REFERENCES

[1] T. Belytschko, Y. Y. Lu, L. Gu. Element-free Galerkin methods. **International Journal for Numerical Methods in Engineering** 1994; 37(2): 229-256.

[2] W.K. Liu, S. Jun, Y. F. Zhang. Reproducing kernel particle methods. Finite elements in fluids-new trends and applications (Barcelona, 1993). **International Journal for Numerical Methods in Fluids** 1995; 20(8-9): 1081-1106.

[3] Y.X. Mukherjee, S. Mukherjee. The boundary node method for potential problems. **International Journal for Numerical Methods in Engineering** 1997; 40(5): 797-815.

[4] S.N. Atluri, T. Zhu. A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics. **Computational Mechanics** 1998; 22(2): 117-127.

[5] _____, S. Shen. **The meshless Local Petrov-Galerkin in (MLPG) method**. Forsyth: Tech Science Press; 2002.

[6] V. Sladek, J. Sladek, S.N. Atluri, R. Van Keer. Numerical integration of singularities in meshless implementation of local boundary integral equations. **Computational Mechanics** 2000; 25(4): 394-403.

[7] M.A. Golberg, C.S. Chen, M. Ganesh. Particular solutions of 3D Helmholtz-type equations using compactly supported radial basis functions. **Engineering Analysis with Boundary Elements** 2000; 24(7-8): 539-547.

[8] W. Toutip. **The dual reciprocity boundary element method for linear and nonlinear** [Ph.D. Dissertation in applied mathematics]. Hertfordshire: University of Hertfordshire; 2001.

[9] R.N.L. Smith. Direct Gauss quadrature formulae for logarithmic singularities on isoparametric elements. **Journal of Engineering Analysis with Boundary Elements** 2000; 24(2): 161-167.

[10] C.A. .Brebbia. **The Boundary Element Method for Engineers**. London: Pentech Press; 1978

[11] _____, L.C. Wrobel. **Boundary Element Methods**. London: CRC Press; 1995.

[12] G.E. Fasshauer. **Meshfree approximation methods with MATLAB**. Hackensack (NJ): World Scientific Publishing; 2007.

[13] _____. Solving Partial Differential Equations by Collocation with Radial Basis Functions. In : A. Le M; chaut'e, C. Rabut, L. L. Schumaker, editors. **Surface Fitting and Multiresolution Methods**. Proceedings of the 3rd International Conference on Curves and Surfaces; 1996 June 27-July 3; Chamonix Mont Blanc. Nashville (TN): Vanderbilt University Press; 1997. p. 131-138.

[14] _____. **Meshfree methods: moving beyond the finite element method**. Boca Raton: CR press; 2007.

[15] Prem K. Kythe. **An introduction to boundary element methods**. Boca Raton (FL): CRC Press; 1995.

[16] R.D. Ciskawski, C.A. Brebbia. **Boundary element method in acoustics**. Southampton: Computational Mechanics Publications; 1991.

[17] G.R. Liu. **Meshfree methods Moving beyond the Finite Element Method**. London: CRC Press; 2002.

[18] J. Lucha. **The boundary element method using quadratic element for two-dimensional with MATLAB** [Master of Science Thesis in applied mathematics]. Khon Kaen: The Graduate School, Khon Kaen University; 2009. [in Thai].

[19] R. Franke. Scattered data interpolation: Tests of some methods. **Mathematics of Computation** 1982; 38(157): 181-200.

[20] E.J. Kansa. Multiquadrics-a scattered data approximation scheme with applications to computational fluid-dynamics-I: Surface approximations and partial derivative estimates. **Computers & Mathematics with Applications** 1990; 19(8-9): 127-145.

[21] E.J. Kansa. Multiquadrics-a scattered data approximation scheme with applications to computational fluid dynamics-II: Solutions to parabolic, hyperbolic and elliptic partial differential equations. **Computers & Mathematics with Applications** 1990; 19(8-9): 147-161.

[22] Y.C. Hon, R. schaback. On unsymmetric collocation by radial basis functions. **Applied Mathematics and Computation** 2001; 119(2-3): 177-186.

[23] H. Power, V. Barraco. Comparison analysis between unsymmetric and symmetric radial basis function collocation methods for the numerical solution of partial differential equations. **Computers & Mathematics with Applications** 2002; 43(3-5): 551-583.

[24] J. Li, A. H. D. Cheng, C. Cheri. A comparison of efficiency and error convergence of multiquadric collocation method and finite element method. **Engineering Analysis with Boundary Elements** 2003; 27(3): 251-257.

[25] M. Sharan, E. J. Kansa, S. Gupta. Application of the multiquadric method for numerical solution of elliptic partial differential equations. **Applied Mathematics and Computation** 1997; 84(2-3): 275-302.

[26] B. Sarler, R. Vertnik. Meshfree explicit local radial basis function collocation method for diffusion problems. **Computers & Mathematics with Applications** 2006; 51(8): 1269-1282.

[27] H. Wendlend. Meshless galerkin methods using radial basis functions. **Mathematics of Computation** 1999; 68(228): 1521-1531.

[28] E. Divo, A. J. Kassab. Efficient localized meshless modeling of natural convective viscous flows. In: R. Hogan, R. Amin, editors. **Computational Heat Transfer I**. 9th AIAA/ASME Joint Thermo physics and Heat Transfer Conference; 2006 June 5-8; San Francisco (CA). Reston (VA): AIAA; 2006. AIAA paper 2006-3089.

[29] _____. A meshless method for conjugate heat transfer problems. **Engineering Analysis with Boundary Elements** 2005; 29(2): 136-149.

[30] P. P. Chinchapatnam, K. Djidjeli, P. B. Nair. Meshless rbf collocation for steady incompressible viscous flows. In: K. Breuer, F. Liu, editors. **Low Speed Flows**. 36th AIAA Fluid Dynamics Conference and Exhibit; 2006 June 5-8; San Francisco (CA). Reston (VA): AIAA; 2006. AIAA paper 2006-3525.

[31] C. Shu, H. Ding, H. Q. Chen, T. G. Wang. An upwind local rbf-dq method for simulation of in viscid compressible flows. **Computer Methods in Applied Mechanics and Engineering** 2005; 194(18-20): 2001-2017.

[32] M.D. Buhmann. **Radial Basis Functions: Theory and Implementations**. Cambridge: Cambridge University Press; 2003.

[33] W. Chen, M. Tanaka. New insights into boundary-only and domain-type RBF methods. **International Journal of Nonlinear Sciences and Numerical Simulation** 2000; 1(3): 145-151.

[34] _____. Boundary knot method: A meshless, exponential convergence, integration-free, and boundary-only RBF technique. **Computers and Mathematics with Applications** 2002; 43: 379-391.

[35] M.A. Golberg, C.S. Chen. The method of fundamental solutions for potential, Helmholtz and diffusion problems. In: M.A. Golberg, editor. **Boundary Integral Methods: Numerical and Mathematical Aspects**; United Kingdom and Billerica. Southampton: Computational Mechanics Publications; 1999. p. 103-176.

[36] A.J. Katz. **Meshless methods for computational fluid dynamics** [Ph.D. Dissertation in aeronautics and astronautics]. Stanford: University of Stanford; 2009.

[37] S. Kaennakhum. **The dual reciprocity boundary element method for two-dimensional Burger's equations using MATLAB** [Master of Science Thesis in Mathematics]. Khon Kaen: The Graduate School, Khon Kaen University; 2004.

[38] O. Chanthawara. **The Dual Reciprocity Boundary Element Method for Two -Dimensional Biharmonic Equations on Irregular Domains Using MATLAB** [Master of Science Thesis in Mathematics]. Khon Kaen: The Graduate School, Khon Kaen University; 2006. [in Thai].

[39] A. Cruise, F. J. Rizzo. A direct formulation and numerical solution of the general transient elasto-dynamic problem. **Journal of Mathematical Analysis and Applications** 1968; 22: 244-259.

[40] M.A. Jaswon. Integral equation methods in potential theory I. **Proceedings of the Royal Society of London. Series A** 1963; 275(1360): 23-32.

[41] T. Symm. Integral equation methods in potential theory II. **Proceedings of the Royal Society of London. Series A** 1963; 275(1360): 33-46.

[42] J.H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. **Numerische Mathematik** 1960; 2(1): 84-90.

[43] T.T. Wong, W.S. Luk, P.A. Heng. Sampling with Hammersley and Halton Points. **Journal of Graphics Tools** 1997; 2(2): 9-24.

APPENDICES

Program (Meshless method)

Program M1

```
%=========================================================
%                  PROGRAM  KANSALAPLACE2D
%=========================================================
%---------------------------------------------------------------------------
%                  PROGRAM 1
% This is a program to solve the problem for example 5.1.1
%---------------------------------------------------------------------------
%---------------------------------------------------------------------------
%              IMQ RBF and its Laplacian
%---------------------------------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
%-------------------------------------------------------
%              its Laplacian for test problem
%-------------------------------------------------------
 Lu=@(x,y) zeros(size(x));
%-------------------------------------------------------
%              Number and type of collocation points
%-------------------------------------------------------
 N = 289; [collpts, N] = CreatePoints(N, 2, 'u');
 indx1 = find(collpts(:,2)==0 & collpts(:,1)~=1);
 indx2 = find(collpts(:,1)==1 & collpts(:,2)~=1);
 indx3 = find(collpts(:,2)==1 & collpts(:,1)~=0);
 indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
 bdydata = collpts([indx1;indx2;indx3;indx4],:);
%-------------------------------------------------------
%              Number of internal points
```

```
%-------------------------------------------------------------
[intdata, N] = CreatePoints(N, 2, 'u');
sn = sqrt(N); h = 1/(sn-1);
%-------------------------------------------------------------------------
%   Create boundary collocation points(outside the domain)
%-------------------------------------------------------------------------
 bdyctrs = bdydata; bdyctrs = (1+2*h)*bdyctrs-h;
 ctrs = [intdata; bdyctrs];
%-------------------------------------------------------------
%         Create the evaluation locations
%-------------------------------------------------------------
 M = 25; epoints = CreatePoints(M,2,'u');
%-------------------------------------------------------------
%         Compute the evaluation matrix
%-------------------------------------------------------------
 DM_eval = DistanceMatrix(epoints,ctrs);
 EM = rbf(ep,DM_eval);
%-------------------------------------------------------------
% Compute blocks for collocation matrix
%-------------------------------------------------------------
  DM_intdata = DistanceMatrix(intdata,ctrs);
  LCM = Lrbf(ep,DM_intdata);
  DM_bdydata = DistanceMatrix(bdydata,ctrs);
  BCM = rbf(ep,DM_bdydata);
  CM = [LCM; BCM];
%-------------------------------------------------------------
% Create right-hand side
%-------------------------------------------------------------
 rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(sn-1,1); ...
        zeros(sn-1,1); 1*ones(sn-1,1);...
        zeros(sn-1,1)];
```

```
%-------------------------------------------------------
% Compute RBF solution
%-------------------------------------------------------
 Pf = EM * (CM\rhs);
%-------------------------------------------------------
% show points(collocation points) in the domain
%-------------------------------------------------------
 figure
 hold on; plot(intdata(:,1),intdata(:,2),'bo');
 plot(bdydata(:,1),bdydata(:,2),'rx');
 plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
%=======================================================
%                        END
%=======================================================
```

Program M2

```
%=======================================================
%            KANSALAPLACE2D
%=======================================================
%-------------------------------------------------------
%            PROGRAM 2
%This is a program to solve problem for example 5.1.2
%-------------------------------------------------------
%-------------------------------------------------------
%        IMQ RBF and its Laplacian
%-------------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
%-------------------------------------------------------
%  Analytical solution and its Laplacian for test problem
%-------------------------------------------------------
```

```
u = @(x,y) x.^2-y.^2;
Lu=@(x,y) zeros(size(x));
%---------------------------------------------------------
%       Number and type of collocation points
%---------------------------------------------------------
N = 289; [collpts, N] = CreatePoints(N, 2, 'u');
indx1 = find(collpts(:,2)==0 & collpts(:,1)~=1);
indx2 = find(collpts(:,1)==1 & collpts(:,2)~=1);
indx3 = find(collpts(:,2)==1 & collpts(:,1)~=0);
indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
bdydata = collpts([indx1;indx2;indx3;indx4],:);
%---------------------------------------------------------
%       Number of internal points
%---------------------------------------------------------
[intdata, N] = CreatePoints(N, 2, 'u');
sn = sqrt(N); h = 1/(sn-1);
%-------------------------------------------------------------------------
%  Create boundary collocation points(outside the domain)
%-------------------------------------------------------------------------
bdyctrs = bdydata; bdyctrs = (1+2*h)*bdyctrs-h;
ctrs = [intdata; bdyctrs];

%---------------------------------------------------------
%       Create the evaluation locations
%---------------------------------------------------------
M = 25; epoints = CreatePoints(M,2,'u');
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);
%---------------------------------------------------------
%       The Exact solution
%---------------------------------------------------------
 exact = u(epoints(:,1),epoints(:,2));
```

```
%-------------------------------------------------------
%       Compute blocks for collocation matrix
%-------------------------------------------------------
DM_intdata = DistanceMatrix(intdata,ctrs);
LCM = Lrbf(ep,DM_intdata);
DM_bdydata = DistanceMatrix(bdydata,ctrs);
BCM = rbf(ep,DM_bdydata);
CM = [LCM; BCM];
%-------------------------------------------------------
%       Create right-hand side
%-------------------------------------------------------
 X=bdydata(1:sn-1,1);
 Y=bdydata(sn:2*sn-2,2);
 XX=bdydata(2*sn-1:3*sn-3,1);
 YY=bdydata(3*sn-2:end,2);
 rhs = [Lu(intdata(:,1),intdata(:,2)); X.^2; ...
     1-Y.^2; XX.^2-1;...
      -YY.^2];
%-------------------------------------------------------
% Compute RBF solution
%-------------------------------------------------------
Pf = EM*(CM\rhs);
%-------------------------------------------------------
% Compute maximum error on evaluation grid
%-------------------------------------------------------
maxerr = norm(Pf-exact,inf);
rms_err = norm(Pf-exact)/5;
fprintf('RMS error:     %e\n', rms_err)
fprintf('Maximum error: %e\n', maxerr)
%-------------------------------------------------------
% show points in the domain
```

```
%-------------------------------------------------------
figure
hold on; plot(intdata(:,1),intdata(:,2),'bo');
plot(bdydata(:,1),bdydata(:,2),'rx');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
%-------------------------------------------------------
%=======================================================================
%                                    END
%=======================================================================
```

Program M3

```
%=======================================================================
%                KANSALAPLACE2D
%=======================================================================

%-----------------------------------------------------------------
%                PROGRAM 3
% This is a program to solve problem for example 5.1.3
%-----------------------------------------------------------------

%-------------------------------------------------------
%          IMQ RBF and its Laplacian
%-------------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 1;
%dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);

%-----------------------------------------------------------------
%  Analytical solution and its Laplacian for test problem
%-----------------------------------------------------------------
```

```
u = @(x,y) x.^3-3*(x.*y.^2);
Lu = @(x,y) zeros(size(x));
%-------------------------------------------------------------
%Create boundary points
%-------------------------------------------------------------
N = 289; [collpts, N] = pointuniform(N, 2); %create boundary
indx1 = find(collpts(:,2)==0 & collpts(:,1)~=2);
indx2 = find(collpts(:,1)==2 & collpts(:,2)~=4);
indx3 = find(collpts(:,2)==4 & collpts(:,1)~=0);
indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
bdypts = collpts([indx1;indx2;indx3;indx4],:);

%-------------------------------------------------------------
%Create internal points
%-------------------------------------------------------------
 n=289;
 intpts=pointuniform(n,2);%create internal points

%-------------------------------------------------------------
%define boundary domain(outside)
%-------------------------------------------------------------
sn = sqrt(N);  h = 1/(sn-1);
%NN=size(pts);
%SNN=sqrt(NN(1,1));hh=1/(SNN-1);
%bdyctrs =bdypts;
bdyctrs =(1+h)*bdypts-1.5*h;
%bdyctrs = bdypts;  bdyctrs = (1+2*h)*bdyctrs;
ctrs = [intpts; bdyctrs];

%-------------------------------------------------------------
% Create evaluation locations
% in the unit square
```

```
%-------------------------------------------------------
 M = 289;  Node = CreatePoints4(M,2,'u');  %Node = 2*Node-2;
Nch=find(Node(:,1)<=2);
epoints=Node(Nch,:);

%-------------------------------------------------------
%Exact solution
%-------------------------------------------------------
exact = u(epoints(:,1),epoints(:,2));

%-------------------------------------------------------
% Compute blocks for collocation matrix
%-------------------------------------------------------
 DM_eval = DistanceMatrix(epoints,ctrs);
 EM = rbf(ep,DM_eval);
 DM_int = DistanceMatrix(intpts,ctrs);
 DM_bdy = DistanceMatrix(bdypts,ctrs);
LCM = Lrbf(ep,DM_int);
 BCM = rbf(ep,DM_bdy);
 CM=[LCM;BCM];

%-------------------------------------------------------
%Create right-hand side
%-------------------------------------------------------
x1=bdypts(1:sn-1,1);y1=bdypts(1:sn-1,2);
x2=bdypts(sn:2*sn-2,1);y2=bdypts(sn:2*sn-2,2);
x3=bdypts(2*sn-1:3*sn-3,1);y3=bdypts(2*sn-1:3*sn-3,2);
x4=bdypts(3*sn-2:end,1);y3=bdypts(3*sn-2:end,2);
rhs = [Lu(intpts(:,1),intpts(:,2)); x1.^3; ...
8-6*y2.^2; x3.^3-48*x3;zeros(sn-1,1)];

%-------------------------------------------------------
```

```
%Compute RBF solutions
%-------------------------------------------------------------
 Pf = EM * (CM\rhs);

%-------------------------------------------------------------
% Compute maximum error on evaluation grid
%-------------------------------------------------------------
  maxerr = norm(Pf-exact,inf);
  rms_err = norm(Pf-exact)/sqrt(M);
  fprintf('RMS error:    %e\n', rms_err)
  fprintf('Maximum error: %e\n', maxerr)

 %-------------------------------------------------------------
 %              show points in domain
 %-------------------------------------------------------------
 figure
 hold on
plot(intpts(:,1),intpts(:,2),'bo');
 plot(bdypts(:,1),bdypts(:,2),'rx');
 plot(bdyctrs(:,1),bdyctrs(:,2),'gx');
hold off
 %=============================================================
 %                        END
 %=============================================================
```

Program M4

```
%=============================================================
%                KANSALAPLACE2D
%=============================================================
%-------------------------------------------------------------
%                PROGRAM 4
```

```
%This is a program to solve the problem for example 5.1.4
%-------------------------------------------------------------------
%-----------------------------------------------------
%            IMQ RBF and its Laplacian
%-----------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);

%-------------------------------------------------------------------
%  Analytical solution and its Laplacian for test problem
%-------------------------------------------------------------------
u=@(x,y) x.^2+y.*(1-y);
Lu=@(x,y) zeros(size(x));

%-----------------------------------------------------
%Create boundary points
%-----------------------------------------------------
N = 289; [collpts, N] = CreatePoints(N, 2, 'u');
indx1 = find(collpts(:,2)==0 & collpts(:,1)~=1);
indx2 = find(collpts(:,1)==1 & collpts(:,2)~=1);
indx3 = find(collpts(:,2)==1 & collpts(:,1)~=0);
indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
bdydata = collpts([indx1;indx2;indx3;indx4],:);%boundary domain
%-----------------------------------------------------
% Create internal points
%-----------------------------------------------------
[intdata, N] = CreatePoints(N, 2, 'h');
sn = sqrt(N); h = 1/(sn-1);
bdyctrs = bdydata; bdyctrs = (1+2*h)*bdyctrs-h;% boundary domain (outside)
ctrs = [intdata; bdyctrs];
```

```
%-------------------------------------------------------------
%Create evaluation locations
%-------------------------------------------------------------
M = 36; epoints = CreatePoints(M,2,'u');
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);

%-------------------------------------------------------------
%        Exact solution
%-------------------------------------------------------------
 exact = u(epoints(:,1),epoints(:,2));

%-------------------------------------------------------------
% Compute blocks for collocation matrix
%-------------------------------------------------------------
DM_intdata = DistanceMatrix(intdata,ctrs);
LCM = Lrbf(ep,DM_intdata);
DM_bdydata = DistanceMatrix(bdydata,ctrs);
BCM = rbf(ep,DM_bdydata);
CM = [LCM; BCM];

%-------------------------------------------------------------
% Create right-hand side
%-------------------------------------------------------------
 X=bdydata(1:sn-1,1);
 Y=bdydata(sn:2*sn-2,2);
 XX=bdydata(2*sn-1:3*sn-3,1);
 YY=bdydata(3*sn-2:end,2);
 rhs = [Lu(intdata(:,1),intdata(:,2)); X.^2; ...
    1+Y.*(1-Y); XX.^2;...
```

```
    YY.*(1-YY)];

%-------------------------------------------------------------
% Compute RBF solution
%-------------------------------------------------------------
Pf = EM*(CM\rhs);

%-------------------------------------------------------------------
% Compute maximum error on evaluation grid
%-------------------------------------------------------------------
maxerr = norm(Pf-exact,inf);
rms_err =norm(Pf-exact)/sqrt(M);
fprintf('RMS error:    %e\n', rms_err)
fprintf('Maximum error: %e\n', maxerr)

%-------------------------------------------------------------
%show poits in domain
%-------------------------------------------------------------
figure
hold on; plot(intdata(:,1),intdata(:,2),'bo');
plot(bdydata(:,1),bdydata(:,2),'rx');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off

%=============================================================
%                    END
%=============================================================
```

Program M5

```
%==========================================================
%                  KANSALAPLACE2D
%==========================================================

%----------------------------------------------------------
%                  PROGRAM 5
%This is a program to solve problem for example 5.1.5
%----------------------------------------------------------

%----------------------------------------------------------
%            IMQ RBF and its Laplacian
%----------------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 0.5;
dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);

%----------------------------------------------------------
%   Analytical solution and its Laplacian for test problem
%----------------------------------------------------------
u = @(x,y) 300-50*x;
Lu = @(x,y) zeros(size(x));

%----------------------------------------------------------
%Create boundary points
%----------------------------------------------------------
N = 325; [collpts, N] = CreatePoints611(N, 2, 'u');
indx1 = find(collpts(:,2)==0 & collpts(:,1)~=6);
indx2 = find(collpts(:,1)==6 & collpts(:,2)~=6);
```

```
indx3 = find(collpts(:,2)==6 & collpts(:,1)~=0);
indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
bdypts = collpts([indx1;indx2;indx3;indx4],:);

%---------------------------------------------------------
% Create internal points
%---------------------------------------------------------
[intpts, N] = CreatePoints611(N, 2, 'h');
sn = sqrt(N); h = 6/(sn-1);
bdyctrs = bdypts; bdyctrs =(1+h)*bdyctrs-3*h;
ctrs = [intpts; bdyctrs];

%---------------------------------------------------------
%Create evaluation locations
%---------------------------------------------------------
M = 16; epoints = CreatePoints611(M,2,'u');
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);

%---------------------------------------------------------
%       Exact solution
%---------------------------------------------------------
exact = u(epoints(:,1),epoints(:,2));

%---------------------------------------------------------
% Compute blocks for collocation matrix
%---------------------------------------------------------
DM_int = DistanceMatrix(intpts,ctrs);
DM_bdy = DistanceMatrix(bdypts,ctrs);
dy_bdy = Differencematrix(bdypts(:,2),ctrs(:,2));
LCM = Lrbf(ep,DM_int);
BCM=rbf(ep,DM_bdy);
```

```
CM = [LCM; BCM];

 %-------------------------------------------------------
 % Create right-hand side
 %-------------------------------------------------------
 X=bdypts(1:sn-1,1);
 XX=bdypts(2*sn-1:3*sn-3,1);
 rhs = [Lu(intpts(:,1),intpts(:,2));300-50*X; ...
zeros(sn-1,1);300-50*XX ;300*ones(sn-1,1)];

 %-------------------------------------------------------
 % Compute RBF solution
 %-------------------------------------------------------
 Pf = EM * (CM\rhs);

%-------------------------------------------------------------
% Compute maximum error on evaluation grid
%-------------------------------------------------------------
  maxerr = norm(Pf-exact,inf);
  rms_err =norm(Pf-exact)/sqrt(M);
  fprintf('RMS error:     %e\n', rms_err)
  fprintf('Maximum error: %e\n', maxerr)
%-----------------------------------------------------
%show poits in domain
%-----------------------------------------------------
 figure
 hold on;
 plot(intpts(:,1),intpts(:,2),'bo');
 plot(bdypts(:,1),bdypts(:,2),'rx');
 plot(bdyctrs(:,1),bdyctrs(:,2),'gx');
 hold off
 %=====================================================
```

```
%                    END
%=================================================================
```

Program M6

```
%=================================================================
%                KANSALAPLACE2D
%=================================================================

%-----------------------------------------------------------
%                PROGRAM 6
%This is a program to solve prblem for example 5.2.1
%-----------------------------------------------------------

%-----------------------------------------------------
%          IMQ RBF and its Laplacian
%-----------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
 Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);

%-------------------------------------------------------------
%   Analytical solution and its Laplacian for test problem
%-------------------------------------------------------------
 u = @(x,y) 1+x+2*y;
 Lu = @(x,y) zeros(size(x));

 %-----------------------------------------------------
 %Create boundary points
 %-----------------------------------------------------
 N = 289; [collpts, N] = CreatePoints(N, 2, 'u');
 indx1 = find(collpts(:,2)==0 & collpts(:,1)~=1);
 indx2 = find(collpts(:,1)==1 & collpts(:,2)~=1);
```

```
indx3 = find(collpts(:,2)==1 & collpts(:,1)~=0);
indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
bdypts = collpts([indx1;indx2;indx3;indx4],:);

%---------------------------------------------------------
% Create internal points
%---------------------------------------------------------
[intpts, N] = CreatePoints(N, 2, 'u');
sn = sqrt(N); h = 1/(sn-1);
bdyctrs = bdypts; bdyctrs = (1+2*h)*bdyctrs-h;
ctrs = [intpts; bdyctrs];

%---------------------------------------------------------
%Create evaluation locations
%---------------------------------------------------------
M = 25; epoints = CreatePoints(M,2,'u');
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);

%---------------------------------------------------------
%       Exact solution
%---------------------------------------------------------
exact = u(epoints(:,1),epoints(:,2));

 %---------------------------------------------------------
 % Compute blocks for collocation matrix
 %---------------------------------------------------------
DM_int = DistanceMatrix(intpts,ctrs);
DM_bdy = DistanceMatrix(bdypts,ctrs);
dy_bdy = Differencematrix(bdypts(:,2),ctrs(:,2));
LCM = Lrbf(ep,DM_int);
BCM1 = -dyrbf(ep,DM_bdy(1:sn-1,:),dy_bdy(1:sn-1,:));
```

```
BCM2 = rbf(ep,DM_bdy(sn:2*sn-2,:));
BCM3 = dyrbf(ep,DM_bdy(2*sn-1:3*sn-3,:),...
dy_bdy(2*sn-1:3*sn-3,:));
BCM4 = rbf(ep,DM_bdy(3*sn-2:end,:));
CM = [LCM; BCM1; BCM2; BCM3; BCM4];

 %-------------------------------------------------------------
 % Create right-hand side
 %-------------------------------------------------------------
 Y=bdypts(sn:2*sn-2,2);
 YY=bdypts(3*sn-2:end,2);
 rhs = [Lu(intpts(:,1),intpts(:,2)); -2*ones(sn-1,1); ...
 2+Y.*2; 2*ones(sn-1,1); 1+YY.*2];

 %-------------------------------------------------------------
 % Compute RBF solution
 %-------------------------------------------------------------
Pf = EM * (CM\rhs);

%-----------------------------------------------------------------
% Compute maximum error on evaluation grid
%-----------------------------------------------------------------
  maxerr = norm(Pf-exact,inf);
  rms_err =norm(Pf-exact)/sqrt(M);
  fprintf('RMS error:     %e\n', rms_err)
  fprintf('Maximum error: %e\n', maxerr)

%-------------------------------------------------------------
%show poits in domain
%-------------------------------------------------------------
 figure
 hold on;
```

```
plot(intpts(:,1),intpts(:,2),'bo');
plot(bdypts(:,1),bdypts(:,2),'ro');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx');
hold off
figure
hold on ;
plot(Pf,'r-o');
plot(exact,'b.');
axis([0 25 1 4]);
hold off
%=========================================================================
%                        END
%=========================================================================
```

Program M7

```
%=========================================================================
%                   KANSALAPLACE2D
%=========================================================================

%-------------------------------------------------------------------
%                   PROGRAM 7
%This is a promgram to solve problem for example 5.2.2
%-------------------------------------------------------------------

%-----------------------------------------------------
%            IMQ RBF and its Laplacian
%-----------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 1.5;
 dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
```

```
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);

%-------------------------------------------------------------------
%  Analytical solution and its Laplacian for test problem
%-------------------------------------------------------------------
u = @(x,y) 300-50*x;
Lu = @(x,y) zeros(size(x));

 %---------------------------------------------------------
%Create boundary points
%---------------------------------------------------------
N = 289; [collpts, N] = CreatePoints6(N, 2, 'u');
indx1 = find(collpts(:,2)==0 & collpts(:,1)~=6);
indx2 = find(collpts(:,1)==6 & collpts(:,2)~=6);
indx3 = find(collpts(:,2)==6 & collpts(:,1)~=0);
indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
bdypts = collpts([indx1;indx2;indx3;indx4],:);

 %---------------------------------------------------------
% Create internal points
%---------------------------------------------------------
[intpts, N] = CreatePoints6(N, 2, 'u');
sn = sqrt(N); h = 6/(sn-1);
bdyctrs = bdypts; bdyctrs =(1+h)*bdyctrs-3*h;
ctrs = [intpts; bdyctrs];

 %---------------------------------------------------------
%Create evaluation locations
%---------------------------------------------------------
M = 16; epoints = CreatePoints6(M,2,'u');
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);
```

```
%-------------------------------------------------------------
%       Exact solution
%-------------------------------------------------------------
exact = u(epoints(:,1),epoints(:,2));

%-------------------------------------------------------------
% Compute blocks for collocation matrix
%-------------------------------------------------------------
DM_int = DistanceMatrix(intpts,ctrs);
DM_bdy = DistanceMatrix(bdypts,ctrs);
dy_bdy = Differencematrix(bdypts(:,2),ctrs(:,2));
LCM = Lrbf(ep,DM_int);
BCM1 = dyrbf(ep,DM_bdy(1:sn-1,:),dy_bdy(1:sn-1,:));
BCM2 = rbf(ep,DM_bdy(sn:2*sn-2,:));
BCM3 = dyrbf(ep,DM_bdy(2*sn-1:3*sn-3,:),...
dy_bdy(2*sn-1:3*sn-3,:));
BCM4 = rbf(ep,DM_bdy(3*sn-2:end,:));
CM = [LCM; BCM1; BCM2; BCM3; BCM4];

 %-------------------------------------------------------------
 % Create right-hand side
 %-------------------------------------------------------------
 Y=bdypts(sn:2*sn-2,2);
 YY=bdypts(3*sn-2:end,2);
rhs = [Lu(intpts(:,1),intpts(:,2)); zeros(sn-1,1); ...
zeros(sn-1,1); zeros(sn-1,1); 300*ones(sn-1,1)];

 %-------------------------------------------------------------
 % Compute RBF solution
 %-------------------------------------------------------------
```

```
Pf = EM .* (CM\rhs);

%-------------------------------------------------------------
% Compute maximum error on evaluation grid
%-------------------------------------------------------------
 maxerr = norm(Pf-exact,inf);
 rms_err =norm(Pf-exact)/sqrt(M);
 fprintf('RMS error:     %e\n', rms_err)
 fprintf('Maximum error: %e\n', maxerr)

%-------------------------------------------------------------
%show poits in domain
%-------------------------------------------------------------
 figure
 hold on;
 plot(intpts(:,1),intpts(:,2),'bo');
 plot(bdypts(:,1),bdypts(:,2),'rx');
 plot(bdyctrs(:,1),bdyctrs(:,2),'gx');
 hold off
 %============================================================
%                      END
 %============================================================
```

Program M8

```
%============================================================
%                KANSALAPLACE2D
%============================================================

%-------------------------------------------------------------
%                PROGRAM 8
%This is a program to solve the problem for example 5.2.3
```

```
%-------------------------------------------------------------------

%-------------------------------------------------------------
%              IMQ RBF and its Laplacian
%-------------------------------------------------------------
clear;
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 2.6;
dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
 Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);

%-------------------------------------------------------------
%    its Laplacian for test problem
%-------------------------------------------------------------
 Lu=@(x,y) zeros(size(x));

 %-------------------------------------------------------------
 %Create boundary points
 %-------------------------------------------------------------
  N =400 ; [collpts, N] = CreatePoints8(N, 2, 'u');
 indx1 = find(collpts(:,2)==0 & collpts(:,1)~=4&collpts(:,1)<=3);
 indx2 = find(collpts(:,1)==4 & collpts(:,2)~=2&collpts(:,2)>=1);
 indx3 = find(collpts(:,2)==2 & collpts(:,1)~=0);
 indx4 = find(collpts(:,1)==0 & collpts(:,2)~=0);
 %bdydata = collpts([indx1;indx2;indx3;indx4],:);
 p=find(collpts(:,1)>=3&collpts(:,2)<=1);
 pp=collpts(p,:);
n=17;
 r=1;
 XL=createX(r,n)+4;
 YL=createY(r,n);
 bdypts=[XL YL];
  c=find(bdypts(:,1)>=3&bdypts(:,2)<=1&bdypts(:,1)<=4&bdypts(:,2)>=0);
```

```
cc=bdypts(c,:);
bdydata=[collpts(indx1,:);cc;collpts(indx2,:);collpts(indx3,:);collpts(indx4,:)];
%-------------------------------------------------------------
% Create internal points
%-------------------------------------------------------------
[node, N] = CreatePoints8(N, 2, 'u');
Nd=find(node(:,1)<=3 | node(:,2)>=1);
intdata=node(Nd,:);
sn = sqrt(N); h = 1/(sn-1);
bdyctrs = bdydata; bdyctrs = (1+h)*bdyctrs-h;
ctrs = [intdata; bdyctrs];

%-------------------------------------------------------------
%Create evaluation locations
%-------------------------------------------------------------
M = 289;  Point= CreatePoints8(M,2,'u');
NP=find(Point(:,1)<=3 | Point(:,2)>=1);
epoints=Point(NP,:);

%-------------------------------------------------------------
% Compute blocks for collocation matrix
%-------------------------------------------------------------
 DM_eval = DistanceMatrix(epoints,ctrs);
 EM = rbf(ep,DM_eval);
 DM_intdata = DistanceMatrix(intdata,ctrs);
 LCM = Lrbf(ep,DM_intdata);
 DM_bdydata = DistanceMatrix(bdydata,ctrs);
 dy_bdy = Differencematrix(bdydata(:,2),ctrs(:,2));
 a=size(indx1);
 b=size(cc);
 c=size(indx2);
 d=size(indx3);
```

```
f=size(indx4);
t=a(1,1)+b(1,1)+c(1,1)+d(1,1)+f(1,1);

BCM1=rbf(ep,DM_bdydata(1:a(1,1),:));
BCM2=rbf(ep,DM_bdydata(a(1,1)+1:a(1,1)+b(1,1),:));
BCM3=dyrbf(ep,DM_bdydata(a(1,1)+b(1,1)+1:a(1,1)+b(1,1)+c(1,1),:),
dy_bdy(a(1,1)+b(1,1)+1:a(1,1)+b(1,1)+c(1,1),:));
BCM4=rbf(ep,DM_bdydata(a(1,1)+b(1,1)+c(1,1)+1:a(1,1)+b(1,1)+c(1,1)+d(1,1),:));
BCM5=rbf(ep,DM_bdydata(a(1,1)+b(1,1)+c(1,1)+d(1,1)+1:t,:));
BCM=[BCM1;BCM2;BCM3;BCM4;BCM5];
CM = [LCM; BCM];

%-------------------------------------------------------------
% Create right-hand side
%-------------------------------------------------------------
 Y=bdydata(a(1,1)+b(1,1)+c(1,1)+d(1,1)+1:t,2);
rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(a(1,1),1); ...
      zeros(b(1,1),1); zeros(c(1,1),1);2*ones(d(1,1),1);Y];

%-------------------------------------------------------------
% Compute RBF solution
%-------------------------------------------------------------
 Pf = EM*(CM\rhs);

%-------------------------------------------------------------
%show poits in domain
%-------------------------------------------------------------
 figure
 hold on; plot(intdata(:,1),intdata(:,2),'bx');
 plot(bdydata(:,1),bdydata(:,2),'rx');
 plot(bdyctrs(:,1),bdyctrs(:,2),'gx');
 plot(epoints(:,1),epoints(:,2),'go');
```

```
hold off
figure;
hold on
plot(Point(:,1),Point(:,2),'rx');
plot(epoints(:,1),epoints(:,2),'bo');
hold off
%=========================================================
%                    END
%=========================================================
```

Program M9

```
%=========================================================
%                KANSALAPLACE2D
%=========================================================

%-------------------------------------------------------------------------
%                PROGRAM 9
%This is a program to solve the Laplace equation for example 5.2.4
%-------------------------------------------------------------------------

%-----------------------------------------------------
%          IMQ RBF and its Laplacian
%-----------------------------------------------------
clear;
 % Gaussian RBF and its derivatives
 %rbf = @(e,r) exp(-(e*r).^2); ep = 3;
 %dyrbf = @(e,r,dy) -2*dy*e^2.*exp(-(e*r).^2);
 %Lrbf = @(e,r) 4*e^2*exp(-(e*r).^2).*((e*r).^2-1);
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 1.5;
dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
```

```
%-----------------------------------------------------------
%    its Laplacian for test problem
%-----------------------------------------------------------
 Lu = @(x,y) zeros(size(x));

 %-----------------------------------------------------------
 %Create boundary points
 %-----------------------------------------------------------
 N =100 ; [collpts, N] = CreatePoints12(N, 2, 'u');
indx1 = find(collpts(:,2)==0 & collpts(:,1)<2);
 indx2 = find(collpts(:,2)>=0 & collpts(:,1)==0);
 p=collpts(indx1,:);
 pp=collpts(indx2,:);

%******points are choosen for approximation*******
 n=36;
 a=2;
 b=1;
XL=createXE(a,n);
YL=createYE(b,n);
b=[XL YL];
bb=find(b(:,1)>=0&b(:,2)>=0);
bbb=b(bb,:);
bdypts=[p;bbb;pp];
%-----------------------------------------------
%Create internal points
%-----------------------------------------------
 [intpts, N] = CreatePoints12(N, 2, 'u');
 CI=((intpts(:,1)).^2)/4+((intpts(:,2)).^2)/1;
 ip=find(CI<=1);
```

```
pts=intpts(ip,:);
intpts=pts;intctrs=pts;
%-----------------------------------------------
%define oudside of domain
%-----------------------------------------------
sn = sqrt(N);  h = 1/(sn-1);
NN=size(pts);
SNN=sqrt(NN(1,1));hh=2/(SNN-1);
%bdyctrs =bdypts;
bdyctrs =(1+hh)*bdypts-0.5*hh;
%bdyctrs = bdypts;  bdyctrs = (1+2*h)*bdyctrs;
ctrs = [intpts; bdyctrs];

%-------------------------------------------------------------
%Create evaluation locations
%-------------------------------------------------------------
M = 1089;  Node = CreatePoints12(M,2,'u');  %Node = 2*Node-2;
pch=((Node(:,1)).^2)/4+((Node(:,2)).^2)/1;
PP1=find(pch<=1);
epoints=Node(PP1,:);

%-------------------------------------------------------------
% Compute blocks for collocation matrix
%-------------------------------------------------------------
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);
DM_int = DistanceMatrix(intpts,ctrs);
DM_bdy = DistanceMatrix(bdypts,ctrs);
dy_bdy = Differencematrix(bdypts(:,2),ctrs(:,2));
LCM = Lrbf(ep,DM_int);
b1=size(p);
b2=size(bbb);
```

```
b3=size(pp);
b4=b1(1,1)+b2(1,1);
b5=b(1,1)+b4;
BCM1 = dyrbf(ep,DM_bdy(1:b1,:),dy_bdy(1:b1,:));
BCM2 = rbf(ep,DM_bdy(b1(1,1)+1:b4,:));
BCM3 = dyrbf(ep,DM_bdy(b4+1:end,:),...
dy_bdy(b4+1:end,:));
CM = [LCM; BCM1; BCM2; BCM3];

%-------------------------------------------------------
% Create right-hand side
%-------------------------------------------------------
x1=bdypts(b1(1,1)+1:b4,1);y1=bdypts(b1(1,1)+1:b4,2);
rhs = [Lu(intpts(:,1),intpts(:,2)); zeros(b1(1,1),1); ...
(x1.^2+y1.^2)/2; zeros(b3(1,1),1)];

%-------------------------------------------------------
% Compute RBF solution
%-------------------------------------------------------
Pf = EM * (CM\rhs);

%-------------------------------------------------------
%show poits in domain
%-------------------------------------------------------
figure
%hold on;
%plot(bdypts(:,1),bdypts(:,2),'ro');
%hold off
figure
hold on
plot(intpts(:,1),intpts(:,2),'bo');
plot(bdypts(:,1),bdypts(:,2),'ro');
```

```
plot(bdyctrs(:,1),bdyctrs(:,2),'gx');
% plot(x1(:,1),y1(:,1),'bx');
 %plot(px,py,'bo');
 %axis([0 6 0 6]);
 %plot(xx,'bo')
 hold off
%=========================================================
%                    END PROGRAM
%=========================================================
```

Subroutines (Meshless method)


```
%-----------------------------------------------------
function [points, N] = CreatePoints(N,s,gridtype)
%-----------------------------------------------------
% Computes a set of N points in [0,1]^s
% Inputs:
% N: number of interpolation points
% s: space dimension
% gridtype:
%   'h'=Halton,
%   'u'=uniform grid
% Outputs:
% points: an Nxs matrix (each row contains one s-D point)
% N: might be slightly less than original N for
%   Chebyshev and gridded uniform points
%-----------------------------------------------------
switch gridtype
   case 'h'
      points = haltonseq(N,s);
   case 'u'
      ppd = zeros(1,s);
```

```
    for j=1:s
      ppd(j) = floor(nthroot(N,s+1-j));
      N = N/ppd(j);
    end
    points = gridsamp([zeros(1,s); ones(1,s)], ppd);
    N = prod(ppd);
  otherwise
    error('Please use h or u data types')
end
%-------------------------------------------------------------

%/////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------
function DM = DistanceMatrix(dsites,ctrs)
%-------------------------------------------------------------
%Inputs
%dsites: Mxs matrix representing a set of M data sites in R^s
%ctrs: Nxs matrix representing a set of N centers in R^s
%Output
%DM: MxN matrix whose i,j position contains the Euclidean distance between
%the i-th data site and j-th center
%-------------------------------------------------------------------
[M,s] = size(dsites); [N,s] = size(ctrs);
DM = zeros(M,N);
for d=1:s
[dr,cc] = ndgrid(dsites(:,d),ctrs(:,d));
DM = DM + (dr-cc).^2;
end
DM = sqrt(DM);
%-------------------------------------------------------------
```

```
%////////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------
function [points,N]=pointuniform(N,s)
%Input
%N: number of points
%s: s-dimension
ppd = zeros(1,s);
      for j=1:s
         ppd(j) = floor(nthroot(N,s+1-j));
         N = N/ppd(j);
      end
      points = gridsamp([zeros(1,s); 2*ones(1,1) 4*ones(1,1)], ppd);
      N = prod(ppd);
%-------------------------------------------------------------

%////////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------
function [points, N] = CreatePoints4(N,s,gridtype)
% Computes a set of N points in [0,1]^s
% Inputs:
% N: number of interpolation points
% s: space dimension
% gridtype:
%    'h'=Halton
%    'u'=uniform grid
% Outputs:
% points: an Nxs matrix (each row contains one s-D point)
%-------------------------------------------------------------
switch gridtype
   case 'h'
```

```
        points = haltonseq2(N,s);
    case 'u'
        ppd = zeros(1,s);
        for j=1:s
            ppd(j) = floor(nthroot(N,s+1-j));
            N = N/ppd(j);
        end
        points = gridsamp([zeros(1,s); 4*ones(1,s)], ppd);
        N = prod(ppd);
    otherwise
        error('Please use h or u data types')
end
%------------------------------------------------------------

 %////////////////////////////////////////////////////////////////////////////////////////
%--------------------------------------------------------
% DM(j,k) = datacoord_j - centercoord_k .
 function DM = DifferenceMatrix(datacoord,centercoord)
 % The ndgrid command produces two MxN matrices:
 %   dr, consisting of N identical columns
 %       (each containing the M data sites)
 %   cc, consisting of M identical rows
 %       (each containing the N centers)
 [dr,cc] = ndgrid(datacoord(:),centercoord(:));
 DM = dr-cc;
%--------------------------------------------------------

%////////////////////////////////////////////////////////////////////////////////////////
%--------------------------------------------------------
function H = haltonseq(NUMPTS,NDIMS)
%HALTONSEQ(NUMPTS,NDIMS,) Generate a Halton sequence in NDIMS
dimensional space
```

```
% containing NUMPTS. The output is between 0 and 1. NUMPTS may be a vector
% of integers in which they indicate which elements of the Halton sequence to
% compute.
%-------------------------------------------------------
if (NDIMS < 12)
   P = [2 3 5 7 11 13 17 19 23 29 31];
else
   P = primes(1.3*NDIMS*log(NDIMS));
   P = P(1:NDIMS);
end
if isequal(size(NUMPTS),[1 1])
   int_pts = [1:NUMPTS];
else %User has put in the points to sample.
   int_pts = NUMPTS;
   NUMPTS = length(int_pts);
end
H = zeros(NUMPTS,NDIMS);
for i = 1:NDIMS %Generate the components for each dimension.
   V = fliplr(dec2bigbase(int_pts,P(i)));
   pows = -repmat([1:size(V,2)],size(V,1),1);
   H(:,i) = sum(V.*(P(i).^pows),2);
end
%-------------------------------------------------------
```

Program (BEM)

Program B1

```
%==========================================================================
%                    PROGRAM LINBEMS
%==========================================================================

%--------------------------------------------------------------------------
%This is a program to solve the Laplace equation for example 5.1.1 and 5.2.2
%for the case of problem with following conditions:
% 1. Consider the Laplace's equation with boundary condition
%(Neumann and Dirichlet conditions)
% 2. Linear element with square domain
%( 1x2 ) ,we let all of the point at the corners be boundary
%nodes so, the number of boundary node which will be entered
%must be in multiple of 4 form.(4,8,12,16,... )
% 3. We use 3-point Gaussian quadrature method to compute
%integration values
%--------------------------------------------------------------------------

%-----------------------------------------------------------------
%      Set ep(g) and w(g) values for 3-point Gaussian quadrature
%-----------------------------------------------------------------
clear
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];
%ep=[-0.86113631,-0.33998104,0.33998104,0.86113631];
%w=[0.34785485,0.65214515,0.65214515,0.34785485];

%-----------------------------------------------------------------
```

```
%          Input data and compute coordinate matrix
%-------------------------------------------------------------------
disp('=============================================================')
disp('              Welcome to LINBEMS program                     ')
disp('=============================================================')
disp('-------------------------------------------------------------')
disp('         Please choose the kind of boundary problem          ')
disp(' choose: ')
disp('         1 the Dirichlet problem                             ')
disp('         2 for the Nuemann problem                           ')
disp('         3 for the mixed problem1                            ')
disp('         4 for the mixed problem2                            ')
disp('-------------------------------------------------------------')
fprintf('\n')
kk=input('You choose number : ');
pp=input('Enter the length of x: ');
rr=input('Enter the length of y: ');
fprintf('\n')
n=input('Enter the number of boundary node: ');
disp('------------------------------------------------------------- ')
fprintf('\n\n')

%-------------------------------------------------------------------
%          we obtain the coordinate of boundary nodes
%-------------------------------------------------------------------
X=coorx(n,pp);%coordinate of boundary nodes
Y=coory(n,rr);%coordinate of boundary nodes

%-------------------------------------------------------------------
%          shows the coordinate of boundary nodes
%-------------------------------------------------------------------
nod=1:1:n;
```

```
node=(nod');
for i=1:n
solu(i,1)=node(i,1);
solu(i,2)=X(i,1);
solu(i,3)=Y(i,1);
%solu(i,4)=U(i,1);
%solu(i,5)=Q(i,1);
end
disp('-----------------------------------------------------------------')
disp('          coordinate of boundary nodes                ')
disp('-----------------------------------------------------------------')
disp('      node        x        y        ')
disp(solu)
fprintf('\n\n')

%-----------------------------------------------------------------------
%     Construct distance [d(i,j)] from a node i to an element j
%     and L(j) is a lenght of element j
%-----------------------------------------------------------------------
d=dist(n,X,Y);
L=lenet(n,X,Y);

%-----------------------------------------------------------------------
%           Construct the matrix H and G
%          for solving equation system HU=GQ
%-----------------------------------------------------------------------
Hhat=matrixh(n,X,Y,d,ep,w,L); % This is the matrix H-hat
G=matrixg(n,X,Y,d,ep,w,L); % This is the matrix G

%-----------------------------------------------------------------------
%     Input known values of U and Q on boundary of square
%-----------------------------------------------------------------------
```

```
if kk==1
nvl1=input('Enter value of U on lower side of boundary of square:');
nvl2=input('Enter value of U on right side of boundary of square:');
nvl3=input('Enter value of U on upper side of boundary of square:');
nvl4=input('Enter value of U on left side of boundary of square:');
elseif kk==2
nvl1=input('Enter value of Q on lower side of boundary of square:');
nvl2=input('Enter value of Q on right side of boundary of square:');
nvl3=input('Enter value of Q on upper side of boundary of square:');
nvl4=input('Enter value of Q on left side of boundary of square:');
elseif kk==3
nvl1=input('Enter value of U on lower side of boundary of square:');
nvl2=input('Enter value of Q on right side of boundary of square:');
nvl3=input('Enter value of U on upper side of boundary of square:');
nvl4=input('Enter value of Q on left side of boundary of square:');
else
nvl1=input('Enter value of Q on lower side of boundary of square:');
nvl2=input('Enter value of U on right side of boundary of square:');
nvl3=input('Enter value of Q on upper side of boundary of square:');
nvl4=input('Enter value of U on left side of boundary of square:');
end
N=NNcode1(kk,n);
t=NNcodv1(nvl1,nvl2,nvl3,nvl4,n,kk);

%will be used in computing B of Ax=B steps
%-------------------------------------------------------------------------
%        Change an system of equation to be in formAx=B
%       and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------------
A=cheqlL(kk,N,Hhat,G,n);
b=cheqrR(kk,N,Hhat,G,n);
B=b*t;
```

```
XX=(inv(A))*B; % XX is x in Ax=B

%-------------------------------------------------------------------
%       Show the matrix U and Q (first, construct U and Q
%       by alternating the value in above matrices)
%-------------------------------------------------------------------
U=matrixuU2(kk,t,n,XX);
Q=matrixqQ2(kk,t,n,XX);
nod=1:1:n;
node=(nod');
for i=1:n
solu(i,1)=node(i,1);
solu(i,2)=X(i,1);
solu(i,3)=Y(i,1);
solu(i,4)=U(i,1);
solu(i,5)=Q(i,1);
end
solu;
fprintf('\n\n')
disp('-------------------------------------------------------------')
disp('              The system has been solve                      ')
disp('-------------------------------------------------------------')
fprintf('\n')
disp('The result is ')
fprintf('\n')
disp('-------------------------------------------------------------')
disp('       node        X         Y        U         Q')
disp('-------------------------------------------------------------')
disp(solu)


%-------------------------------------------------------------------
```

```
%       Next, we start to compute for internal nodes
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%          step 1 : input the data to compute
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-------------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n\n')
disp('-------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
Xin=Xin1';
Yin=Yin1';

%-------------------------------------------------------------------
%          step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
din=distin(n,n1,Xin,Yin,X,Y);
H2=matrixh2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixg2(n,n1,X,Y,Xin,Yin,din,ep,w,L);

%-------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
```

```
%-------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));
%-------------------------------------------------------------------------------
%    construct matrix UU, XX1,YY1 for compare with analytic solution
%-------------------------------------------------------------------------------
M=n+n1;
UU=zeros(M,1);
XX1=zeros(M,1);
YY1=zeros(M,1);
An=zeros(M,1);
UU(1:n)=U(1:n);
 XX1(1:n)=X(1:n);
 YY1(1:n)=Y(1:n);
 UU(n+1:M)=Usol(1:n1);
 XX1(n+1:M)=Xin(1:n1);
 YY1(n+1:M)=Yin(1:n1);

%-------------------------------------------------------------------
%                The analytic solution
%-------------------------------------------------------------------
for i=1:M
 An(i)=50*YY1(i);
end

%-------------------------------------------------------------------
%              The solution are shown (internal node)
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-----------------------------------------------------------')
disp('   The solutions are shown as follows (internal node)          ')
disp('-----------------------------------------------------------')
for i=1:n1
```

```
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')

%-------------------------------------------------------------------
%              Compare numerical solution with analytic solution
%-------------------------------------------------------------------
nod1=1:1:M;
node=(nod1');
for i=1:M
solu1(i,1)=node(i,1);
solu1(i,2)=XX1(i,1);
solu1(i,3)=YY1(i,1);
solu1(i,4)=UU(i,1);
%solu1(i,5)=An(i,1);
end
disp('=============================================================')
fprintf('\n')
disp('        Compare the numerical solution with analytic solution          ')
fprintf('\n')
disp('-------------------------------------------------------------')
disp('       int.node       X        Y        solution    ')
disp('-------------------------------------------------------------')
fprintf('\n')
disp(solu1)
fprintf('\n\n')
disp('=============================================================')
%=============================================================
%                    END PROGRAM
%=============================================================
```

ProgramB2

```
%==========================================================================
%                    PROGRAM LINBEMS
%==========================================================================

%--------------------------------------------------------------------------
%This is a program to solve the Laplace equation for Ex 5.1.2
%By using Linear element
%--------------------------------------------------------------------------
clear
%--------------------------------------------------------------------------
%      Set ep(g) and w(g) values for 3-point Gaussian quadrature
%--------------------------------------------------------------------------
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];

%--------------------------------------------------------------------------
%             Input data and compute coordinate matrix
%--------------------------------------------------------------------------
disp('==========================================================')
disp('              Welcome to LINBEMS program                  ')
disp('==========================================================')
fprintf('\n')
rr=input('Enter the length of x: ');
pp=input('Enter the length of y: ');
n=input('Enter the number of boundary node: ');
disp('---------------------------------------------------------- ')
fprintf('\n\n')
%--------------------------------------------------------------------------
%             we obtain the coordinate of boundary nodes
%--------------------------------------------------------------------------
```

```
X=coordx(rr,n);
Y=coordy(pp,n);
%-------------------------------------------------------------------
%     Construct distance [d(i,j)] from a node i to an element j
%     and L(j) is a lenght of element j
%-------------------------------------------------------------------
d=distT(n,X,Y);
L=lenetT(n,X,Y);

%-------------------------------------------------------------------
%            Construct the matrix H and G
%          for solving equation system HU=GQ
%-------------------------------------------------------------------
Hhat=matrixhH(n,X,Y,d,ep,w,L);  % This is the matrix H-hat
G=matrixgG(n,X,Y,d,ep,w,L);     % This is the matrix G
%-------------------------------------------------------------------
%Assigned to node type of the boundary condition
%-------------------------------------------------------------------
N=NCode6(n);
t=NCodv6(X,Y,n);

%-------------------------------------------------------------------
%        Change an system of equation to be in formAx=B
%       and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------
A=cheqLL(N,Hhat,G,n);
bb=cheqRR(N,Hhat,G,n);
B=bb*t;
XX=(inv(A))*B;

%-------------------------------------------------------------------
%Rearranges the boundary values and forms the matrices t and XX
```

```
%-------------------------------------------------------------------
U=matrixuU6(t,n,XX);
Q=matrixqQ6(t,n,XX);

%-------------------------------------------------------------------
%         Next, we start to compute for internal nodes
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%            step 1 : input the data to compute
%-------------------------------------------------------------------
fprintf('\n')
disp('-------------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n')
disp('-------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
Xin=Xin1';
Yin=Yin1';

%-------------------------------------------------------------------
%            step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
din=distinN(n,n1,Xin,Yin,X,Y);
```

```
H2=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
%---------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
%---------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));

%---------------------------------------------------------------------
%                The solution are shown (internal node)
%---------------------------------------------------------------------
fprintf('\n\n')
disp('-----------------------------------------------------------')
disp('   The solutions are shown as follows (internal node)          ')
disp('-----------------------------------------------------------')
for i=1:n1
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')

%=============================================================
%                    END PROGRAM
%=============================================================
```

Program B3

```
%=============================================================
%                    PROGRAM LINBEMS
%=============================================================

%---------------------------------------------------------------
%This is a program to solve the Laplace equation for Ex 5.1.3
%By using Linear element
```

```
%-------------------------------------------------------------------
clear
%-------------------------------------------------------------------
%      Set ep(g) and w(g) values for 3-point Gaussian quadrature
%-------------------------------------------------------------------
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];

%-------------------------------------------------------------------
%              Input data and compute coordinate matrix
%-------------------------------------------------------------------
disp('=============================================================')
disp('              Welcome to LINBEMS program                     ')
disp('=============================================================')
fprintf('\n')
rr=input('Enter the length of x: ');
pp=input('Enter the length of y: ');
n=input('Enter the number of boundary node: ');
disp('------------------------------------------------------------- ')
fprintf('\n\n')

%-------------------------------------------------------------------
%              we obtain the coordinate of boundary nodes
%-------------------------------------------------------------------
X=coordx(rr,n);
Y=coordy(pp,n);

%-------------------------------------------------------------------
%      Construct distance [d(i,j)] from a node i to an element j
%      and L(j) is a lenght of element j
%-------------------------------------------------------------------
d=distT(n,X,Y);
L=lenetT(n,X,Y);
```

```
%-------------------------------------------------------------------
%              Construct the matrix H and G
%            for solving equation system HU=GQ
%-------------------------------------------------------------------
Hhat=matrixhH(n,X,Y,d,ep,w,L);  % This is the matrix H-hat
G=matrixgG(n,X,Y,d,ep,w,L);     % This is the matrix G

%-------------------------------------------------------------------
%Assigned to node type of the boundary condition
%-------------------------------------------------------------------
N=NCode9(n);
t=NCodv9(X,Y,n);

%-------------------------------------------------------------------
%          Change an system of equation to be in formAx=B
%         and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------
A=cheqLL(N,Hhat,G,n);
bb=cheqRR(N,Hhat,G,n);
B=bb*t;
XX=(inv(A))*B;

%-------------------------------------------------------------------
%Rearranges the boundary values and forms the matrices t and XX
%-------------------------------------------------------------------
U=matrixuU9(t,n,XX);
Q=matrixqQ9(t,n,XX);

%-------------------------------------------------------------------
%          Next, we start to compute for internal nodes
%-------------------------------------------------------------------
```

```
%-------------------------------------------------------------------
%          step 1 : input the data to compute
%-------------------------------------------------------------------
fprintf('\n')
disp('-------------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n')
disp('-------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
Xin=Xin1';
Yin=Yin1';

%-------------------------------------------------------------------
%          step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
din=distinN(n,n1,Xin,Yin,X,Y);
H2=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L);

%-------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
%-------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));
```

```
%-------------------------------------------------------------------
%                The solution are shown (internal node)
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-------------------------------------------------------------')
disp('   The solutions are shown as follows (internal node)          ')
disp('-------------------------------------------------------------')
for i=1:n1
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')
%===================================================================
%                    END PROGRAM
%===================================================================
```

ProgramB4

```
%===================================================================
%                    PROGRAM LINBEMS
%===================================================================

%-------------------------------------------------------------------
%This is a program to solve the boundary problem for Ex 5.1.4
%By using Linear element
%-------------------------------------------------------------------
clear
%-------------------------------------------------------------------
%     Set ep(g) and w(g) values for 3-point Gaussian quadrature
%-------------------------------------------------------------------
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];

%-------------------------------------------------------------------
%           Input data and compute coordinate matrix
```

```
%-------------------------------------------------------------------
disp('=============================================================')
disp('          Welcome to LINBEMS program                         ')
disp('=============================================================')
fprintf('\n')
rr=input('Enter the length of x: ');
pp=input('Enter the length of y: ');
n=input('Enter the number of boundary node: ');
disp('------------------------------------------------------------ ')
fprintf('\n\n')

%-------------------------------------------------------------------
%          we obtain the coordinate of boundary nodes
%-------------------------------------------------------------------
X=coordx(rr,n);
Y=coordy(pp,n);

%-------------------------------------------------------------------
%     Construct distance [d(i,j)] from a node i to an element j
%     and L(j) is a lenght of element j
%-------------------------------------------------------------------
d=distT(n,X,Y);
L=lenetT(n,X,Y);

%-------------------------------------------------------------------
%           Construct the matrix H and G
%          for solving equation system HU=GQ
%-------------------------------------------------------------------
Hhat=matrixhH(n,X,Y,d,ep,w,L);  % This is the matrix H-hat
G=matrixgG(n,X,Y,d,ep,w,L);     % This is the matrix G

%-------------------------------------------------------------------
```

```
%Assigned to node type of the boundary condition
%-------------------------------------------------------------------
N=NCode7(n);
t=NCodv7(X,Y,n);

%-------------------------------------------------------------------
%         Change an system of equation to be in formAx=B
%        and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------
A=cheqLL(N,Hhat,G,n);
bb=cheqRR(N,Hhat,G,n);
B=bb*t;
XX=(inv(A))*B;

%-------------------------------------------------------------------
%Rearranges the boundary values and forms the matrices t and XX
%-------------------------------------------------------------------
U=matrixuU7(t,n,XX);
Q=matrixqQ7(t,n,XX);

%-------------------------------------------------------------------
%         Next, we start to compute for internal nodes
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%           step 1 : input the data to compute
%-------------------------------------------------------------------
fprintf('\n')
disp('-----------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n')
```

```
disp('-----------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
Xin=Xin1';
Yin=Yin1';

%-------------------------------------------------------------------
%          step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
din=distinN(n,n1,Xin,Yin,X,Y);
H2=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L);

%-------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
%-------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));

%-------------------------------------------------------------------
%                The solution are shown (internal node)
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-----------------------------------------------------------------')
disp('   The solutions are shown as follows (internal node)          ')
disp('-----------------------------------------------------------------')
for i=1:n1
```

```
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')

%=================================================================
%                    END PROGRAM
%=================================================================
```

ProgramB5

```
%=================================================================
%                    PROGRAM  LINBEMS
%=================================================================

%-----------------------------------------------------------------
%This is a program to solve the problem for Ex 5.1.5
%By using Linear element
%-----------------------------------------------------------------
clear
%-----------------------------------------------------------------
%      Set ep(g) and w(g) values for 3-point Gaussian quadrature
%-----------------------------------------------------------------
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];

%-----------------------------------------------------------------
%            Input data and compute coordinate matrix
%-----------------------------------------------------------------
disp('=============================================================')
disp('              Welcome to LINBEMS program                     ')
disp('=============================================================')
fprintf('\n')
rr=input('Enter the length of x: ');
pp=input('Enter the length of y: ');
```

```
n=input('Enter the number of boundary node: ');
disp('------------------------------------------------------------ ')
fprintf('\n\n')

%-------------------------------------------------------------------
%            we obtain the coordinate of boundary nodes
%-------------------------------------------------------------------
X=coordx(rr,n);
Y=coordy(pp,n);

%-------------------------------------------------------------------
%     Construct distance [d(i,j)] from a node i to an element j
%     and L(j) is a lenght of element j
%-------------------------------------------------------------------
d=distT(n,X,Y);
L=lenetT(n,X,Y);

%-------------------------------------------------------------------
%             Construct the matrix H and G
%           for solving equation system HU=GQ
%-------------------------------------------------------------------
Hhat=matrixhH(n,X,Y,d,ep,w,L);  % This is the matrix H-hat
G=matrixgG(n,X,Y,d,ep,w,L);     % This is the matrix G

%-------------------------------------------------------------------
%Assigned to node type of the boundary condition
%-------------------------------------------------------------------
N=NCode7(n);
t=NCodv11(X,Y,n);

%-------------------------------------------------------------------
%         Change an system of equation to be in formAx=B
```

```
%       . and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------------
A=cheqLL(N,Hhat,G,n);
bb=cheqRR(N,Hhat,G,n);
B=bb*t;
XX=(inv(A))*B;

%-------------------------------------------------------------------------
%Rearranges the boundary values and forms the matrices t and XX
%-------------------------------------------------------------------------
U=matrixuU7(t,n,XX);
Q=matrixqQ7(t,n,XX);

%-------------------------------------------------------------------------
%          Next, we start to compute for internal nodes
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%             step 1 : input the data to compute
%-------------------------------------------------------------------------
fprintf('\n')
disp('-------------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n')
disp('-------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
```

```
Xin=Xin1';
Yin=Yin1';
%-------------------------------------------------------------------
%          step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
din=distinN(n,n1,Xin,Yin,X,Y);
H2=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L);

%-------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
%-------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));

%-------------------------------------------------------------------
%                The solution are shown (internal node)
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-------------------------------------------------------------')
disp('    The solutions are shown as follows (internal node)          ')
disp('-------------------------------------------------------------')
for i=1:n1
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')

%===================================================================
%                    END PROGRAM
%===================================================================
```

ProgramB6

```
%==========================================================================
%                    PROGRAM  LINBEMS
%==========================================================================

%--------------------------------------------------------------------------
%This is a program to solve the problem for Ex 5.2.1
%By using Linear element
%--------------------------------------------------------------------------
clear
%--------------------------------------------------------------------------
%      Set ep(g) and w(g) values for 3-point Gaussian quadrature
%--------------------------------------------------------------------------
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];

%--------------------------------------------------------------------------
%             Input data and compute coordinate matrix
%--------------------------------------------------------------------------
disp('============================================================')
disp('                Welcome to LINBEMS program                  ')
disp('============================================================')
fprintf('\n')
n=input('Enter the number of boundary node: ');
rr=input('Enter the length of x: ');
pp=input('Enter the length of y: ');
disp('------------------------------------------------------------ ')
fprintf('\n\n')
%--------------------------------------------------------------------------
%             we obtain the coordinate of boundary nodes
%--------------------------------------------------------------------------
```

```
X=coordx(rr,n);
Y=coordy(pp,n);
%-------------------------------------------------------------------
%      Construct distance [d(i,j)] from a node i to an element j
%      and L(j) is a lenght of element j
%-------------------------------------------------------------------
d=distT(n,X,Y);
L=lenetT(n,X,Y);

%-------------------------------------------------------------------
%              Construct the matrix H and G
%            for solving equation system HU=GQ
%-------------------------------------------------------------------
Hhat=matrixhH(n,X,Y,d,ep,w,L); % This is the matrix H-hat
G=matrixgG(n,X,Y,d,ep,w,L); % This is the matrix G

%-------------------------------------------------------------------
%Assigned to node type of the boundary condition
%-------------------------------------------------------------------
N=NCode2(n);
t=NCodv2(Y,n);

%-------------------------------------------------------------------
%         Change an system of equation to be in formAx=B
%        and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------
A=cheqLL(N,Hhat,G,n);
bb=cheqRR(N,Hhat,G,n);
B=bb*t;
XX=(inv(A))*B;

%-------------------------------------------------------------------
```

```
%Rearranges the boundary values and forms the matrices t and XX
%-------------------------------------------------------------------
U=matrixuU2(t,n,XX);
Q=matrixqQ2(t,n,XX);

%-------------------------------------------------------------------
%       Next, we start to compute for internal nodes
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%          step 1 : input the data to compute
%-------------------------------------------------------------------
fprintf('\n')
disp('-------------------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n')
disp('-------------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
Xin=Xin1';
Yin=Yin1';

%-------------------------------------------------------------------
%          step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
```

```
din=distinN(n,n1,Xin,Yin,X,Y);
H2=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L);

%-------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
%-------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));

%-------------------------------------------------------------------
%                The solution are shown (internal node)
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-----------------------------------------------------------')
disp('    The solutions are shown as follows (internal node)          ')
disp('-----------------------------------------------------------')
for i=1:n1
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')

%===================================================================
%                    END PROGRAM
%===================================================================
```

ProgramB7

```
%===================================================================
%                    PROGRAM LINBEMS
%===================================================================
%-------------------------------------------------------------------
%This is a program to solve the problem for Ex 5.2.3
```

```
%By using Linear element
%-------------------------------------------------------------------------
clear
%-------------------------------------------------------------------------
%      Set ep(g) and w(g) values for 3-point Gaussian quadrature
%-------------------------------------------------------------------------
w=[5/9;8/9;5/9]; ep=[-sqrt(3/5);0;sqrt(3/5)];
n=60;
X=[0;0.2;0.4;0.6;0.8;1;1.2;1.4;1.6;1.8;
2;2.2;2.4;2.6;2.8;3;3.1;3.2;3.3;3.4;3.5;3.6;
3.7;3.8;3.9;4;4;4;4;4;4;3.8;
3.6;3.4;3.2;3.0;2.8;2.6;2.4;2.2;2;1.8;
1.6;1.4;1.2;1;.8;.6;.4;.2;0;0;
0;0;0;0;0;0;0;0];
Y=[0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;.46;.6;.71;.8;
.87;.92;.95;.98;.99;1;1.2;1.4;1.6;1.8;
2;2;2;2;2;2;2;2;2;2;
2;2;2;2;2;2;2;2;2;2;2;
1.8;1.6;1.4;1.2;1;0.8;0.6;0.4;0.2];
%-------------------------------------------------------------------------
%      Construct distance [d(i,j)] from a node i to an element j
%      and L(j) is a lenght of element j
%-------------------------------------------------------------------------
d=distT(n,X,Y);
L=lenetT(n,X,Y);

%-------------------------------------------------------------------------
%              Construct the matrix H and G
%            for solving equation system HU=GQ
%-------------------------------------------------------------------------
Hhat=matrixhH(n,X,Y,d,ep,w,L);  % This is the matrix H-hat
```

```
G=matrixgG(n,X,Y,d,ep,w,L);     % This is the matrix G

%-------------------------------------------------------------------------
%Assigned to node type of the boundary condition
%-------------------------------------------------------------------------
N=NCode5(n);
t=NCodv5(Y,n);
%-------------------------------------------------------------------------
%          Change an system of equation to be in formAx=B
%         and solve to obtain the unknown values of U and Q
%-------------------------------------------------------------------------
A=cheqLL4(N,Hhat,G,n);
bb=cheqRR4(N,Hhat,G,n);
B=bb*t;
XX=(inv(A))*B;

%-------------------------------------------------------------------------
%Rearranges the boundary values and forms the matrices t and XX
%-------------------------------------------------------------------------
U=matrixuU5(t,n,XX);
Q=matrixqQ5(t,n,XX);

%-------------------------------------------------------------------------
%          Next, we start to compute for internal nodes
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%             step 1 : input the data to compute
%-------------------------------------------------------------------------
fprintf('\n\n')
disp('-----------------------------------------------------------')
n1=input(' Enter the number of internal nodes: ');
fprintf('\n\n')
```

```
disp('Next ,input the coordinate(X,Y) of each int node')
fprintf('\n\n')
disp('-----------------------------------------------------------------')
for i=1:n1
fprintf('X-coordinate of node %.0f is',i);
Xin1(i)=input(':');
fprintf('Y-coordinate of node %.0f is',i);
Yin1(i)=input(':');
end
Xin=Xin1';
Yin=Yin1';

%-------------------------------------------------------------------
%          step 2 : Construct matrix H and G
%          both H2 and G2 are constructed in the same
%          way to construct Hhat and G
%-------------------------------------------------------------------
din=distinN(n,n1,Xin,Yin,X,Y);
H2=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L);
G2=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L);

%-------------------------------------------------------------------
%          step 3 : compute the solutions at each internal node
%-------------------------------------------------------------------
Usol=(1/(2*pi))*((G2)*(Q)-(H2)*(U));

%-------------------------------------------------------------------
%                The solution are shown (internal node)
%-------------------------------------------------------------------
fprintf('\n\n')
disp('-----------------------------------------------------------')
disp('   The solutions are shown as follows (internal node)          ')
```

```
disp('-----------------------------------------------------------------')
for i=1:n1
fprintf('\n U(%.2f,%.2f)=%.6f\n',Xin(i),Yin(i),Usol(i));
end
fprintf('\n\n')
%=================================================================
%                    END PROGRAM
%=================================================================
```

Subroutines (BEM)

```
%//////////////////////////////////////////////////////////////////////////////////////////
%----------------------------
function [X]=coordx(rr,n)
%Inputs
%rr: lenght of x
%n: number of elements
k=n/4;
X=zeros(n,1);
for i=1:k
X(i+1)=X(i+1)+(i/k)*rr;
end
for i=k+2:2*k+1
X(i)=rr;
end
for i=2:k
X(2*k+i)=X(k-(i-2));
end
%-------------------------
%//////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------
function [Y]=coordy(pp,n)
%Inputs
```

```
%pp:lenght of y
%n: number of elements
k=n/4;
Y=zeros(n,1);
for i=1:k
Y(k+(i+1))=Y(k+(i+1))+(i/k)*pp;
end
for i=2*k+2:3*k+1
Y(i)=pp;
end
for i=2:k
Y(3*k+(i))=Y(2*k-(i-2));
end
%---------------------------------------
%///////////////////////////////////////////////////////////////////////////////////////
%---------------------------------------
function [t]=NCodv(Y,n)
k=n/4;
t=zeros(n,1);
for i=1:k
t(i)=-2;
end
for i=k+1:2*k
t(i)=2+2*Y(i);
end
for i=2*k+1:3*k
t(i)=2;
end
for i=3*k+1:n
t(i)=1+2*Y(i);
end
%---------------------------------------
```

```
%////////////////////////////////////////////////////////////////////////////////////////
function [N]=NCode(n)
%-----------------------------------------------
% N(i)=0 means at node i the value of Q is known
% N(i)=1 means at node i the value of U is known
%-----------------------------------------------
N=zeros(n,1);
k=n/4;
for i=1:k
N(i)=0;
end
for i=k+1:2*k
N(i)=1;
end
for i=2*k+1:3*k
N(i)=0;
end
for i=3*k+1:n
N(i)=1;
end
%-----------------------------------------------
%////////////////////////////////////////////////////////////////////////////////////////
%-----------------------------------------------
function [bb]=cheqRR(N,Hhat,G,n)
bb=zeros(n);
for i=1:n
for j=1:n
if N(j)==1
C(i,j)=-Hhat(i,j);
Hhat(i,j)=-G(i,j);
G(i,j)=C(i,j);
end
```

```
end
end
bb=G;
%////////////////////////////////////////////////////////////////////////////////////////////
function [A]=cheqLL(N,Hhat,G,n)
A=zeros(n);
for i=1:n
for j=1:n
if N(j)==1
C(i,j)=-Hhat(i,j);
Hhat(i,j)=-G(i,j);
G(i,j)=C(i,j);
end
end
 end
A=Hhat;
%////////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------
function [U]=matrixuU(t,n,XX)
U=zeros(n,1);
k=n/4;
 for i=1:k
 U(i)=XX(i);
 end
 for i=k+1:2*k
 U(i)=t(i);
 end
 for i=2*k+1:3*k
 U(i)=XX(i);
 end
 for i=3*k+1:n
 U(i)=t(i);
```

```
end
%-----------------------------------------------------
%/////////////////////////////////////////////////////////////////////////////////////////
%-----------------------------------------------------
function [Q]=matrixqQ(t,n,XX)
Q=zeros(n,1);
k=n/4;
for i=1:k
Q(i)=t(i);
end
for i=k+1:2*k
Q(i)=XX(i);
end
for i=2*k+1:3*k
Q(i)=t(i);
end
for i=3*k+1:n
Q(i)=XX(i);
end
%-----------------------------------------------------
%/////////////////////////////////////////////////////////////////////////////////////////
%-----------------------------------------------------
% Construct distance matrix d(i,j)
function [d]=distT(n,X,Y)
d=zeros(n);
for i=1:n
for j=1:n-1
ax=X(j+1)-X(j);
ay=Y(j+1)-Y(j);
if abs(ax)>=0.0001
ta=(ay/ax);
d(i,j)=abs(ta*X(i)-Y(i)+Y(j)-ta*X(j))/(sqrt(ta^2+1));
```

```
else
d(i,j)=abs(X(i)-X(j));
end
end
end
for i=1:n
ax=X(1)-X(n);
ay=Y(1)-Y(n);
if abs(ax)>=0.0001
ta=(ay/ax);
d(i,n)=abs(ta*X(i)-Y(i)+Y(n)-ta*X(n))/(sqrt(ta^2+1));
else
d(i,n)=abs(X(i)-X(n));
end
end
%-------------------------------------------------------------
%/////////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------
function L=lenetT(n,X,Y)
L=zeros(n,1);
for j=2:n-1
L(j)=L(j)+sqrt((X(j+1)-X(j))^2+(Y(j+1)-Y(j))^2);
end
L(n)=sqrt((X(1)-X(n))^2+(Y(1)-Y(n))^2);
L(1)=sqrt((X(2)-X(1))^2+(Y(2)-Y(1))^2);
% ******Construct distance matrix d(i,j)******
function [din]=distinN(n,n1,Xin,Yin,X,Y)
din=zeros(n1,n);
for i=1:n1
for j=1:n-1
ax=X(j+1)-X(j);
ay=Y(j+1)-Y(j);
```

```
if abs(ax)>=0.0001
ta=(ay/ax);
din(i,j)=abs(ta*Xin(i)-Yin(i)+Y(j)-ta*X(j))/(sqrt(ta^2+1));
else
 din(i,j)=abs(Xin(i)-X(j));
end
end
end
 for i=1:n1
ax=X(1)-X(n);
ay=Y(1)-Y(n);
 if abs(ax)>=0.0001
 ta=(ay/ax);
 din(i,n)=abs(ta*Xin(i)-Yin(i)+Y(n)-ta*X(n))/(sqrt(ta^2+1));
 else
 din(i,n)=abs(Xin(i)-X(n));
 end
 end
 %-------------------------------------------------------------
 %//////////////////////////////////////////////////////////////
 %-------------------------------------------------------------
 function [Hhat]=matrixhH(n,X,Y,d,ep,w,L)
 Hhat=zeros(n);
 x=zeros(n,1);
 y=zeros(n,1);
 %-------------------------------------------------------------
 % Compute x,y middpoint of each element (see p.39 Davies )
 %-------------------------------------------------------------
 for i=1:n-1
 x(i)=x(i)+(X(i)+X(i+1))/2;
 y(i)=y(i)+(Y(i)+Y(i+1))/2;
 end
```

```
x(n)=x(n)+(X(n)+X(1))/2;
y(n)=y(n)+(Y(n)+Y(1))/2;
%-------------------------------------------------------------
% Next we shall construct H-hat by condidering in 2 parts
% first for R(i,j) and second, for R(i,j-1)
% ***** For the element 1 to n-1 we obtain *******
%-------------------------------------------------------------
for i=1:n
%********** when j=2...n-1*********
for j=2:n-1
%********** first consider for R(i,j)********
for g=1:3
x1(g)=x(j)+(1/2)*((X(j+1)-X(j))*ep(g));
y1(g)=y(j)+(1/2)*((Y(j+1)-Y(j))*ep(g));
R2ij1(g)=(X(i)-x1(g))^2+(Y(i)-y1(g))^2;
S1(g)=w(g)*((1-ep(g))/R2ij1(g));
end
s1=sum(S1);
ss1=(-L(j)*d(i,j)/(4))*(s1);
%-------------------------------------------------------------
% **** next consider for R(i,j-1)
%-------------------------------------------------------------
for g=1:3
x3(g)=x(j-1)+(1/2)*((X(j)-X(j-1))*ep(g));
y3(g)=y(j-1)+(1/2)*((Y(j)-Y(j-1))*ep(g));
R2ij3(g)=(X(i)-x3(g))^2+(Y(i)-y3(g))^2;
S3(g)=w(g)*((1+ep(g))/R2ij3(g));
end
s3=sum(S3);
ss2=(-L(j)*d(i,j-1)/(4))*(s3);
%*********
Hhat(i,j)=(1/(2*pi))*(ss1+ss2);
```

```
end
end
%-------------------------------------------------------------
% ********* For an element j=1 **********
% first consider for R(i,j)
%-------------------------------------------------------------
for i=1:n
for g=1:3
x1(g)=x(1)+(1/2)*((X(2)-X(1))*ep(g));
y1(g)=y(1)+(1/2)*((Y(2)-Y(1))*ep(g));
R2ij1(g)=(X(i)-x1(g))^2+(Y(i)-y1(g))^2;
S1(g)=w(g)*((1-ep(g))/R2ij1(g));
end
s1=sum(S1);
ss1=(-L(1)*d(i,1)/(4))*(s1);
%-------------------------------------------------------------
% **** next consider for R(i,j-1)
%-------------------------------------------------------------
for g=1:3
x3(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y3(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
R2ij3(g)=(X(i)-x3(g))^2+(Y(i)-y3(g))^2;
S3(g)=w(g)*((1+ep(g))/R2ij3(g));
end
s3=sum(S3);
ss2=(-L(1)*d(i,n)/(4))*(s3);
%*********
Hhat(i,1)=(1/(2*pi))*(ss1+ss2);
end
%-------------------------------------------------------------
%******* For an element j=n ***************
%-------------------------------------------------------------
```

```
for i=1:n
% ******first for R(i,j)*******
for g=1:3
x1(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y1(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
R2ij1(g)=(X(i)-x1(g))^2+(Y(i)-y1(g))^2;
S1(g)=w(g)*((1-ep(g))/R2ij1(g));
end
s1=sum(S1);
ss1=(-L(n)*d(i,n)/(4))*(s1);
%-------------------------------------------------------------
% Next for R(i,j)
%-------------------------------------------------------------
for g=1:3
x3(g)=x(n-1)+(1/2)*((X(n)-X(n-1))*ep(g));
y3(g)=y(n-1)+(1/2)*((Y(n)-Y(n-1))*ep(g));
R2ij3(g)=(X(i)-x3(g))^2+(Y(i)-y3(g))^2;
S3(g)=w(g)*((1+ep(g))/R2ij3(g));
end
s3=sum(S3);
ss2=(-L(n)*d(i,n-1)/(4))*(s3);
%*********
Hhat(i,n)=(1/(2*pi))*(ss1+ss2);
end
%-------------------------------------------------------------
% Next we consider Hhat's values when i=j
%-------------------------------------------------------------
Hhat1=sum(Hhat');
for i=1:n
Hhat(i,i)=-Hhat1(1,i);
end
%-------------------------------------------------------------
```

```
%////////////////////////////////////////////////////////////////////////////////////////
%--------------------------------------------------------
function [G]=matrixgG(n,X,Y,d,ep,w,L)
G=zeros(n);
x=zeros(n,1);
y=zeros(n,1);
%--------------------------------------------------------
% Compute x,y middpoint of each element (see p.39 Davies )
%--------------------------------------------------------
for i=1:n-1
x(i)=x(i)+(X(i)+X(i+1))/2;
y(i)=y(i)+(Y(i)+Y(i+1))/2;
end
x(n)=x(n)+(X(n)+X(1))/2;
y(n)=y(n)+(Y(n)+Y(1))/2;
%--------------------------------------------------------
% Next we shall construct H-hat by condidering in 2 parts
% first for R(i,j) and second, for R(i,j-1)
% ***** For the element 1 to n-1 we obtain *******
%--------------------------------------------------------
for i=1:n
% ********when j=2...n-1********
for j=2:n-1
% ********first consider for R(i,j)********
for g=1:3
x1(g)=x(j)+(1/2)*((X(j+1)-X(j))*ep(g));
y1(g)=y(j)+(1/2)*((Y(j+1)-Y(j))*ep(g));
Rij1(g)=sqrt((X(i)-x1(g))^2+(Y(i)-y1(g))^2);
S1(g)=w(g)*(1-ep(g))*log(Rij1(g));
end
s1=sum(S1);
ss1=(-L(j)/4)*(s1);
```

```
%-----------------------------------------------------------
% **** next consider for R(i,j-1)*******
%-----------------------------------------------------------
for g=1:3
x3(g)=x(j-1)+(1/2)*((X(j)-X(j-1))*ep(g));
y3(g)=y(j-1)+(1/2)*((Y(j)-Y(j-1))*ep(g));
Rij3(g)=sqrt((X(i)-x3(g))^2+(Y(i)-y3(g))^2);
S3(g)=w(g)*(1+ep(g))*log(Rij3(g));
end
s3=sum(S3);
ss2=(-L(j)/4)*(s3);
%***********
G(i,j)=(1/(2*pi))*(ss1+ss2);
end
end



%-----------------------------------------------------------
% ********* For an element j=1 **********
% first consider for R(i,j)
%-----------------------------------------------------------
for i=1:n
for g=1:3
x1(g)=x(1)+(1/2)*((X(2)-X(1))*ep(g));
y1(g)=y(1)+(1/2)*((Y(2)-Y(1))*ep(g));
Rij1(g)=sqrt((X(i)-x1(g))^2+(Y(i)-y1(g))^2);
S1(g)=w(g)*(1-ep(g))*log(Rij1(g));
end
s1=sum(S1);
ss1=(-L(1)/4)*(s1);
%-----------------------------------------------------------
```

```
% **** next consider for R(i,j-1)
%-------------------------------------------------------
for g=1:3
x3(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y3(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
Rij3(g)=sqrt((X(i)-x3(g))^2+(Y(i)-y3(g))^2);
S3(g)=w(g)*(1+ep(g))*log(Rij3(g));
end
s3=sum(S3);
ss2=(-L(1)/4)*(s3);
%*********
G(i,1)=(1/(2*pi))*(ss1+ss2);
end
%-------------------------------------------------------
%******* For an element j=n ****************
%-------------------------------------------------------
for i=1:n
%***** first for R(i,j)*****
for g=1:3
x1(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y1(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
Rij1(g)=sqrt((X(i)-x1(g))^2+(Y(i)-y1(g))^2);
S1(g)=w(g)*(1-ep(g))*log(Rij1(g));
end
s1=sum(S1);
ss1=(-L(n)/4)*(s1);
% ******Next for R(i,j)******
for g=1:3
x3(g)=x(n-1)+(1/2)*((X(n)-X(n-1))*ep(g));
y3(g)=y(n-1)+(1/2)*((Y(n)-Y(n-1))*ep(g));
Rij3(g)=sqrt((X(i)-x3(g))^2+(Y(i)-y3(g))^2);
S3(g)=w(g)*(1+ep(g))*log(Rij3(g));
```

```
end
s3=sum(S3);
ss2=(-L(n)/4)*(s3);
%*******
G(i,n)=(1/(2*pi))*(ss1+ss2);
end
%-------------------------------------------------------------
% Next we consider Hhat's values when i=j
%-------------------------------------------------------------
for i=2:n
G(i,i)=(1/(2*pi))*(-(L(i)/2)*(log(L(i))-(3/2))-(L(i-1)/2)*(log(L(i-1))-(3/2)));
end
G(1,1)=(1/(2*pi))*(-(L(1)/2)*(log(L(1))-(3/2))-(L(n)/2)*(log(L(n))-(3/2)));
%-------------------------------------------------------------
%/////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------------------
function [G2]=matrixgG2(n,n1,X,Y,Xin,Yin,din,ep,w,L)
G2=zeros(n1,n);
x=zeros(n,1);
y=zeros(n,1);
%-------------------------------------------------------------------------
% Compute x,y middpoint of each element (see p.39 Davies )
%-------------------------------------------------------------------------
for i=1:n-1
x(i)=x(i)+(X(i)+X(i+1))/2;
y(i)=y(i)+(Y(i)+Y(i+1))/2;
end
x(n)=x(n)+(X(n)+X(1))/2;
y(n)=y(n)+(Y(n)+Y(1))/2;
%-------------------------------------------------------------------------
% Next we shall construct H-hat by condidering in 2 parts
% first for R(i,j) and second, for R(i,j-1)
```

```
% ***** For the element 1 to n-1 we obtain *******
%-------------------------------------------------------------------
for i=1:n1
% ****when j=2...n-1****
for j=2:n-1
% ****first consider for R(i,j)*****
for g=1:3
x1(g)=x(j)+(1/2)*((X(j+1)-X(j))*ep(g));
y1(g)=y(j)+(1/2)*((Y(j+1)-Y(j))*ep(g));
Rij1(g)=sqrt((Xin(i)-x1(g))^2+(Yin(i)-y1(g))^2);
S1(g)=w(g)*(1-ep(g))*log(Rij1(g));
end
s1=sum(S1);
ss1=(-L(j)/4)*(s1);
% **** next consider for R(i,j-1)********
for g=1:3
x3(g)=x(j-1)+(1/2)*((X(j)-X(j-1))*ep(g));
y3(g)=y(j-1)+(1/2)*((Y(j)-Y(j-1))*ep(g));
Rij3(g)=sqrt((Xin(i)-x3(g))^2+(Yin(i)-y3(g))^2);
S3(g)=w(g)*(1+ep(g))*log(Rij3(g));
end
s3=sum(S3);
ss2=(-L(j)/4)*(s3);
%*********
G2(i,j)=(ss1+ss2);
end
end
%-----------------------------------------------------------
% ********* For an element j=1 ***********
% first consider for R(i,j)
%-----------------------------------------------------------
for i=1:n1
```

```
for g=1:3
x1(g)=x(1)+(1/2)*((X(2)-X(1))*ep(g));
y1(g)=y(1)+(1/2)*((Y(2)-Y(1))*ep(g));
Rij1(g)=sqrt((Xin(i)-x1(g))^2+(Yin(i)-y1(g))^2);
S1(g)=w(g)*(1-ep(g))*log(Rij1(g));
end
s1=sum(S1);
ss1=(-L(1)/4)*(s1);
%-------------------------------------------------------------
% **** next consider for R(i,j-1)
%-------------------------------------------------------------
for g=1:3
x3(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y3(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
Rij3(g)=sqrt((Xin(i)-x3(g))^2+(Yin(i)-y3(g))^2);
S3(g)=w(g)*(1+ep(g))*log(Rij3(g));
end
s3=sum(S3);
ss2=(-L(1)/4)*(s3);
%*********
G2(i,1)=(ss1+ss2);
end
%-------------------------------------------------------------
%******* For an element j=n ***************
%-------------------------------------------------------------
for i=1:n1
%**** first for R(i,j)****
for g=1:3
x1(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y1(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
Rij1(g)=sqrt((Xin(i)-x1(g))^2+(Yin(i)-y1(g))^2);
S1(g)=w(g)*(1-ep(g))*log(Rij1(g));
```

```
end
s1=sum(S1);
ss1=(-L(n)/4)*(s1);
% *****Next for R(i,j)****
for g=1:3
x3(g)=x(n-1)+(1/2)*((X(n)-X(n-1))*ep(g));
y3(g)=y(n-1)+(1/2)*((Y(n)-Y(n-1))*ep(g));
Rij3(g)=sqrt((Xin(i)-x3(g))^2+(Yin(i)-y3(g))^2);
S3(g)=w(g)*(1+ep(g))*log(Rij3(g));
end
s3=sum(S3);
ss2=(-L(n)/4)*(s3);
%*********
G2(i,n)=(ss1+ss2);
end
%-------------------------------------------------------------

%////////////////////////////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------
function [H2]=matrixhH2(n,n1,X,Y,Xin,Yin,din,ep,w,L)
H2=zeros(n1,n);
x=zeros(n,1);
y=zeros(n,1);
%-------------------------------------------------------------------------
% Compute x,y middpoint of each element (see p.39 Davies )
%-------------------------------------------------------------------------
for i=1:n-1
x(i)=x(i)+(X(i)+X(i+1))/2;
y(i)=y(i)+(Y(i)+Y(i+1))/2;
end
x(n)=x(n)+(X(n)+X(1))/2;
y(n)=y(n)+(Y(n)+Y(1))/2;
```

```
%-----------------------------------------------------------------
% Next we shall construct H-hat by condidering in 2 parts
% first for R(i,j) and second, for R(i,j-1)
% ***** For the element 1 to n-1 we obtain *******
%-----------------------------------------------------------------
for i=1:n1
% when j=2...n-1
for j=2:n-1
% first consider for R(i,j)
for g=1:3
x1(g)=x(j)+(1/2)*((X(j+1)-X(j))*ep(g));
y1(g)=y(j)+(1/2)*((Y(j+1)-Y(j))*ep(g));
R2ij1(g)=(Xin(i)-x1(g))^2+(Yin(i)-y1(g))^2;
S1(g)=w(g)*((1-ep(g))/R2ij1(g));
end
s1=sum(S1);
ss1=(-L(j)*din(i,j)/(4))*(s1);
%-----------------------------------------------------------
% **** next consider for R(i,j-1)
%-----------------------------------------------------------
for g=1:3
x3(g)=x(j-1)+(1/2)*((X(j)-X(j-1))*ep(g));
y3(g)=y(j-1)+(1/2)*((Y(j)-Y(j-1))*ep(g));
R2ij3(g)=(Xin(i)-x3(g))^2+(Yin(i)-y3(g))^2;
S3(g)=w(g)*((1+ep(g))/R2ij3(g));
end
s3=sum(S3);
ss2=(-L(j)*din(i,j-1)/(4))*(s3);
%*********
H2(i,j)=(ss1+ss2);
end
end
```

```
%-------------------------------------------------------------
% ********* For an element j=1 ***********
% first consider for R(i,j)
%-------------------------------------------------------------
for i=1:n1
for g=1:3
x1(g)=x(1)+(1/2)*((X(2)-X(1))*ep(g));
y1(g)=y(1)+(1/2)*((Y(2)-Y(1))*ep(g));
R2ij1(g)=(Xin(i)-x1(g))^2+(Yin(i)-y1(g))^2;
S1(g)=w(g)*((1-ep(g))/R2ij1(g));
end
s1=sum(S1);
ss1=(-L(1)*din(i,1)/(4))*(s1);
%-------------------------------------------------------------
% **** next consider for R(i,j-1)
%-------------------------------------------------------------
for g=1:3
x3(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y3(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
R2ij3(g)=(Xin(i)-x3(g))^2+(Yin(i)-y3(g))^2;
S3(g)=w(g)*((1+ep(g))/R2ij3(g));
end
s3=sum(S3);
ss2=(-L(1)*din(i,n)/(4))*(s3);
%*********
H2(i,1)=(ss1+ss2);
end
%-------------------------------------------------------------
%******* For an element j=n ***************
%-------------------------------------------------------------
for i=1:n1
% ****first for R(i,j)******
```

```
for g=1:3
x1(g)=x(n)+(1/2)*((X(1)-X(n))*ep(g));
y1(g)=y(n)+(1/2)*((Y(1)-Y(n))*ep(g));
R2ij1(g)=(Xin(i)-x1(g))^2+(Yin(i)-y1(g))^2;
S1(g)=w(g)*((1-ep(g))/R2ij1(g));
end
s1=sum(S1);
ss1=(-L(n)*din(i,n)/(4))*(s1);
% ******Next for R(i,j)******
for g=1:3
x3(g)=x(n-1)+(1/2)*((X(n)-X(n-1))*ep(g));
y3(g)=y(n-1)+(1/2)*((Y(n)-Y(n-1))*ep(g));
R2ij3(g)=(Xin(i)-x3(g))^2+(Yin(i)-y3(g))^2;
S3(g)=w(g)*((1+ep(g))/R2ij3(g));
end
s3=sum(S3);
ss2=(-L(n)*din(i,n-1)/(4))*(s3);
%**********
H2(i,n)=(ss1+ss2);
end
%-------------------------------------------------------------
```