. For N = 20
	
	Cubic Spline Interpolation
	
	Number of divisions: 20
	
	spline =
	
	-0.6446 -1.9339 -0.3631 0.9262
	-0.6288 -1.8910 -0.3245 0.9377
	-0.5974 -1.8158 -0.2643 0.9538
	-0.5513 -1.7191 -0.1966 0.9696
	-0.4917 -1.6117 -0.1322 0.9825
	-0.4200 -1.5041 -0.0784 0.9914
	-0.3379 -1.4056 -0.0390 0.9967
	-0.2475 -1.3242 -0.0146 0.9991
	-0.1510 -1.2663 -0.0030 0.9999
	-0.0507 -1.2362 0.0000 1.0000
	0.0507 -1.2362 -0.0000 1.0000
	0.1510 -1.2663 0.0030 0.9999
	0.2475 -1.3242 0.0146 0.9991
	0.3379 -1.4056 0.0390 0.9967
	0.4200 -1.5041 0.0784 0.9914
	0.4917 -1.6117 0.1322 0.9825
	0.5513 -1.7191 0.1966 0.9696
	0.5974 -1.8158 0.2643 0.9538
	0.6288 -1.8910 0.3245 0.9377
	0.6446 -1.9339 0.3631 0.9262
	
	max_error =
	
	8.3267e-17
	
	rms_error =
	
	1.0041e-09



Cubic Spline at N=4



Cubic Spline at N=10



Cubic Spline at N=20

➢ Result and Conclusion

Method	N	4	10	20
Lagrange	Max Error	0.0017	2.6651e-9	1.7137e-7
	R.M.S. Error	0.0017	1.1454e-9	3.8450e-08
Newton Divided Difference	Max Error	0.0017	2.6648e-9	3.3307e--16
	R.M.S. Error	0.0017	1.1453e-9	1.5651e-16
Least Square Method	Max Error	0.0017	2.6952e-09	4.1445e-09
	R.M.S. Error	0.0017	1.1583e-09	1.8527e-09
Cubic Spline	Max Error	4.6865e-04	4.067e-06	8.3267e-17
	R.M.S. Error	.0078	3.3388e-04	1.0041e-09

➢ Errors in Same Interval

a) Maximum Error

- N = 4 Maximum Error is predicted by Lagrange, Newton divided difference and Least square.
  Error in Lagrange=Newton divided difference=Least square>Cubic Spline

- N = 10 Maximum Error is predicted by Cubic Spline
  Error in Cubic Spline >Least Square Method >Lagrange >Newton divided difference

- N = 20 Maximum Error is predicted by Lagrange
  Error in Lagrange>Least Square>Newton Divided Difference>Cubic Spline

b) R.M.S. Error

- N = 4 Maximum R.M.S. Error is predicted by Lagrange, Newton divided difference and Least square.
  R.M.S Error in Lagrange=Newton divided difference=Least square>Cubic Spline

- N = 10 Maximum R.M.S. Error is predicted by Cubic Spline
  R.M.S Error in Cubic Spline >Least Square Method >Lagrange >Newton divided difference

- N = 20 Maximum R.M.S. Error is predicted by Lagrange
  R.M.S Error in Lagrange>Least Square >Cubic Spline>Newton Divided Difference

- Errors in Method

a) <u>For Lagrange</u>

- Maximum Error is predicted at N = 4
  Maximum Error in N = 4 > N = 20 > N = 10

- Maximum R.M.S. is predicted at N = 4
  R.M.S Error in N = 4 > N = 20 > N = 10

b) <u>For Newton Divided Difference</u>

- Maximum Error is predicted at N = 4
  Maximum Error in N = 4 > N = 10 > N = 20

- Maximum R.M.S. is predicted at N = 4
  R.M.S Error in N = 4 > N = 10 > N = 20

c) <u>For Least Square Method</u>

- Maximum Error is predicted at N = 4
  Maximum Error in N = 4 > N = 20 > N = 10

- Maximum R.M.S. is predicted at N = 4
  R.M.S Error in N = 4 > N = 20 > N = 10

d) <u>For Cubic Spline Method</u>

- Maximum Error is predicted at N = 4
  Maximum Error in N = 4 > N = 10 > N = 20

- Maximum R.M.S. is predicted at N = 4
  R.M.S Error in N = 4 > N = 10 > N = 20