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

```
do
{  /*
    * Send data to four neighbors */
  int num_requests=0;
  if(row!=Top)           /* Send North */
  {
    MPI_Isend(&val[1][1], Width-2, MPI_FLOAT,
              NorthPE(myID), tag, MPI_COMM_WORLD, &requests[0]);
  num_requests++;
  }

  if(col!=Right)         /* Send East */
  {
    for(i=1; i<Height-1; i++)
    {
      buffer1[i-1]=val[i][Width-2];
    }
    MPI_Isend(buffer1, Height-2, MPI_FLOAT,
              EastPE(myID), tag, MPI_COMM_WORLD, &requests[1]);
    num_requests++;
  }

  if(row!=Bottom)        /* Send South */
  {
    MPI_Isend(&val[Height-2][1], Width-2, MPI_FLOAT,
              SouthPE(myID), tag, MPI_COMM_WORLD, &requests[2]);
    num_requests++;
  }

  if(col!=Left)          /* Send West */
  {
    for(i=1; i<Height-1; i++)
    {
      buffer2[i-1]=val[i][1];
    }
    MPI_Isend(buffer2, Height-2, MPI_FLOAT,
              WestPE(myID), tag, MPI_COMM_WORLD, &requests[3]);
  num_requests++;
  }

  /*
   * Receive messages
   */
  if(row!=Top)           /* Receive from North */
  {
    MPI_Irecv(&val[0][1], Width-2, MPI_FLOAT,
              NorthPE(myID), tag, MPI_COMM_WORLD, &requests[4]);
  num_requests++;
  }

  if(col!=Right)         /* Receive from East */
  {
    MPI_Irecv(&buffer3, Height-2, MPI_FLOAT,
              EastPE(myID), tag, MPI_COMM_WORLD, &requests[5]);
    num_requests++;
  }

  if(row!=Bottom)        /* Receive from South */
  {
    MPI_Irecv(&val[Height-1][1], Width-2, MPI_FLOAT,
             SouthPE(myID), tag, MPI_COMM_WORLD, &requests[6]);
  num_requests++;
  }

  if(col!=Left)          /* Receive from West */
  {
    MPI_Irecv(&buffer4, Height-2, MPI_FLOAT,
              WestPE(myID), tag, MPI_COMM_WORLD, &requests[7]);
  num_requests++;

  delta=0.0; /* Calculate average, delta for all points */
  for(i=2; i<Height-2; i++)
  {
    for(j=2; j<Width-2; j++)
    {
      average=(val[i-1][j]+val[i][j+1]+
               val[i+1][j]+val[i][j-1])/4;
      delta=Max(delta, Abs(average - val[i][j]));
      new[i][j]=average;
    }
  }
  MPI_Waitall(num_requests, requests, status);

  if(col!=Right)         /* Receive from East */
    for(i=1; i<Height-1; i++)
    {
      val[i][Width-1]=buffer3[i-1];
    }
  if(col!=Left)          /* Receive from West */
    for(i=1; i<Height-1; i++)
    {
      val[i][0]=buffer4[i-1];
  }
  /* update top and bottom edges, including corners */
  for(j=1; j<Width-1; j++)
  {
    i=1;
    average=(val[i-1][j]+val[i][j+1]+
           val[i+1][j]+val[i][j-1])/4;
    delta=Max(delta, Abs(average-val[i][j]));
    new[i][j]=average;

    i=Height-2;
    average=(val[i-1][j]+val[i][j+1]+
           val[i+1][j]+val[i][j-1])/4;
    delta=Max(delta, Abs(average-val[i][j]));
    new[i][j]=average;
  }

  /* update left and right edges, excluding corners */
  for(i=2; i<Height-2; i++)
  {
    j=1;
    average=(val[i-1][j]+val[i][j+1]+
              val[i+1][j]+val[i][j-1])/4;
    delta=Max(delta, Abs(average - val[i][j]));
    new[i][j]=average;

    j=Width-2;
    average=(val[i-1][j]+val[i][j+1]+
              val[i+1][j]+val[i][j-1])/4;
    delta=Max(delta, Abs(average-val[i][j]));
    new[i][j]=average;
  }
  /* Find maximum diff */
  MPI_Reduce(&delta, &globalDelta, 1, MPI_FLOAT, MPI_MAX,
             RootProcess, MPI_COMM_WORLD);
  Swap(val, new);
} while(globalDelta >= THRESHOLD);
```