This FEM code is to for solving Laplace's equation with primary current distribution considered in Model 1.
This code is based on FEM weak-form. Biquadratic langrance shape functions (9nodes in an element) are used.

```
> restart;
> with(LinearAlgebra):
> Lx:=1: #length in X
> Ly:=1: #length in Y
> nx:=100: #number of elements in X (even numbers only)
> ny:=100: #number of elements in Y, to be kept same as nx in this
  version
> hx:=Lx/nx: #element size x
> hy:=Ly/ny: #element size y
```

Procedure to perform numerical integration on shape functions to obtain local matrices (can be replaced with analytical integration for this particular problem)
-Shape functions are also used as weight functions in applying weak formulation. Numerical integration is done using Simpson's rule.
-Local cartesian coordinates x,y are converted to natural coordinates zeta and eta. This transformation is not required for this simple geomerty but useful in general. zeta and eta are obtained by scaling x and y with hx/2 and hy/2, respectively, in this code.

```
> localmatrices:=proc(a1,a2,a3,b1,b2,b3,q1,q2)
  global Kx,Ky,Nx,Ny,zeta,eta,c;
  local
  A,dAdzeta,dAdeta,y,x,J,terms,i,j,k,l,dx,dy,fx,fy,fxy,fyy,dzeta,det
  a,J1,J2;
> A:=[(1-zeta)*zeta*(1-eta)*eta/4,-(1-(zeta)^2)*(1-eta)*eta/2,-(1+ze
  ta)*zeta*(1-eta)*eta/4,(1-(eta)^2)*(1+zeta)*zeta/2,(1+zeta)*zeta*(
  1+eta)*eta/4,(1-(zeta)^2)*(1+eta)*eta/2,-(1-zeta)*zeta*(1+eta)*eta
  /4,-(1-(eta)^2)*(1-zeta)*(zeta)/2,(1-(zeta)^2)*(1-(eta)^2)]; #bi
  quadratic langrange shape functions
> dAdzeta:=diff(A,zeta);
> dAdeta:=diff(A,eta);
> y:=[-a1,-a2,-a3,0,a3,a2,a1,0,0];x:=[-b1,0,b1,b2,b3,0,-b3,-b2,0];
> J:=simplify(add(dAdzeta[i]*x[i],i=1..9)*add(dAdeta[i]*y[i],i=1..9)
  -add(dAdzeta[i]*y[i],i=1..9)*add(dAdeta[i]*x[i],i=1..9));
> Nx:=[seq((simplify(add(dAdeta[i]*y[i],i=1..9))*dAdzeta[j]-simplify
  (add(dAdzeta[i]*y[i],i=1..9))*dAdeta[j])/simplify(J),j=1..9)];
> Ny:=[seq((-dAdzeta[j]*simplify(add(dAdeta[i]*x[i],i=1..9))+dAdeta[
  j]*simplify(add(dAdzeta[i]*x[i],i=1..9)))/simplify(J),j=1..9)];
> Kx:=Matrix(nops(A),nops(A),datatype=float[8]):
> Ky:=Matrix(nops(A),nops(A),datatype=float[8]):
```

```
  c:=Vector(nops(A),datatype=float[8]):
> terms:=20:#number of terms for numerical integration
  dzeta:=2/terms:
  deta:=2/terms:
> for i from 1 to nops(Nx) do #loop to obtain local matrices

  for j from 1 to nops(Ny) do
  Kx[i,j]:=0;
  Ky[i,j]:=0;

  for k from 0 to terms do #outer loop double integration,
  integration in zeta

  if k = 0 then fx[k]:= subs(zeta=-1,Nx[i]*Nx[j]*J); fy[k]:=
  subs(zeta=-1,Ny[i]*Ny[j]*J);
  elif k = terms then
  fx[k]:= subs(zeta=-1+(k*dzeta),Nx[i]*Nx[j]*J);
  fy[k]:= subs(zeta=-1+(k*dzeta),Ny[i]*Ny[j]*J);
  elif irem(k,2) = 0 then
  fx[k]:= 2*subs(zeta=-1+(k*dzeta),Nx[i]*Nx[j]*J);
  fy[k]:=     2*subs(zeta=-1+(k*dzeta),Ny[i]*Ny[j]*J);
  else fx[k]:= 4*subs(zeta=-1+(k*dzeta),Nx[i]*Nx[j]*J);
  fy[k]:=     4*subs(zeta=-1+(k*dzeta),Ny[i]*Ny[j]*J);  end if;

  for l from 0 to terms do #inner loop double integration,
  integration in eta

  if l = 0 then fxy[l]:= subs(eta=-1,fx[k]); fyy[l]:=
  subs(eta=-1,fy[k]);
  elif l = terms then fxy[l]:= subs(eta=-1+(l*deta),fx[k]); fyy[l]:=
  subs(eta=-1+(l*deta),fy[k]);
  elif irem(l,2) = 0 then fxy[l]:= 2*subs(eta=-1+(l*deta),fx[k]);
  fyy[l]:= 2*subs(eta=-1+(l*deta),fy[k]);
  else fxy[l]:=4*subs(eta=-1+(l*deta),fx[k]);
  fyy[l]:=4*subs(eta=-1+(l*deta),fy[k]); end if;
  Kx[i,j]:=Kx[i,j]+fxy[l];
  Ky[i,j]:=Ky[i,j]+fyy[l];

  end do;

  end do;
  Kx[i,j]:=Kx[i,j]*dzeta*deta/9;
  Ky[i,j]:=Ky[i,j]*dzeta*deta/9;
```

```
  end do;
  end do:

> end proc:
> n:=nx*ny; #total number of elements
                                  (n := 10000)
> Nx1:=nx*2+1: #number of nodes in x in one row

> N:=Nx1*(2*ny+1): # total number of nodes/equations
> K:=Matrix(N,N,storage=sparse): # global K matrix
> C:=Vector(N,storage=sparse): # global c matrix
> L2G:=Matrix(n,9,datatype=integer):  #mapping matrix-each row has
  node numbers for each element
> l:=1:k:=1:
  localmatrices(hy/2,hy/2,hy/2,hx/2,hx/2,hx/2,0,0):
  kx:=copy(Kx):ky:=copy(Ky):c0:=copy(c):
> for i from 1 to n do #modifying,adding and assembling matrices to
  get global matrix

  if i<=nx/2 then
  a1:=copy(kx); a2:=copy(ky); a3:=0;
  a1[1..3,1..9]:=IdentityMatrix(3,9);
  a2[1..3,1..9]:=Matrix(3,9,shape=zero); a4:=a1+a2; c:=copy(c0);
  c[1..3]:=1.0;

  elif i=nx/2+1 then  a1:=copy(kx); a2:=copy(ky); a3:=0;
  a1[1,1..9]:=IdentityMatrix(1,9);
  a2[1,1..9]:=Matrix(1,9,shape=zero); a4:=a1+a2; c:=copy(c0);
  c[1]:=1.0;

  elif i>nx*(ny-1) then a1:=copy(kx); a2:=copy(ky); a3:=0;
  a1[5..7,5..7]:=IdentityMatrix(3,3);
  a1[5..7,1..4]:=ZeroMatrix(3,4);
  a1[5..7,8..9]:=ZeroMatrix(3,2);
  a2[5..7,1..9]:=Matrix(3,9,shape=zero); a4:=a1+a2; c:=copy(c0);
  c[5..7]:=0;

  else a1:=kx; a2:=ky; a3:=0;a4:=a1+a2; c:=c0;  end if;


  L2G[i,1..9]:=Matrix([l,l+1,l+2,l+2+Nx1,l+2+Nx1*2,l+1+Nx1*2,l+Nx1*2
  ,l+Nx1,l+1+Nx1]):
```

```
   k:=k+1:

   if k>nx then k:=1; l:=l+Nx1+3;

   else l:=l+2; end if:

   indx2:=L2G[i,1..9]:
   indx2:=convert(indx2,list):

   C[indx2]:=C[indx2]+c[1..9];
   c[1..9];

   for i1 from 1 to 9 do
   indx1:=L2G[i,i1]:
   K[indx1,indx2]:=K[indx1,indx2]+a4[i1,1..9]:
   end do:

   end do:
> phi:=LinearSolve(K,C,method=SparseLU): #linear set of equations
   solved using Sparse LU solver
> phi_at(1,0):=phi[Nx1];
```

$$(\mathrm{phi\_at}(1, 0) := 0.541986124530508)$$

```
> dNdy:=copy(Ny):
> dNdy_bottom_left:=subs(eta=-1,zeta=-1,dNdy);
```

$$(\mathrm{dNdy\_bottom\_left} := [-300, 0, 0, 0, 0, 0, -100, 400, 0])$$

```
> current_at(0,0):=add(dNdy_bottom_left[i]*phi[L2G[1,i]],i=1..nops(N
   y));
```

$$(\mathrm{current\_at}(0, 0) := -1.16075988781967)$$

```
> if irem(nx/2,2)=0
   then
   current_at(0,0.25):=add(dNdy_bottom_left[i]*phi[L2G[nx/4+1,i]],i=1
   ..nops(Ny));
   else
   dNdy_bottom_center:=subs(eta=-1,zeta=0,dNdy);

   current_at(0,0.25):=add(dNdy_bottom_center[i]*phi[L2G[(nx/2+1)/2,i
   ]],i=1..nops(Ny));
   end if;
```

$$(\mathrm{current\_at}(0, 0.25) := -1.27343509336214)$$

```
>
>
```