Name: Senthamilaruvi Moorthy

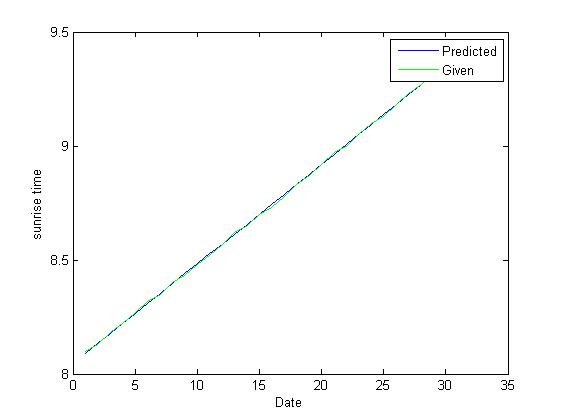
CWID: 10722805

HW : LAB 7

TABLE OF CONTENTS

|  |  |
| --- | --- |
| Content | Page Number |
| 1. Derivation 1.a | 1 |
| 2. Summary 1. b and 1. c | 4 |
| 1. Summary 2.a 2.b and 2.c | 5 |
| 1. Summary 4.a and 4.b | 6 |
| 1. Summary 4.c | 7 |
| 1. Summary 4.d | 8 |
| 1. Summary 4.e | 9 |
| 1. Program 1.b &1.c | 10 |
| 1. Program 2.a &2.b | 12 |
| 1. Program 2.c | 13 |
| 1. Program 4.b | 14 |
| 1. Program 4.c | 15 |
| 1. Program 4.d | 16 |
| 1. Program 4.e | 18 |

1. b& 1. c
2. A function has been created using Matlab for performing least square fit to a given set of data.
3. The function gets the input values of dependant and independent variables in a matrix form and stores the values of dependant and independent variables in a separate matrix.
4. The order of the polynomial was given as user input.
5. A matrix has been created to store the values of [x x^2 x^3….] .Depending on the order of the polynomial given by the user the matrix adds the number of columns.
6. The matrix was augmented to a column of ones to account for constant term in polynomial.
7. The coefficient matrix was found using the A\B function in Matlab.
8. A polynomial function was written using the values of coefficients obtained
9. Then the output value was predicted using the polynomial function obtained.
10. Percentage error was obtained between the real and the value given by the polynomial function fitted to the data. (Plot shown below)
11. The best Mth order polynomial was determined by comparing the percentage error. **The Polynomial of order 1 was found better fit to the given sunrise data.**



2.

i. The approximate solution for u(x) has been assumed as

u(x) = a0 +a1 x +a2 x^2+a3 x^3+a4 x^4+a5 x^5

ii. The constants a0 and a1 are eliminated using given boundary conditions u(0) = 0 and u(1) =1 a0 = 0 and a1 = -(a1 + a2 + a3 + a4 + a5).So four unknown coefficients are left.

iii. The residual R(x) is obtained by substituting the u(x) in the given differential equation.

2.a. **Point collocation**

i. Four collocation points 0.2, 0.4, 0.6 and 0.8 (in the interval (0,1) are substituted in the residual R(x) and four linear equations of the four unknown coefficients are obtained.

ii. Now the numbers of unknowns are equal to the number of equations. Then the linear equations are solved to obtain the solution for coefficients.

iii. **The coefficient matrix [a0 a1 a2 a3 a4 a5] = [0 -0.1490 -0.003 0.1427 -0.0015 0.008]**

2.b**Sub-domain collocation**

i. Four sub-domains 0.1-0.3, 0.3-0.5, 0.5-0.7 and 0.7-0.9 (in the interval (0,1) are chosen.

ii. The residual is integrated over the four sub-domains to obtain four linear equation of unknown coefficients.

iii. **The coefficient matrix [a0 a1 a2 a3 a4 a5] = [0 -0.1490 -0.003 0.1427 -0.0015 0.008]**

2.c**Galerkin method**

i. The residual R(x) is multiplied with weighting functions w2(x) = x^2(1-x), w3(x) = x^3(1-x), w4(x) = x^4 (1-x) and w5(x) = x^5 (1-x) respectively and integrated over the interval (0,1) to obtain four linear equations

ii. Now the numbers of unknowns are equal to the number of equations. Then the linear equations are solved to obtain the solution for coefficients.

iii. **The coefficient matrix [a0 a1 a2 a3 a4 a5] = [0 -0.1490 -0.004 0.1431 -0.0020 0.0083]**

The coefficients of a0 a1 a2 a3 a4 and a5 obtained by all the above three methods are found to be approximately equal

4.aThe approximating function used is

n

∑(AjG(xj;xj’,yj;yj’)

J= 1

The approximate solution satisfies the differential equations but not the boundary conditions.

b.

i.Source and field points are fed into the function as separate matrix. The green function

G(x;x’)= (-1/2π)\*logr, ԀG/Ԁn = (-1/2πr^2) (x-x’)nx +(u-u’)ny are written as separate functions in Matlab and called as and when required.

ii. The green function matrix was populated using the appropriate values and functions for the field points located at x=0, x=1 ,y =0 and y =1.

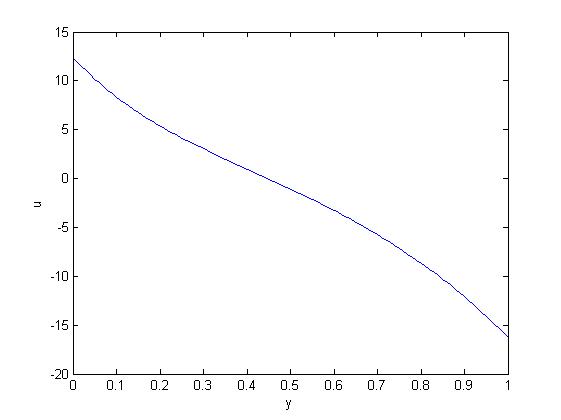
iii. QR decomposition was used to condition the green function matrix.

iv. The coefficients (source strengths) were found using backslash command in Matlab.

v. The approximate solutions u(x,y) was found using source strengths and appropriate values of green’s function.

vi. The u(x,y) could be plotted at x =0.5 and y= 0-1. The solution has errors at boundary.

Shown below.

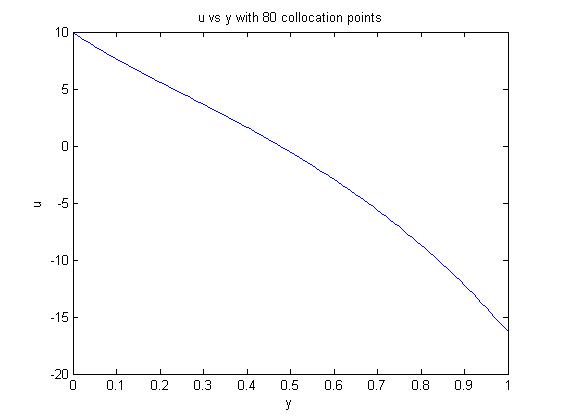


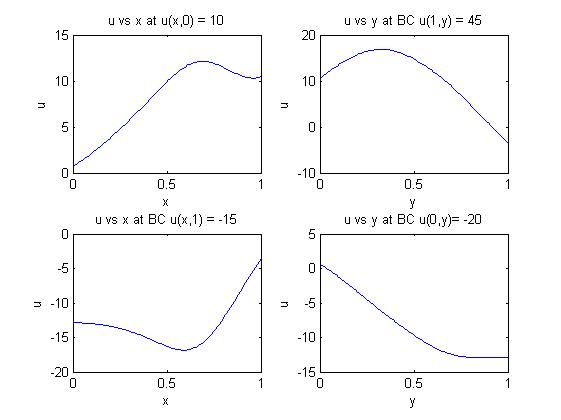
4.C.

i. The source strength is estimated again as mentioned in the previous problem but with an overdeterministic approach with 80 collocation points.

ii. The 80 points are divided as 20 points each on the line of x = 0, x =1, y =0 and y =1.

iii. The solution was plotted again and found that the accuracy of the solution has increased

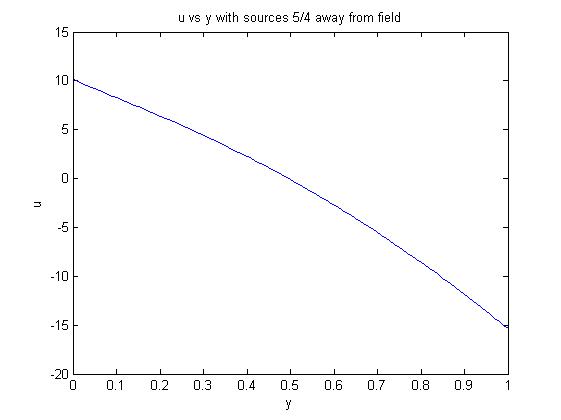


4.d. 

i. The approximate solution u(x,y) and q(x,y) at boundary conditions u(x,0) = 10 , u(x,1) = - 15 , u(1,y) =45 and u(0,y) = -20 were plotted as shown above.

ii. As a general observation at the mid of the interval (0,1),x =0.5 /y=0.5 the approximate solution becomes more nearer to the exact solution than at the boundaries.

4. e.



1. The u(x,y) is plotted against y again but the source points are moved 5/4 away from the field .
2. The solution gets improved as the source points are moved from the field.

Program 1.b.

% Program for fitting a performing a least square fit to a given set of

% data using an Mthordrpolynamial

functionpolfit(B)

% Getting order of polynomial as input from user

k = input('Please enter the order of the polynomial:');

r = length(B);

X = zeros(r,1);

Y = zeros(r,1);

F = zeros(r,k);

error = zeros(r,1);

% Store the values of dependant and independent variables

fori = 1: r

X(i,1) = B(i,1);

Y(i,1) = B(i,2);

end

% creating matrix of values of x x^2 and x^3 ..depending on order of

% polynomial

fori = 1: r

for p = 1:k

F(i,p) = (X(i,1)).^p;

end

end

%finding coefficient matrix

D = ones(r,1);

A = [D F];

C = A\Y;

y = zeros(r,1);

% Finding the curve

fori = 1:r

for j = 1:k+1

y(i,1) = y(i,1)+ C(j,1)\*(X(i,1))^(j-1);

end

end

% Plotting the curve

plot(X,y,'b-')

xlabel('Date')

ylabel('sunrise time')

holdon

plot(X,Y,'g-')

legend('Predicted','Given')

%Calculating percentage error

fori = 1: r

error(i,1) = (((Y(i)-y(i))\*100)/Y(i));

avgpererr = (sum(error(i,1))/length(error));

end

fprintf('The average percentage error is %3.5f\n',avgpererr);

end

INPUT

B = xlsread(‘Sunrise.xlsx’)

OUTPUT

>>polfit(B)

Please enter the order of the polynomial:3

The average percentage error is -0.00258

>>polfit(B)

Please enter the order of the polynomial:2

The average percentage error is -0.00242

>>polfit(B)

Please enter the order of the polynomial:1

The average percentage error is -0.00008

2.a. Point Collocation

%Solving differential equation using point collocation

clear

clc

symsRxa0a1a2a3a4a5u

%Elimiating constants with boundary conditions

a0 = 0;

a1 = -(a2 + a3 + a4 + a5);

u = -a2\*x - a3\*x - a4\*x - a5\*x + a2\*x^2 + a3\*x^3 +a4\*x^4 +a5\*x^5;

% Residual

R(x) = diff(diff(u,x))-u-x;

%choosing collocation

p1 = 0.2;

p2 = 0.4;

p3 = 0.6;

p4 = 0.8;

%Solving for constants

c = solve(R(p1)== 0,R(p2) == 0, R(p3) == 0, R(p4) == 0);

af2 = eval(c.a2);

af3 = eval(c.a3);

af4 = eval(c.a4);

af5 = eval(c.a5);

af1 = -(af2 + af3 + af4 + af5);

fprintf('The coefficient matrix is')

A = [a0 af1 af2 af3 af4 af5]

OUTPUT

The coefficient matrix is

A = 0 -0.1490 -0.0003 0.1427 -0.0015 0.0080

2 b.

%Solving differential equation using subdomain collocation

clear

clc

symsRxa0a1a2a3a4a5u

%Elimiating constants with boundary conditions

a0 = 0;

a1 = -(a2 + a3 + a4 + a5);

u = -a2\*x - a3\*x - a4\*x - a5\*x + a2\*x^2 + a3\*x^3 +a4\*x^4 +a5\*x^5;

% Residual

R(x) = diff(diff(u,x))-u-x;

%subdomain collocation

sb1 = int(R(x),x,0.1,0.3);

sb2 = int(R(x),x,0.3,0.5);

sb3 = int(R(x),x,0.5,0.7);

sb4 = int(R(x),x,0.7,0.9);

%Solving for constants

c = solve(sb1 == 0,sb2 == 0, sb3 == 0, sb4 == 0);

af2 = eval(c.a2);

af3 = eval(c.a3);

af4 = eval(c.a4);

af5 = eval(c.a5);

af1 = -(af2 + af3 + af4 + af5);

fprintf('The coeffecient matrix is')

A = [a0 af1 af2 af3 af4 af5]

OUTPUT

The coefficient matrix is

A = 0 -0.1490 -0.0003 0.1427 -0.0015 0.0080

2 c

%Solving differential equation using subdomain collocation

clear

clc

symsRxa0a1a2a3a4a5u

%Elimiating constants with boundary conditions

a0 = 0;

a1 = -(a2 + a3 + a4 + a5);

u = -a2\*x - a3\*x - a4\*x - a5\*x + a2\*x^2 + a3\*x^3 +a4\*x^4 +a5\*x^5;

% Residual

R(x) = diff(diff(u,x))-u-x;

%Weightingfuction

W2(x) = (x^2)\*(1-x);

W3(x) = (x^3)\*(1-x);

W4(x) = (x^4)\*(1-x);

W5(x) = (x^5)\*(1-x);

%Integrating with weighting function

Ga1 = int((R(x)\*W2(x)),x,0,1);

Ga2 = int(R(x)\*W3(x),x,0,1);

Ga3 = int(R(x)\*W4(x),x,0,1);

Ga4 = int(R(x)\*W5(x),x,0,1);

%Solving for constants

c = solve(Ga1 == 0,Ga2 == 0, Ga3 == 0, Ga4 == 0);

af2 = eval(c.a2);

af3 = eval(c.a3);

af4 = eval(c.a4);

af5 = eval(c.a5);

af1 = -(af2 + af3 + af4 + af5);

fprintf('The coefficient matrix is')

A = [a0 af1 af2 af3 af4 af5]The coefficient matrix is

OUTPUT

A = 0 -0.1490 -0.0004 0.1431 -0.0020 0.0083

.

4. b.

%This script is for finding approximate solution for a PDE

% using deterministic approach

clc

clear

%source points

s = [ 1 -1/4; 2/3 -1/4; 5/4 1/3; 5/4 2/3; 2/3 5/4 ; 1/3 5/4; -1/4 2/3; -1/4 1/3 ];

%field points

f = [ 1/3 0; 2/3 0; 1 1/3; 1 2/3; 2/3 1; 1/3 1; 0 2/3; 0 1/3];

% Boundary condition matrix

bc = [10 10 45 45 -15 -15 -20 -20]';

%initializing gmat

gmat = zeros(length(f), length(s));

%Populating gmat

fori = 1:length(f)

for j = 1:length(s)

x = f(i,1);

y = f(i,2);

x1 = s(j,1);

y1 = s(j,2);

if y == 0 || y == 1

gmat(i,j) = green(x,y,x1,y1);

elseif x == 1

gmat(i,j) = Dgreen(x,y,x1,y1,1,0);

elseif x == 0

gmat(i,j) = Dgreen(x,y,x1,y1,-1,0);

end

end

end

[Q,R]=qr(gmat);

c=Q'\*bc;

A = R\c;

symsuxyu\_exact

u(x,y) = 0;

fori =1: length(s)

u(x,y) = u(x,y) + A(i)\*green(x,y,s(i,1),s(i,2));

end

yplot=(0:0.01:1);

plot(yplot,eval(u(0.5,yplot)))

xlabel('y')

ylabel('u')

4 c.

%This script is for finding approximate solution for a PDE

% using deterministic approach

clc

clear

%source points

s = [ 1 -1/4; 2/3 -1/4; 5/4 1/3; 5/4 2/3; 2/3 5/4 ; 1/3 5/4; -1/4 2/3; -1/4 1/3 ];

%field points

xf = (linspace(0.2,0.8,20))';

yf = zeros(20,1);

a = [xfyf];

xs = ones(20,1);

ys = (linspace(0.2,0.8,20))';

b = [xsys];

yt = ones(20,1);

d = [xfyt];

xl = zeros(20,1);

e = [xlys];

f =[ a; b; d; e];

% Boundary condition matrix

bc = [10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 ]';

%initializing gmat

gmat = zeros(length(f), length(s));

%Populating gmat

fori = 1:length(f)

for j = 1:length(s)

x = f(i,1);

y = f(i,2);

x1 = s(j,1);

y1 = s(j,2);

if y == 0 || y == 1

gmat(i,j) = green(x,y,x1,y1);

elseif x == 1

gmat(i,j) = Dgreen(x,y,x1,y1,1,0);

elseif x == 0

gmat(i,j) = Dgreen(x,y,x1,y1,-1,0);

end

end

end

[Q,R]=qr(gmat);

c=Q'\*bc;

A = R\c;

symsuxyu\_exact

u(x,y) = 0;

fori =1: length(s)

u(x,y) = u(x,y) + A(i)\*green(x,y,s(i,1),s(i,2));

end

yplot=(0:0.01:1);

plot(yplot,eval(u(0.5,yplot)))

xlabel('y')

ylabel('u')

title('u vs y with 80 collocation points')

4 d.

%This script is for finding approximate solution for a PDE

% using deterministic approach

clc

clear

%source points

s = [ 1 -1/4; 2/3 -1/4; 5/4 1/3; 5/4 2/3; 2/3 5/4 ; 1/3 5/4; -1/4 2/3; -1/4 1/3 ];

%field points

xf = (linspace(0.2,0.8,20))';

yf = zeros(20,1);

a = [xfyf];

xs = ones(20,1);

ys = (linspace(0.2,0.8,20))';

b = [xsys];

yt = ones(20,1);

d = [xfyt];

xl = zeros(20,1);

e = [xlys];

f =[ a; b; d; e];

% Boundary condition matrix

bc = [10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 ]';

%initializing gmat

gmat = zeros(length(f), length(s));

%initializing percerror matrix

percerror = zeros(100,1);

%Populating gmat

fori = 1:length(f)

for j = 1:length(s)

x = f(i,1);

y = f(i,2);

x1 = s(j,1);

y1 = s(j,2);

if y == 0 || y == 1

gmat(i,j) = green(x,y,x1,y1);

elseif x == 1

gmat(i,j) = Dgreen(x,y,x1,y1,1,0);

elseif x == 0

gmat(i,j) = Dgreen(x,y,x1,y1,-1,0);

end

end

end

[Q,R]=qr(gmat);

c=Q'\*bc;

A = R\c;

% approximate solution of u(x,y) and q(x,y)

symsuq1q2xy

u(x,y) = 0;

q1(x,y) = 0;

q2(x,y) = 0;

fori =1: length(s)

u(x,y) = u(x,y) + A(i)\*green(x,y,s(i,1),s(i,2));

end

fori = 1: length(s)

q1(x,y) = q1(x,y) + A(i)\*Dgreen(x,y,s(i,1),s(i,2),1,0);

end

fori = 1: length(s)

q2(x,y) = q2(x,y) + A(i)\*Dgreen(x,y,s(i,1),s(i,2),-1,0);

end

%Estimating error at bc 1

xplot = (linspace(0,1,100))';

fori = 1 :100

u\_approx = eval(u(xplot(i,1),0));

percerror(i,1) = abs(((u\_approx -10)/10)\*100);

end

percenterror = ((sum(percerror))/length(percerror));

fprintf('The percentage error at u(x,0) = 10 is %3.3f\n',percenterror);

subplot(2,2,1);

plot(xplot,eval(u(xplot,0)))

xlabel('x')

ylabel('u')

title('u vs x at u(x,0) = 10')

%Estimating error at bc 2

yplot = (linspace(0,1,100))';

fori = 1 :100

q1\_approx = eval(q1(1,yplot(i,1)));

percerror(i,1) = abs(((q1\_approx - 45)/45)\*100);

end

percenterror = (sum(percerror))/length(percerror);

fprintf('The percentage error at q(1,y)= 45 is %3.2f\n',percenterror);

subplot(2,2,2)

plot(yplot,eval(q1(1,yplot)))

xlabel('y')

ylabel('q')

title('q vs y at BC q(1,y) = 45')

%Estimating error at bc 3

xplot = (linspace(0,1,100))';

fori = 1 :100

u\_approx = eval(u(xplot(i,1),1));

percerror(i,1) = abs(((u\_approx +15)/(-15))\*100);

end

percenterror = (sum(percerror))/length(percerror);

fprintf('The percentage error at BC u(x,1) = 10 is %3.2f\n', percenterror);

subplot(2,2,3)

plot(xplot,eval(u(xplot,1)))

xlabel('x')

ylabel('u')

title('u vs x at BC u(x,1) = -15')

%Estimating error at bc 4

yplot = (linspace(0,1,100))';

fori = 1 :100

q2\_approx = eval(q2(0,yplot(i,1)));

percerror(i,1) = abs(((q2\_approx + 20)/(-20))\*100);

end

percenterror = (sum(percerror))/length(percerror);

fprintf('The percentage error at BC q(0,y) = -20 is %3.2f\n',percenterror);

subplot(2,2,4)

plot(yplot,eval(q2(0,yplot)))

xlabel('y')

ylabel('q')

title('q vs y at BC q(0,y)= -20')

4 e.

%This script is for finding approximate solution for a PDE

% using deterministic approach

clc

clear

%source points

s = [ 1 -5/4; 2/3 -5/4; 9/4 1/3; 9/4 2/3; 2/3 9/4 ; 1/3 9/4; -5/4 2/3; -5/4 1/3 ];

% 80 field points

xf = (linspace(0.2,0.8,20))';

yf = zeros(20,1);

a = [xfyf];

xs = ones(20,1);

ys = (linspace(0.2,0.8,20))';

b = [xsys];

yt = ones(20,1);

d = [xfyt];

xl = zeros(20,1);

e = [xlys];

f =[ a; b; d; e];

% Boundary condition matrix

bc = [10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 ]';

%initializing gmat

gmat = zeros(length(f), length(s));

%Populating gmat

fori = 1:length(f)

for j = 1:length(s)

x = f(i,1);

y = f(i,2);

x1 = s(j,1);

y1 = s(j,2);

if y == 0 || y == 1

gmat(i,j) = green(x,y,x1,y1);

elseif x == 1

gmat(i,j) = Dgreen(x,y,x1,y1,1,0);

elseif x == 0

gmat(i,j) = Dgreen(x,y,x1,y1,-1,0);

end

end

end

[Q,R]=qr(gmat);

c=Q'\*bc;

A = R\c;

symsuxyu\_exact

u(x,y) = 0;

fori =1: length(s)

u(x,y) = u(x,y) + A(i)\*green(x,y,s(i,1),s(i,2));

end

yplot=(0:0.01:1);

plot(yplot,eval(u(0.5,yplot)))

xlabel('y')

ylabel('u')

title('u vs y with sources 5/4 away from field')

function G = green( x,y,x1,y1 )

%The function calculates green function values for the given two points

r = sqrt((x - x1)^2 + (y - y1)^2);

G = -(log(r)/(2\*pi));

End

function DG = Dgreen(x,y,x1,y1,nx,ny)

%Thsi function calculates partial derivative of Green funtion with

% respect to normal unit vector

r = sqrt((x-x1)^2 + (y-y1)^2);

DG = ((-1/(2\*pi\*r^2)\*((x-x1)\*nx+(y-y1)\*ny)));

end