```
real grid[0:n+1,0:n+1], new[0:n+1,0:n+1];
int HEIGHT = n/PR;   # assume PR evenly divides n
real maxdiff[1:PR] = ([PR] 0.0);

procedure barrier(int id) {
  # efficient barrier algorithm from Section 3.4
}

process worker[w = 1 to PR] {
  int firstRow = (w-1)*HEIGHT + 1;
  int lastRow = firstRow + HEIGHT - 1;
  real mydiff = 0.0;
  initialize my strips of grid and new, including boundaries;
  barrier(w);
  for [iters = 1 to MAXITERS by 2] {
    # compute new values for my strip
    for [i = firstRow to lastRow, j = 1 to n]
      new[i,j] = (grid[i-1,j] + grid[i+1,j] +
                  grid[i,j-1] + grid[i,j+1]) * 0.25;
    barrier(w);
    # compute new values again for my strip
    for [i = firstRow to lastRow, j = 1 to n]
      grid[i,j] = (new[i-1,j] + new[i+1,j] +
                   new[i,j-1] + new[i,j+1]) * 0.25;
    barrier(w);
  }
  # compute maximum difference for my strip
  for [i = firstRow to lastRow, j = 1 to n]
    mydiff = max(mydiff, abs(grid[i,j]-new[i,j]));
  maxdiff[w] = mydiff;
  barrier(w);
  # maximum difference is the max of the maxdiff[*]
}
```

Figure 11.3 Jacobi iteration using shared variables.