```matlab
1 function fe2dx_nd
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %                              Discussion
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 % 'fe2dx_nd.m'   2D finite element Matlab code for Scheme 1 applied
6 % to the predator-prey system with Kinetics 1. The nodes and elements
7 % of the unstructured grid are loaded from external files 't_triang.dat'
8 % and 'p_coord.dat' respectively, as are the list of nodes on which
9 % Dirichlet and Neumann b.c.'s are to be imposed (from 'bn1_nodes.dat' and
10 % 'bn2_nodes.dat' respectively).
11 %
12 % Boundary conditions:
13 %    Gamma1: Dirichlet
14 %    Gamma2: Neumann
15 %
16 % (C) 2009 Marcus R. Garvie. See 'mycopyright.txt' for details.
17 %
18 % Modified April 7, 2014
19 %
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 %                       Enter data for mesh geometry
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 % Read in 'p(2,n)', the 'n' coordinates of the nodes
24 load p_coord.dat -ascii
25 p = (p_coord)';
26 % Read in 't(3,no_elems)', the list of nodes for 'no_elems' elements
27 load t_triang.dat -ascii
28 t = (round(t_triang))';
29 % Read in 'bn1(1,isn1)', the nodes on Gamma1
30 load bn1_nodes.dat -ascii
31 bn1 = (round(bn1_nodes))';
32 % Read in 'bn2(1,isn2)', the nodes on Gamma2
33 load bn2_nodes.dat -ascii
34 bn2 = (round(bn2_nodes))';
35 % Construct the connectivity for the nodes on Gamma2
36 cpp = subsetconnectivity (t', p', bn2');
37 % Number of edges on Gamma2
38 [e2,junk] = size(cpp);
39 % Degrees of freedom per variable (n)
40 [junk,n]=size(p);
41 % Number of elements (no_elems)
42 [junk,no_elems]=size(t);
43 % Number of nodes on boundary Gamma1 (isn1)
44 [junk,isn1]=size(bn1);
45 % Extract vector of 'x' and 'y' values
46 x = p(1,:); y = p(2,:);
47 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48 %                         Enter data for model
49 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50 % User inputs of parameters
51 alpha = input('Enter parameter alpha   ');
52 beta = input('Enter parameter beta    ');
53 gamma = input('Enter parameter gamma    ');
54 delta = input('Enter parameter delta    ');
55 T = input('Enter maximum time T    ');
```

```matlab
56 delt = input('Enter time-step Delta t   ');
57 % User inputs of initial data
58 u0_str = input('Enter initial data function u0(x,y)   ','s');
59 u0_anon = @(x,y)eval(u0_str);   % create anonymous function
60 u = arrayfun(u0_anon,x,y)';
61 v0_str = input('Enter initial data function v0(x,y)   ','s');
62 v0_anon = @(x,y)eval(v0_str);   % create anonymous function
63 v = arrayfun(v0_anon,x,y)';
64 % Enter the boundary conditions
65 g1u_str = input('Enter the Dirichlet b.c. g1u(x,y,t) for u  ','s');
66 g1u = @(x,y,t)eval(g1u_str);   % create anonymous function
67 g1v_str = input('Enter the Dirichlet b.c. g1v(x,y,t) for v  ','s');
68 g1v = @(x,y,t)eval(g1v_str);   % create anonymous function
69 g2u_str = input('Enter the Neumann b.c. g2u(x,y,t) for u ','s');
70 g2u = @(x,y,t)eval(g2u_str);   % create anonymous function
71 g2v_str = input('Enter the Neumann b.c. g2v(x,y,t) for v ','s');
72 g2v = @(x,y,t)eval(g2v_str);   % create anonymous function
73 % Calculate and assign some constants
74 N=round(T/delt);
75 % Degrees of freedom per variable (n)
76 [junk,n]=size(p);
77 % Number of elements (no_elems)
78 [junk,no_elems]=size(t);
79 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80 %                                    Assembly
81 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82 m_hat=zeros(n,1);
83 K=sparse(n,n);
84 for elem = 1:no_elems
85     % Identify nodes ni, nj and nk in element 'elem'
86     ni = t(1,elem);
87     nj = t(2,elem);
88     nk = t(3,elem);
89     % Identify coordinates of nodes ni, nj and nk
90     xi = p(1,ni);
91     xj = p(1,nj);
92     xk = p(1,nk);
93     yi = p(2,ni);
94     yj = p(2,nj);
95     yk = p(2,nk);
96     % Calculate the area of element 'elem'
97     triangle_area = abs(xj*yk-xk*yj-xi*yk+xk*yi+xi*yj-xj*yi)/2;
98     % Calculate some quantities needed to construct elements in K
99     h1 = (xi-xj)*(yk-yj)-(xk-xj)*(yi-yj);
100     h2 = (xj-xk)*(yi-yk)-(xi-xk)*(yj-yk);
101     h3 = (xk-xi)*(yj-yi)-(xj-xi)*(yk-yi);
102     s1 = (yj-yi)*(yk-yj)+(xi-xj)*(xj-xk);
103     s2 = (yj-yi)*(yi-yk)+(xi-xj)*(xk-xi);
104     s3 = (yk-yj)*(yi-yk)+(xj-xk)*(xk-xi);
105     t1 = (yj-yi)^2+(xi-xj)^2;  % g* changed to t*
106     t2 = (yk-yj)^2+(xj-xk)^2;
107     t3 = (yi-yk)^2+(xk-xi)^2;
108     % Calculate local contributions to m_hat
109     m_hat_i = triangle_area/3;
110     m_hat_j = m_hat_i;
111     m_hat_k = m_hat_i;
112     % calculate local contributions to K
```

```matlab
113     K_ki = triangle_area*s1/(h3*h1);
114     K_ik = K_ki;
115     K_kj = triangle_area*s2/(h3*h2);
116     K_jk = K_kj;
117     K_kk = triangle_area*t1/(h3^2);
118     K_ij = triangle_area*s3/(h1*h2);
119     K_ji = K_ij;
120     K_ii = triangle_area*t2/(h1^2);
121     K_jj = triangle_area*t3/(h2^2);
122     % Add contributions to vector m_hat
123     m_hat(nk)=m_hat(nk)+m_hat_k;
124     m_hat(nj)=m_hat(nj)+m_hat_j;
125     m_hat(ni)=m_hat(ni)+m_hat_i;
126     % Add contributions to K
127     K=K+sparse(nk,ni,K_ki,n,n);
128     K=K+sparse(ni,nk,K_ik,n,n);
129     K=K+sparse(nk,nj,K_kj,n,n);
130     K=K+sparse(nj,nk,K_jk,n,n);
131     K=K+sparse(nk,nk,K_kk,n,n);
132     K=K+sparse(ni,nj,K_ij,n,n);
133     K=K+sparse(nj,ni,K_ji,n,n);
134     K=K+sparse(ni,ni,K_ii,n,n);
135     K=K+sparse(nj,nj,K_jj,n,n);
136 end
137 % Construct matrix L
138 ivec=1:n;
139 IM_hat=sparse(ivec,ivec,1./m_hat,n,n);
140 L=delt*IM_hat*K;
141 % Construct fixed parts of matrices A_{n-1} and C_{n-1}
142 A0=L+sparse(1:n,1:n,1-delt,n,n);
143 C0=delta*L+sparse(1:n,1:n,1+delt*gamma,n,n);
144 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
145 %                         Time-stepping procedure
146 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
147 for nt=1:N
148     tn = nt*delt;
149     % Update coefficient matrices of linear system
150     diag = abs(u);
151     diag_entries = u./(alpha + abs(u));
152     A = A0 + delt*sparse(1:n,1:n,diag,n,n);
153     B = delt*sparse(1:n,1:n,diag_entries,n,n);
154     C = C0 - delt*beta*sparse(1:n,1:n,diag_entries,n,n);
155     % Initialize RHS functions
156     rhs_u = u;
157     rhs_v = v;
158     % Impose Neumann boundary condition on Gamma2
159     for i = 1:e2
160         node1 = cpp(i,1);
161         node2 = cpp(i,2);
162         x1 = p(1,node1);
163         y1 = p(2,node1);
164         x2 = p(1,node2);
165         y2 = p(2,node2);
166         im_hat1 = 1/m_hat(node1);
167         im_hat2 = 1/m_hat(node2);
168         gamma12 = sqrt((x1-x2)^2 + (y1-y2)^2);
169         rhs_u(node1) = rhs_u(node1) + delt*g2u(x1,y1,tn)*im_hat1*gamma12/2;
```

```matlab
170            rhs_u(node2) = rhs_u(node2) + delt*g2u(x2,y2,tn)*im_hat2*gamma12/2;
171            rhs_v(node1) = rhs_v(node1) + delt*g2v(x1,y1,tn)*im_hat1*gamma12/2;
172            rhs_v(node2) = rhs_v(node2) + delt*g2v(x2,y2,tn)*im_hat2*gamma12/2;
173        end
174        % Impose dirichlet boundary conditions on Gamma1
175        for i = 1:isn1
176            node = bn1(i);
177            xx = p(1,node);
178            yy = p(2,node);
179            C(node,:)=0;
180            C(node,node)=1;
181            rhs_v(node)=g1v(xx,yy,tn);
182            A(node,:)=0;
183            A(node,node)=1;
184            B(node,:)=0;
185            rhs_u(node)=g1u(xx,yy,tn);
186        end
187        % Do the incomplete LU factorisation of A and C
188        [LC,UC] = ilu(C,struct('type','ilutp','droptol',1e-5));
189        [LA,UA] = ilu(A,struct('type','ilutp','droptol',1e-5));
190        % Solve for v using GMRES
191        [v,flagv,relresv,iterv]=gmres(C,rhs_v,[],1e-6,[],LC,UC,v);
192        if flagv~=0 flagv,relresv,iterv,error('GMRES did not converge'),end
193        r=rhs_u - B*v;
194        % Solve for u using GMRES
195        [u,flagu,relresu,iteru]=gmres(A,r,[],1e-6,[],LA,UA,u);
196        if flagu~=0 flagu,relresu,iteru,error('GMRES did not converge'),end
197 end
198 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
199 %                              Plot solutions
200 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
201 % Plot solution for u
202 figure;
203 set(gcf,'Renderer','zbuffer');
204 trisurf(t',x,y,u,'FaceColor','interp','EdgeColor','interp');
205 colorbar;axis off;title('u');
206 view ( 2 );
207 axis equal on tight;
208 % Plot solution for v
209 figure;
210 set(gcf,'Renderer','zbuffer');
211 trisurf(t',x,y,v,'FaceColor','interp','EdgeColor','interp');
212 colorbar;axis off;title('v');
213 view ( 2 );
214 axis equal on tight;
```