

Lin/Snyder, *Principles of Parallel Programming*, Figure 7.8, Correct

```
101 do
102 { /*
103  * Send data to four neighbors */
104  int num_requests=0;
105  if(row!=Top)          /* Send North */
106  {
107      MPI_Isend(&val[1][1], Width-2, MPI_FLOAT,
108              NorthPE(myID), tag, MPI_COMM_WORLD, &requests[0]);
109  num_requests++;
110  }
111
112  if(col!=Right)       /* Send East */
113  {
114      for(i=1; i<Height-1; i++)
115      {
116          buffer1[i-1]=val[i][Width-2];
117      }
118      MPI_Isend(buffer1, Height-2, MPI_FLOAT,
119              EastPE(myID), tag, MPI_COMM_WORLD, &requests[1]);
120  num_requests++;
121  }
122
123  if(row!=Bottom)     /* Send South */
124  {
125      MPI_Isend(&val[Height-2][1], Width-2, MPI_FLOAT,
126              SouthPE(myID), tag, MPI_COMM_WORLD, &requests[2]);
127  num_requests++;
128  }
129
130  if(col!=Left)       /* Send West */
131  {
132      for(i=1; i<Height-1; i++)
133      {
134          buffer2[i-1]=val[i][1];
135      }
136      MPI_Isend(buffer2, Height-2, MPI_FLOAT,
137              WestPE(myID), tag, MPI_COMM_WORLD, &requests[3]);
138  num_requests++;
139  }
140
141  /*
142  * Receive messages
143  */
144  if(row!=Top)        /* Receive from North */
145  {
146      MPI_Irecv(&val[0][1], Width-2, MPI_FLOAT,
147              NorthPE(myID), tag, MPI_COMM_WORLD, &requests[4]);
148  num_requests++;
149  }
150
151  if(col!=Right)     /* Receive from East */
152  {
```

```

153     MPI_Irecv(&buffer3, Height-2, MPI_FLOAT,
154             EastPE(myID), tag, MPI_COMM_WORLD, &requests[5]);
155     num_requests++;
156 }
157
158 if(row!=Bottom)      /* Receive from South */
159 {
160     MPI_Irecv(&val[Height-1][1], Width-2, MPI_FLOAT,
161             SouthPE(myID), tag, MPI_COMM_WORLD, &requests[6]);
162     num_requests++;
163 }
164
165 if(col!=Left)        /* Receive from West */
166 {
167     MPI_Irecv(&buffer4, Height-2, MPI_FLOAT,
168             WestPE(myID), tag, MPI_COMM_WORLD, &requests[7]);
169     num_requests++;
170
171     delta=0.0; /* Calculate average, delta for all points */
172     for(i=2; i<Height-2; i++)
173     {
174         for(j=2; j<Width-2; j++)
175         {
176             average=(val[i-1][j]+val[i][j+1]+
177                     val[i+1][j]+val[i][j-1])/4;
178             delta=Max(delta, Abs(average - val[i][j]));
179             new[i][j]=average;
180         }
181     }
182     MPI_Waitall(num_requests, requests, status);
183
184     if(col!=Right)    /* Receive from East */
185     for(i=1; i<Height-1; i++)
186     {
187         val[i][Width-1]=buffer3[i-1];
188     }
189     if(col!=Left)     /* Receive from West */
190     for(i=1; i<Height-1; i++)
191     {
192         val[i][0]=buffer4[i-1];
193     }
194     /* update top and bottom edges, including corners */
195     for(j=1; j<Width-1; j++)
196     {
197         i=1;
198         average=(val[i-1][j]+val[i][j+1]+
199                 val[i+1][j]+val[i][j-1])/4;
200         delta=Max(delta, Abs(average-val[i][j]));
201         new[i][j]=average;
202
203         i=Height-2;
204         average=(val[i-1][j]+val[i][j+1]+
205                 val[i+1][j]+val[i][j-1])/4;

```

```
206     delta=Max(delta, Abs(average-val[i][j]));
207     new[i][j]=average;
208 }
209
210 /* update left and right edges, excluding corners */
211 for(i=2; i<Height-2; i++)
212 {
213     j=1;
214     average=(val[i-1][j]+val[i][j+1]+
215             val[i+1][j]+val[i][j-1])/4;
216     delta=Max(delta, Abs(average - val[i][j]));
217     new[i][j]=average;
218
219     j=Width-2;
220     average=(val[i-1][j]+val[i][j+1]+
221             val[i+1][j]+val[i][j-1])/4;
222     delta=Max(delta, Abs(average-val[i][j]));
223     new[i][j]=average;
224 }
225 /* Find maximum diff */
226 MPI_Reduce(&delta, &globalDelta, 1, MPI_FLOAT, MPI_MAX,
227           RootProcess, MPI_COMM_WORLD);
228 Swap(val, new);
229 } while(globalDelta >= THRESHOLD);
```