**Creating Root Architectures with RootBox**

Contents

[Introduction 2](#_Toc460594333)

[Abbreviations used: 2](#_Toc460594334)

[References: 2](#_Toc460594335)

[Defining Branch Length 2](#_Toc460594336)

[Definitions 2](#_Toc460594337)

[Simulation time 3](#_Toc460594338)

[Defining Branching Angles 4](#_Toc460594339)

[Tropism 5](#_Toc460594340)

[Types of Tropism 5](#_Toc460594341)

[Parameters 6](#_Toc460594342)

[Output Formats 8](#_Toc460594343)

[L-System String 8](#_Toc460594344)

[Cartesian Coordinates 10](#_Toc460594345)

[RSWMS Format 11](#_Toc460594346)

[Troubleshooting 11](#_Toc460594347)

[Issue 1: Non-zero origin 11](#_Toc460594348)

**Version History:**

9/2/2016 – Adding branching angle section. Updated tropism definitions. Added additional references

2/29/2016 – Added tropism section and started version history. Branch length section was created in 2015.

# Introduction

RootBox is a growth-based root architecture model created by Daniel Leitner for Matlab. The model uses Lindenmayer-system (L-system) strings which propagate root growth through a series of set production rules. These production rules, including growth rate, branching distances and frequency, are defined by the user before each model simulation. Root architectural modeling has been a challenging process, but I hope this guide will serve to jumpstart new students or to serve as a reference guide for those with more experience. It is recommended that all new users first read Leitner et al. 2010 for an introduction to the model.

## Abbreviations used

|  |  |
| --- | --- |
| *K* | maximum length of branch [L] |
| *lb* | length of basal zone (portion which begins at branching node) [L] |
| *la* | length of apical zone (portion which includes root tip) [L] |
| *nob* | number of branches which will spawn from this parent [-] |
| *ln* | distance between spawned branches [L] |
| *dx* | general segment resolution [L] |
| *gf* | growth function (negative exponential or linear) |
| *t* | time [days] |
| *r* | initial growth rate [L/day] |

## References

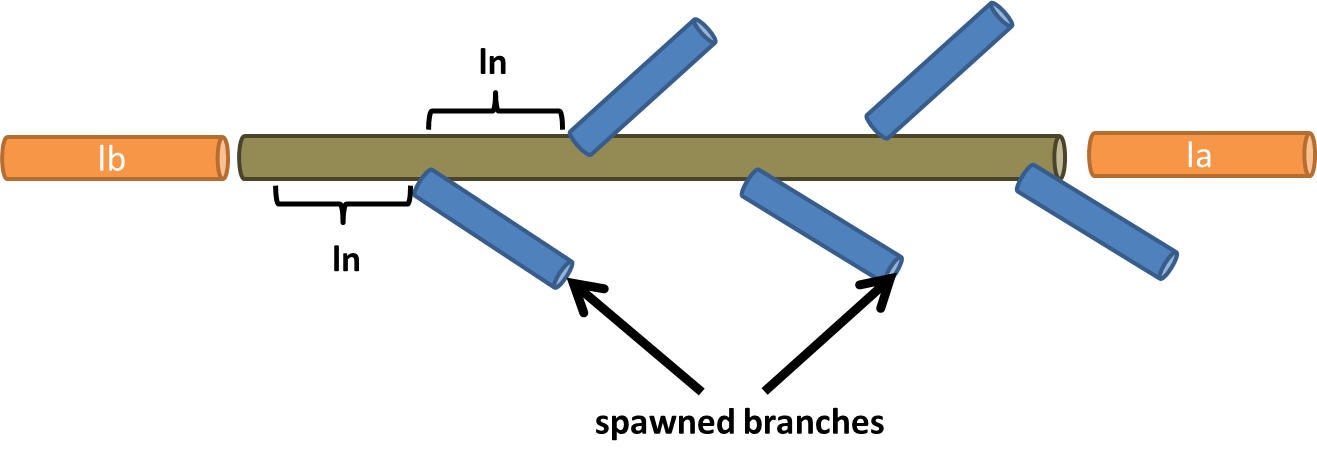
Leitner, D., S. Klepsch, G. Bodner, and A. Schnepf (2010), A dynamic root system growth model based on L-Systems, *Plant Soil*, *332*(1-2), 177–192, doi:10.1007/s11104-010-0284-7.

Dunbabin, V. M., J. A. Postma, A. Schnepf, L. Pagès, M. Javaux, L. Wu, D. Leitner, Y. L. Chen, Z. Rengel, and A. J. Diggle (2013), Modelling root–soil interactions using three–dimensional models of root growth, architecture and function, Plant and Soil, 372(1-2), 93–124, doi:10.1007/s11104-013-1769-y.

Leitner, D., F. Meunier, G. Bodner, M. Javaux, and A. Schnepf (2014), Impact of contrasted maize root traits at flowering on water stress tolerance – A simulation study, Field Crops Research, 165, 125–137, doi:10.1016/j.fcr.2014.05.009.

# Defining Branch Length

## Nomenclature and anatomy

**

The length of a branch can be calculated using the following formulation:

Note that this maximum length is the path length of the branch and may not coordinate to the Cartesian coordinates, especially in the case of tropisms. Also, maximum length can only be achieved with a sufficiently large simulation period and/or growth rate. To achieve a desired branch length, the *nob* or the *ln* parameters (whilst keeping the other constant) can be set to values based on the formulations:

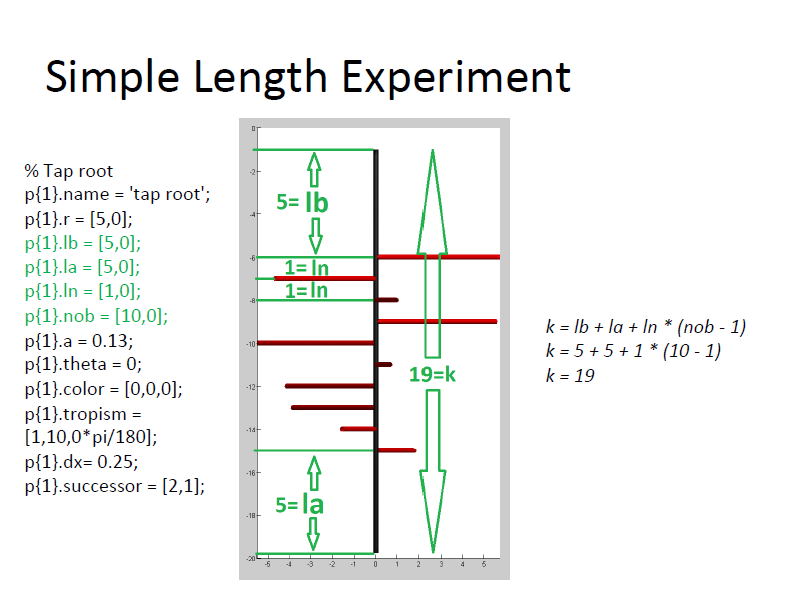
Note that the segment lengths will make up a combination of the *dx* resolution and the *ln, lb,*and *la* parameters.

## Simulation time

The current implementation of RootBox includes two growth functions: negative exponential and linear. The length of a branch, *k*, at time *t*, can be calculated as

for negative exponential growth for linear growth

To achieve a desired branch length *K*, simply solve for time and add to the simulation time at which the branch was born (e.g., if a lateral branch was spawned at simulation time 5 days and it requires 10 days of growth for its branch to reach maximum length, the total simulation time would need to be greater than or equal to 15 days). Also, if you want the branch to reach its maximum length and no consideration is needed for branch age, you can set an arbitrarily high growth rate (p{n}.r = [mean, std]) for near immediate results.



# Defining Branching Angles

The angle between the parent branch and the child is determined by the theta parameter. The angle is defined within the child order’s parameter set and can be exact or defined by the normal distribution given by the standard deviation parameter. Theta is the angle between the vector defined by the parent’s direction at the time of spawning and the initial heading of the child branch.

|  |  |
| --- | --- |
| 𝜃 | 𝜃 |



𝜋/2

𝜋/3

𝜋/4



2𝜋/3

**Figure. Simple tap root with a single branching child for different mean values of theta. No standard deviation is applied in this example and the tap root is oriented in the negative z direction.**

p{n}.theta = [angle\_radians, std\_dev]

# Tropism

Tropism is used to incorporate the plasticity of the root system to the natural environment, mimicking the growth responses to gravity, resource searching, etc. We currently utilize tropism to add additionally path length within a given soil volume.

|  |  |
| --- | --- |
|  | 1. Straight line root with no added tropism with radius Ra ­and path length La 2. Curved root caused by tropism with radius Rb and path length Lb |
| **Figure. Example of root segments with and without tropism. Ra = Rb, but Lb > La. This illustrates tropism’s ability to add additional length within a given radial distance.** | |

|  |
| --- |
| **Tropism = [t, N, s]**   * + t: type     - 0 = plagiotropism: growth at an oblique or horizontal angle     - 1 = gravitropism: growth in response to gravity     - 2 = exotropism: growth away from the main axis     - 3 = chemotropism: growth navigated by chemical stimulus (not implemented)   + N: number of trials to find the optimal angles for rotation   + s: maximal angular deviation in root heading |

## Types of Tropism

|  |  |  |
| --- | --- | --- |
| **Plagiotropism** | **Gravitropism** | **Exotropism** |
|  |  |  |

*0 – Plagiotropism*: Growth at an oblique or horizontal angle. Pushes growth into the XY horizontal plane.

*1 – Gravitropism:* Growth downwards in response to gravity.

*2 – Exotropism*: Growth away from the main axis (e.g., parent branch). Growth is not necessarily horizontal and depends on the heading of the parent branch.

The three tropisms are characterized by the following objective functions:

|  |
| --- |
| % for plagiotropism r – beta, t - alpha  function z=testH(r,t,y)  R = y.R \* [ 1 0 0; 0 cos(r) -sin(r); 0 sin(r) cos(r) ]; % roll  R = R \* [ cos(t) sin(t) 0; -sin(t) cos(t) 0; 0 0 1]; % turn  z = abs(R(3,1)); % minimize abs(z)  % for gravitropism  function z=testZ(r,t,y)  R = y.R \* [ 1 0 0; 0 cos(r) -sin(r); 0 sin(r) cos(r) ]; % roll  R = R \* [ cos(t) sin(t) 0; -sin(t) cos(t) 0; 0 0 1]; % turn  z = R(3,1); % minimize z    % for exotropism  function z=testExo(r,t,y,iR)  R = y.R \* [ 1 0 0; 0 cos(r) -sin(r) ;0 sin(r) cos(r) ]; % roll  R = R \* [ cos(t) sin(t) 0; -sin(t) cos(t) 0; 0 0 1]; % turn  s = (R(:,1)'\*iR') / norm(R(:,1)) / norm(iR);  z = acos(s); |

where y.R is the position vector, rotated using standard rotation matrices for Euclidean space. TODO: do any example rotation using this array to visually illustrate the process the objective function is going through.

## Parameters

N – the number of trials to find the optimal angles α and β for the rotation R where α is a normally distributed random number with mean 0 and standard deviation σdx and β is uniformly distributed from [ -π, π]. Optimal refers to minimizing an objective function depending on the type of tropism.

S – defines the standard deviation of the normal distribution for angle α, such that , where dx is the segment length.

|  |  |
| --- | --- |
|  |  |

β is defined with the applyRules function on line 226, within the dynamic letter ‘303’ or the ‘Root Creation’ section of the code.

*Effects on total length:*

To maintain the same radial distance from the collar but to increase the path length, use the above curves to modify the main lateral. The *maximum* total length on a lateral branch, including its daughter branches can be found using:

where ki is the length of a branch of order *i* and nobi is the number of daughter roots order i will spawn when allowed to grow to maximum length. Recall that the branch length can be represented by base parameters as:

|  |  |
| --- | --- |
|  |  |

*Troubleshooting:*

1. “Could not respect boundaries” – if tropism occurs too close to the ‘surface’; I fixed this issue by adding a slight basal distance to move the system down.

# Output Formats

## L-System String

The default format for all RootBox root systems is an L-system string, held in an [N x 6] matrix. Each letter is represented as a matrix of Nx6 double values, with the structure [ID,N,p1,p2,p3,p4; pos; ap], where ID is the identification number of the letter, N is the total number of rows, p1-p4 are parameter values, pos contains the position (see updatePos), and ap contains additional parameters.

In the str matrix, the first entry of the l-system string is an ID. If the ID < 300 it is a static letter whose value corresponds to a column in the names matrix:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| F | f | C | # | [ | ] | + | - | & | ^ | \ | / | | | r |

**Parameters:**

name

'F' Forward p(1) = length

'f' Noline forward p(1) = length

'C' Color p(1:3) = RGB

'#' Diameter p(1) = diameter

'[' Push turtle state

']' Pop turtle state

'+' Turn left p(1) = angle

'-' Turn right p(1) = angle

'&' Pitch up p(1) = angle

'^' Pitch down p(1) = angle

'\' Roll left p(1) = angle

'/' Roll right p(1) = angle

'|' Turn around

'r' Roll and turn p(1) = angle p(2) = angle

300 Section growth

301 Delay

302 Branching

303 Create root

304 Create successor

305 Root tip that stopped growing

The format of larger growth “letters” are:

*Section Growth (300)*

l=[300,N,length,age,roottype,0;

pos;

pgf,ptr;

t0,tend,l0,lend,rlt,dx;

iheading 0 0 0;

successor];

*Delay growth (301)*  
l=[301,N,t,tend,0,0;

pos;

successor];

*Branching (302)*

l=[302,N,cbn,nob,0,0;

pos;

pgf,ptr;

rlt,dx,0,heading;

times;

delays;

lateral;

next]

*Create root (303)*

l=[303,N,type,0,0,0;

pos];

*Create successor (304)*

l=[304,N,not,0,0,0;

pos;

probabilites]

*Key*

N ... total number of rows of the letter (including successor)

pos ... [3x6], contains:[[postion[1x3];color[1x3];radius,age,0], local axis[3x3]]

pgf ... parameters for the growth function [1x3]

ptf ... parameters for the tropism function [1x3]

t0 ... local initiation time

tend ... local final time

l0 ... arclength of section start (along the root)

lend ... arclength of sections end (along the root)

rlt ... root life time

iheading ... initial heading [1x3]

successor ... [nx6]

cbn ... current branch number

nob ... number of branches

For a more rigorous definition of the L-system growing parameters:

* **F**: the system draws a segment here
* **f**: move to the next position without drawing a segment
* **+**: rotate counterclockwise by *lsystem.angle*.
* **-**: rotate clockwise by *lsystem.angle*.
* **|**: rotate 180 degrees.
* **[**: works like the push() command, starting a subbranch.
* **]**: works like the pop() command, ending the subbranch

REF: <https://www.nodebox.net/code/index.php/L-system>

When dynamic growth ceases (i.e., the branch has exhausted its maximum length), the dynamic letter will be replaced by the ‘]’ pop letter (number 6 within str(:,1)).

**Table: Letters and their array structure**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| letter | letter ID | # of rows |  |  |  |  |
| F | 1 | 1 | length | age | root type | 0 |
| f | 2 | 1 | length |  |  |  |
| C | 3 | 1 | R | G | B |  |
| # | 4 | 1 | diameter |  |  |  |
| [ | 5 | 1 | 0 | 0 | 0 | 0 |
| ] | 6 | 1 | 0 | 0 | 0 | 0 |
| + | 7 | 1 | alpha | 0 | 0 | 0 |
| - | 8 | 1 | alpha | 0 | 0 | 0 |
| & | 9 | 1 | angle? | 0 | 0 | 0 |
| ^ | 10 | 1 | angle? | 0 | 0 | 0 |
| \ | 11 | 1 | beta | 0 | 0 | 0 |
| / | 12 | 1 | beta | 0 | 0 | 0 |
| | | 13 | 1 | 0 | 0 | 0 | 0 |
| r | 14 | 1 | alpha | beta | 0 | 0 |
| root section growth | 300 | N | length | age | root type | 0 |
|  | pos |  |  |  |  |  |
|  | pgf | ptr |  |  |  |  |
|  | t0 | tend | l0 | lend | rlt | dx |
|  | iheading |  |  |  |  |  |
|  | successor |  |  |  |  |  |
| delay | 301 | N | t | tend | 0 | 0 |
|  | pos |  |  |  |  |  |
|  | successor |  |  |  |  |  |
| branching | 302 | N | cbn | nob | 0 | 0 |
|  | pos |  |  |  |  |  |
|  | pgf | ptr |  |  |  |  |
|  | rlt | dx | 0 | heading |  |  |
|  | times |  |  |  |  |  |
|  | delays |  |  |  |  |  |
|  | lateral |  |  |  |  |  |
|  | next |  |  |  |  |  |
| create root | 303 | N | type | 0 | 0 | 0 |
|  | pos |  |  |  |  |  |
| create successor | 304 | N | not | 0 | 0 | 0 |
|  | pos |  |  |  |  |  |
|  | probabilities | |  |  |  |  |
| myc section growth | 400 | N | length | age | roottype | 0 |
|  | pos |  |  |  |  |  |
|  | pgf | ptr |  |  |  |  |
|  | t0 | tend | l0 | lend | rlt | dx |
|  | iheading |  |  | b\_age | theta | thetas |
|  | successor? |  |  |  |  |  |
| create myc | 403 | N | type | 0 | 0 | 0 |
|  | pos |  |  |  |  |  |

## Cartesian Coordinates

The L-system string can be converted into Cartesian coordinates using the getSegments function which has the following form:

[x1,x2,r,color,time,type,indS]=getSegments(str, pos)

where x1 is the x, y, and z coordinates of one segment endpoint, x2 the coordinates of the other endpoint, r is segment radii, color is the user defined color, time is creation time, type is the index of root type, and indS is the index within the L-system string.

## RSWMS Format

The RSWMS format is the formatting required for the calculation of the standard uptake fraction and effective root system resistance. RootBox now comes with a function which converts the L-system string into RSWMS format:

[seg\_info, tip\_info]=convert2RSWMSinput(str, pos)

where seg\_info and tip\_info are defined as:

seg\_info = [Nseg, X, Y, Z, connected, order, Nbranch, length, surface, mass (=0), age]

tip\_info = [Nseg, X, Y, Z, segment, order, edge, length\_branch]

# Troubleshooting

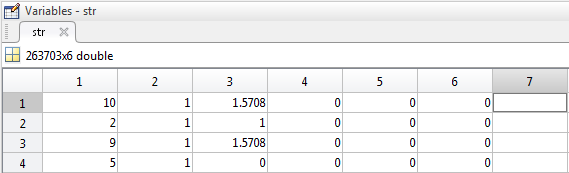
## Issue 1: Non-zero origin

It is unknown why this occurs, but in some simulations, the root systems begin at (0, 0, -1) instead of (0, 0, 0). This can be corrected by modifiying the ‘str’ matrix. The origin point will occur within the first few lines of the str matrix.

*Example:*

Row 1: Pitch down at a 90o angle.

Row 2: No line move forward 1m



Which produces a root system that is shifted gravitropically down by 1m. By changing the value in (2, 3) to 0, this repositions the root system at the origin. Note, this may cause some limbs to be above the “surface.” This can be prevented by changing (2,3) to an appropriate non-zero z-value.

*Code fix for this issue:*

filename: createRootSystem.m

|  |
| --- |
| 80 %  81 % set up root system  82 %  83 rootsystem = [pitchdown;letter('f',0);pitchup]; %planting depth 1 cm  Make sure the highlighted value is 0 or whatever you want your zero-order offset to be. |