```
function pois36
%=====================================================================
%
%
%            Program POIS36
%
%            A finite element program for the solution of poissons
%            equation using either 3 or 6-node elements.
%
%     equation: dx(rx * dx(phi)) + dy(ry * dy(phi)) = qv(x, y)
%
%            FEM representation
%
%               sk(i, j) * phi(j) = q(i)
%
%            Input data:
%
%            numel    = number of elements
%            numnp    = number of nodes
%            ib       = band width of sk-matrix for current mesh
%            irz      = axi-symmetric specification
%                          irz=0: standard x,y coordinates
%                          irz=1: cylindrical coordinates with
%                                 axial symmetry: x=r, y=z
%                                 note: all x-coordinates must be
%                                 greater or equal:zero.
%            nnpe     = number of nodes per element (must be 3 or 6)
%            contr    = contour interval desired for plotting
%            xord(i)  = x-coordinate of node i
%            yord(i)  = y-coordinate of node i
%            np(i,j)  = nodal point j of element i (must be listed
%                         in counterclockwise order)
%            mat(i)   = material number for element i
%            numbc    = number of boundary conditions:be specified
%                         which differ from the default specification
%            npbc(i)  = nodal point boundary condition for node i
%                           npbc = 0: q(i)   is known      ***
%                                     phi(i) is unknown
%                           npbc = 1: q(i)   is unknown    ***
%                                     phi(i) is known
%
%            In addition:the above input data, the following
%            functions must be specified.
%
%               rx(x, y, lment, mj) = rx
%               ry(x, y, lment, mj) = ry
%               qv(x, y, lment, mj) = qv
%
%            where:   (x,y) = coordinate of the centroid of element
%                     lment = element number
%                     mj    = material number for element
%
%            The following dimensions are used in pois36:
%
%            mxbb = maximum band width of sk-matrix
%            mxee = maximum number of elements
%            mxnp = maximum number of nodes
%            mxne = number of nodes per element(3 or 6)
%
%=====================================================================
%            program POIS36
%=====================================================================
clear all
clc

mxbb = 1000;    % Max band width
mxee = 2000;    % Max number of elements
mxnp = 2000;    % Max number of nodes
mxne = 6;       % Max number of nodes per element

%===================
% OPEN INPUT FILES =
%===================
disp('PROGRAM POIS36')

fileID = -1;
errmsg = '';
while fileID < 0
   disp(errmsg);
   file = input('Open file: ', 's');
```

```
   filein  = strcat(file,'.in');
   [fileID,errmsg] = fopen(filein);
end

fileout = strcat(file,'.out');

%==================
% READ INPUT DATA =
%==================
A = fscanf(fileID,'%d %d %d %d %d',5);

numnp = A(1);
numel = A(2);
ib    = A(3);
irz   = A(4);
nnpe  = A(5);

for ii = 1:numnp
    A = fscanf(fileID,'%d %f %f',3);
    i = A(1);
    xord(i) = A(2);
    yord(i) = A(3);
end % ii

for ii = 1:numel
    if nnpe == 3
        A = fscanf(fileID,'%d %d %d %d',4);
        i = A(1);
        np(i,1:3)= A(2:4);
    else
        A = fscanf(fileID,'%d %d %d %d %d %d %d',7);
        i = A(1);
        np(i,1:6)= A(2:7);
    end
end  % ii

fclose(fileID);

%==================
% ECHO INPUT DATA =
%==================
fileID = fopen(fileout,'w');

printheader(fileID,0, nnpe)

disp(' ')
disp(['Number of nodes    : ',num2str(numnp)])
disp(['Number of elements : ',num2str(numel)])
disp(['Band Width         : ',num2str(ib)])
disp(['Nodes/elements     : ',num2str(nnpe)])
disp(['Axisymmetric flag  : ',num2str(irz)])
disp(' ')

fprintf(fileID,'%s\r\n',['Number of nodes    : ',num2str(numnp)]);
fprintf(fileID,'%s\r\n',['Number of elements : ',num2str(numel)]);
fprintf(fileID,'%s\r\n',['Band Width         : ',num2str(ib)]);
fprintf(fileID,'%s\r\n',['Nodes/elements     : ',num2str(nnpe)]);
fprintf(fileID,'%s\r\n',['Axisymmetric flag  : ',num2str(irz)]);

printheader(fileID,1,nnpe);

for i = 1:numnp
    fprintf(fileID,'%5i %17.5f %13.5f\r\n', i, xord(i), yord(i) );
end % end

printheader(fileID,2,nnpe)
for i = 1:numel
    if nnpe == 3
        fprintf(fileID,'%7i %10i %13i %13i\r\n', i, np(i,1:3) );
    else
        fprintf(fileID,'%7i %10i %13i %13i %13i %13i %13i\r\n', i, np(i,1:6) );
    end
end %i

%=========================================
%=  Call boundary condition subroutine  =
%=========================================
```

```
[npbc, qs, hs, phi, mat] = init(numnp, numel);

printheader(fileID,3,nnpe)
for i = 1:numnp
    if npbc(i) == 1
        fprintf(fileID,'%5i %12i %13.5f %13.5f %13.5f\r\n', i, npbc(i), qs(i), phi(i), hs(i) );
    end
end % i

%====================================
%=  Call shape function subroutine  =
%====================================
[sf, wt, numqp] = shafac(nnpe);


sk(i:numnp,1:ib) = 0;

q(1:numnp)  = 0;
qs(1:numnp) = 0;
hs(1:numnp) = 0;

%===================================
%=  formation of element matrices  =
%===================================
for i = 1:numel

    s(1:nnpe,1:nnpe) = 0;
    qe(1:nnpe) = 0;

    % Begin volume quadrature
    jend = numqp(1);

    %numqp(1) equals 7 for a 6-node element

    for j = 1:jend

        mj = mat(i);
        xj = 0;
        yj = 0;
        rjac(1:2,1:2) = 0;

        for k = 1:nnpe
            npk = np(i, k);
            xj = xj + sf(1, k, j) * xord(npk);
            yj = yj + sf(1, k, j) * yord(npk);
            rjac(1, 1) = rjac(1, 1) + sf(2, k, j) * xord(npk);
            rjac(1, 2) = rjac(1, 2) + sf(3, k, j) * xord(npk);
            rjac(2, 1) = rjac(2, 1) + sf(2, k, j) * yord(npk);
            rjac(2, 2) = rjac(2, 2) + sf(3, k, j) * yord(npk);
        end %k

        detj = rjac(1, 1) * rjac(2, 2) - rjac(2, 1) * rjac(1, 2);

        if detj < 0
            disp(['Determinant of J < 0 for element : ',num2str(i)])
        end

        rjaci(1, 1) = rjac(2, 2) / detj;
        rjaci(1, 2) = rjac(1, 2) / detj * (-1);
        rjaci(2, 1) = rjac(2, 1) / detj * (-1);
        rjaci(2, 2) = rjac(1, 1) / detj;

        for k = 1:nnpe
            dndx(k) = rjaci(1, 1) * sf(2, k, j) + rjaci(2, 1) * sf(3, k, j);
            dndy(k) = rjaci(1, 2) * sf(2, k, j) + rjaci(2, 2) * sf(3, k, j);
        end %k

        rxj = rx(xj, yj, i, mj);
        ryj = ry(xj, yj, i, mj);
        qvj = qv(xj, yj, i, mj);

        dv = detj;

        if irz == 1
            dv = 2 * pi * xj * dv;
        end
```

```
        for k = 1:nnpe
            qe(k) = qe(k) + wt(1, j) * sf(1, k, j) * qvj * dv;
            for L = k:nnpe
                s(k, L) = s(k, L) + wt(1, j) * (dndx(k) * rxj * dndx(L) + dndy(k) * ryj * dndy(L)) *
dv;
            end %L
        end %k

    end %j

    %Since we only calculated half of the matrix, fill in
    %the rest of it (it's symmetric).

    for j = 1:nnpe
        for k = j:nnpe
            s(k, j) = s(j, k);
        end %k
    end %j

    %Begin surface quadrature
    for j = 1:3
        if nnpe == 6
            j1 = np(i, 2 * j);
            chkh = hs(j1);
            chkq = qs(j1);
            npside = 3;
        else
            j1 = np(i, j);
            j2 = j + 1;
            if j2 > 3
                j2 = 1;
            end
            j2 = np(i, j2);
            chkh = hs(j1) * hs(j2);
            chkq = qs(j1) * qs(j2);
            npside = 2;
        end

        if (chkh ~= 0) && (chkq ~= 0)

            kend = numqp(2);
            for k = 1:kend
                xk = 0;
                dxdxi = 0;
                dydxi = 0;
                hsk = 0;
                qsk = 0;
                phik = 0;

                for L = 1:npside
                    L1 = (j - 1) * (npside - 1) + L;
                    if L1 > nnpe
                        L1 = 1;
                    end
                    npl = np(i, L1);
                    xk = xk + sf(4, L, k) * xord(npl);
                    dxdxi = dxdxi + sf(5, L, k) * xord(npl);
                    dydxi = dydxi + sf(5, L, k) * yord(npl);
                    qsk = qsk + sf(4, L, k) * qs(npl);
                    hsk = hsk + sf(4, L, k) * hs(npl);
                    phik = phik + sf(4, L, k) * phi(npl);
                end %L

                ds = Sqr(dxdxi ^ 2 + dydxi ^ 2);
                if irz == 1
                    ds = 2 * pi * xk * ds;
                end

                for L = 1:npside
                    L1 = (j - 1) * (npside - 1) + L;
                    if L1 > nnpe
                        L1 = 1;
                    end
                    if chkq ~= 0
                        qe(L1) = qe(L1) + wt(2, k) * sf(4, L, k) * qsk * ds;
                    end
                    if chkh ~= 0
```

```
                        qe(L1) = qe(L1) + wt(2, k) * sf(4, L, k) * hsk * phik * ds;
                        for m = 1:npside
                            m1 = (j - 1) * (npside - 1) + m;
                            if m1 > nnpe
                                m1 = 1;
                            end
                            s(L1, m1) = s(L1, m1) + wt(2, k) * sf(4, L, k) * hsk * sf(4, m, k) * ds;
                        end %m
                    end
                end %L
            end %k
        end %if
    end %j

    %End of quadrature

    %Place completed element matrices in global sk and q matrices

    for j = 1:nnpe
        j1 = np(i, j);
        q(j1) = q(j1) + qe(j);
        for k = 1:nnpe
            k1 = np(i, k);
            if k1 >= j1
                k2 = k1 - j1 + 1;
                sk(j1, k2) = sk(j1, k2) + s(j, k);
            end
        end %k
    end %j

end %i

% All elements are accounted for.
% Global matrices are now totally assembled.
% Specify known values of phi(blast the matrix)

for i = 1:numnp
    if npbc(i) == 1
        sk(i, 1) = sk(i, 1) * 1E+40;
        q(i) = phi(i) * sk(i, 1);
    end
end %Next i

phi(1:numnp) = q(1:numnp);

% =================================
% =  Solving equations using ULU  =
% =================================
numeq = numnp;
lu = 1;

[ phi ] = udu(sk, phi, numeq, ib, lu);

printheader(fileID,4,nnpe)
for i = 1:numnp
    fprintf(fileID,'%5i %16.5f %13.5f %13.5f %7i\r\n', i, xord(i), yord(i), phi(i), npbc(i) );
end % i

fclose(fileID);

plotcontour(xord, yord, np, nnpe, numel, numnp, phi)


disp('POIS36 analysis complete')

%End of POIS36 fucntion =====================================================
end

function printheader(fileID,np,nnpe)

    switch np
        case 0
            fprintf(fileID,'%s\r\n',' ');
            fprintf(fileID,'%s\r\n', ' ************************************************************');
            fprintf(fileID,'%s\r\n', ' **                                                        **');
            fprintf(fileID,'%s\r\n', ' ***          POIS36 - The University of Memphis          ***');
            fprintf(fileID,'%s\r\n', ' **                                                        **');
            fprintf(fileID,'%s\r\n', ' ************************************************************');
            fprintf(fileID,'%s\r\n',' ');
        case 1
```

```
            fprintf(fileID,'%s\r\n', ' ');
            fprintf(fileID,'%s\r\n', ' --------------------------------------------------');
            fprintf(fileID,'%s\r\n', '  Node             x             y ');
            fprintf(fileID,'%s\r\n', ' --------------------------------------------------');
        case 2
            if nnpe == 3
                fprintf(fileID,'%s\r\n', ' ');
                fprintf(fileID,'%s\r\n', ' --------------------------------------------------');
                fprintf(fileID,'%s\r\n', '  Element     Node1         Node2         Node3');
                fprintf(fileID,'%s\r\n', ' --------------------------------------------------');
            else
                fprintf(fileID,'%s\r\n', ' ');
                fprintf(fileID,'%s\r\n', ' --------------------------------------------------------------------------------------------');
                fprintf(fileID,'%s\r\n', '  Element     Node1         Node2         Node3         Node4         Node5         Node6');
                fprintf(fileID,'%s\r\n', ' --------------------------------------------------------------------------------------------');
            end
        case 3
            fprintf(fileID,'%s\r\n', ' ');
            fprintf(fileID,'%s\r\n', ' -------------------------------------------------------------------');
            fprintf(fileID,'%s\r\n', '  Node         npbc         qs           phi            hs');
            fprintf(fileID,'%s\r\n', ' -------------------------------------------------------------------');
        case 4
            fprintf(fileID,'%s\r\n', ' ');
            fprintf(fileID,'%s\r\n', ' ----------------------------------------------------------------');
            fprintf(fileID,'%s\r\n', '  Node           x            y             phi       npbc');
            fprintf(fileID,'%s\r\n', ' ----------------------------------------------------------------');
    end
end

function [npbc, qs, hs, phi, mat] = init(numnp,numel)

%      This subroutine must be written specifically for each problem
%      for the purpose of specifying nodal values of phi or q
%      which differ from the default value of zero, and for the
%      specification of boundary conditions and surface parameters
%      as designated by the arrays: npbc, qs, and hs.
%      the material number (type) for each element must also be specified
%      in the array mat(i).

phi (1:numnp,1)=0;
qs  (1:numnp,1)=0;
hs  (1:numnp,1)=0;
npbc(1:numnp,1)=0;
mat (1:numel,1)=1;

% Problem p95
    npbc(4:6) = 1;
    phi(1:6)  = 0;
    mat(1:6)  = 1

end


function frx = rx(x, y, lment, mj)
    frx = 1;
end

function fry = ry(x, y, lment, mj)
    fry = 1;
end

function fqv = qv(x, y, lment, mj)
    fqv =1;
end
```