APÉNDICE

CÓDIGO DEL PROGRAMA EMPLEADO PARA LA SIMULACIÓN DE RIPPLES

```
Function[Hnew,M]=ripples2_original(H,hopX,windX,hopY,windY,grain,gravity,critAng,
numsteps, dim1,dim2)

[rows cols] = size (H);
Heven = H;
Hodd = H;
for currstep = 1:numsteps

Heven = Hodd;
for x = 1:rows
        for y = 1:cols

%Aquí se definen las coordenadas de los puntos vecinos al punto dado

if (x==1)
        xUp=rows;
else
        xUp = x-1;
end

if (x==rows)
        xDown=1;
else
        xDown = x+1;
end

if (y==1)
        yLeft=cols;
else
        yLeft = y-1;
end

if (y==cols)
        yRt=1;
else
        yRt = y+1;
end

%Aquí define los valores de la altura de la celda de centro (x,y) y sus 8 vecinos
%correspondeintes a las celdas adyacentes a los cuatro lados y los cuatro vértices.

h = Hodd(x,y);
hR = Hodd(x,yRt); hRD = Hodd(xDown,yRt); hRU = Hodd(xUp,yRt);
hL = Hodd(x,yLeft); hLD = Hodd(xDown,yLeft); hLU = Hodd(xUp,yLeft);
hD = Hodd(xDown,y); hU = Hodd(xUp,y);
```

```
%%%%%%%%%%%%%%%%%%%%
% SALTACIÓN

%Se definen los gradientes

delHx = Hodd(xDown,y)-Hodd(x,y);
delHy = Hodd(x,yRt) - Hodd(x,y);
delH  = sign(delHx)*sqrt(delHx^2 + delHy^2);

%La longitud de salto depende de la altura y la pendiente
hopLengthX = (hopX + windX*Hodd(x,y))*(1-tanh(delHx));
hopLengthY = (hopY + windY*Hodd(x,y))*(1-tanh(delHy));

%La cantidad de granos transportada depende de la pendiente y de delH
grainAmt = (grain)*(1+tanh(delH));

%se define el punto de aterrizaje de los granos
blowToX = round(x + hopLengthX);
blowToY = round(y + hopLengthY);

%Cuando se acaba el dominio vuelve al inicio
if (blowToX > rows)
blowToX = blowToX - rows;
end

if (blowToY > cols)
blowToY = blowToY - cols;
end

Heven(x,y) = Heven(x,y) - grainAmt;
Heven(blowToX,blowToY) = Heven(blowToX,blowToY) + grainAmt;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REPTACIÓN debida a la gravedad

%Aqui se hace una suma ponderada de los 4 vecinos al punto en perpendicular
FirstNbrSum = 1/6*( hU + hD + hL + hR );

%Aqui se hace una suma ponderada de los 4 vecinos de las esquinas
SecondNbrSum = 1/12*( hLU + hRU + hLD + hRD);

Heven(x,y) = Heven(x,y) + gravity*(FirstNbrSum+SecondNbrSum - h);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%AVALANCHA si la pendiente sobrepasa el ángulo crítico en reposo

sandIndex = [];
b = [];
```

```
if ((h-hR)/1 > tan(critAng))
        sandIndex = [sandIndex;1];
        b = [b;tan(critAng)+hR-h];
end

f (h-hRD)/(sqrt(2)) > tan(critAng)
        sandIndex = [sandIndex;2];
        b = [b;sqrt(2)*tan(critAng)+hRD-h];
end

if (h-hD)/1 > tan(critAng)
        sandIndex = [sandIndex;3];
        b = [b;tan(critAng)+hD-h];
end

if (h-hLD)/(sqrt(2)) > tan(critAng)
        sandIndex = [sandIndex;4];
        b = [b;sqrt(2)*tan(critAng)+hLD-h];
end

n = length(b);
A = -ones(n);

for i = 1:n
        A(i,i)= -2;
end

sandShift = A\b;

        for j = 1:n
            if sandIndex(j) == 1
                Heven(x,yRt)=hR + sandShift(j);
            elseif sandIndex(j) == 2
                Heven(xDown,yRt)=hRD + sandShift(j);
            elseif sandIndex(j) == 3
                Heven(xDown,y)=hD + sandShift(j);
            elseif sandIndex(j) == 4
                Heven(xDown,yLeft)=hLD + sandShift(j);
            end
         end

         Heven(x,y)-sum(sandShift);

         end
      end
      Hodd = Heven;
```

```
        meshz(Hodd)
        axis([0 100 0 100 0 50])
        pause(.01)

        if mod(currstep,3)==0
            M(currstep/3)=getframe;
        end
end
Hnew = Hodd;
```