L-Systems in MATLAB

Paulo Alves
Universidade do Minho
Braga, Portugal
pauloalves2250@gmail.com

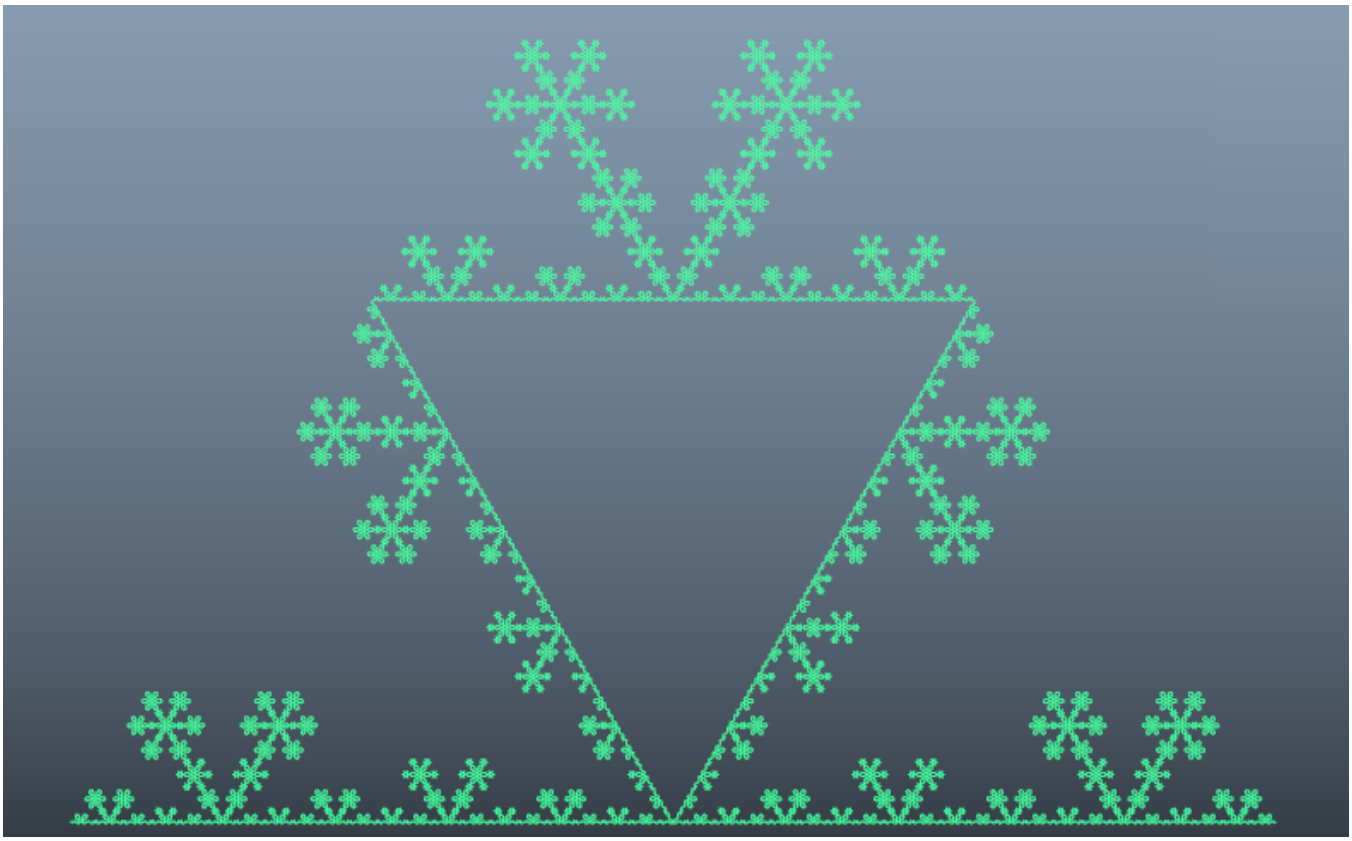

Figure 1

ABSTRACT

This article introduces some basic L-System's theory, what they are how can we describe them and some known examples. This article will also cover a basic implementation of an L-System in *MATLAB* as well as plot some results and discuss the performance using the tool. Lastly a brief discussion about why to use **MATLAB** and what we could achieve with it.

KEYWORDS

MATLAB, L-Systems, image recognition, feature extraction, edge detection

1 INTRODUCTION

L-Systems were originally created to describe the growth of some organisms, such as fungi, later on they were expanded to describe some more complex structures like plant, their structure and how it branches .

This had nothing to do with computer graphics or any graphic at all considering an L-System is basically a grammar, with no contex,t where we have a set of letters or alphabet and a set of rules that expand that alphabet into a bigger set over a number of iterations. Nowadays however, besides its basic application of modeling plants and replication of branches, it is used to create some very interesting fractals, model rivers, roads or even create music.
It has many uses outside and inside computer graphics.

2 L-SYSTEM STRUCTURE

Formally, an L-System is a triplet of the form **L=(ALPHABET,INITIALSTATE,RULES)**, where the **ALPHABET** is a non-empty set of symbols, the **INITIALSTATE** is our seed or initial state of how the L-SYSTEM is before the first iteration and the **RULES** are a set of rules that expand the initial seed into more and more symbols of the alphabet.
An extra resource is also defined, the constants or terminals, they

are basically any symbol that appear in the set of rules but does not in the alphabet. Commonly used constants are **'+'** and **'-'**, they are used to define rotations and together with symbols like **'F'** that is used as 'draw forward' we can draw and visualize some very interesting fractals .
The set of rules has to be finite and for computational and time purpose the number of iterations, besides being finite is usually a small number as too many iterations usually make the generation of the result have a exponential increase in time.

3 EVOLUTION OF AN L-SYSTEM

The basic evolution of an L-System is the following:

- We take the initial state.
- Given a rule of the type **R : A->AB**, we rewrite the symbol A of the alphabet into 'AB'.
- Apply the same for every rule.
- Repeat for **n** number of iterations.

3.1 Interpreting the string result

After expanding the alphabet and generating a string with the various symbols over a number of iterations, we can as mentioned before, interpret the results in different ways. The usual way is to draw it using a number of lines to visualize the result.
For that purpose we can define that the symbols that were in the alphabet means 'draw forward' and use constants like **'+'** and **'-'** that mean 'turn right' and 'turn left', that is however not the only way, as said before we could interpret the resulting string and create music with it or whatever else we like.
There are many options.

Following is an example of a basic L-System alphabet as well as its rules and initial set, also known as axiom, its result over 'n' iterations and the resulting drawing.

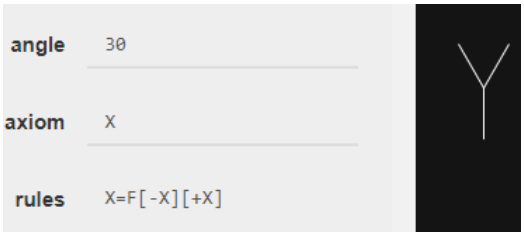

Figure 2: Binary tree using L-Systems

In this example we have the triplet of the L-System being **L=(ALPHABET,INITIALSTATE,RULES)** where the **ALPHABET** is 'X', the **INITIALSTATE** also 'X' and just one rule, that translates into "If you find an 'X' draw a line foward", thats what the 'F' constant means, then "if you find the '[' constant, save the current angle and turn left with an angle of 30º", and "when you find the ']' constant go back to the angle that was saved and turn 30º to the right", this is important because we want to create and separate branches from leaves.
The '-' and '+' indicates normal rotations to the left and to the right of 30º angle aswell.
For any following iteration we just apply the same rule to the resulting string, in this case we started with X and the progression was :

Figure 3: 1st iteration

Figure 4: 2nd iteration

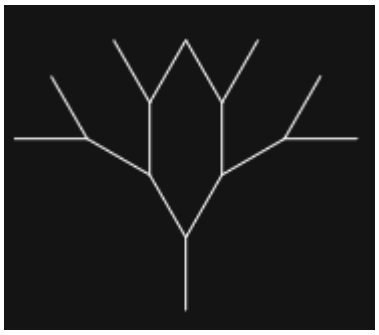

Figure 5: 3rd iteration

4 EXAMPLES OF L-SYSTEMS

In this chapter we are going to present some well-known examples of L-Systems.

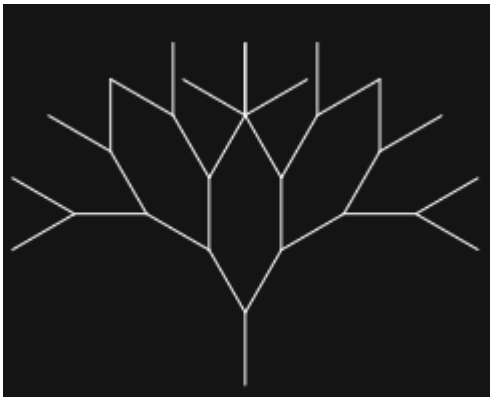

Figure 6: 4th iteration of a binary tree L-System using a 30º angle.

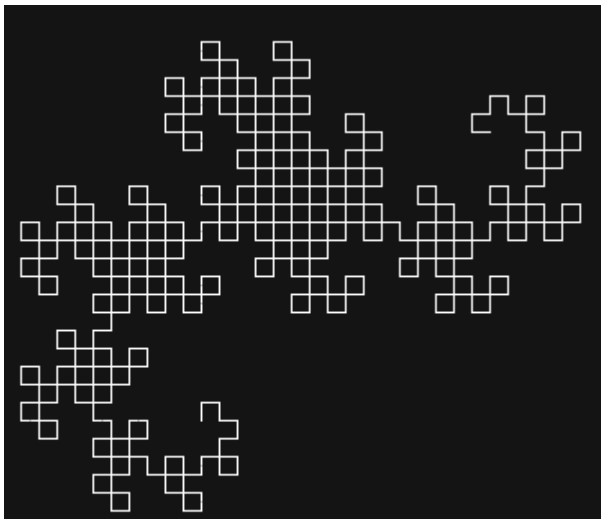

Figure 7: 9th iteration of the dragon curve L-System using a 90º angle.

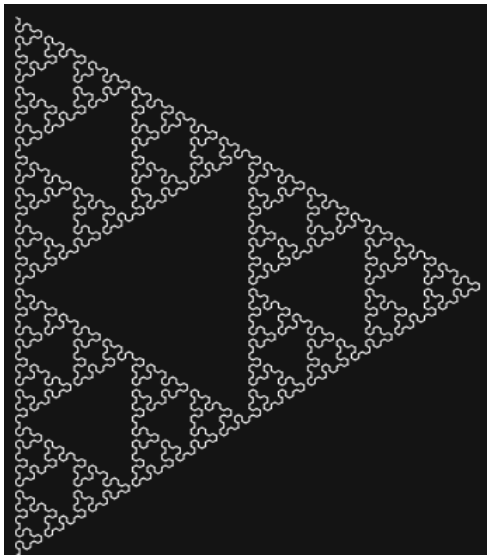

Figure 8: 7th iteration of an L-System that simulated the Sierpinski's triangle.

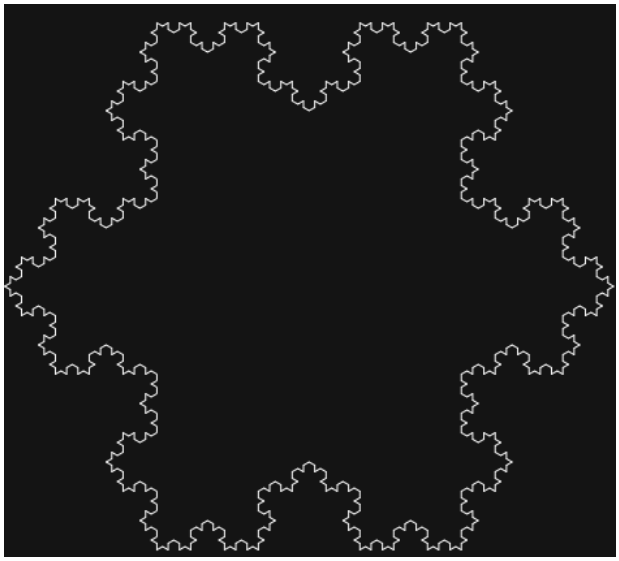

Figure 9: 4th iteration of the koch snowflake or koch curve L-System that uses a 60º angle.

As previously discussed , L-Systems were originally created to describe organisms and plant's growing behaviour.The following are some L-Systems that are very similar to some plants.

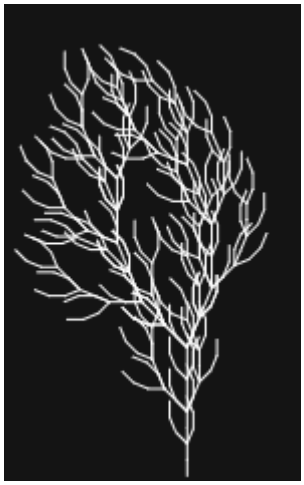

Figure 10: L-System that is very similar to a seaweed's structure.

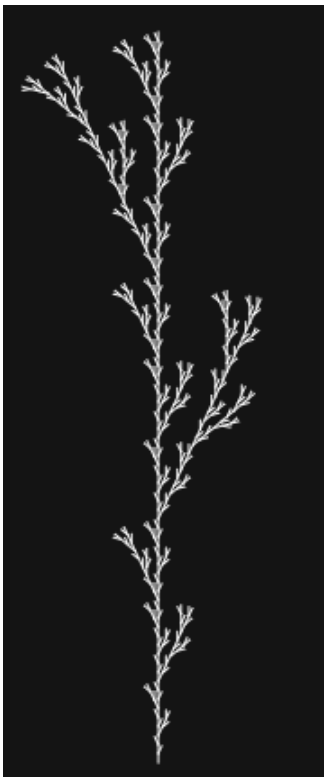

Figure 11: L-System of a tall seaweed.

Note that there are other plants that look like this , that's because many plants have the same structure, as they say "plants are recursive" but we could also say that the DNA is as well.

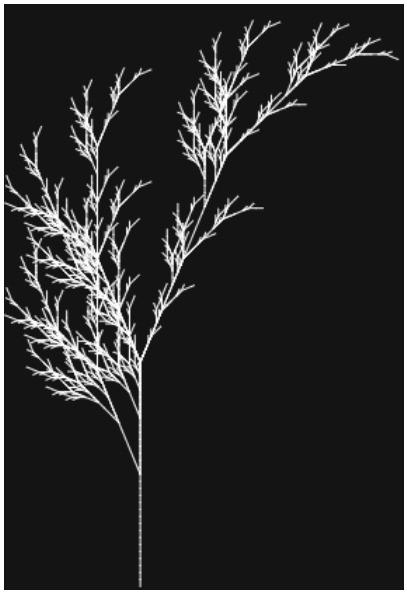

Figure 12: L-System designated as fuzzy weed.

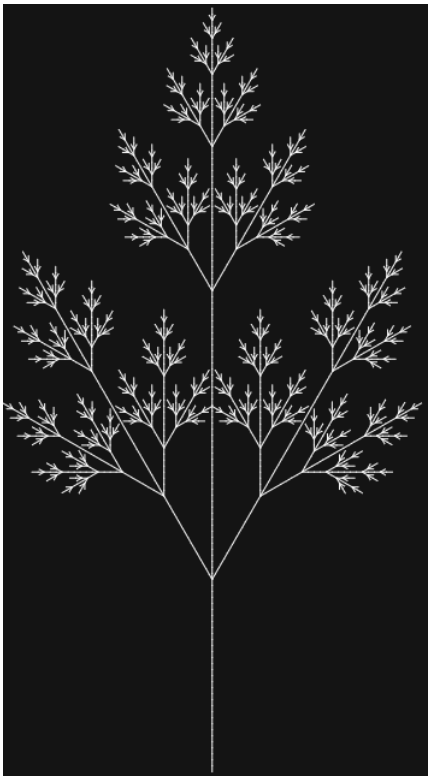

Figure 13: Arrowhead L-System.

Just by changing the color of the drawn lines we can achieve some interesting images.

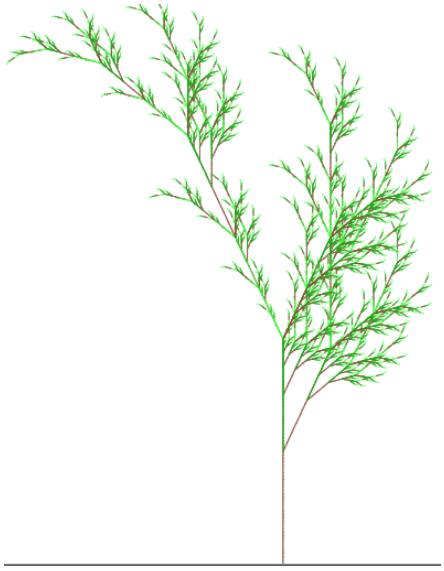

Figure 14: Fuzzy weed but colored depending on the symbol of the alphabet.

5 L-SYSTEMS IN MATLAB

Now that we know what L-Systems are, how to create them and how to interpret the resulting strings, lets take a look on how simple it can be to implement some and plot the values using **MATLAB**.
MATLAB is a widely known and used programming platform for scientists and engineers and it has it's own programming language, a matrix-based language that is very power full when we are dealing with mathematical expressions.
Because a lot of L-Systems can describe fractals and other geometry based on mathematical formulas it should, in theory, be very easy to implement and draw the results using **MATLAB**.
Indeed it is very simple to implement, just like using anything else, we define the L-System by using the same triplet as before **L=(ALPHABET,INITIALSTATE,RULES)** like seen in the next screenshot.

```
rule(1).before = 'F';
rule(1).after = 'FRH';

rule(2).before = 'H';
rule(2).after = 'FLH';

nRules = length(rule);

%starting seed
axiom = 'F';
```

Figure 15: Dragon Curve L-System description in MATLAB.

The **ALPHABET** in this case is 'A=FH' and the rules are: 'R=F->FRH' and 'H->FLH'.
Now we just choose a number of iterations and apply the rules creating a result string as seen here.

```
>> dragon(4)

ans =

    'FRHRFLHRFRHLFLHRFRHRFLHLFRHLFLH'
```

Figure 16: 4th iteration of the Dragon Curve L-System using MATLAB.

Now we just have to choose an angle and feed the resulting string into a plotting function/script and we have the following result:

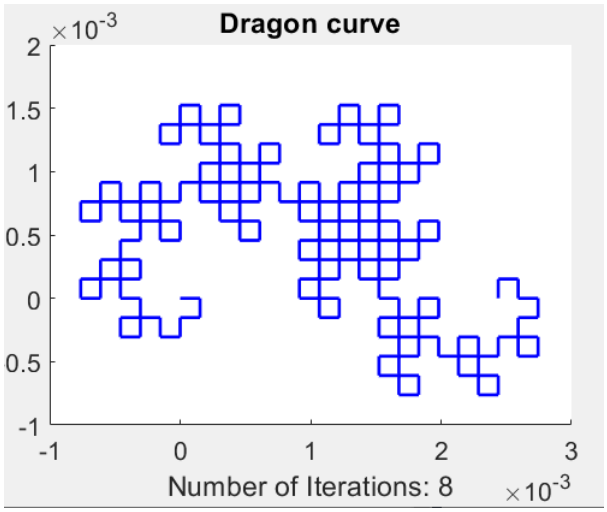

Figure 17: 8th iteration of the Dragon Curve plotted using MATLAB.

5.1 Performance

Performance-wise, the generation of the L-System string after a number of iterations is quiet fast, as we would expect, yet plotting the results not so much.
Perhaps its more due to the method chosen of making a plot rather then printing it directly to the screen as plain pixels, whatever the case may be, for general usage of defining and drawing L-Systems, it would be a better idea to use another tool, language or even using **MATLAB** but with another method for drawing, perhaps drawing it directly to an bitmap image.
For a dragon curve of 15 iterations it took around 25 seconds to plot to the screen while in other tools and languages it took me no more than 1 or 2 seconds.

6 WHY MATLAB

So why would we actually want to use **MATLAB** then? we have seen that it does the job, slowly but it does, and we have not seen any special feature that would make us want to use it for this over another tool or language, so why then?
Well **MATLAB** has a very powerful computer vision toolkit, it is very easy to do things like image recognition and edge detection, and that's exactly the reason why we would want to use L-Systems

in **MATLAB**.
Before going further in, lets also mention that L-Systems don't have to be generated and interpret as 2D images, we can also do it in 3D. here is an example:

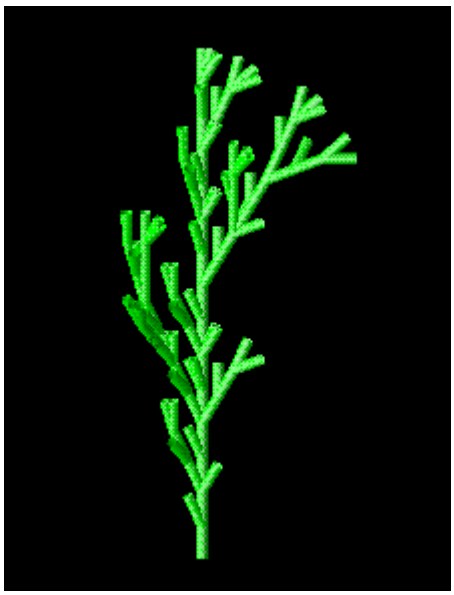

Figure 18: Tall seaweed L-System in 3D

Now, we know we can describe and generate plant-like structures in 2D and 3D, we know that we can describe a variety of different plants using L-Systems, we also know that we can take a picture of a plant and search online to find out which plant it is, but what if we could combine everything, image recognition/edge detection, L-Systems, maybe some sort of machine learning and without using the internet or anything find out what plant it mostly likely is.
Lets for example say, someone is lost in a forest, and he needs to eat, he does have his cellphone with some battery but no internet connection and no knowledge of botany or anything. Lastly he is hungry and needs to eat something, for example a plant.
The process would be like this:

- Having a database of L-systems in 2D/3D in my phone or maybe just the descriptions themselves.
- He takes a picture of a plant that he doesn't know if he should or could eat, maybe the plant has some fruits or maybe not but he is desperate for food, and some plants as we know are indeed edible.
- The phone runs an image recognition script on the picture.
- After having the result, the phone that also had some machine learning process and was trained to recognize patters of plants, takes the result of the image recognition and compare it to the L-System's rules and find out a probable match, having an L-System description takes a lot less space than having a full image database of plants/fruits on a phone, besides on some cases it wouldn't even need to generate the L-System string, it could just look at the rules and find a match.
- After all the process the phone says that the plant matches a poisonous plant, saving him from certain death, not from hunger though.

Considering that machine learning is being used more and more nowadays, and so is image recognition, and that we could indeed describe a lot of plants with L-Systems, what is the DNA if no more then an L-System.
Considering all this, it is very possible that in the future phones do indeed have a similar system like this. Thas the theory, study and describe plants as L-Systems and be able to recognize patterns without needing a degree in botany.
Here is an example of a poisonous plant and a basic edge detection and feature extraction generated using **MATLAB** that could be used to compare to a variety of L-Systems and try to find a possible match.

Figure 19: Edge Detection and feature extraction of a *Lythrum Salicaria* poisonous weed.

A good Botanist could probably tell that those edges have nothing to do with our previous drawn L-Systems, and that this could be in fact a poisonous weed. A machine with some training in pattern recognition, using pictures of plants and some L-Systems could probably do the same with good accuracy, besides on that hypothetical case of someone lost in the wood, he wouldn't need a 100% accuracy, just enough to tell him that its better to try and find someone else because there is imminent risk.
Another useful idea would be to use a system like this to help a botany student, or who knows maybe one day we will find other planets with life and we could use the same tool to try to find similarities with our planet's flora.

7 CONCLUSION

L-Systems are indeed very interesting, some just because they can be drawn to make some visually impressive fractals but others could prove to be very useful for other reasons, maybe in the future we can in fact take some DNA and find a perfect L-System match for that DNA and use it to recognize a variety of organisms, maybe even replicate some with some sort of carbon-based L-System 3D printer. For a future work i would like to actually try to take pictures of plants, run some feature extraction algorithms using **MATLAB** and try to train a system to detect what kind of plant it is based on L-Systems.
Another possibility would be to use machine learning to create new L-Systems based on images of plants and draw them in 3D.

8 REFERENCES

REFERENCES

[1] Marek Fišer. *L-systems online thesis.[Department of Software and Computer Science Education]*, Thesis, Prague, 2012.

[2] Przemyslaw Prusinkiewicz. *Graphical applications of Lsystems.*

[3] Lila Kari, Grzegorz Rozenberg, and Arto Salomaa . *L Systems* .

[4] Lindenmayer systems,
https://draemm.li/various/fractals/l-systems/

[5] L-System renderer online,
http://piratefsh.github.io/p5js-art/public/lsystems/

[6] **MATLAB** features extraction,
https://www.mathworks.com/help/vision/ref/extractfeatures.html

[7] What is **MATLAB**,
https://www.mathworks.com/discovery/what-is-matlab.html

[8] L-Systems in **MATLAB**,
https://github.com/girardimatthew/L-system-MATLAB

[9] Purple loosestrife (Lythrum salicaria) L.,
https://www.invasive.org/