Homework 1.(b)

Modeling Complex Systems, Javier Lobato

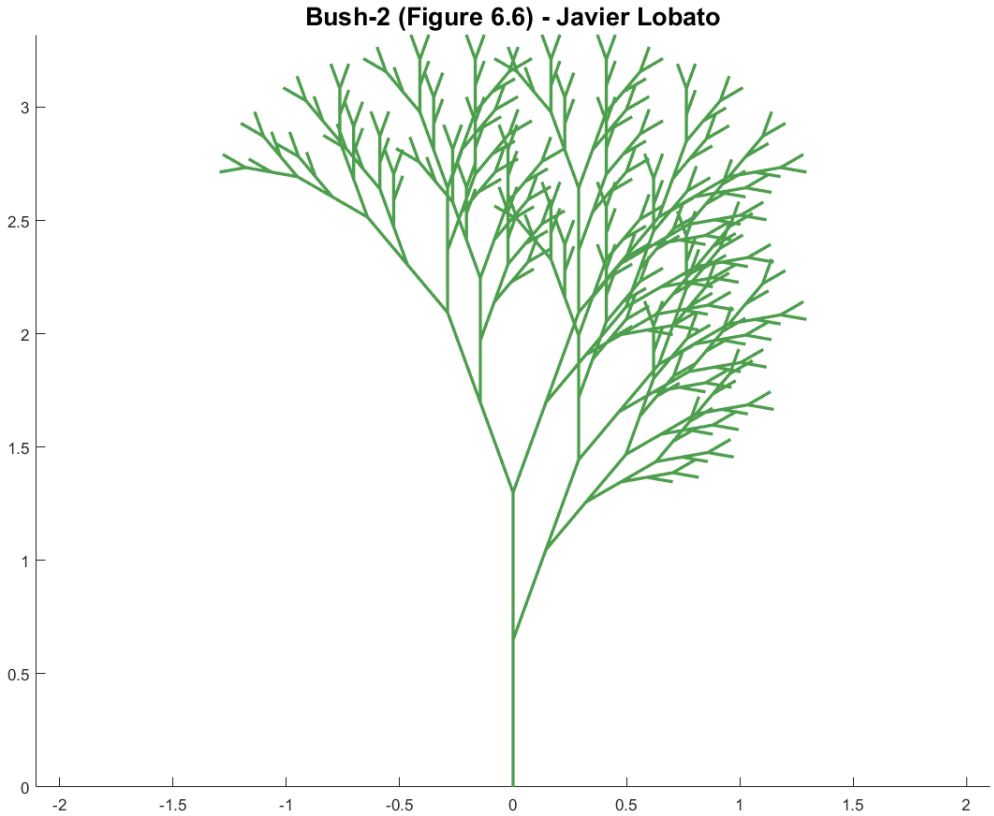
Due date: Thursday, February 8, 2018

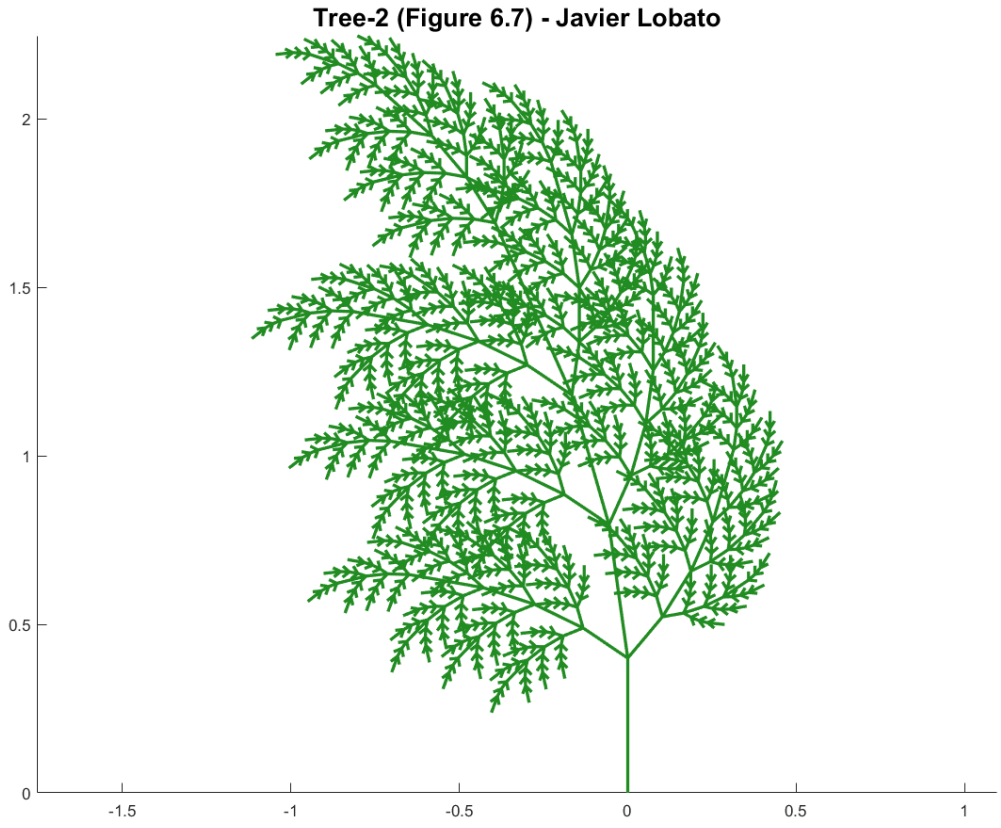
1 Implementation of the operator '|'

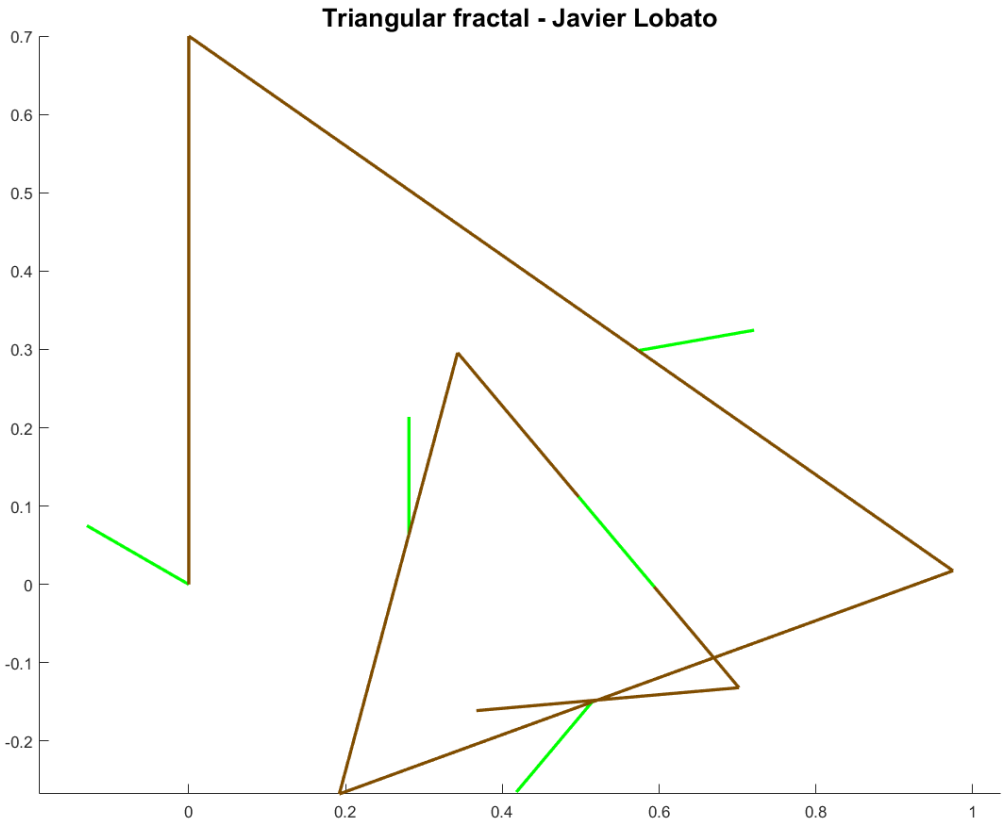
In order to implement the operator `|` a modification in the function `LsysExpandB.m` was done. The operator `|` creates a line whose length depends on the depth n in which the operator has been inserted in the whole string (having δ^n, where δ is the base length). To save the depth of each element, when the `rule().after` is substituted, a string with the length of the `rule().after` filled with the numerical value of the depth is included in a cell-array. In this way, the function `LsysExpandB.m` will return two strings: one with the expansion of the L-system (`resultingString`) and another one with the depth in which each element of `resultingString` has been inserted (called `depthLevel`).

This `depthLevel` string is afterwards used as input in the `LsysDrawB.m` function, allowing it to know in which depth was each symbol inserted. The length of `|` and F will depend on the depth each `|` and F have been inserted, meanwhile the length of G, M and N is constant. When `LsysDrawB.m` reads F or `|`, it will search on `depthLevel` in which level that operator has been inserted, computing the length that should be drawn.

Below are the figures of the different L-systems represented:



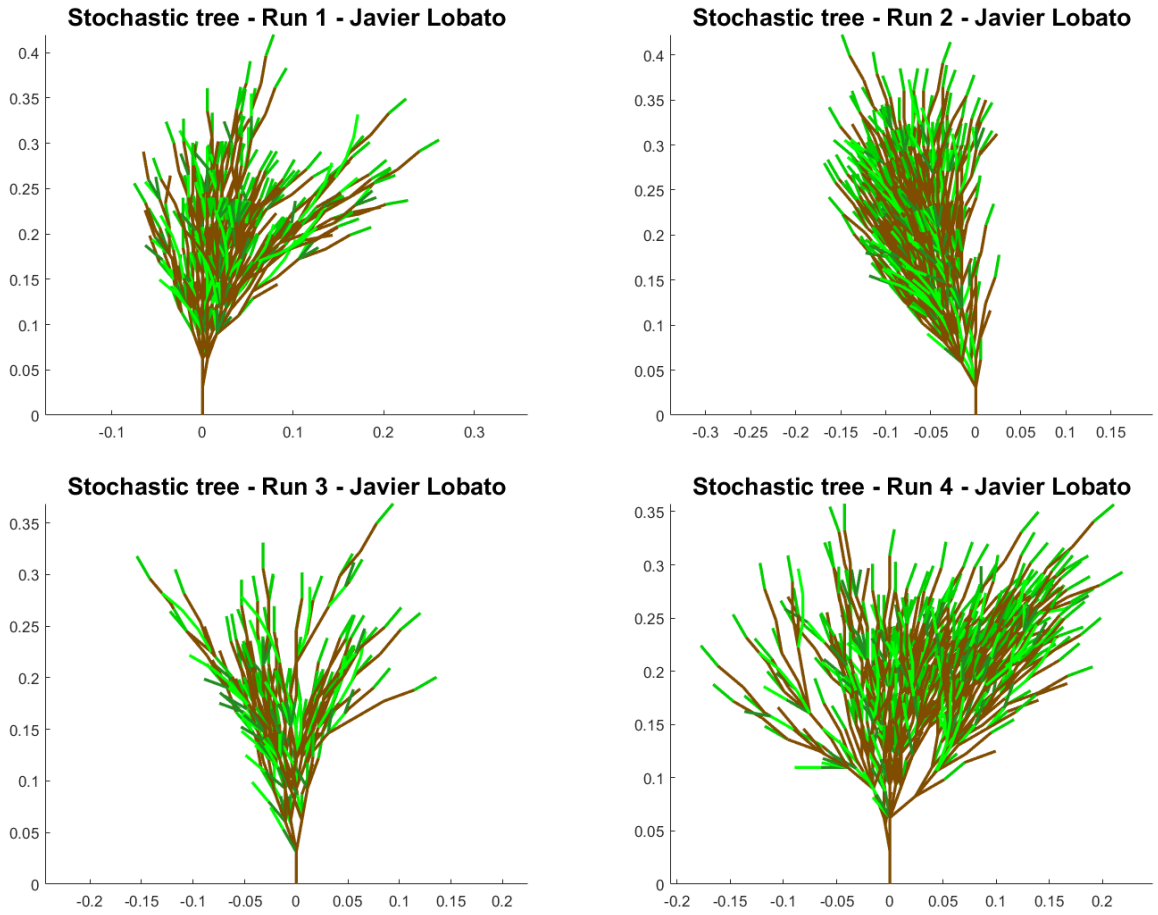

Tree-2 (Figure 6.7) - Javier Lobato
2
1.5
1
0.5
0
-1.5
-1
-0.5
0
0.5
1


Triangular fractal - Javier Lobato
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
0.2
0.4
0.6
0.8
1

2 Stochastic L-system

To implement a stochastic type L-system, some minor modifications in the code were made. The first one is to include another element in the struct array called `rule().prob` that will have associated the probability of some rule. To define a set of rules, each one should be written in a different element of the struct - `rule().before` can be the same for different rules. This will be taken into account later when a $n \times n$ array is created with the probability of each case. This will allow the function to check if all probabilities add up to one. Using that array, cumulative sum function, a random number generator and function find, a random element of the probability matrix can be extracted, applying a different rule each time that `rule().before` appears.

Some examples of this system can be seen below. These representations will vary each time the code is executed, having different fractals:



3 Listing of the code used

3.1 `LsysExpandB.m` function

```
function [ResultingString, depthLevel] = LsysExpandB(nReps, axiom, rule, mode)
% LsysExpand: Apply a set of rules of an L-system on the input axiom during
% a specified number of repetitions to give a resulting string
%
%INPUTS:
%nReps = integer value, number of repetitions of the L-system
%axiom = starting seed for the L-system
%rule = set of rules that define the L-system
%mode = receives the working mode (stochastic or deterministic)
%
%OUTPUTS:
%ResultingString = modified string after applying the rules to the axiom
%depthLevel = string with the depth in which each value has been inserted
%
%Sample test call:
%[ResultingString, depthLevel] = LsysExpandB(nReps, axiom, rule, mode);
%
%Original code at: http://courses.cit.cornell.edu/bionb441/LSystem/index.html
%Modified by Javier Lobato and Veronica Saz (02/01/2018)
%Re-modified by Javier Lobato (02/08/2018)

%Get the number of rules from the input set
nRules = length(rule);

%Copy the axiom to the resulting string, for the initialization of the loop
ResultingString = axiom;

%JL step 6: lets check if the sume of the probabilities is one (for stochastic case)
prob = zeros([1,nRules,nRules]);
if strcmp(mode, 'stoc')
    for i = 1:nRules
        for j = 1:nRules
            %JL step 6: if the rule().before for i and j are the same, the
            %probability is stored in an array
            if strcmp(rule(i).before, rule(j).before)
                prob(1,j,i) = rule(j).prob;
            end
        end
    end
    if any(sum(prob)-1) == 1
        %JL step 6: in case the probability doesn't add up to 1 it exits
        disp('Bad probability array')
        return
    end
end

%JL step 6: get the cumulative sum of the probability matrix
prob = cumsum(prob);

%JL step 5: create a string with the length of the axiom replaced with zeros
depthLevel = num2str(zeros([1,length(axiom)]));
```

```
%JL step 5: given that num2str returns a string with spaces, let's erase them
depthLevel = depthLevel(find(~isspace(depthLevel)));

for i = 1:nReps
    %Convert ResultingString (char) to a cell array called RScells
    RScells = cellstr(ResultingString');
    %JL step 5: convert depthLevel (char) to a cell array called DLcells
    DLcells = cellstr(depthLevel');
    for j = 1:nRules
        %Find occurences of the set of rules in the ResultingString
        hit = strfind(ResultingString, rule(j).before);
        if (length(hit)>=1) %If ocurrences are found
            for k = hit %This will apply each rule to the occurrences
                if strcmp(mode, 'stoc')
                    %JL step 6: if the mode is stochastic, this will get a
                    %random value in the matrix (according to their
                    %probabilities) and apply that rule
                    index = find(rand<prob(:,:,j), 1, 'first');
                    RScells{k} = rule(index).after;
                    %JL step 5: in addition to replacing the value of
                    %RScells, the same length of characters as rule.after
                    %but with the numerical value of (depth) will be
                    %inserted in DLcells
                    DLcells{k} = num2str(i*ones([1,length(rule(index).after)]));
                else
                    %JL step 6: in case the mode is deterministic it will
                    %apply each rule inside the set
                    RScells{k} = rule(j).after;
                    %JL step 5: in addition to replacing the value of
                    %RScells, the same length of characters as rule.after
                    %but with the numerical value of (depth) will be
                    %inserted in DLcells
                    DLcells{k} = num2str(i*ones([1,length(rule(j).after)]));
                end
            end
        end
    end
    %Convert RScells from cell array to string (no prellocation required)
    ResultingString = [RScells{:}];
    %JL step 5: convert DLcells from cell array to string(no prellocation required)
    depthLevel = [DLcells{:}];
    %JL step 5: remove the spaces in the string
    depthLevel = depthLevel(find(~isspace(depthLevel)));
end

end
```

3.2 `LsysDrawB.m` function

```
function [] = LsysDrawB(LsysString, depthLevel, plotParameters, plotTitle, figNo)
% LsysDraw: Draw a string obtained from an L-system with some parameters and
% gives a figure with the specified title. Turtle graphics
%
%INPUTS:
%LsysString = string that contains the result of the function LsysExpand
%depthLevel = string with the depth in which each element of LsysString has
%   been inserted in the string
%plotParameters = structured array with the length and color of each case
%   and the specified delta angle
%plotTitle = string that contains the title of the plot
%figNo = will create different figures to avoid them to overwrite
%
%OUTPUTS:
%No other output than the figure
%
%Sample test call:
%LsysDrawB(LsysString, depthLevel, plotParameters, plotTitle, figNo)
%
%Original code at: http://courses.cit.cornell.edu/bionb441/LSystem/index.html
%Modified by Javier Lobato ans Veronica Saz (02/01/2018)
%Re-modified by Javier Lobato (02/08/2018)

%Initial state (position and angle) of the turtle
xT = 0;
yT = 0;
aT = 0;

%Convert the specified angle to radians
da = deg2rad(plotParameters(1).delta);

%Init the turtle stack with the required preallocation
stack = struct('xT', cell(length(LsysString), 1), 'yT', cell(length(LsysString), 1),'aT',
 ↪  cell(length(LsysString), 1));

%Stack counter definition
stckCounter = 1;

%Variable to add on the cumulative turnings (for the cases with digits)
turnNo = 0;

%Create a figure and keep it open until it is completed
figure(figNo)
hold on

% JL step 5: If the dimension of LsysString doesn't match the dimension of
% depthLevel, it will exit from the function
if length(LsysString) ~= length(depthLevel)
    display('Bad array input!');
    return
end

for i=1:length(LsysString)
    stringElement = LsysString(i);
```

```
54
55      %Different case separation
56      switch stringElement
57
58      %Letter case definition
59      case {'F', 'G', 'M', 'N', '|'} %JL step 6: | has been included in the case
60          %Assign an index for each letter corresponding to one index of the
61          %structured array of the input
62          if stringElement == 'F'
63              j = 1;
64              %JL step 5: F lengths will decrease with depth
65              exponent = str2num(depthLevel(i));
66          elseif stringElement == 'G'
67              j = 2;
68              %JL step 5: as G lengths will not decrease with depth, the exponent is 1
69              exponent = 1;
70          elseif stringElement == 'M'
71              j = 3;
72              %JL step 5: as M lengths will not decrease with depth, the exponent is 1
73              exponent = 1;
74          elseif stringElement == 'N'
75              j = 4;
76              %JL step 5: as N lengths will not decrease with depth, the exponent is 1
77              exponent = 1;
78          elseif stringElement == '|' %JL step 6: F and | will follow the same rules
79              j = 1;
80              %JL step 5: | lengths will decrease with depth
81              exponent = str2num(depthLevel(i));
82          end
83
84          %JL step 5: compute the new location of the X and Y
85          newxT = xT + ((plotParameters(j).length)^exponent)*cos(aT);
86          newyT = yT + ((plotParameters(j).length)^exponent)*sin(aT);
87          plot([yT newyT], [xT newxT],'color',plotParameters(j).color, 'linewidth',2);
88          xT = newxT;
89          yT = newyT;
90
91      case {'X', 'Y'}
92          %Do nothing!
93
94      case '+' %Clockwise turning angle
95          %In case the number is zero (initialization value) it will be one
96          %to make a turning equal to delta
97          if turnNo == 0
98              turnNo = 1;
99          end
100         aT = aT + turnNo*da; %Multiply the delta angle times the specified digit number
101         turnNo = 0; %Assign the value of turnings to zero
102
103     case '-' %Counterclockwise turning angle
104         %In case the number is zero (initialization value) it will be one
105         %to make a turning equal to delta
106         if turnNo == 0
107             turnNo = 1;
108         end
109         aT = aT - turnNo*da; %Multiply the delta angle times the specified digit number
110         turnNo = 0; %Assign the value of turnings to zero
```

```
111
112     case '[' %Push the stack with current values
113         stack(stckCounter).xT = xT ;
114         stack(stckCounter).yT = yT ;
115         stack(stckCounter).aT = aT ;
116         stckCounter = stckCounter +1 ;
117
118     case ']' %Pop the stack taking the last values
119         stckCounter = stckCounter-1 ;
120         xT = stack(stckCounter).xT ;
121         yT = stack(stckCounter).yT ;
122         aT = stack(stckCounter).aT ;
123
124     case {'0','1','2','3','4','5','6','7','8','9'} %Digit case
125         %Takes the digit value
126         turnNo = turnNo + str2num(LsysString(i));
127         %Checks the next element of the string. In case it is another
128         %digit, it multiplies the value of turnNo by 10 and it will add the
129         %next digit in the following for-loop repetition
130         if ~mod(str2num(LsysString(i+1)),1) == 1
131             turnNo = turnNo*10;
132         end
133
134     otherwise
135             disp('error')
136         return
137     end
138 end
139
140 hold off
141
142 %plot configuration and title
143 axis equal
144 title(plotTitle, 'FontSize',16)
145
146 end
```

3.3 `LsystemDriverB.m` script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                HOMEWORK #1.B                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This code was originally downloaded from the following web site
%   http://courses.cit.cornell.edu/bionb441/LSystem/index.html
%
%Given by Margaret Eppstein for the course
%    CSYS 302 'Modeling Complex Systems'
%
%Modified by Javier Lobato (02/08/2018)

%% BUSH-2 Figure 6.6
clear all

%axiom
axiom = 'F';

%mode selection: 'stoc' or 'nonStoc'
mode = 'nonStoc';

%set of rules
rule(1).before = 'F';
rule(1).after = '|[+F]|[-F]+F';

%number of repetitions
nReps = 5;

%calculation of the
[resultingString, depthLevel] = LsysExpandB(nReps, axiom, rule, mode);

%plot parameters definition
plotParameters = struct('length', cell(1, 1), 'color', cell(1, 1),'delta',cell(1,1));

plotParameters(1).length = 0.65; %length of case F
plotParameters(1).color = [0.30 0.62 0.30]; %dark green to the bush
plotParameters(1).delta = 20;

%turtle graphic plotter
LsysDrawB(resultingString, depthLevel, plotParameters, 'Bush-2 (Figure 6.6) - Javier Lobato', 1);


%% TREE-2 Figure 6.7
clear all

%axiom
axiom = 'F';

%mode selection: 'stoc' or 'nonStoc'
mode = 'nonStoc';

%set of rules
rule(1).before = 'F';
rule(1).after = '|[5+F][7-F]-|[4+F][6-F]-|[3+F][5-F]-|F';
```

```

%number of repetitions
nReps = 4;

%string calculation
[resultingString, depthLevel] = LsysExpandB(nReps, axiom, rule, mode);

%plot parameters definition
plotParameters = struct('length', cell(1, 1), 'color', cell(1, 1),'delta',cell(1,1));

plotParameters(1).length = 0.4; %length of case F
plotParameters(1).color = [0.13 0.55 0.13]; %forest green for the tree
plotParameters(1).delta = 8;

%turtle graphic plotter
LsysDrawB(resultingString, depthLevel, plotParameters, 'Tree-2 (Figure 6.7) - Javier Lobato', 2);

%% Triangular fractal
clear all

%axiom
axiom = 'F';

%mode selection: 'stoc' or 'nonStoc'
mode = 'nonStoc';

%set of rules
rule(1).before = 'F';
rule(1).after = 'G|25+|F';
rule(2).before = 'G';
rule(2).after = '[3-G]';

%number of repetitions
nReps = 5;

%string calculation
[resultingString, depthLevel] = LsysExpandB(nReps, axiom, rule, mode);

%plot parameters definition
plotParameters = struct('length', cell(2, 1), 'color', cell(2, 1));

plotParameters(1).length = 0.7; %length of case F
plotParameters(2).length = 0.15; %length of case G
plotParameters(1).color = [0.5 0.3 0.0]; %brown (F)
plotParameters(2).color = [0.0 1.0 0.0]; %green (G)
plotParameters(1).delta = 5;

%turtle graphic plotter
LsysDrawB(resultingString, depthLevel, plotParameters, 'Triangular fractal - Javier Lobato', 3);

%% Stochastic tree
clear all

%axiom
axiom = 'F';

%mode selection: 'stoc' or 'nonStoc'
```

```
mode = 'stoc';

%set of rules
rule(1).before = 'F';
rule(1).after = 'F+[-F+F-G]';
rule(1).prob = 0.3;
rule(2).before = 'F';
rule(2).after = 'F-[G2+F]';
rule(2).prob = 0.3;
rule(3).before = 'F';
rule(3).after = 'F-[F+G]+[-F2+F-F+G]';
rule(3).prob = 0.4;
rule(4).before = 'G';
rule(4).after = '[+M][2-N][-M]';
rule(4).prob = 0.25;
rule(5).before = 'G';
rule(5).after = '[2-MN]';
rule(5).prob = 0.25;
rule(6).before = 'G';
rule(6).after = '[N2-M]';
rule(6).prob = 0.25;
rule(7).before = 'G';
rule(7).after = '[N-N-N]';
rule(7).prob = 0.25;

%number of repetitions
nReps = 5;

%plot parameters definition
plotParameters = struct('length', cell(4, 1), 'color', cell(4, 1), 'delta', cell(1, 1));

plotParameters(1).length = 0.5; %length of case F
plotParameters(2).length = 0.025; %length of case G
plotParameters(3).length = 0.025; %length of case M
plotParameters(4).length = 0.025; %length of case N
plotParameters(1).color = [0.5 0.3 0.0]; %brown (case F)
plotParameters(2).color = [0.0 0.81 0.0]; %darker green (case G)
plotParameters(3).color = [0.13 0.55 0.13]; %forest green (case M)
plotParameters(4).color = [0.0 1.0 0.0]; %green (case N)
plotParameters(1).delta = 10;

%let's generate various plot to demonstrate the randomness of the model
for i = 1:4
    %string calculation for each repetition
    [resultingString, depthLevel] = LsysExpandB(nReps, axiom, rule, mode);

    %turtle graphic plotter
    LsysDrawB(resultingString, depthLevel, plotParameters, sprintf('Stochastic tree - Run %i -
     ↪  Javier Lobato', i), i+3);
end
```