FORFIRE

Table of Contents

```
function [data]= ForFire (GSize, f, p, t, d_rate,output)
```

```
% Syntax: ForFire(Grid Size, Lightning parameter, Growth parameter,
% Simulation time, Diagnostics rate, Output option)
```

Input variables

The input variables are: GSize (for creating a grid GSize x GSize f, the fire parameter (chance of spontaneous fire) p, the growth parameter t, the simulation time d_rate, the diagnostics rate (set it to 50 or more for performance issues) output, the option of having a command window output (set it to 1) which tells the current state of the simulation, or having no output (set it to 0) which is much more convenient when doing parameter sweeps. Note that the output does not have a relevant effect on the simulation duration.

```
if output
fprintf('Initializing Simulation...\n');
tic
end
```

```
Initializing Simulation...
```

Initialization

Both the original and the shadow grid are initialized empty and global. The original grid contains the information about the state of a cell The shadow grid contains the information about the cluster Index of a cell.

```
global grid
grid=zeros(GSize);
global shgrid
shgrid=grid;
% The index matrix contains all the possible indexes and their availability
% where 1 means that the index is already in use and 0 means that the index
% is currently not in use
% There are GSize^2/2 possible clusters (in a perfect
% chessboard-configuration)
% Index(1,k) returns the k-th index
% Index(2,k) returns the availability
indexes=1:ceil(GSize^2/2);
availability=zeros(1,ceil(GSize^2/2));
global Index
Index=[indexes;availability];

% initialize the vector which holds all the sizes of all the clusters over
% all timesteps. Since we do not know the amount of clusters we are going
% to get, it is impossible to initialize it to the correct size. The
% Performance issues are marginal.
sizevec=0;
% initialize the vector which holds all the radii of all the clusters over
% all timesteps. Same problem as above applies.
radvec=0;
% Initialize the vector which will store the amount of alive trees over all
% timesteps. Since we only measure this property at distinct intervals, its
% length is well known.
Ntrees=zeros(1,t/d_rate);
% Ugly help variable for later...
op=0;
```

Loop over all timesteps

```
for i=1:t

    % Pick a random cell to work on
    x=ceil(GSize*rand());
    y=ceil(GSize*rand());
    % Generate random number for growth and fire possibilities
    valrand=rand();
    % Check if site is empty
    if grid(x,y)==0
        % Check if growth is possible
        if valrand<p
            % Grow a tree
            grid(x,y)=1;
            % Updating the shadowgrid
            [a,b,c,d]=GetNcIndex(x,y);
            % if all neighbors are empty, set the index to the smallest
            % possible
            if ((a+b+c+d)==0)
                shgrid(x,y)=lowIndex();
                %fprintf('cell at %d,%d has no indexed neighbors and has been set
            end
            % if only one neighbor is already indexed
            % note that the program does NOT need to know which one it is,
            % since the condition is satisfied iff exactly one of them is
            % different from zero which automatically gives the max value
            if ((max([a,b,c,d])==(a+b+c+d) )&& max([a,b,c,d])~=0)
                shgrid(x,y)=max([a,b,c,d]);
                %fprintf('cell at %d, %d has one indexed neighbor and has been set
            end
            % if more than one neighbors are already indexed, the lowest
            % will be chosen and the other indexed cluster will be changed
            % to this one too
            if (max([a,b,c,d])<(a+b+c+d))
                % check for empty ones
                % the resulting vector cInd holds all the relevant indexes.
                % meaning that the empty ones are thrown out
                %fprintf('cell at %d,%d has more than two indexed neighbors ', x,y
                cInd=0;
                if a>0
                    cInd=a;
```

```
             end
             if b>0
                 cInd=[cInd b];
             end
             if c>0
                 cInd=[cInd c];
             end
             if d>0
                 cInd=[cInd d];
             end
             % ugly workaround to check if a was zero --> gives nasty
             % errors if not checked.
             if cInd(1)==0
                 cInd=cInd(2:size(cInd,2));
             end
             % set the current grid cell to the lowest available Index
             shgrid(x,y)=min(cInd);


             % for all the other clusters, the indices are changed to
             % the new, common index.
             for k=1:size(cInd,2)
                 changeIndex(cInd(k),min(cInd));
             end
         end
     end
 end
 %check if Grid point is a tree
 if grid(x,y)==1
     % if the condition for fire is satisfied
     if valrand<f
         % burn the cluster. See description in the function for closer
         % insight.
     clusterburn(x,y);
     end
 end

 % Since the diagnostics function is by far the most time-consuming part
 % of the simulation, it is only called at distinct timesteps with a
 % period of d_rate.
 if mod(i,d_rate)==0
     % run the diagnostics and retrieve the data.
 [cSize,cRadius,N]=diagnostics();
 % store the data in a bigger vector. It is not necessary to sort this,
 % since it will only be used in histograms later on. Note that it is
 % impossible to determine the size of these vectors at the beginning of
 % the program, therefore they change their size in every loop. The
 % effects on the overall performance of the program are though
 % marginal.
sizevec=[sizevec; cSize];
radvec=[radvec; cRadius];
Ntrees(i/d_rate)=N;
% for every percent the simulation advances, there is a print at the
% command window. this can be suppressed with setting the output option
% to zero.
op=op+1;
if mod(op,t/d_rate/100)==0
    if output
fprintf('Simulation is at %d percent, estimated time left: %d seconds\n', round
    end
op=0;
end
 end
```

```
  % commented out section for live-plotting of the grid, the shadow grid
  % and histograms of the radius and size distribution
%    subplot(2,2,1);
%    image(shgrid*3);
%    subplot(2,2,2);
%    image(10*grid);
%    subplot(2,2,3);
%    hist(cRadius);
%    subplot(2,2,4);
%    hist(cSize);
%    getframe;
%    end
  %
  %


end

% applying a low-pass filter to the tree evolution data for smoother
% visualization
fourierTrees=fft(Ntrees);
for i=50:size(fourierTrees,2)
    fourierTrees(i)=0;
end
cleanTrees=ifft(fourierTrees);
```

```
Simulation is at 1 percent, estimated time left: 13 seconds
Simulation is at 2 percent, estimated time left: 17 seconds
Simulation is at 3 percent, estimated time left: 21 seconds
Simulation is at 4 percent, estimated time left: 23 seconds
Simulation is at 5 percent, estimated time left: 25 seconds
Simulation is at 6 percent, estimated time left: 25 seconds
Simulation is at 7 percent, estimated time left: 25 seconds
Simulation is at 8 percent, estimated time left: 24 seconds
Simulation is at 9 percent, estimated time left: 23 seconds
Simulation is at 10 percent, estimated time left: 22 seconds
Simulation is at 11 percent, estimated time left: 21 seconds
Simulation is at 12 percent, estimated time left: 19 seconds
Simulation is at 13 percent, estimated time left: 19 seconds
Simulation is at 14 percent, estimated time left: 18 seconds
Simulation is at 15 percent, estimated time left: 18 seconds
Simulation is at 16 percent, estimated time left: 18 seconds
Simulation is at 17 percent, estimated time left: 18 seconds
Simulation is at 18 percent, estimated time left: 18 seconds
Simulation is at 19 percent, estimated time left: 18 seconds
Simulation is at 20 percent, estimated time left: 18 seconds
Simulation is at 21 percent, estimated time left: 17 seconds
Simulation is at 22 percent, estimated time left: 17 seconds
Simulation is at 23 percent, estimated time left: 16 seconds
Simulation is at 24 percent, estimated time left: 16 seconds
Simulation is at 25 percent, estimated time left: 16 seconds
Simulation is at 26 percent, estimated time left: 16 seconds
Simulation is at 27 percent, estimated time left: 15 seconds
Simulation is at 28 percent, estimated time left: 15 seconds
Simulation is at 29 percent, estimated time left: 15 seconds
Simulation is at 30 percent, estimated time left: 15 seconds
Simulation is at 31 percent, estimated time left: 14 seconds
Simulation is at 32 percent, estimated time left: 14 seconds
Simulation is at 33 percent, estimated time left: 14 seconds
Simulation is at 34 percent, estimated time left: 14 seconds
Simulation is at 35 percent, estimated time left: 14 seconds
Simulation is at 36 percent, estimated time left: 14 seconds
Simulation is at 37 percent, estimated time left: 13 seconds
Simulation is at 38 percent, estimated time left: 13 seconds
Simulation is at 39 percent, estimated time left: 13 seconds
Simulation is at 40 percent, estimated time left: 13 seconds
Simulation is at 41 percent, estimated time left: 12 seconds
Simulation is at 42 percent, estimated time left: 12 seconds
Simulation is at 43 percent, estimated time left: 12 seconds
Simulation is at 44 percent, estimated time left: 12 seconds
Simulation is at 45 percent, estimated time left: 11 seconds
Simulation is at 46 percent, estimated time left: 11 seconds
Simulation is at 47 percent, estimated time left: 11 seconds
Simulation is at 48 percent, estimated time left: 11 seconds
Simulation is at 49 percent, estimated time left: 11 seconds
Simulation is at 50 percent, estimated time left: 10 seconds
Simulation is at 51 percent, estimated time left: 10 seconds
Simulation is at 52 percent, estimated time left: 10 seconds
Simulation is at 53 percent, estimated time left: 10 seconds
Simulation is at 54 percent, estimated time left: 10 seconds
Simulation is at 55 percent, estimated time left: 9 seconds
Simulation is at 56 percent, estimated time left: 9 seconds
Simulation is at 57 percent, estimated time left: 9 seconds
Simulation is at 58 percent, estimated time left: 9 seconds
Simulation is at 59 percent, estimated time left: 9 seconds
Simulation is at 60 percent, estimated time left: 9 seconds
Simulation is at 61 percent, estimated time left: 8 seconds
Simulation is at 62 percent, estimated time left: 8 seconds
Simulation is at 63 percent, estimated time left: 8 seconds
Simulation is at 64 percent, estimated time left: 8 seconds
Simulation is at 65 percent, estimated time left: 7 seconds
Simulation is at 66 percent, estimated time left: 7 seconds
Simulation is at 67 percent, estimated time left: 7 seconds
Simulation is at 68 percent, estimated time left: 7 seconds
Simulation is at 69 percent, estimated time left: 7 seconds
Simulation is at 70 percent, estimated time left: 6 seconds
Simulation is at 71 percent, estimated time left: 6 seconds
Simulation is at 72 percent, estimated time left: 6 seconds
Simulation is at 73 percent, estimated time left: 6 seconds
Simulation is at 74 percent, estimated time left: 6 seconds
Simulation is at 75 percent, estimated time left: 5 seconds
Simulation is at 76 percent, estimated time left: 5 seconds
Simulation is at 77 percent, estimated time left: 5 seconds
Simulation is at 78 percent, estimated time left: 5 seconds
Simulation is at 79 percent, estimated time left: 5 seconds
Simulation is at 80 percent, estimated time left: 4 seconds
Simulation is at 81 percent, estimated time left: 4 seconds
Simulation is at 82 percent, estimated time left: 4 seconds
Simulation is at 83 percent, estimated time left: 4 seconds
Simulation is at 84 percent, estimated time left: 3 seconds
Simulation is at 85 percent, estimated time left: 3 seconds
Simulation is at 86 percent, estimated time left: 3 seconds
Simulation is at 87 percent, estimated time left: 3 seconds
Simulation is at 88 percent, estimated time left: 3 seconds
Simulation is at 89 percent, estimated time left: 2 seconds
Simulation is at 90 percent, estimated time left: 2 seconds
Simulation is at 91 percent, estimated time left: 2 seconds
Simulation is at 92 percent, estimated time left: 2 seconds
Simulation is at 93 percent, estimated time left: 2 seconds
Simulation is at 94 percent, estimated time left: 1 seconds
Simulation is at 95 percent, estimated time left: 1 seconds
Simulation is at 96 percent, estimated time left: 1 seconds
Simulation is at 97 percent, estimated time left: 1 seconds
Simulation is at 98 percent, estimated time left: 0 seconds
Simulation is at 99 percent, estimated time left: 0 seconds
Simulation is at 100 percent, estimated time left: 0 seconds
```

Data Preview

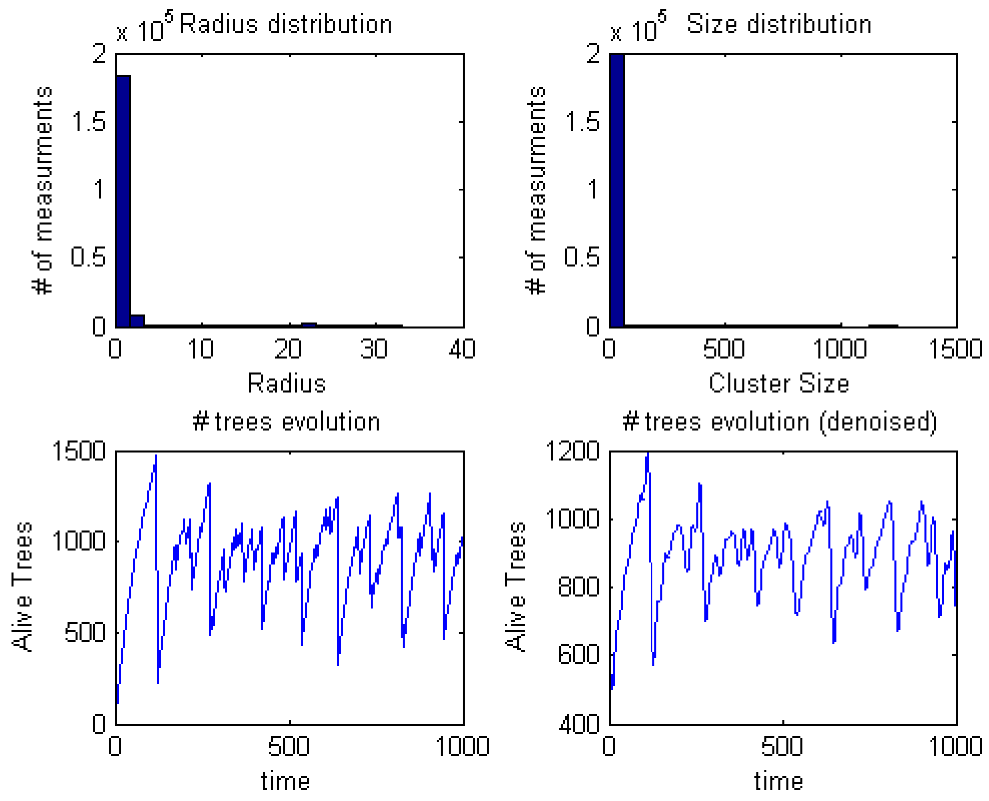
```
if output
fprintf('preparing data preview...\n');
    figure('Name','Data Preview');

    subplot(2,2,1);
    hist(radvec,20);
    title('Radius distribution');
    xlabel('Radius');
    ylabel('# of measurments');
    subplot(2,2,2);
    hist(sizevec,20);
    title('Size distribution');
    xlabel('Cluster Size');
    ylabel('# of measurments');
    subplot(2,2,3);
    plot(Ntrees);
    title('# trees evolution');
    xlabel('time'); % note that this is not the simulation time but the time divid
    ylabel('Alive Trees');
    subplot(2,2,4);
    plot(real(cleanTrees));
    title('# trees evolution (denoised)')
    xlabel('time');
    ylabel('Alive Trees');

    fprintf('saving data...\n');

end
```

```
preparing data preview...
saving data...
```



Saving Data in a custom struct

```
data=struct('rad_dist',radvec,'size_dist',sizevec,'trees',Ntrees,'d_period',d_rate
Theta=p/f;
a=num2str(GSize);
b=num2str(Theta);
c=num2str(t);
i=0;
cd Results
% making a custom folder with information about the grid size, theta and
% the simulation time
folder_name=[a '_' b '_' c];
% if such a simulation has already been run, it just adds increasing
% numbers to the end of the folder name.
while exist(folder_name)==7
    d=num2str(i);
    folder_name=[a '_' b '_' c '_' d];
    i=i+1;
end
mkdir (folder_name)
cd (folder_name)
save(folder_name,'data');
cd ..
cd ..
if output
fprintf('simulation finished. \n');
end
```

```
simulation finished.
```

```
end
```

```
ans =

     rad_dist: [200791x1 double]
    size_dist: [200791x1 double]
        trees: [1x1000 double]
     d_period: 20
```

Published with MATLAB® 7.10