

[illegible]

```

56 % User inputs of parameters
57 alpha = input('Enter parameter alpha ');
58 beta = input('Enter parameter beta ');
59 gamma = input('Enter parameter gamma ');
60 delta = input('Enter parameter delta ');
61 T = input('Enter maximum time T ');
62 delt = input('Enter time-step Delta t ');
63 % User inputs of initial data
64 u0_str = input('Enter initial data function u0(x,y) ','s');
65 u0_anon = @(x,y)eval(u0_str); % create anonymous function
66 u = arrayfun(u0_anon,x,y)';
67 v0_str = input('Enter initial data function v0(x,y) ','s');
68 v0_anon = @(x,y)eval(v0_str); % create anonymous function
69 v = arrayfun(v0_anon,x,y)';
70 % Enter the boundary conditions
71 k1 = input('Enter the parameter k1 in the Robin b.c. for u ');
72 k2 = input('Enter the parameter k2 in the Robin b.c. for v ');
73 g2u_str = input('Enter the Neumann b.c. g2u(x,y,t) for u ','s');
74 g2u = @(x,y,t)eval(g2u_str); % create anonymous function
75 g2v_str = input('Enter the Neumann b.c. g2v(x,y,t) for v ','s');
76 g2v = @(x,y,t)eval(g2v_str); % create anonymous function
77 % Calculate and assign some constants
78 N=round(T/delt);
79 % Degrees of freedom per variable (n)
80 [junk,n]=size(p);
81 % Number of elements (no_elems)
82 [junk,no_elems]=size(t);
83 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
84 %                               Assembly
85 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
86 m_hat=zeros(n,1);
87 K=sparse(n,n);
88 for elem = 1:no_elems
89     % Identify nodes ni, nj and nk in element 'elem'
90     ni = t(1,elem);
91     nj = t(2,elem);
92     nk = t(3,elem);
93     % Identify coordinates of nodes ni, nj and nk
94     xi = p(1,ni);
95     xj = p(1,nj);
96     xk = p(1,nk);
97     yi = p(2,ni);
98     yj = p(2,nj);
99     yk = p(2,nk);
100    % Calculate the area of element 'elem'
101    triangle_area = abs(xj*yk-xk*yj-xi*yk+xk*yi+xi*yj-xj*yi)/2;
102    % Calculate some quantities needed to construct elements in K
103    h1 = (xi-xj)*(yk-yj)-(xk-xj)*(yi-yj);
104    h2 = (xj-xk)*(yi-yk)-(xi-xk)*(yj-yk);
105    h3 = (xk-xi)*(yj-yi)-(xj-xi)*(yk-yi);
106    s1 = (yj-yi)*(yk-yj)+(xi-xj)*(xj-xk);
107    s2 = (yj-yi)*(yi-yk)+(xi-xj)*(xk-xi);
108    s3 = (yk-yj)*(yi-yk)+(xj-xk)*(xk-xi);
109    t1 = (yj-yi)^2+(xi-xj)^2; % g* changed to t*
110    t2 = (yk-yj)^2+(xj-xk)^2;
111    t3 = (yi-yk)^2+(xk-xi)^2;
112    % Calculate local contributions to m_hat

```

```

113     m_hat_i = triangle_area/3;
114     m_hat_j = m_hat_i;
115     m_hat_k = m_hat_i;
116     % calculate local contributions to K
117     K_ki = triangle_area*s1/(h3*h1);
118     K_ik = K_ki;
119     K_kj = triangle_area*s2/(h3*h2);
120     K_jk = K_kj;
121     K_kk = triangle_area*t1/(h3^2);
122     K_ij = triangle_area*s3/(h1*h2);
123     K_ji = K_ij;
124     K_ii = triangle_area*t2/(h1^2);
125     K_jj = triangle_area*t3/(h2^2);
126     % Add contributions to vector m_hat
127     m_hat(nk)=m_hat(nk)+m_hat_k;
128     m_hat(nj)=m_hat(nj)+m_hat_j;
129     m_hat(ni)=m_hat(ni)+m_hat_i;
130     % Add contributions to K
131     K=K+sparse(nk,ni,K_ki,n,n);
132     K=K+sparse(ni,nk,K_ik,n,n);
133     K=K+sparse(nk,nj,K_kj,n,n);
134     K=K+sparse(nj,nk,K_jk,n,n);
135     K=K+sparse(nk,nk,K_kk,n,n);
136     K=K+sparse(ni,nj,K_ij,n,n);
137     K=K+sparse(nj,ni,K_ji,n,n);
138     K=K+sparse(ni,ni,K_ii,n,n);
139     K=K+sparse(nj,nj,K_jj,n,n);
140 end
141 % Construct matrix L
142 ivec=1:n;
143 IM_hat=sparse(ivec,ivec,1./m_hat,n,n);
144 L=delt*IM_hat*K;
145 % Construct fixed parts of matrices A_{n-1} and C_{n-1}
146 A0=L+sparse(1:n,1:n,1-delt,n,n);
147 C0=delta*L+sparse(1:n,1:n,1+delt*gamma,n,n);
148 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149 %                               Time-stepping procedure
150 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
151 for nt=1:N
152     tn = nt*delt;
153     % Initialize right-hand-side functions
154     rhs_u = u;
155     rhs_v = v;
156     % Update coefficient matrices of linear system
157     diag = abs(u);
158     diag_entries = u./(alpha + abs(u));
159     A = A0 + delt*sparse(1:n,1:n,diag,n,n);
160     B = delt*sparse(1:n,1:n,diag_entries,n,n);
161     C = C0 - delt*beta*sparse(1:n,1:n,diag_entries,n,n);
162     % Do the incomplete LU factorisation of C and A
163     [LC,UC] = ilu(C,struct('type','ilutp','droptol',1e-5));
164     [LA,UA] = ilu(A,struct('type','ilutp','droptol',1e-5));
165     % Impose Robin boundary condition on Gamma1
166     for i = 1:el
167         node1 = cpp1(i,1);
168         node2 = cpp1(i,2);
169         x1 = p(1,node1);

```

```

170     y1 = p(2,node1);
171     x2 = p(1,node2);
172     y2 = p(2,node2);
173     im_hat1 = 1/m_hat(node1);
174     im_hat2 = 1/m_hat(node2);
175     gamma12 = sqrt((x1-x2)^2 + (y1-y2)^2);
176     rhs_u(node1) = rhs_u(node1) + deltt*k1*u(node1)*im_hat1*gamma12/2;
177     rhs_u(node2) = rhs_u(node2) + deltt*k1*u(node2)*im_hat2*gamma12/2;
178     rhs_v(node1) = rhs_v(node1) + deltt*k2*v(node1)*im_hat1*gamma12/2;
179     rhs_v(node2) = rhs_v(node2) + deltt*k2*v(node2)*im_hat2*gamma12/2;
180 end
181 % Impose Neumann boundary condition on Gamma2
182 for i = 1:e2
183     node1 = cpp2(i,1);
184     node2 = cpp2(i,2);
185     x1 = p(1,node1);
186     y1 = p(2,node1);
187     x2 = p(1,node2);
188     y2 = p(2,node2);
189     im_hat1 = 1/m_hat(node1);
190     im_hat2 = 1/m_hat(node2);
191     gamma12 = sqrt((x1-x2)^2 + (y1-y2)^2);
192     rhs_u(node1) = rhs_u(node1) + deltt*g2u(x1,y1,tn)*im_hat1*gamma12/2;
193     rhs_u(node2) = rhs_u(node2) + deltt*g2u(x2,y2,tn)*im_hat2*gamma12/2;
194     rhs_v(node1) = rhs_v(node1) + deltt*g2v(x1,y1,tn)*im_hat1*gamma12/2;
195     rhs_v(node2) = rhs_v(node2) + deltt*g2v(x2,y2,tn)*im_hat2*gamma12/2;
196 end
197 % Solve for v using GMRES
198 [v,flagv,relresv,iterv]=gmres(C,rhs_v,[],1e-6,[],LC,UC,v);
199 if flagv~=0 flagv,relresv,iterv,error('GMRES did not converge'),end
200 r=rhs_u - B*v;
201 % Solve for u using GMRES
202 [u,flagu,relresu,iteru]=gmres(A,r,[],1e-6,[],LA,UA,u);
203 if flagu~=0 flagu,relresu,iteru,error('GMRES did not converge'),end
204 end
205 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
206 %                               Plot solutions
207 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
208 % Plot solution for u
209 figure;
210 set(gcf,'Renderer','zbuffer');
211 trisurf(t',x,y,u,'FaceColor','interp','EdgeColor','interp');
212 colorbar;axis off;title('u');
213 view ( 2 );
214 axis equal on tight;
215 % Plot solution for v
216 figure;
217 set(gcf,'Renderer','zbuffer');
218 trisurf(t',x,y,v,'FaceColor','interp','EdgeColor','interp');
219 colorbar;axis off;title('v');
220 view ( 2 );
221 axis equal on tight;

```

