

```

1  //-----
2  // Figure 3.7 Page 60
3  float Exp_FD_Euro_Call(float K, float T, float S, float sig, float r, float div,
4                          int N, int Nj, float dx)
5  {
6      int i, j;
7      float dt, nu, edx, pu, pm, pd;
8      float St[51], C[51][51];
9
10     dt = T/N;
11     nu = r-div-0.5*sig*sig;
12     edx = exp(dx);
13
14     pu = 0.5*dt*(sig*sig/(dx*dx)+nu/dx);
15     pm = 1.0-dt*(sig*sig/(dx*dx))-r*dt;
16     pd = 0.5*dt*(sig*sig/(dx*dx)-nu/dx);
17
18     St[Id(-Nj)] = S*exp(-Nj*dx);
19     for (j=-Nj+1;j<=Nj;j++)
20     {
21         St[Id(j)] = St[Id(j-1)]*edx;
22     }
23
24     for (j=-Nj;j<=Nj;j++)
25     {
26         C[Id(N)][Id(j)] = max(0, St[Id(j)]-K);
27     }
28
29     for (i=N-1;i>=0;i--)
30     {
31         for (j=-Nj+1;j<=Nj-1;j++)
32         {
33             C[Id(i)][Id(j)] = pu*C[Id(i+1)][Id(j+1)]+pm*C[Id(i+1)][Id(j)]+
34                             pd*C[Id(i+1)][Id(j-1)];
35         }
36         C[Id(i)][Id(-Nj)] = C[Id(i)][Id(-Nj+1)];
37         C[Id(i)][Id(Nj)] = C[Id(i)][Id(Nj-1)]+St[Id(Nj)]-St[Id(Nj-1)];
38     }
39     return( C[Id(0)][Id(0)] );
40 }
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```

```

69  //-----
70  // Figure 3.9 Page 62
71  float Exp_FD_Amer_Put(float K, float T, float S, float sig, float r, float div,
72                        int N, int Nj, float dx)
73  {
74      int i, j;
75      float dt, nu, edx, pu, pm, pd;
76      float St[51], C[51][51];
77
78      dt = T/N;
79      nu = r-div-0.5*sig*sig;
80      edx = exp(dx);
81
82      pu = 0.5*dt*(sig*sig/(dx*dx)+nu/dx);
83      pm = 1.0-dt*(sig*sig/(dx*dx))-r*dt;
84      pd = 0.5*dt*(sig*sig/(dx*dx)-nu/dx);
85
86      St[Id(-Nj)] = S*exp(-Nj*dx);
87      for (j=-Nj+1;j<=Nj;j++)
88      {
89          St[Id(j)] = St[Id(j-1)]*edx;
90      }
91
92      for (j=-Nj;j<=Nj;j++)
93      {
94          C[Id(N)][Id(j)] = max(0, K-St[Id(j)]);
95      }
96
97      for (i=N-1;i>=0;i--)
98      {
99
100         C[Id(i)][Id(Nj)] = C[Id(i)][Id(Nj-1)]+St[Id(Nj)]-St[Id(Nj-1)];
101         C[Id(i)][Id(-Nj)] = C[Id(i)][Id(-Nj+1)];
102
103         for (j=-Nj+1;j<=Nj-1;j++)
104         {
105             C[Id(i)][Id(j)] = pu*C[Id(i+1)][Id(j+1)]+pm*C[Id(i+1)][Id(j)]+
106                             pd*C[Id(i+1)][Id(j-1)];
107         }
108
109         for (j=-Nj;j<=Nj;j++)
110         {
111             C[Id(i)][Id(j)] = max( C[Id(i)][Id(j)], K-St[Id(j)] );
112         }
113     }
114     return( C[Id(0)][Id(0)] );
115 }
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136

```

```

137 //-----
138 // Figure 3.13 Page 69
139 float Imp_FD_Amer_Put(float K, float T, float S, float sig, float r, float div,
140                      int N, int Nj, float dx)
141 {
142     int i, j;
143     float dt, nu, edx, pu, pm, pd;
144     float lamda_L, lamda_U;
145     float St[51], C[51][51];
146
147     void solve_implicit_tridiagonal_system(float C[51][51], float pu, float pm,
148                                           float pd, float lamda_L, float lamda_U, int Nj );
149
150     dt = T/N;
151     nu = r-div-0.5*sig*sig;
152     edx = exp(dx);
153
154     pu = -0.5*dt*(sig*sig/(dx*dx)+nu/dx);
155     pm = 1.0+dt*(sig*sig/(dx*dx))+r*dt;
156     pd = -0.5*dt*(sig*sig/(dx*dx)-nu/dx);
157
158     St[Id(-Nj)] = S*exp(-Nj*dx);
159     for (j=-Nj+1;j<=Nj;j++)
160     {
161         St[Id(j)] = St[Id(j-1)]*edx;
162     }
163
164     for (j=-Nj;j<=Nj;j++)
165     {
166         C[Id(N)][Id(j)] = max(0, K-St[Id(j)]);
167     }
168
169     lamda_L = -1*( St[-Nj+1] - St[-Nj] );
170     lamda_U = 0.0;
171
172     solve_implicit_tridiagonal_system( C, pu, pm, pd, lamda_L, lamda_U, Nj );
173
174     for (i=N-1;i>=0;i--)
175     {
176         for (j=-Nj;j<=Nj;j++)
177         {
178             C[Id(i)][Id(j)] = max( C[Id(i)][Id(j)], K-St[Id(j)] );
179         }
180     }
181     return( C[Id(0)][Id(0)] );
182 }
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204

```

```
205 void solve_implicit_tridiagonal_system(float C[51][51], float pu, float pm, float pd,
206                                         float lamda_L, float lamda_U, int Nj )
207 {
208     int j;
209     float pmp[51], pp[51];
210
211     pmp[Id(-Nj+1)] = pm + pd;
212     pp[Id(-Nj+1)] = C[0][-Nj+1] + pd*lamda_L;
213
214     for(j=-Nj+2; j<=Nj-1; j++)
215     {
216         pmp[Id(j)] = pm-pu*pd/pmp[Id(j-1)];
217         pp[Id(j)] = C[Id(0)][Id(j)]-pp[Id(j-1)]*pd/pmp[Id(j-1)];
218     }
219
220     C[Id(1)][Id(Nj)] = (pp[Id(Nj-1)]+pmp[Id(Nj-1)]*lamda_U)/(pu+pmp[Id(Nj-1)]);
221     C[Id(1)][Id(Nj-1)] = C[Id(1)][Id(Nj)] - lamda_U;
222
223     for(j=Nj-2; j>=-Nj+1; j--)
224     {
225         C[Id(1)][Id(j)] = (pp[Id(j)]-pu*C[Id(1)][Id(j+1)])/pmp[Id(j)];
226     }
227 }
228
```