Program burn3d.c

/* This software was developed at the National Institute of */
/* Standards and Technology by employees of the Federal Government */
/* in the course of their official duties. Pursuant to title 17 */
/* Section 105 of the United States Code this software is not */
/* subject to copyright protection and is in the public domain. */
/* NIST assumes no */
/* responsibility whatsoever for its use by other parties, and */
/* makes no guarantees, expressed or implied, about its quality, */
/* reliability, or any other characteristic. We would appreciate */
/* acknowledge if the software is used. This software can be */
/* redistributed and/or modified freely provided that any */
/* derivative works bear some notice that they are derived from it, */
/* and any modified versions bear some notice that they have been */
/* modified. */

#define BURNT 70    /* label for a burnt pixel */
#define SIZE2D 49000 /* size of matrices for holding burning locations */
/* functions defining coordinates for burning in any of three directions */
#define cx(x,y,z,a,b,c) ((1-b-c)*x+(1-a-c)*y+(1-a-b)*z)
#define cy(x,y,z,a,b,c) ((1-a-b)*x+(1-b-c)*y+(1-a-c)*z)
#define cz(x,y,z,a,b,c) ((1-a-c)*x+(1-a-b)*y+(1-b-c)*z)

/* routine to assess the connectivity (percolation) of a single phase */
/* Two matrices are used here: one to store the recently burnt locations */
/* the other to store the newly found burnt locations */
int burn3d(npix,d1,d2,d3)
    int npix;    /* ID of phase to perform burning on */
    int d1,d2,d3;   /* directional flags */
{    
    long ntop,nthrough,ncur,nnew,ntot,nphc;
    int i,inew,j,k,nmatx[SIZE2D],nmaty[SIZE2D],nmatz[SIZE2D];
    int x1,xh,j1,k1,px,py,pz,qx,qy,qz,xcn,ycn,zcn;
    int x1,y1,z1,igood,nnewx[SIZE2D],nnewy[SIZE2D],nnewz[SIZE2D];
    int jnew,icur;
    int bflag;
    float mass_burn=0.0,alpha_burn=0.0,con_frac;
    FILE *fileperc;

    /* counters for number of pixels of phase accessible from surface #1 */
    /* and number which are part of a percolated pathway to surface #2 */
    ntop=0;
    bflag=0;
    nthrough=0;
    nphc=0;

    /* percolation is assessed from top to bottom only */
    /* and burning algorithm is periodic in other two directions */
    /* use of directional flags allow transformation of coordinates */
    /* to burn in direction of choosing (x, y, or z) */
    i=0;

    for(k=0;k<SYSIZE;k++){
for (j = 0; j < SYSIZE; j++) {
    igood = 0;
    ncur = 0;
    ntot = 0;
    /* Transform coordinates */
    px = cx(i, j, k, d1, d2, d3);
    py = cy(i, j, k, d1, d2, d3);
    pz = cz(i, j, k, d1, d2, d3);
    if (mic[px][py][pz] == npix) {
        /* Start a burn front */
        mic[px][py][pz] = BURNT;
        ntot += 1;
        ncur += 1;
        /* burn front is stored in matrices nmat* */
        /* and nnew* */
        nmatx[ncur] = i;
        nmaty[ncur] = j;
        nmatz[ncur] = k;
        /* Burn as long as new (fuel) pixels are found */
        do {
            nnew = 0;
            for (inew = 1; inew <= ncur; inew++) {
                xcn = nmatx[inew];
                ycn = nmaty[inew];
                zcn = nmatz[inew];
                /* Check all six neighbors */
                for (jnew = 1; jnew <= 6; jnew++) {
                    x1 = xcn;
                    y1 = ycn;
                    z1 = zcn;
                    if (jnew == 1) {x1 -= 1;}
                    if (jnew == 2) {x1 += 1;}
                    if (jnew == 3) {y1 -= 1;}
                    if (jnew == 4) {y1 += 1;}
                    if (jnew == 5) {z1 -= 1;}
                    if (jnew == 6) {z1 += 1;}
                    if ((x1 >= 0) && (x1 < SYSIZE)) {
                        /* Transform coordinates */
                        px = cx(x1, y1, z1, d1, d2, d3);
                        py = cy(x1, y1, z1, d1, d2, d3);
                        pz = cz(x1, y1, z1, d1, d2, d3);
                        if (mic[px][py][pz] == npix) {
                            ntot += 1;
                            mic[px][py][pz] = BURNT;
                            nnew += 1;
                            if (nnew >= SIZE2D) {
                                printf("error in size of nnew \n");
                            }
                        }
                    }
                }
            }
        } while (nnew > 0);
    }
}

218
nnewx[nnew]=xl;
nnewy[nnew]=yl;
nnewz[nnew]=zl;
}
}
}
if(nnew>0){
ncur=nnew;
/* update the burn front matrices */
for(icur=1;icur<=ncur;icur++){
  nmatx[icur]=nnewx[icur];
nmaty[icur]=nnewy[icur];
nmatz[icur]=nnewz[icur];
}
}
while (nnew>0);
ntop+=ntot;
xl=0;
xh=SYSIZE-1;
/* See if current path extends through the microstructure */
for(j1=0;j1<SYSIZE;j1++){
  for(k1=0;k1<SYSIZE;k1++){
    px=cx(xl,j1,k1,d1,d2,d3);
    py=cy(xl,j1,k1,d1,d2,d3);
    pz=cz(xl,j1,k1,d1,d2,d3);
    qx=cx(xh,j1,k1,d1,d2,d3);
    qy=cy(xh,j1,k1,d1,d2,d3);
    qz=cz(xh,j1,k1,d1,d2,d3);
    if((mic [px] [py] [pz]==BURNT)&&(mic [qx] [qy] [qz]==BURNT)){
      igood=2;
    }
    if(mic [px] [py] [pz]==BURNT){
      mic [px] [py] [pz]=BURNT+1;
    }
    if(mic [qx] [qy] [qz]==BURNT){
      mic [qx] [qy] [qz]=BURNT+1;
    }
  }
}
if(igood==2){
  nthrough+=ntot;
}
/* return the burnt sites to their original phase values */
for(i=0;i<SYSIZE;i++){
  for(j=0;j<SYSIZE;j++){
    for(k=0;k<SYSIZE;k++){
      if(mic [i] [j] [k]>=BURNT){
        nphc+=1;
        mic [i] [j] [k]=npix;
      }
    }
  }
}
else if(mic[i][j][k]==npix){
    nphc+=1;
}

printf("Phase ID= %d \n",npix);
printf("Number accessible from first surface = %ld \n",ntop);
printf("Number contained in through pathways= %ld \n",nthrough);
fileperc=fopen(ppsname,"a");
mass_burn+=specgrav[C3S]*count[C3S];
mass_burn+=specgrav[C2S]*count[C2S];
mass_burn+=specgrav[C3A]*count[C3A];
mass_burn+=specgrav[C4AF]*count[C4AF];
alpha_burn=1.-(mass_burn/cemmass);
con_frac=0.0;
if(nphc>0){
    con_frac=(float)nthrough/(float)nphc;
}
fprintf(fileperc,"%ld %f %f %ld %ld %f\n",cyccnt,time_cur+(2.*(float)(cyccnt)-1.0)*beta/krate,alpha_burn,nthrough,nphc,con_frac);
fclose(fileperc);
if(nthrough>0){
    bflag=1;
}
return(bflag);