

Probabilistic Models to Predict the Mass and Shapes of Leaves in a Tree

Control Number 17164

February 13, 2012

Abstract

The task is to create a mathematical model to describe and classify leaves. We have come up with probabilistic models that are able to predict the mass and shapes of leaves in a given tree. We begin by constructing a model of a tree based on physical principles of branching algorithm and bifurcation concepts to approximate the number of leaves in a tree. We then improve on this model to further emulate the physical appearance of a real tree, while enabling us to vary the shape of the tree. Furthermore, research and further analysis of photosynthetic rate and light irradiance from meteorological database data [1] have enabled us to relate light intensity and Leaf Mass per Area, enabling us to calculate the most likely mass of a leaf in a tree. Using the data obtained and our model, we are able to predict the mass of leaves in a given tree. This model is useful to be used for swift approximation in botanical and ecological research. Moreover, we have also created a completely different probabilistic model to analyze which leaf shape is the best for a specific type of tree in a given region. Simulations show that the sword leaf shape is the most economical to be employed by any tree in any region. Since we know that not all trees have sword-shaped leaves, this proves the point that plants do not always aim to have leaves that maximize surface area exposed to sunlight, as there are other factors that must be taken into consideration [2]. Lastly, we show that our model can be further calibrated and extended to more environmental conditions into account.

Table of Contents

1 Introduction	3
2 Outline of Our Approach	3
2.1 Key Definitions.....	4
2.2 Assumptions.....	4
3 Constructing the Model of a Tree.....	4
3.1 Vertical Bifurcation Model	5
3.2 Vertical Trunk – Horizontal Branch Bifurcation Model	6
3.3 Fountain Tree Model	7
3.4 Justification of Variables.....	8
4 Seasonal and Regional Variations in Leaf Mass	9
4.1 Relationship between Photosynthetic Rate and LMA.....	9
4.2 Relationship between Light Irradiance and Photosynthetic Rate.....	11
4.3 Regional and Seasonal Leaf Mass Calculations.....	12
5 Predicting the Most Efficient Leaf Shape.....	13
5.1 Light Irradiance Probabilistic Model A.....	13
5.2 Light Irradiance Probabilistic Model B.....	15
5.3 Justification of Variables.....	17
6 Results	21
6.1 Leaf Mass Predictions	21
6.2 Leaf Shapes Classifications	23
7 Further Considerations	25
8 Recommendations and Conclusions	25
9 References.....	26
10 Appendices.....	27
10.1 MATLAB Code.....	27
10.2 Leaf Shapes.....	67
10.3 Branching Models.....	68
10.4 Letter to a Scientific Journal Editor.....	69

1 Introduction

The term “leaf” most often corresponds to the foliage leaves, which are the broad, flattened, mostly green structures that are directly responsible for providing energy for the plant through photosynthesis. Absorbing carbon dioxide from the air and using sunlight as the energy source, the chlorophyll in leaves produces sugar and oxygen, two components that are essential to human survival. However, photosynthesis is not the only function of a leaf, for there are other functions as well, such as preventing excessive water loss from a plant and protecting the plant from fungi and bacteria [2]. Considering such significance of leaves for the plant in specific and the entire biome in general, it is of significant use to be able to model leaf shapes and the total mass of leaves in a tree to facilitate swift approximation models to be used in botanical and ecological research. Therefore, our goal is to design and test a mathematical model to describe and classify leaves.

2 Outline of Our Approach

- Our aim is to build mathematical models that are able to predict, within reasonable limits, the total mass of the leaves in a tree, given specific parameters, as well as the efficiency of the different shapes of leaves in a tree, to predict how the leaf shapes correlate with the different shapes of trees and environmental conditions.
- We begin by constructing the model of a tree using simple vertical bifurcation algorithm based on physical characteristics of tree trunks and branches.
- We decide to improve on this model to make it more realistic by implementing a Vertical Trunk – Horizontal Branch Bifurcation (VT-HBB) model. This enables us to replicate a real physical tree with greater accuracy and enables us to vary the modeled tree shape to emulate the different physical tree shapes.
- Using this model, we can examine the number of leaves that a particular tree can support by determining the length of branch in the tree.
- We construct a mathematical equation relating the light irradiance in a particular region in a particular season with the Leaf Mass per Area (LMA) to predict how heavy a leaf is per unit area, provided a specific light irradiance.
- Using the calculated LMA and the number of leaves that a tree can support, we are thus able to predict the mass of leaves in a tree based on the given parameters (solar irradiation, geographical region, and tree shape).
- Our next step is to build a model to predict the most likely leaf shape in a given shape of a tree, based on the light irradiation-efficiency concept. This enables us to predict which leaf shapes are more likely to be present in a tree with a specified shape in a

specific region at a given season and analyze what impacts the different leaf shapes have on the tree.

- Lastly, we extend our model to explore several exciting possibilities, including using the interaction between rainfall, solar irradiation, and soil condition to better predict the leaf mass and shapes.

2.1 Key Definitions

Throughout this paper, we are using terms that are prone to misinterpretation or confusion. Hence, we have defined the following terms so as to enhance the clarity of our paper.

- The term 'leaf' is assumed as a foliage leaf, which characteristically mostly green in color, broad, and flattened or leaves of conifers, which are needle-like. Other forms of leaves such as sclerophyllous leaves, or succulent leaves are not within the scope of this paper.
- The 'shape' of a leaf refers to the characteristic outline of the broadest cross-section of the leaf. It does not take the vertical cross-section into account.

2.2 Key Assumptions

- We assume that in a tree, the size and shape of leaves are roughly the same. Although this is definitely not true in a real tree, the averaging process takes these variations into account.
- We assume that the regions of the world can be broadly classified into three based on temperature and solar irradiation: alpine, temperate, and tropics. Since we do not include desert vegetation in our analysis, we choose not to include desert region as one of our regions.
- We have decided to narrow down the seasons in temperate and alpine climate into two: spring/summer and autumn. This is because the variations in inter-geographical environmental conditions far outweigh the seasonal variations in a place [1]. Hence, we just take into account two distinct seasons: spring, when the plants are at its peak, and autumn, when seasonal plants start to shed their leaves.

3 Constructing the Model of a Tree

To be able to predict the mass of the leaves in a tree, we are using equation (1). The equation simply states that the mass of leaves in a tree is the product of the mass of the leaves (M_{leaf}) with the number of leaves the tree has ($NumLeaf$).

$$M_{treeleaf} = M_{leaf} * NumLeaf \quad (1)$$

Hence, we have come up with two models to represent a real physical tree in order to calculate NumLeaf.

3.1 Vertical Bifurcation Model

Our first model is the Vertical Bifurcation (VB) model. In this model, a tree can be represented by a branching structure, in which each branch splits into two sub-branches at the end. This model is proposed by Honda [3] and it is suggested that the overall shape of a tree is basically determined by two parameters, branch angle and ratio of branch lengths.

The observed branch angles are very similar to optimal theoretical angles that produce maximum effective leaf area [3]. This suggests that branch angles are adapted for maximum exposure of leaf surface to sunlight. In another paper [4], the observed values for optimum branch angle are 24.6 and -41.4 with respect to the branch before it. This makes the average angle separation between branches to be about 61-66 degrees.

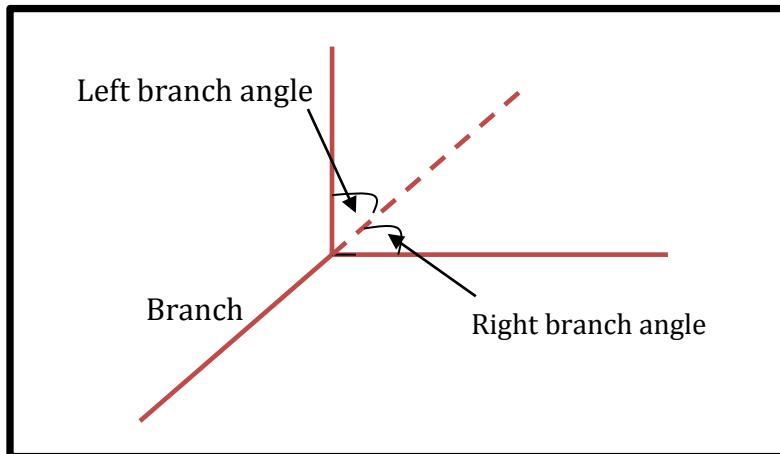


Figure 1: Branch bifurcation angles

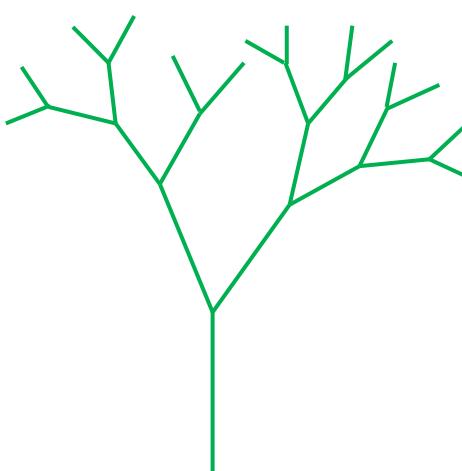


Figure 2: Vertical Bifurcation Model

The strength of this model is that we are able to vary the length of each sub-branch by using a ratio of branch lengths ($\text{Ratio}_{\text{branch}}$). This corresponds to the real physical trees, in

which the branches decrease in length as the branch gets further from the main stem. Furthermore, the branching reflects the real branching structure of leaves by using the aforementioned branch angles.

3.2 Vertical Trunk – Horizontal Branch Bifurcation Model (VT-HBB)

However, we realized that the VB model is only a 2D model that does not represent the trees in real life. Hence, we come up with an improved model, which is the Vertical Trunk – Horizontal Branch Bifurcation Model (VT-HBB). In this model, we utilize our VB model as the shape of the branch in the VT-HBB models. We assume that our branches are fully horizontal, and that the main stem is fully perpendicular to the ground. This way, we are able to model a 3D tree, with the main trunk pointing upwards, and the branches radiating outwards from the trunk.

Furthermore, we are able to vary the approximate radius of every branch level, so as to give us the flexibility to model different shapes of trees, such as columnar, full-crowned, and pyramidal. Figure 3 shows the different shapes of trees.

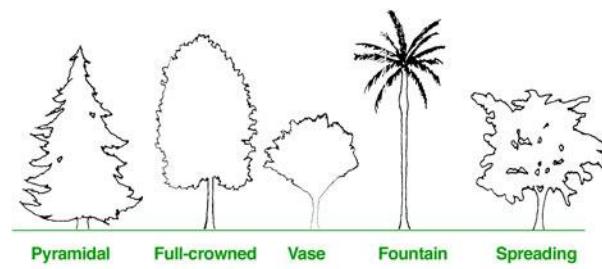
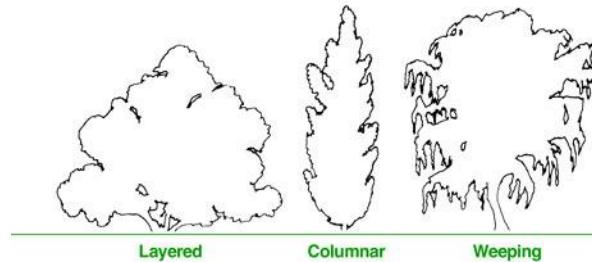
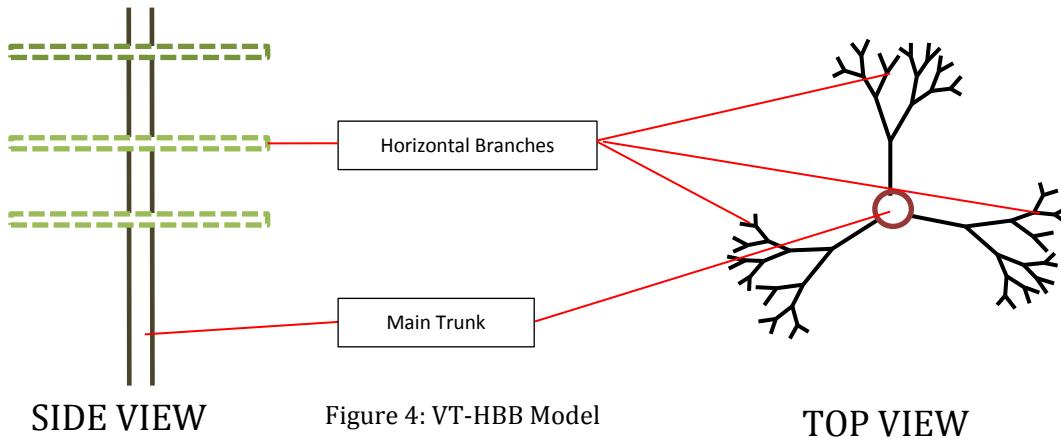


Figure 3: Shapes of Leaves [5]



By utilizing the VT-HBB model, we can predict, with reasonable accuracy, the overall branching structure of the tree. From there, we are able to determine the total branch lengths, and thus the possible number of leaves growing on the tree. Moreover, by varying the height of the tree and the horizontal spacing between each branching layer, Figure 4 shows the VT-HBB model.



However, even from this model, we have to make some assumptions. The first assumption is that the leaves between the horizontal layers do not interact with each other (one example would be that leaves in layer A will not block the space of a potential leaf in layer B).

3.3 Fountain Tree Model

For completeness, we decide to model the fountain type tree. The growth of the fountain type tree is different from the previous two groups of trees, and our VT-HBB model cannot be realistically used to model the fountain type tree. As a result, we created a new model that examines the top of the fountain type tree, where the branching of leaves occurs. We model this branching base as a top hemisphere, and we make a few assumptions about the branching characteristics. Firstly, we define the fountain type leaves as the large leaf containing the main stem of the branch and all the smaller leaves attached to this main stem. Secondly, this hemisphere is filled maximally with leaves. Next, we once again make the assumption that all the leaves are exactly the same.

This model does not require the height of tree as it predicts that the circumference of the trunk is the most important factor that affects the total number of leaves on the tree and hence the total leaf mass. This is physically consistent as fountain type trees grow out directly from the trunk of the tree. Hence, the larger the trunk, the more available space there is for leaves to grow. Other inputs for this model mostly include the characteristics of the fountain type leaf, such as the arc length of the leaf base, which is essential the length of the leaf base, and the thickness of the leaf base. These two inputs directly affect the number of leaves that can grow on the hemisphere.

This model operates in a systematic way. First, it uses the thickness of the leaf base and the radius of the hemisphere (which is also the radius of the tree trunk) to calculate the number of possible “rings” that leaves can be created around. Subsequently, the model

creates the maximum number of leaves it can fit along every ring. It then totals up the number of leaves that the tree can fit onto its hemisphere.

3.4 Justification of Variables

We calculate the number of leaves on a tree according to the following variables:

For columnar and pyramidal trees we need to know the height of the first level of branches from the ground, the height of the second branch level from the first, the length of the first branch level, length of first branch in the first branch level, ratio of the length of $(n+1)$ -th branch to n -th branch, average number of leaves per length on the first branch in the first branch level, ratio of number of leaves per unit length on $(n+1)$ -th branch to n -th branch. We define n as an increasing number of layers going up the tree.

Table 1: Variables used in the VT-HBB model for columnar and pyramidal trees

Type of Tree	Columnar	Pyramidal
H_{first}	$\frac{h}{9}$	$\frac{h}{7}$
H_{disecond}	$\frac{2H_{\text{first}}}{9}$	$\frac{3H_{\text{first}}}{5}$
$L_{\text{firstlevel}}$	$\frac{h}{5}$	$\frac{h}{4}$
$L_{\text{firstbranch}}$	$\frac{h}{60}$	$\frac{h}{40}$
$\text{Ratio}_{\text{branch}} [6]$	$\frac{L_{\text{firstbranch}}}{2}$	$\frac{3L_{\text{firstbranch}}}{5}$
Leaf_{ave}	60	200
$\text{Ratio}_{\text{leaf}}$	1.1	1.2

H_{first} = Height of the first level of branches from the ground.

H_{disecond} = Height of the second branch level from the first.

$L_{\text{firstlevel}}$ = Length of the first branch level.

$L_{\text{firstbranch}}$ = Length of the first branch in the first level.

$\text{Ratio}_{\text{branch}}$ = Ratio of the length of $(n+1)$ -th branch to n -th branch.

Leaf_{ave} = Average number of leaves per length on the first branch.

$\text{Ratio}_{\text{leaf}}$ = Ratio of the number of leaves per unit length on $(n+1)$ th branch to n th branch.

h = Height of the tree

For the other types, the variables are in this paper are height of trunk, ratio of the length of $(n+1)$ th branch to n th branch, average number of leaves per unit length on trunk, ratio of number of leaves per unit length on $(n+1)$ th branch to n th branch. Since the relationship of

these variables have the same relationship with h for spreading, layered and vase trees, we combine into one genre and call them SLV.

Table 2: Variables used in the VT-HBB model for other kinds of trees

Type of Tree	H_{first}	Ratio _{branch}	Leaf _{ave}	Ratio _{leaf}
SLV	$\frac{h}{5}$	$\frac{3h}{5}$	60	1.1
Weeping	$\frac{h}{3}$	$\frac{h}{2}$	150	1.1
Full-crowned	$\frac{h}{4}$	$\frac{3h}{5}$	75	1.1

Through observation, we find that the difference between $n+1$ and n branch levels is approximately the same. Thus, if H stands for the total height of the tree, F_1 stands for the height of first branch level from the ground, F_2 stands for the height of the second branch level from the ground, Total number of branch levels = $\frac{H - F_1}{F_2 - F_1}$

The variable *heightSecondBranchLevel* in the MATLAB code is equivalent to $F_2 - F_1$.

4 Seasonal and Regional Variations in Leaf Mass

In order to determine the variations in leaf mass (M_{leaf}), we use the fact that leaf mass is the product of the Leaf Mass per Area (LMA) with the total surface area of the leaf (SA). By determining both LMA and SA, we are able to calculate the mass of a single leaf, and even the mass of the leaves in a tree.

$$M_{\text{leaf}} = \text{LMA} * \text{SA} \quad (2)$$

4.1 Relationship between Photosynthetic Rate and LMA

From the paper by Jin et al. [7], there is positive correlation between A_{mass} , or mass-based photosynthetic rate and Specific Leaf Area (SLA) [Figure 5].

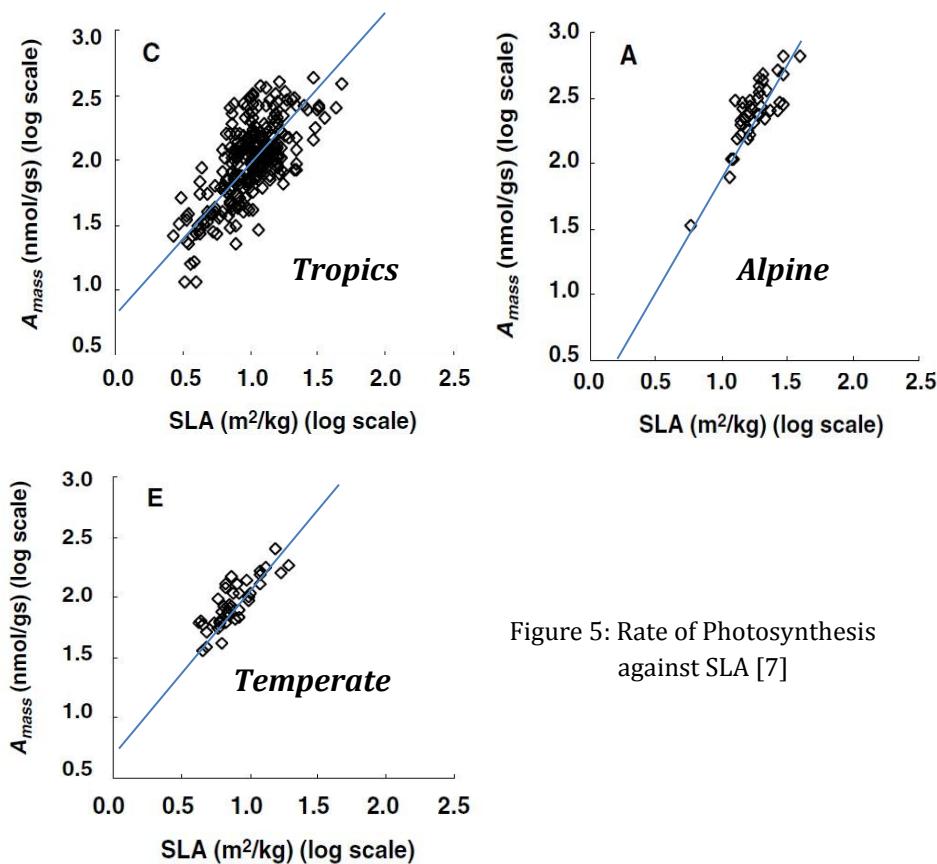


Figure 5: Rate of Photosynthesis
against SLA [7]

SLA is just the reciprocal of LMA, so we can deduce that there is negative correlation between A_{mass} and LMA. From the paper mentioned, we are also able to vary the correlations based on the geographical region in the world. For the purpose of this paper, we are only concerned about three region types: tropics, alpine, and temperate. We find three different equations, one each for a specific region [7]:

$$\log A_{\text{mass}}(\text{Tropics}) = 1.23(\log \text{SLA}) + 0.857 \quad (2)$$

$$\log A_{\text{mass}}(\text{Temperate}) = 1.41(\log \text{SLA}) + 0.575 \quad (3)$$

$$\log A_{\text{mass}}(\text{Alpine}) = 1.66(\log \text{SLA}) + 0.317 \quad (4)$$

From the paper done by Stephen Hunt [8], there is always a positive correlation between solar irradiance and photosynthetic rate, although up to a certain limit, the light saturation point is reached and photosynthetic rate no longer increases. Using this deduction, we further obtain data from Jin et al. [7] to determine the solar radiation range used in the data above and use it to construct the model that predicts the relationship between solar irradiation and photosynthetic rate.

Table 3: Environmental Data [7]

Biome	Solar Irradiation (W/m ²)	Temperature (°C)
Tropical	131-192	-2.8 – 8.7
Temperate	98-188	-1.5 – 20.5
Alpine	118-156	15.8 – 27.3

4.2 Relationship between Photosynthetic Rate and Solar Irradiance

From the data obtained above, we know that there is a light saturation point (LSP), which is a point where increasing light irradiance does not increase the rate of photosynthesis. Hence, we model the relationship between the photosynthetic rate and solar irradiance using an exponential decay equation. By using the solar irradiation data from table 3 and specific points from figure 5, we have come up with the following equations:

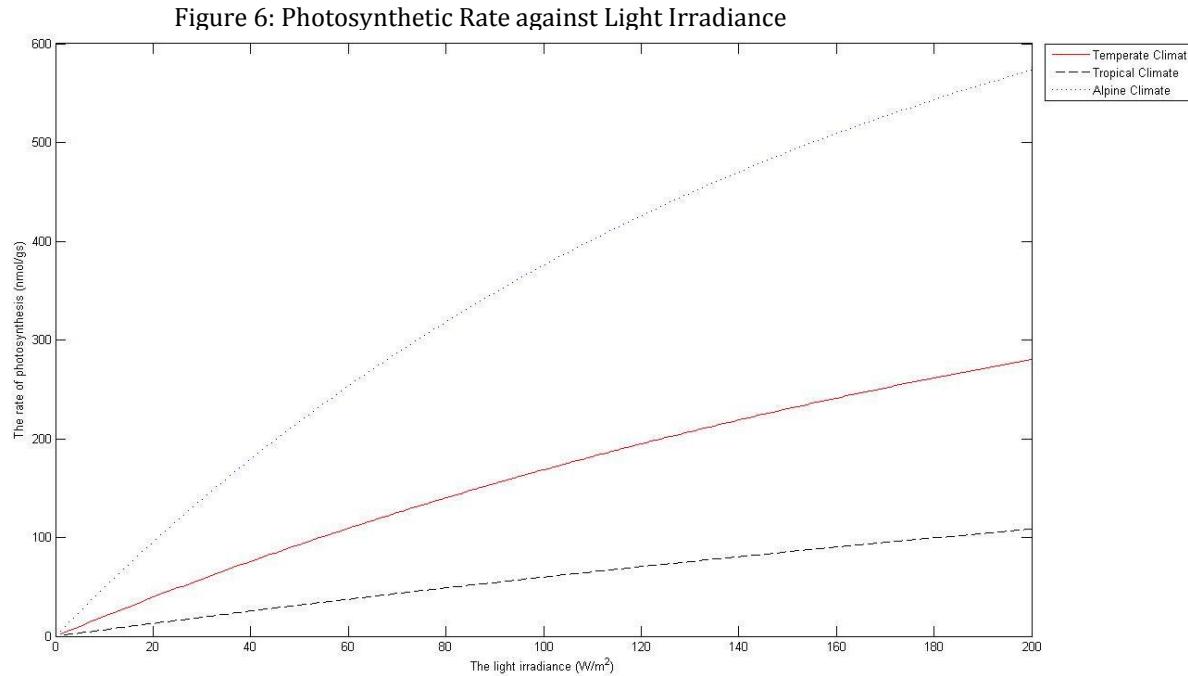
$$PhRTemp = 10^{2.7} * (1 - e^{-0.0041LIr}) \quad (5)$$

$$PhRTrop = 10^{2.5} * (1 - e^{-0.0021LIr}) \quad (6)$$

$$PhRALp = 10^{2.9} * (1 - e^{-0.0064LIr}) \quad (7)$$

PhRTemp is the photosynthetic rate in the temperate region; PhRTrop is the photosynthetic rate in the tropical region; PhRALp is the photosynthetic rate in the alpine region. LIr is the light irradiance that is varied from 0 to 200 W/m².

It is worth noting that based on our equations and graph (figure 6), the leaves of trees that grow in the alpine conditions are more efficient in performing photosynthesis than the trees that grow in other region. On the flip side, the leaves of trees that grow in the tropical region are the least efficient in performing photosynthesis amongst all the trees across the regions. This interesting finding might be due to the fact that the leaves of trees in the tropical regions need not be concerned about the amount of light reaching the leaves as the level of solar radiation is mostly high perennially. This is in contrast with the leaves of alpine trees that must maximize the intake of solar radiation to perform photosynthesis as the amount of radiation reaching the leaves is much lower as compared to the radiation reaching the tropical regions (due to the lesser angle of impact of sunlight and due to the longer distance required for sunlight to travel to the higher latitude regions rather than equatorial regions).



4.3 Regional and Seasonal Leaf Mass Calculations

Using figure 6 and equations 2, 3, and 4, we can determine the LMA of leaves in a particular region, based on the most common light intensity in that region. Going further, using 120cm² as the most common leaf area, we can determine the approximate mass of a single leaf.

We realize that the values of LMA and SLA use the dried mass of a leaf. Hence, we have to calculate the mass of fresh leaf from the dried mass using the formula below, where DtFR is the Dry to Fresh Mass Ratio:

$$DtFR = \frac{\text{Mass of dry leaf}}{\text{Mass of fresh leaf}} \quad (8)$$

Table 4: Dry to Fresh Mass Ratio Based on Seasons [9]

Biome/Season	DtFR
Alpine/Autumn	0.5
Alpine/Spring	0.3
Temperate/Autumn	0.5
Temperate/Spring	0.3
Tropics	0.3

The values of DtFR are obtained from [9], with several adjustments to take into account varying seasons. In the autumn, the leaves begin to dry, so DtFR would increase. In the spring or summer, DtFR would be lower. The calculation of M_{leaf} is as follows:

$$M_{leaf} = \frac{LMA * SA}{DtFR} \quad (9)$$

We utilized the meteorological data in different cities which are located in different locations in China as a basis of our solar irradiation values. Based on our calculations, we come up with the predicted leaf mass, given a tree at a particular geographic region (biome) and a particular season (table 5). The result means that the mass of a leaf in the tropics region is the greatest, whereas the mass of a leaf in the alpine is the least. This result makes sense, as in the tropics, the leaves tend to be much bigger than the ones in other regions, while the leaves of trees in the alpine region tend to be much smaller.

Table 5: Predicted Leaf Mass for trees in a particular biome at a particular season

Biome/Season	Leaf Mass (g)
Alpine/Autumn	1.14
Alpine/Spring	1.67
Temperate/Autumn	1.55
Temperate/Spring	2.30
Tropics	4.72

5 Predicting the Most Efficient Leaf Shapes

5.1 Light Irradiance Probabilistic Model A (LeafSA1 Model)

In the determination of the leaf shape, we decided that two very crucial factors that the tree considers are the total rate of photosynthesis and total rate of transpiration. Since these two factors are largely dependent on the surface area of the leaves, we decided to make a model to find out how much surface area of leaves there are on a specified branching level of the tree, given a specific leaf shape. We define branching level as a level of the tree that branching from the trunk occurs. Our main goal is to qualitatively evaluate our results of the total surface area of leaves on a branching level for different leaf shapes. This can then help us to analyze which shape would produce the most optimum results for the tree, either through high rate of photosynthesis or low rate of transpiration. Our first model, 'leafSA1' examines one level of branching of the tree.

This model requires the radius of the branching level (in cm), the probability that a leaf grows at a point within the level, and a matrix that is the 1s and 0s make-up of a leaf. This matrix has to be pre-made. We create five different matrices that represent the shape of five different types of leaves, specifically: 'sword', 'elliptical', 'maple', 'round' and

'bipinnate'. (Refer to appendix for an illustration of these matrices). One cell of the matrix represents 1 cm^2 is area. These matrices are made in proportion to each other, giving a relatively good comparison of surface area. The radius of the branching level will be given in units of cm, and the model uses this radius to create an array that represents a branching level of the tree. Lastly, the probability that a leaf grows at a point within the level is determined externally with respect to the intensity of sunlight. The graphic representation of this model can be seen in figure 7.

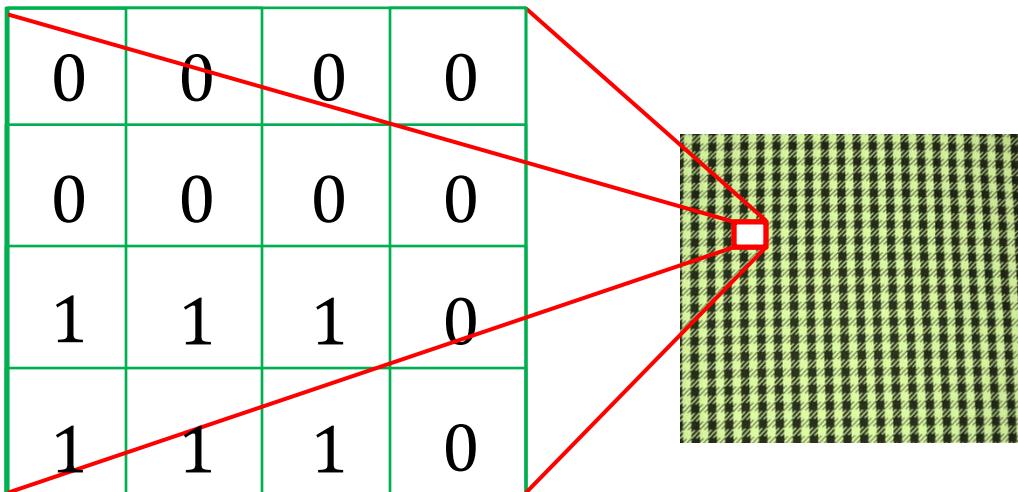


Figure 7: LeafSA1 and LeafSA2 Algorithm

Before we simulate the leaves growing on the branching level, we attempt to make the model more realistic by creating branches in the level, which may obstruct the growth of leaves. We model the growth of branches in the level using a formula that is detailed and justified in a later section. Once the branches are set, we then proceed to grow the leaves. The model searches for areas in the level where there are sufficient space for the leaf to grow, then a leaf either grows or not according to the given probability that a leaf will grow there. Since the leaves are provided in a 1s and 0s matrix, where the 1s represent part of the leaf, the model checks if leaves will overlap with one another or with the branches. If there is clearly no overlap, the model allows for three leaves to grow, one in each of the x, y, and z direction, so as to model a three dimensional situation.

The surface area of each leaf is calculated at the beginning of the simulation. This is a straightforward process since each 1 in the leaf shape matrix represents 1 cm^2 . As the simulation runs, the total surface area of all the leaves is summed. The output of this model is the total surface area of all the leaves in that particular branching level.

With this total surface area, we could then proceed to find the total rate of photosynthesis and transpiration of the leaves in this branching level. Since the total surface area would

change with different leaf shapes as they have different individual orientations and surface area, we can then compare the effect of the leaf shape on the total rate of photosynthesis and transpiration. If we manage to weigh the importance of photosynthesis versus transpiration, we would then know what leaf shape would be the best for a tree given a set of specified conditions (i.e. radius of branching level and probability that the leaf would grow, which is directly related to the intensity of sunlight). At this point, we decided that the model is limited as it only models the growth of leaves on an arbitrary branching level. We decided to expand on ‘LeafSA1’ in order to obtain a realistic model and hence a more accurate conclusion.

5.2 Light Irradiance Probabilistic Model B (LeafSA2 Model)

In our second model, we decided to scale up our assumptions, variables, and simulations. We extended our LeafSA2 by using it to calculate the total surface area of all the leaves in a tree given a set of specified conditions. Initially, we modeled a columnar type tree, which has a fixed radius with each branching level. In this model, we required inputs such as the height of the tree (in meters), the intensity of sunlight (in Watts per hour), and the shape of the leaf which growth is to be simulated.

In this model, new variables are considered. These variables are unique to the leaf shape. They include the following: the transmittance of the leaf, which is related to the amount of sunlight that is able to pass through it; a proportionality constant, which scales the sunlight intensity.

Before the model simulates the growth of the tree, it first extracts the 1s and 0s matrix corresponding to the leaf shape, and prepares it to be used in ‘LeafSA1’. It then calculates the radius of each of the branching level, which is a simple relation with respect to the height of the tree. Using the height of this columnar tree, a separate function ‘treeData’ then provides this model with information such as the number of levels of branching. With that, it begins growing the tree level-by-level, starting from the top level (where the radius is the smallest), and ending at the base level (where the radius is the largest). At each branching level, the intensity is calculated through a formula (detailed in later sections) relating the intensity of sunlight, transmittance and the probability that a leaf will grow at a point in the branching level above it. At each branching level, ‘LeafSA1’ is called to obtain the surface area of leaves in that particular branching level. After running through all the levels of the tree, the total surface area of all the leaves in all the branching levels is summed.

After this model for columnar type trees was completed, we proceeded to create a model for pyramidal type trees. This new model for Pyramidal type trees was similar, except that the radius of each branching level now varies with the level, and the intensity of the sunlight on that particular branching level is a different function of new variables. The

radius of each branching level is an extensive formula (detailed in later sections) that relates the total number of levels, the current level, the height of the tree, the height of the first branching level, and the height of the second branching level. The intensity of sunlight at a particular branching level is also a function of these variables together with transmittance, a scaled initial intensity of sunlight, and the probability that a leaf will grow at a point in the branching level above it. Similarly, 'LeafSA1' is called for each level, and the total surface area of leaves in all branching levels of the tree is summed.

For completeness, we decided to make another similar model for all other types of tree types, namely, Spreading, Layered, Vase, Weeping and Full Crowned. We deliberately left out Fountain type trees as their leaf shapes were too drastically different from the other trees. Furthermore, the fountain type leaves are too huge in surface area to be created in a 1s and 0s matrix alongside the other leaf shapes. Since the leaves for these remaining tree types are roughly similar, we decided that it is a good approximation to have simply one model to simulate the growth of leaves on such trees.

For this new model, we decide that the levels increase in radius from the top level to the middle level. Around the middle, the radius stays constant for a few levels, before decreasing again from the middle level to the bottom level. One consideration we made was that the rate of increase in radius from the top to middle level is not as sharp as in the pyramidal case. The rate of increase is somewhat "rounded" in such trees instead. Similarly, the rate of decrease in radius from the middle to the base level is also not sharp. As a result, we appropriately scaled the formula for calculating change in radius in a pyramidal case and used it in this model. When scaling the formula for radius, we also noted that a tree of these types has a more rounded top and a sharper bottom. The formula for calculating intensity in this model was imported from both the Columnar and Pyramidal model. When the radius was changing, the intensity followed that of the Pyramidal model, while when the radius was constant, the intensity followed that of the Columnar model. Once again, the simulation creates the different branching levels and the total surface area of the leaves is summed.

LeafSA2 is in fact a collection of three different models for three different groups of trees. With this model, we can then integrate additional information to make robust conclusions about the total rate of photosynthesis and transpiration of the trees with different leaf shapes. In order to do this, we created various support functions to provide the necessary information. We aim to only require inputs of region, season, type of tree, and height of tree.

Firstly, we created a function, 'intensitySunlight' that will provide the intensity of sunlight given a region and season. On top of that, we created two other functions that will provide the rate of photosynthesis (nanomols per gram of leaf per second) and transpiration (gram

per hour per 100 cm²) given the region, season, and the shape of the leaf that is being tested. A separate function, 'massPerArea', also provides the grams per cm² of leaf. This is necessary as the photosynthesis and transpiration data is provided in mass and not in surface area. The data inputted into these functions are found through research and will be detailed and explained in later sections. Two other functions, 'treeData' and 'leafShapes' simply store the data for all types of trees and leaves, which will be used to eventually compare the effect of the different leaf shapes given specific conditions.

All of these culminates in a user-friendly script, 'PlotOfPandT', which prompts the user for 4 inputs: 'region', 'season', 'type of tree', and 'height of tree'. With these inputs, the script then models the growth of leaves of all five leaf shapes on a tree. The initial output of this model is the total surface area of leaves using each of the five different leaf shapes. With this output, the total leaf mass can then be calculated, which will then provide the total rate of photosynthesis and transpiration. The final output is a bar chart that compares the total rate of photosynthesis and transpiration of each type of leaf shape with the given input conditions.

This neat model allows the user to compare the effectiveness of each leaf shape in different conditions, hence aiding the user in making a qualitative analysis of why leaves have various shapes in different conditions, and why trees choose to have different leaf shapes. With this model, one can model any combination of region, season, type of tree, and height of tree, in order to find out which leaf shape is the most optimum in terms of rate of photosynthesis or rate of transpiration. A separate script, 'PlotLM2', shows the user how the total leaf mass of the tree differs with the different leaf shapes. Deciding which leaf shape is optimum hence depends on the many conditions provided, and how the results are interpreted and weighed. In the discussion of our results, we attempt to use this model to examine the Alpine tree and the Rubber tree, and compare our results with real life situations.

5.3 Justification of Variables

The probability that a leaf will grow on a random place on the *n*th branch level is governed by the following equation [14]:

$$Pr_n = \frac{I_n - 1.5\% \times I_{fullsummersunlight}}{I_n} \quad (10)$$

Pr_n = the probability that a leaf will grow at a place on the *n*th branch level

I_n = the average intensity of sunlight on the *n*th branch level

$$I_{fullsummersunlight} = 200 \frac{W}{m^2}, \text{ as determined by the data in [1].}$$

Since in our models, we assume that on each level of leaves cannot overlap, the average intensity of the nth layer is the average of the sum of the intensity of sunlight it receives directly from the sun and the intensity that goes through the leaves on the (n-1)th level and reach the nth level.

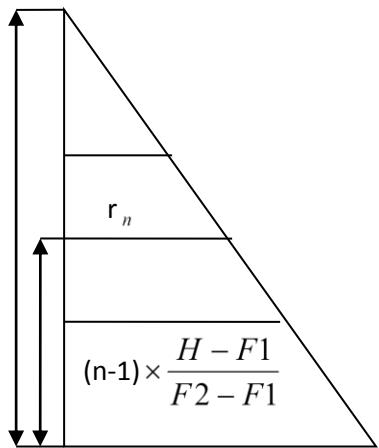


Table 7: Side view of a tree model

$$I_n = \frac{I_1 \times (\pi \times r_{n-1}^2 - \pi \times r_n^2) + I_1 \times \pi \times r_{n-1}^2 \times (1 - Pr_{n-1}) + p \times I_{n-1} \times \pi \times r_{n-1}^2 \times Pr_{n-1}}{\pi \times r_n^2} \quad (11)$$

Where p is the percentage of intensity that shines through a layer, also called transmittance rate.

$I_1 = \frac{I_o}{A}$, where I_o is the intensity that strikes the first layer and A is a proportionality constant that takes into account the effect of leaf area on average light intensity of the layers below.

$$\frac{r_{n-1}}{r_n} = \left(\frac{H - F_1 - (n-2) \times (F_2 - F_1)}{H - F_1 - (n-1) \times (F_2 - F_1)} \right) \quad (12)$$

For the branch structure, we determine a function to represent a physical branch. In order to represent a surface level with branches in a square array, we need to find a function whose value is restricted to $-n$ and n when the variable is from 0 to n . Also, we need to find a function that approximately retains the same shape when n varies. Therefore, we use a

function of sin and cosine whose amplitude and frequency change according to n. The function we find suitable is as follows:

$$Y = \frac{n}{2} \times \left[\sin\left(\frac{0.5}{n^2} \times x^2\right) + \cos\left(\frac{0.15}{n^2} \times x^2\right) \right] \quad (12)$$

For further data, refer to appendix regarding plot for n=100, 1000, 10000.

We run this function once and then exchange x and y to make the branch in the vertical direction to simulate branches in all four directions.

Table 6: Transpiration Rates of Leaves

Leaf type	Sword	Elliptic	Maple	Round	Bipinnate
Transpiration rate [11]	$0.19 \times f$	$0.44 \times f$	$0.21 \times f$	$0.3 \times f$	$0.19 \times f$
Transmittance value [12][13]	0.65	0.6	0.55	0.5	0.7
Proportionality Constant	5	9	7	10	14

The function f is related to temperature and is given in table 7.

The proportionality constant is the approximate ratio of area of the 5 different kinds of leaves

Table 7: Function f [10]

Region	Alpine	Temperate	Tropics
Function f	$\text{Max}\left(0, \frac{t}{10}\right)$	$\frac{3}{20}t$	$\frac{t}{5}$

The above data is determined by an average of values from various sources

For a pyramidal tree, the cross section is a triangle. Looking at figure 9 on the next page,

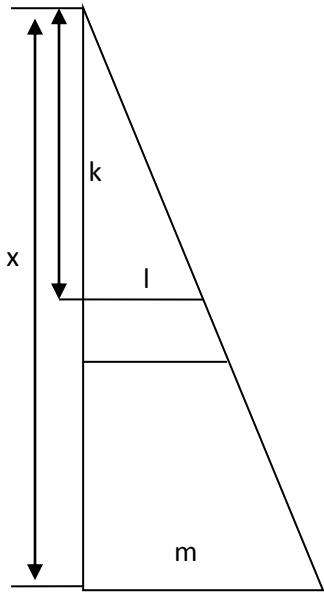


Figure 9: Cross-section of a pyramidal tree

For this kind of pyramidal tree, we assume that the ratio of the height to the base of the triangle is three.

$$\text{Since } \frac{k}{l} = \frac{x}{m}, l = \frac{k}{x} \times m = \frac{k}{3}$$

k is found by subtracting x_n from x , where n goes from 1 to $n-1$; $x_n = (n-1) \times m \times x$

$$\text{so, } l = \sum_{a=1}^{n-1} \frac{x(1 + m - a \times m)}{3} \quad (13)$$

6 Results

6.1 Leaf Mass Predictions

Based on our models, we have run simulations to predict the approximate total mass of leaves in a tree, given specific geographical region and season. The graphs are as follows:

Figure 10: Leaf Mass of Spreading/Layered/Vase Tree (kg) against Its Height (m)

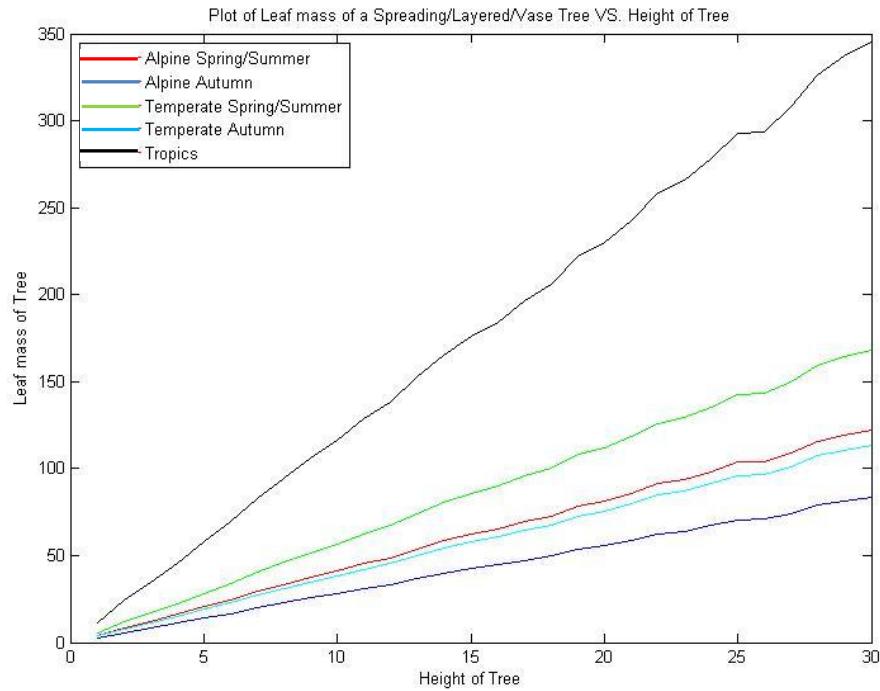
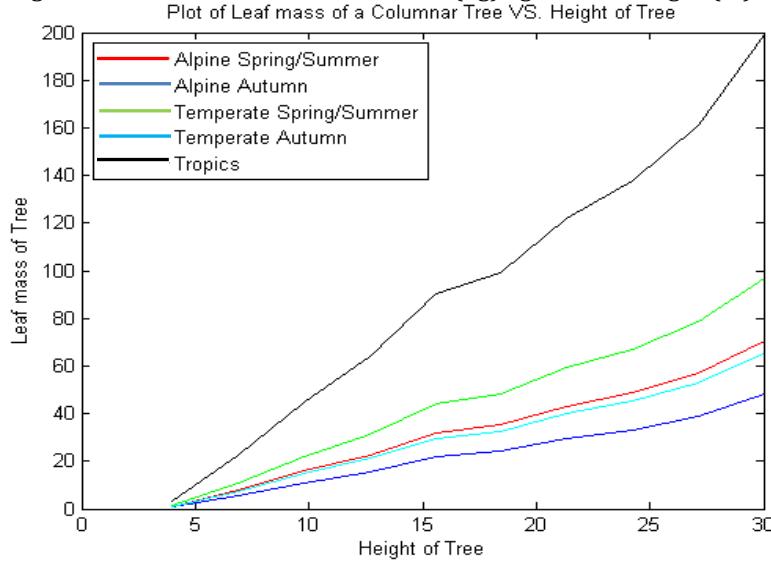


Figure 11: Leaf Mass of Columnar Tree (kg) against Its Height (m)



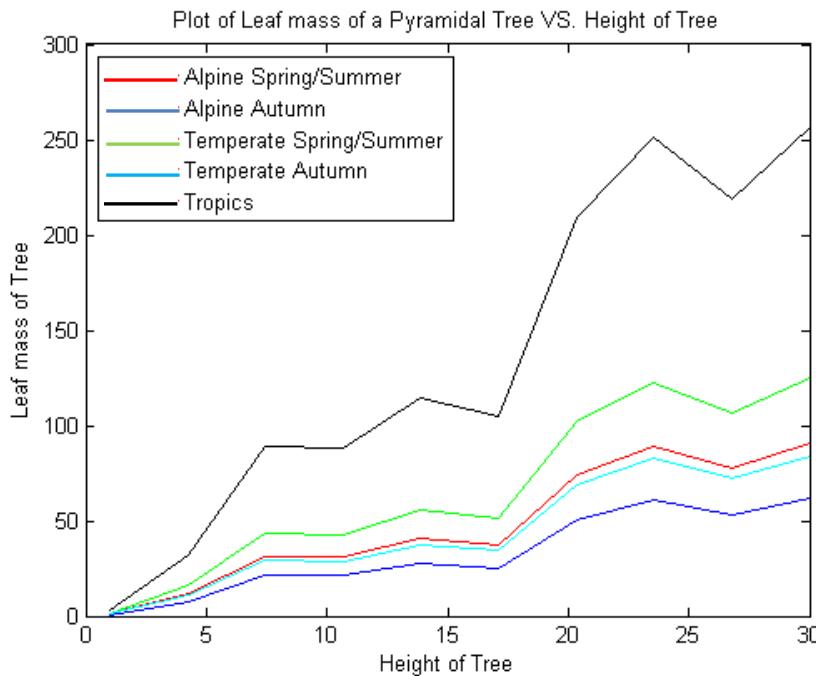
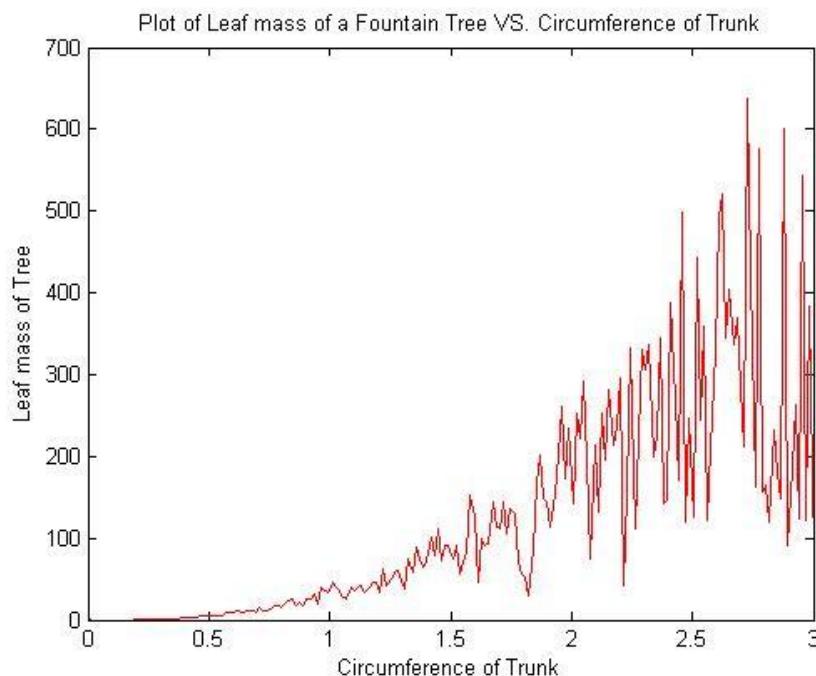


Figure 12: Leaf Mass of Pyramidal Tree (kg) against Its Height (m)

From the above figures, we can see that tropical trees have bigger tree leaf mass for a tree with a given height. In contrast, the mass of the leaves in an Alpine tree during autumn season is the least compared to others. Furthermore, it is evident that trees in spring/summer have larger total leaf mass compared to the same trees in the autumn.

Comparing between figures, our VT-HBB model has shown that a tree with a spreading/layered/vase gives more tree leaf mass as compared to trees with other shapes.

Figure 13: Leaf Mass of Tree (kg) against Circumference of Trunk of Fountain Tree



From the graphs, we can deduce the most probable total mass of leaves in a tree. However, since there is no data that can be found regarding the relationship between tree height and the total mass of leaves of that tree, we can only approximate the validity of our graphs. From qualitative analysis, most of the tree models with 10m height have around 10-30 kilograms of leaves at any one time, which we feel falls within reasonable bounds. Hence, we can say with certain confidence that our VT-HBB model is justified. Moreover, for our fountain tree model, it is shown that the circumference of the tree's trunk positively (although non-linearly) to the tree's mass. From figure 13, the variation in the mass of the leaves varies only slightly when the circumference of trunk is small. However, there is sharp variation in the tree leaf mass when the circumference of trunk is larger.

6.2 Leaf Shape Predictions

Although our simulation is able to perform various combinations of parameters, including tree shapes, seasons, and geographical regions, we have chosen the two most interesting findings that we have obtained. Referring to figure 14, the total rate of photosynthesis in an alpine tree is the maximum when sword-shaped leaf is present. This shape of leaf also minimizes the water loss through evapotranspiration, which is ideal for alpine conditions as the amount of liquid water available in the ground is scarce.

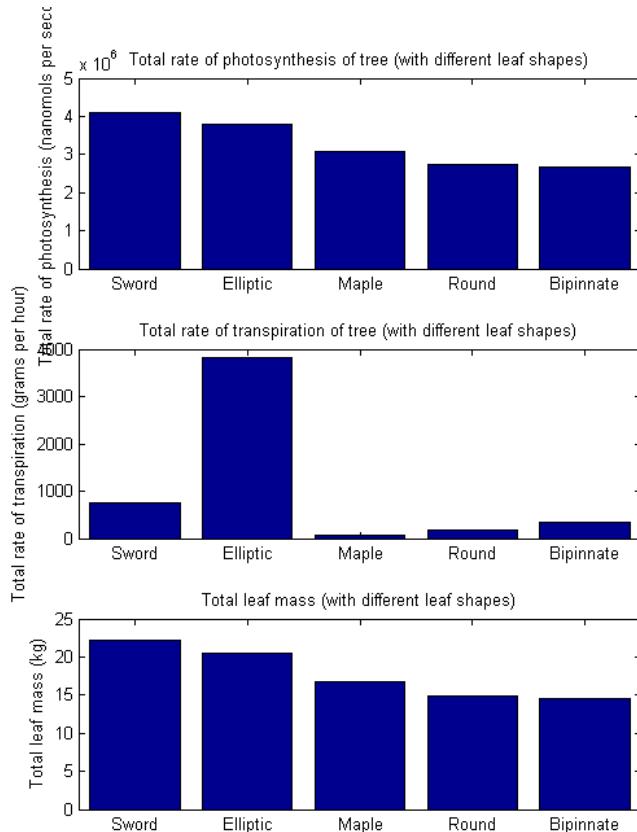


Figure 14: Alpine Leaf Shape Predictions using our LeafSA2 Model

We also run our simulation using tropics weather and summer as our parameters. We also obtain that sword-shaped leafs emerge first in terms of the total rate of photosynthesis allowed within a tree. However, it is evident that the real physical tree does not often have this characteristic, as the type of leaves that characterize tropical region is generally broad and elliptical in shape. We explain this phenomenon by arguing that the leaves in a tropical rainforests do not aim only to maximize photosynthesis and minimize evapotranspiration as these areas have high annual rainfall. According to Mauseth [2], many structural and physiological aspects of leaves that make them waterproof, pathogen resistant, and other beneficial properties in some way interfere with photosynthesis. Therefore, there is an unaccounted interplay of other factors that we have not taken into account.

Not only that, our model shows that elliptic leaf is not far behind sword-shaped leaf in terms of rate of photosynthesis per tree. Moreover, the high rate of water loss here is also characteristic of the leaves that grow in the tropical region. Hence, although our model does not fully predict the most likely leaf to be employed in the tropical region, our model still successfully describes and classifies leaves according to their characteristics.

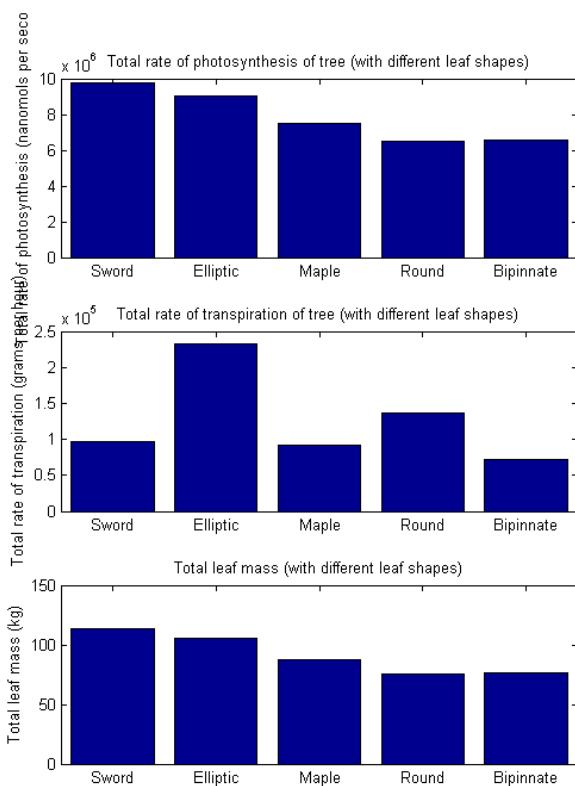


Figure 15: Tropical Leaf Shape Predictions using our LeafSA2 Model

7 Further Considerations

One of the main strengths of our models lies in the fact that it represents a physical tree as much as possible. It can be calibrated with different parameters in the future to resemble the physical tree more closely. Moreover, it can be used very easily by field researchers to roughly estimate the mass of leaves that a tree has.

Moreover, we take into account different determinants that ultimately set the mass of leaves in a tree, such as amount of sunlight irradiation received, height of the tree, and temperature. Moreover, based on the comparison between our VT-HBB model and our LIP model, the mass of leaves predicted agrees with each other. Although this does not yet necessarily mean that our model is 100% valid, it gives more credibility to both of our models as both generate roughly similar results.

We recognize several shortcomings of these models. For one, the leaf mass and surface area varies very broadly across different plants, and even within the same plant. However, although our models do not directly address this issue, we have used an average leaf mass per unit area and approximate average leaf surface area in our calculation. Hence, the impact of this shortcoming is minimized in our models. Furthermore, some of our assumptions might not hold true, such as the fact that only light intensity determines the rate of photosynthesis in a geographical region at a particular season. We recognize that there are other factors that determine the rate of photosynthesis, such as the amount of CO₂, the permeability of the epidermal cells to solar irradiation, as well as the number of stomata in the leaf. However, we decided that ultimately, solar irradiation is the only determinant of photosynthetic rate that varies widely across regions and seasons. Hence, this assumption, although not perfectly valid, is still reasonable.

8 Recommendations and Conclusions

Based on our models and computational results, we are able to predict the mass of the leaves in a tree, given the geographic location, shape, and season as parameters. Moreover, we attempted to find out the most efficient leaf shape for leaves of a tree in a given environmental condition. Our result shows that the sword-shaped leaf is generally the best as it gives the most surface area of leaves exposed to the sun, hence maximizing the potential for photosynthesis, yet minimizing the effect of evapotranspiration. However, it is worth recognizing that leaves have other functions as well besides performing photosynthesis. Hence, our model can be expanded in the future to include these functions as well so as to explain better why different trees have different leaf shapes, thus enabling us to classify and describe leaves better.

9 References

- [1] The Standard of Engineering Construction of China. "Radiation in Summer and Winter for Major Cities in China". Web. 11 Feb. 2012.
[<http://wenku.baidu.com/view/ce19960b581b6bd97f19eaf3.html>](http://wenku.baidu.com/view/ce19960b581b6bd97f19eaf3.html).
- [2] Mauseth, J. D. *Botany, an introduction to plant biology*. 4th. Sudburry: Jones & Bartlett Learning, 2009. Print.
- [3] Honda, H, and J Fisher. "Ratio of Tree Branch Lengths: The Equitable Distribution of Leaf Clusters on Branches." *Proceedings of the National Academy of Sciences of the United States of America*. 76.8 (1979): 3875-3879. Web. 11 Feb. 2012.
[<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC383938/?page=4>](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC383938/?page=4).
- [4] Honda, H, and J Fisher. "Tree Branch Angle: Maximizing Effective Leaf Area." *AAAS: Science*. 199. (1978): 888-890. Web. 11 Feb. 2012.
[<http://www.uvm.edu/~pdodds/files/papers/others/1978/honda1978a.pdf>](http://www.uvm.edu/~pdodds/files/papers/others/1978/honda1978a.pdf).
- [5] Trees Are Good. "Tree Care Information." (2012). Web. 12 Feb 2012.
[<http://www.treesaregood.com/treecare/tree_selection.aspx>](http://www.treesaregood.com/treecare/tree_selection.aspx).
- [6] Honda, H, and J Fisher. "Branch Geometry and Effective Leaf Area: A Study of Terminalia-Branching Pattern." *American Journal of Botany*. 66.6 (1979): 633-644. Web. 11 Feb. 2012.
[<http://www.jstor.org/stable/2442408?seq=1>](http://www.jstor.org/stable/2442408?seq=1).
- [7] Jin, Dongmei, Yiqiang Dai, Li Sun, and Shucun Sun. "Is Mass-based Metabolism Rate Proportional to Surface Area in Plant Leaves? A Data Re-analysis." *Journal of Investigative Plant Biology*. 50.6 (2008): 673-681. Web. 11 Feb. 2012.
- [8] Hunt, Stephen. "Effects of Irradiance on Photosynthetic CO₂ Uptake and Chlorophyll Fluorescence." *Association for Biology Laboratory Education*. (2000). Web. 12 Feb. 2012.
[<http://www.ableweb.org/volumes/vol-21/11-hunt.pdf>](http://www.ableweb.org/volumes/vol-21/11-hunt.pdf).
- [9] Niinemets, Ulo. "Research Review: Components of Leaf Dry Mass Per Area-Thickness and Density-Alter Leaf Photosynthetic Capacity in Reverse Directions in Woody Plants." *New Phytologist*. 144.1 (1999): 35-47. Print.
- [10] "Evapotranspiration." *FAO Corporate Document Repository*. Web. 12 Feb 2012.
[<http://www.fao.org/docrep/x0490e/x0490e04.htm>](http://www.fao.org/docrep/x0490e/x0490e04.htm)
- [11] Meyer, Bernard. "The Measurement of the Rate of Water-Vapor Loss From Leaves Under Standard Conditions." *American Journal of Botany*. 14.10 (1927): 582-591. Web. 12 Feb. 2012. [<http://www.jstor.org/stable/2446298?seq=8>](http://www.jstor.org/stable/2446298?seq=8).
- [12] Woolley, Joseph. "Reflectance and Transmittance of Light by Leaves." *Plant Physiology*. 47. (1971): 656-662. Web. 12 Feb 2012.
[<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC396745/pdf/plntphys00184-0062.pdf>](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC396745/pdf/plntphys00184-0062.pdf)

- [13] Chen. "Leaf-Level Measurements." University of Toronto. (2005). Web. 12 Feb 2012. <http://faculty.geog.utoronto.ca/Chen/Chen%27s%20homepage/res_leaf.htm>
- [14] Hershey, David. "Percentage of sunlight needed by plants for photosynthesis." (2006). Web. 12 Feb 2012. <<http://www.molecularstation.com/forum/botany-forum/25481-percentage-sunlight-needed-plants-photosynthesis.html>>
- [15] NASA. "First-of-its-Kind Map Depicts Global Forest Heights." (2010). Web. 12 Feb 2012. <<http://www.nasa.gov/topics/earth/features/forest-height-map.html>>

10 Appendices

10.1 MATLAB Code

1. Models for Leaf Mass

1.a. Vertical Bifurcation Model

1.a.i.

```

function n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k)
% Obtain the total number of leaves, n on the subTree through recursion.
% This subTree has a main stem and it bifurcates into 2 branches.
% These branches then bifurcate into 2 branches each. This process
% continues until the subTree reaches vertical height h.
% The main stem of the subTree has angle stemAngle from the horizontal.
% The main stem of the subTree has the length stemLength.
% The shortest branch occurs when h<=0.
% meanLeaf is the mean number of leaves in a unit length of stem.
% The number of leaves on a stem is determined using a normal distribution
% about the meanLeaf.
% For h>0, the main stem of the subTree bifurcates into two branches.
% These branches have stem length of branchLengthRatio*stemLength, and
% number of leaves on their branch determined using a normal distribution
% about meanLeaf*leafRatio.
% These two branches differ in the following way:
% Left branch has an angle of its stem angle + (normally distributed
% avgBranchAngleL)
% Right branch has an angle of its stem angle + (normally distributed
% avgBranchAngleR)
% n is the counter for the number of leaves during recursion. It should be
% initialized as 0.
% k is the recursion limit counter, that restricts the maximum number of
% times a series of branches can bifurcate.
% Assumptions:
% - All branches (and the main stem) bifurcate.
% - Height h is in meters, angles are all in radian.
% - It is possible for two or more branches to intersect.
% - When calculating the vertical height of the branches, only 2-D angles
% are considered.

if k <= 0 || h <= 0                                % If bifurcation limit reached
or branch is at maximum height

```

```

% Base case: Last and shortest possible branch -- no further bifurcation
r1= max(0,meanLeaf*(randn+1)); % Using a normal distribution
about meanLeaf, determine r1, number of leaves in a unit length of stem
nL= r1*stemLength; % Determine nL, the total
number of leaves on the stem
n= n+nL; % Update the total number of
leaves
else
  % While subTree still can grow
  % Left branch
  height= stemLength*sin(stemAngle)/2; % Calculate the vertical
height of the stem (divided by 2 as it is the average of the height of both
the left and right branch)
  h= h-max(0,height); % Update the height left for
the tree to grow
  rL= avgBranchAngleL*(randn+1); % Using a normal distribution
about avgBranchAngleL, determine rL, the angle of the left branch from its
stem
  store1= h; % Store h
  store2= stemAngle; % Store stemAngle
  store4= meanLeaf; % Store meanLeaf
  stemAngle= store2+rL; % Determine the stem angle for
the left branch
  stemLength= branchLengthRatio*stemLength; % Update stemLength of
branches
  store3= stemLength; % Store stemLength
  meanLeaf= leafRatio*meanLeaf; % Update meanLeaf for branches
  r1= max(0,meanLeaf*(randn+1)); % Using a normal distribution
about meanLeaf, determine r1, number of leaves in a unit length of stem
  nL= r1*stemLength*2; % Determine nL, the total
number of leaves on the stem
  n= n+nL; % Update counter n
  k= k-1; % Update counter k
  store5= k; % Store k
  n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngl
eR,meanLeaf,leafRatio,n,k);
  % Right branch
  h= store1; % Restore h for right branch
  stemLength= store3; % Update stemLength for right
branch
  meanLeaf= store4; % Restore meanLeaf for right
branch
  rR= avgBranchAngleL*(randn+1); % Using a normal distribution
about avgBranchAngleR, determine rR, the angle of the right branch from its
stem
  stemAngle= store2-rR; % Determine the stem angle for
the right branch
  k= store5; % Restore counter k for right
branch
  n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngl
eR,meanLeaf,leafRatio,n,k);
end

```

1.a.ii

```

function makeTree1
% Use subTree to obtain the total number of leaves on a specified tree
% This model is used to model the following types of tree:
% - Spreading Tree
% - Weeping Tree
% - Layered Tree

```

```
% - Vase Tree
% - Full-crowned Tree

% Request input from user
h= input('Height of Tree: ');
stemLength= input('Height of main trunk: ');
stemAngle= input('Angle (radians) of main trunk from the ground: ');
branchLengthRatio= input('Ratio of branch length to height of main trunk (<0): ');
avgBranchAngleL= input('Average anticlockwise angle (radians) of branch from main trunk: ');
avgBranchAngleR= input('Average clockwise angle (radians) of branch from main trunk: ');
meanLeaf= input('Average number of leaves per unit length of main trunk: ');
leafRatio= input('Ratio of the number of leaves on a newly bifurcated branch to its parent branch (>0): ');

% Run model for 10 trials (trees)
totaln= 0; % Initialize counter to sum all the leaves in all trials
for j=1:10
    % Initialize n, the initial number of leaves, and k, the maximum number of bifurcations of a series of branches
    n= 0;
    k= 10;
    % Call function subTree to determine number of leaves on specified tree
    n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
    n= n-meanLeaf; % Subtract the number of leaves on the main trunk, since physically, leaves do not grow on the main trunk
    totaln= totaln+n;
end

% Determine average number of leaves on the specified tree
avgn= totaln/10;

% Print number of leaves on tree
fprintf('This tree has %d leaves.\n',avgn)
```

1.b. Vertical Trunk – Horizontal Bifurcation Model

1.b.i.

```
function makeTree2
% Use subTree to obtain the total number of leaves on a specified tree.
% In this model, subTree represents branches instead of a tree.
% This model is used to model the following types of tree:
%   - Columnar Tree
% Assumptions:
%   - At each level that branching occurs, there is an equal probability that
%     there are 3 to 6 main branches.
%   - There is a deterministic ratio (<0) of the gaps in the branching levels
from
%   the second branching level all the way to the maximum branching level
%   - Since it is a columnar tree, the lengths of the main branches remain
%     constant throughout all the branching levels

% Request input from user
heightTree= input('Height of tree: ');
heightFirstBranchLevel= input('Height of the first branching level from the ground:');
```

```

heightSecondBranchLevel= input('Height of the second branching level from the
first branching level: ');
h= input('Length of main branch: ');
stemLength= input('Length of main stem of the main branch: ');
stemAngle= input('Angle (radians) of main branch from the trunk: ');
branchLengthRatio= input('Ratio of length of newly bifurcated branch to length
of parent branch (<0): ');
avgBranchAngleL= input('Average anticlockwise angle (radians) of newly
bifurcated branch from main branch: ');
avgBranchAngleR= input('Average clockwise angle (radians) of newly bifurcated
branch from main branch: ');
meanLeaf= input('Average number of leaves per unit length of main branch: ');
leafRatio= input('Ratio of the number of leaves on a newly bifurcated branch
to its parent branch (>0): ');

% Calculate number of levels that branching occurs. Determination of
% equation is detailed in report.
levels= round((heightTree-heightFirstBranchLevel)/(heightSecondBranchLevel));

% Run model for 3 trials (trees)
totaln= 0; % Initialize counter to sum all the leaves in
all trials
for j=1:3
  % For each level of branching
  for l=1:levels
    r= round(3*rand+3); % Equal probability that there are 3 to 6 main
branches
    % For each main branch
    for b=1:r
      % Initialize n, the initial number of leaves, and k, the maximum
number of
      % bifurcations of a series of branches
      n= 0;
      k= 10;
      % Call function subTree to determine number of leaves on main
branch, b
      n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngl
eR,meanLeaf,leafRatio,n,k);
      n= n-meanLeaf; % Subtract the number of leaves on the main
branch, since physically, leaves do not grow on the main branch
      totaln= totaln+n;
    end
  end
end

% Determine average number of leaves on the specified tree
avgn= totaln/3;

% Print number of leaves on tree
fprintf('This tree has %d leaves.\n',avgn)

```

1.b.ii.

```

function makeTree3
% Use subTree to obtain the total number of leaves on a specified tree.
% In this model, subTree represents branches instead of a tree.
% This model is used to model the following types of tree:
% - Pyramidal Tree
% Assumptions:
% - At each level that branching occurs, there is an equal probability that
% there are 3 to 6 main branches.
% - There is a deterministic ratio (<0) of the gaps in the branching levels

```

```

from
%   the second branching level all the way to the maximum branching level.
%   - Since it is a pyramidal tree, the lengths of the main branches vary by
%   a deterministic ratio (<0) all the way from the first branching level to
the
%   maximum branching level.

% Request input from user
heightTree= input('Height of tree: ');
heightFirstBranchLevel= input('Height of the first branching level from the
ground: ');
heightSecondBranchLevel= input('Height of the second branching level from the
first branching level: ');
h= input('Length of main branch of the first branching level: ');
stemLength= input('Length of main stem of the main branch of the first
branching level: ');
stemAngle= input('Angle (radians) of main branch from the trunk: ');
branchLengthRatio= input('Ratio of length of newly bifurcated branch to length
of parent branch (<0): ');
avgBranchAngleL= input('Average anticlockwise angle (radians) of newly
bifurcated branch from main branch: ');
avgBranchAngleR= input('Average clockwise angle (radians) of newly bifurcated
branch from main branch: ');
meanLeaf= input('Average number of leaves per unit length of main branch: ');
leafRatio= input('Ratio of the number of leaves on a newly bifurcated branch
to its parent branch (>0): ');

% Store stemLength for the different trials
store= stemLength;

% Calculate number of levels that branching occurs. Determination of
% equation is detailed in report.
% m is the difference between the height of the first and second branching
% levels divided by the height of the first branching level from the ground.
% Its derivation is detailed in the report.
m= heightSecondBranchLevel/heightFirstBranchLevel;
levels= round((heightTree-heightFirstBranchLevel)/(heightSecondBranchLevel));

% Run model for 3 trials (trees)
totaln= 0;                                     % Initialize counter to sum
all the leaves in all trials
for j=1:3
    stemLength= store;                         % Restore original value of
stemLength
    % For each level of branching
    for l=1:levels
        r= round(3*rand+3);                   % Equal probability that there
are 3 to 6 main branches
        % Below is a mathematical method to calculate the subsequent
        % lengths of the branches at each higher branching level. The
        % detailed derivation is done in the report.
        sum= 0;                                % Initialize sum
        for i=1:(l-1)
            s= (heightSecondBranchLevel*(1-levels*m+m))/3;
            sum= sum+s;                        % Update sum
        end
        stemLength= store-sum;                 % Update stemLength
        % For each main branch
        for b=1:r
            % Initialize n, the initial number of leaves, and k, the maximum
number of
            % bifurcations of a series of branches
            n= 0;
            k= 10;

```

```

    % Call function subTree to determine number of leaves on main
branch, b
n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
n= n-meanLeaf;           % Subtract the number of leaves on the
main branch, since physically, leaves do not grow on the main branch
totaln= totaln+n;
end
end
end

% Determine average number of leaves on the specified tree
avgn= totaln/3;

% Print number of leaves on tree
fprintf('This tree has %d leaves.\n',avgn)

```

1.c. Palm Tree Model

1.c.i.

```

function n= subTree2(perimeterOfTrunk,arcLengthLeafBase,thicknessLeafBase)
% This model looks at how many leafs branch out of a circular fountain tree
% top. We focus on the top of the tree trunk, where the branching occurs
% about a hemisphere. We imagine that the leafs branch along rings,
% starting from the outer-most ring (perimeter) of the hemisphere, all
% the way until the center-most ring.
% n is the number of leaves of the specified tree.
% perimeterOfTrunk is the perimeter of the tree trunk.
% arcLengthLeafBase is the average arc length of the base of the leaf.
% thicknessLeafBase is the average thickness of the base of the leaf.
% Assumptions:
% - The "rings" are maximally filled with leafs.
% - All leaves only branch out of the "hemisphere-like" top of
% the tree, and no other part of the tree.
% - All leaves along the same ring have the same arc length.
% - All leaves have the same thickness.

% Initialize n
n= 0;

% Calculate the radius of the trunk
r= perimeterOfTrunk/(2*pi);

% Calculate the maximum number of rings of branching
thickness= max(0.03,thicknessLeafBase*(randn+1));           % Normal distribution
(3cm minimum)
rings= r/thickness;

for k=1:rings
    % Use a normal distribution to determine the arc length of the base of
    % the leaf in the k-th ring (15cm minimum)
    arcLength= max(0.15,arcLengthLeafBase*(randn+1));
    % Calculate the perimeter of the k-th ring; the ring falls on the midpoint
    % of the thickness of the leaf base so that a leaf base can fully fit
    % onto the hemisphere
    perimeter= 2*pi*((r-k*thickness+thickness)-thickness/2);
    % Calculate the number of leafs that can fit around the k-th ring
    nn= perimeter/arcLength;
    n= n+nn;
end

```

1.c.ii.

```

function makeTree4
% Call subTree2 to obtain the total number of leaves on a specified tree.
% This model is used to model the following types of tree:
% - Fountain Tree
% This model looks at how many leafs branch out of a circular fountain tree
% top. We focus on the top of the tree trunk, where the branching occurs
% about a hemisphere. We imagine that the leafs branch along rings,
% starting from the outer-most ring (perimeter) of the hemisphere, all
% the way until the center-most ring.
% Assumptions:
% - The "rings" are maximally filled with leafs.
% - All leaves only branch out of the "hemisphere-like" top of
% the tree, and no other part of the tree.
% - All leaves along the same ring have the same arc length.
% - All leaves have the same thickness.

% Request input from user
perimeterOfTrunk= input('Perimeter of trunk: ');
arcLengthLeafBase= input('Arc length of the base of the leaf: ');
thicknessLeafBase= input('Thickness of the base of the leaf: ');

% Run model for 10 trials (trees)
totaln= 0; % Initialize counter to sum all the leaves in
all trials
for k=1:10
    n= subTree2(perimeterOfTrunk,arcLengthLeafBase,thicknessLeafBase);
    totaln= totaln+n; % Update counter
end

% Determine average number of leaves on the specified tree
avgn= totaln/10;

% Print number of leaves on tree
fprintf('This tree has %d leaves.\n',avgn)

```

1.d. Integration (Leaf Mass)1.d.i.

```

% Script
% Returns the number of leaves of the specified tree

% Create database of the type of trees
T= {'Full-crowned'; 'Vase'; 'Spreading'; 'Layered'; 'Weeping'; 'Pyramidal';
'Columnar'; 'Fountain'};
disp('List of tree types:')
disp(T)

% Request user input
treetype= input('Type of tree: ','s');

if strcmp(treetype,'Full-crowned') || strcmp(treetype,'Vase') ||
strcmp(treetype,'Spreading') || strcmp(treetype,'Weeping')
    makeTree1
elseif strcmp(treetype,'Columnar')
    makeTree2
elseif strcmp(treetype,'Pyramidal')
    makeTree3
elseif strcmp(treetype,'Fountain')

```

```
makeTree4
end
```

1.d.ii.

```
function plotSLVTree(region,season)
% Modeling the relationship between the leaf mass of a Spreading, Layered,
% Vase tree and the height of the tree
% Plot leaf mass Vs height of tree

% Initialize 30 regularly spaced heights of tree between 0 and 30
height= linspace(0,30,30);

% Initialize lm, a vector bin to store 30 values of the leaf mass of the tree
% corresponding to the respective height
lm= zeros(30);

% Determine mass of 1 leaf
if strcmp(region,'Alpine')
    if strcmp(season,'Spring')
        mass=1.67/1000;
    elseif strcmp(season,'Autumn')
        mass=1.14/1000;
    end
elseif strcmp(region,'Temperate')
    if strcmp(season,'Spring')
        mass=2.30/1000;
    elseif strcmp(season,'Autumn')
        mass=1.55/1000;
    end
elseif strcmp(region,'Tropics')
    mass= 4.72/1000;
end

for j=1:30
    h= height(j);
    stemAngle= pi/2;
    stemLength= 1/5*j;
    branchLengthRatio= 0.6;
    avgBranchAngleL= 41.4/360*2*pi;
    avgBranchAngleR= 24.6/360*2*pi;
    meanLeaf= 60;
    leafRatio= 1.1;
    n=0;
    k=15;
    n=
    subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
    lm(j)= mass*n;
end

% Create plot
figure
plot(height,lm,'r')
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Spreading/Layered/Vase Tree VS. Height of Tree')
```

1.d.iii.

```
function plotWeepingTree(region,season)
% Modeling the relationship between the leaf mass of a Weeping tree and
% the height of the tree
```

```
% Plot leaf mass Vs height of tree

% Initialize 30 regularly spaced heights of tree between 0 and 20
height= linspace(0,20,30);

% Initialize lm, a vector bin to store 30 values of the leaf mass of the tree
% corresponding to the respective height
lm= zeros(30);

% Determine mass of 1 leaf
if strcmp(region,'Alpine')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=1.67/1000;
    elseif strcmp(season,'Autumn')
        mass=1.14/1000;
    end
elseif strcmp(region,'Temperate')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=2.30/1000;
    elseif strcmp(season,'Autumn')
        mass=1.55/1000;
    end
elseif strcmp(region,'Tropics')
    mass= 4.72/1000;
end

for j=1:30
    h= height(j);
    stemAngle= pi/2;
    stemLength= 1/3*j;
    branchLengthRatio= 0.55;
    avgBranchAngleL= 41.4/360*2*pi;
    avgBranchAngleR= 24.6/360*2*pi;
    meanLeaf= 150;
    leafRatio= 1.1;
    n=0;
    k=15;
    n=
    subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
    lm(j)= mass*n;
end

% Create plot
figure
plot(height,lm,'r')
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Weeping Tree VS. Height of Tree')
```

1.d.iv.

```
function plotFullCrownedTree(region,season)
% Modeling the relationship between the leaf mass of a Full-crowned
% tree and the height of the tree
% Plot leaf mass Vs height of tree

% Initialize 30 regularly spaced heights of tree between 0 and 30
height= linspace(0,30,30);

% Initialize lm, a vector bin to store 30 values of the leaf mass of the tree
% corresponding to the respective height
lm= zeros(30);
```

```
% Determine mass of 1 leaf
if strcmp(region,'Alpine')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=1.67/1000;
    elseif strcmp(season,'Autumn')
        mass=1.14/1000;
    end
elseif strcmp(region,'Temperate')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=2.30/1000;
    elseif strcmp(season,'Autumn')
        mass=1.55/1000;
    end
elseif strcmp(region,'Tropics')
    mass= 4.72/1000;
end

for j=1:30
    h= height(j);
    stemAngle= pi/2;
    stemLength= 1/4*j;
    branchLengthRatio= 0.58;
    avgBranchAngleL= 41.4/360*2*pi;
    avgBranchAngleR= 24.6/360*2*pi;
    meanLeaf= 75;
    leafRatio= 1.1;
    n=0;
    k=15;
    n=
    subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
    lm(j)= mass*n;
end

% Create plot
figure
plot(height,lm,'r')
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of Full-crowned Tree VS. Height of Tree')
```

1.d.v.

```
function plotColumnarTree(region,season)
% Modeling the relationship between the leaf mass of a columnar tree and
% the height of the tree
% Plot leaf mass Vs height of tree
% Assumptions
% - Mass of leaf (in kg) only depends on its region and season; it is the
% same for all types of trees

% Initialize 10 regularly spaced heights of tree between 0 and 30
height= linspace(1,30,10);

% Initialize lm, a vector bin to store 10 values of the leaf mass of the tree
% corresponding to the respective height
lm= zeros(10);

% Determine mass of 1 leaf
if strcmp(region,'Alpine')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=1.67/1000;
```

```

elseif strcmp(season,'Autumn')
    mass=1.14/1000;
end
elseif strcmp(region,'Temperate')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=2.30/1000;
    elseif strcmp(season,'Autumn')
        mass=1.55/1000;
    end
elseif strcmp(region,'Tropics')
    mass= 4.72/1000;
end

for j=1:10
    % Use subTree to obtain the total number of leaves on a specified tree.
    % In this model, subTree represents branches instead of a tree.
    % This model is used to model the following types of tree:
    % - Columnar Tree
    % Assumptions:
    % - At each level that branching occurs, there is an equal probability
that
    % there are 3 to 6 main branches.
    % - There is a deterministic ratio (<0) of the gaps in the branching
levels from
    % the second branching level all the way to the maximum branching level
    % - Since it is a columnar tree, the lengths of the main branches remain
    % constant throughout all the branching levels

    % Initialize variables
heightTree= height(j);                                % Height of tree
heightFirstBranchLevel= 1/9*heightTree;                % Height of the first
branching level from the ground
heightSecondBranchLevel= 0.4*heightFirstBranchLevel;% Height of the second
branching level from the first branching level
h= 1/5*heightTree;                                    % Length of main branch
stemLength= 1/5*heightTree/12;                         % Length of main stem of the
main branch of the first branching level
stemAngle= pi/2;                                       % Angle (radians) of main
branch from the trunk
branchLengthRatio= 0.5;                                % Ratio of length of newly
bifurcated branch to length of parent branch (<0)
avgBranchAngleL= 41.4/360*2*pi;                      % Average anticlockwise angle
(radians) of newly bifurcated branch from main branch
avgBranchAngleR= 24.6/360*2*pi;                      % Average clockwise angle
(radians) of newly bifurcated branch from main branch
meanLeaf= 60;                                         % Average number of leaves per
unit length of main branch
leafRatio= 1.1;                                        % Ratio of the number of
leaves on a newly bifurcated branch to its parent branch (>0)

    % Calculate number of levels that branching occurs. Determination of
    % equation is detailed in report.
    levels= round((heightTree-
heightFirstBranchLevel)/(heightSecondBranchLevel));

    % Initialize totaln to keep count of total number of leaves on tree
totaln= 0;

    % For each level of branching
    for l=1:levels
        r= round(3*rand+3);                            % Equal probability that there
are 3 to 6 main branches
        % For each main branch
        for b=1:r

```

```

    % Initialize n, the initial number of leaves, and k, the maximum
number of bifurcations of a series of branches
n= 0;
k= 10;
% Call function subTree to determine number of leaves on main
branch, b
n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
n= n-meanLeaf;           % Subtract the number of leaves on the
main branch, since physically, leaves do not grow on the main branch
totaln= totaln+n;        % Update total number of leaves on j-th
tree
end
lm(j)= mass*totaln;      % Store in counter lm(j) the leaf mass of
tree
end

% Create plot
figure
plot(height,lm,'r')
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Columnar Tree VS. Height of Tree')

```

1.d.vi.

```

function plotPyramidalTree(region,season)
% Modeling the relationship between the leaf mass of a pyramidal tree and
% the height of the tree
% Plot leaf mass Vs height of tree
% Assumptions
% - Mass of leaf (in kg) only depends on its region and season; it is the
% same for all types of trees

% Initialize 10 regularly spaced heights of tree between 0 and 30
height= linspace(1,30,10);

% Initialize lm, a vector bin to store 10 values of the leaf mass of the tree
% corresponding to the respective height
lm= zeros(10);

% Determine mass of 1 leaf
if strcmp(region,'Alpine')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=1.67/1000;
    elseif strcmp(season,'Autumn')
        mass=1.14/1000;
    end
elseif strcmp(region,'Temperate')
    if strcmp(season,'Spring') || strcmp(season,'Summer')
        mass=2.30/1000;
    elseif strcmp(season,'Autumn')
        mass=1.55/1000;
    end
elseif strcmp(region,'Tropics')
    mass= 4.72/1000;
end

for j=1:10
    % Use subTree to obtain the total number of leaves on a specified tree.

```

```

% In this model, subTree represents branches instead of a tree.
% This model is used to model the following types of tree:
%   - Pyramidal Tree
% Assumptions:
%   - At each level that branching occurs, there is an equal probability
that
%   there are 3 to 6 main branches.
%   - There is a deterministic ratio (<0) of the gaps in the branching
levels from
%   the second branching level all the way to the maximum branching level.
%   - Since it is a pyramidal tree, the lengths of the main branches vary
by
%   a deterministic ratio (<0) all the way from the first branching level
to the
%   maximum branching level.

% Initialize variables
heightTree= height(j);
heightFirstBranchLevel= 1/7*heightTree;
branching level from the ground
heightSecondBranchLevel= 0.6*heightTree;
branching level from the first branching level
h= 1/4*heightTree;
first branching level
stemLength= 1/4*heightTree/10;
main branch of the first branching level
stemAngle= pi/2;
branch from the trunk
branchLengthRatio= 0.6;
bifurcated branch to length of parent branch (<0)
avgBranchAngleL= 41.4/360*2*pi;
(radians) of newly bifurcated branch from main branch
avgBranchAngleR= 24.6/360*2*pi;
(radians) of newly bifurcated branch from main branch
meanLeaf= 200;
unit length of main branch
leafRatio= 1.2;
leaves on a newly bifurcated branch to its parent branch (>0)

% Calculate number of levels that branching occurs. Determination of
% equation is detailed in report.
% m is the difference between the height of the first and second branching
% levels divided by the height of the first branching level from the
ground.
% Its derivation is detailed in the report.
m= heightSecondBranchLevel/heightFirstBranchLevel;
levels= round((heightTree-
heightFirstBranchLevel)/(heightSecondBranchLevel));

% Initialize totaln to keep count of total number of leaves on tree
totaln= 0;

% For each level of branching
for l=1:levels
  r= round(3*rand+3);                                % Equal probability that there
are 3 to 6 main branches
  % Below is a mathematical method to calculate the subsequent
  % lengths of the branches at each higher branching level. The
  % detailed derivation is done in the report.
  sum= 0;
  for i=1:(l-1)
    s= (heightSecondBranchLevel*(1-l*m+m))/3;
    sum= sum+s;                                     % Update sum
  end

```

```

stemLength= stemLength-sum;                                % Update stemLength
% For each main branch
for b=1:r
    % Initialize n, the initial number of leaves, and k, the maximum
number of
    % bifurcations of a series of branches
    n= 0;
    k= 10;
    % Call function subTree to determine number of leaves on main
branch, b
    n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
    n= n-meanLeaf;           % Subtract the number of leaves on the
main branch, since physically, leaves do not grow on the main branch
    totaln= totaln+n;       % Update total number of leaves on j-th
tree
end
end
lm(j)= mass*totaln;                                     % Store in counter lm(j) the leaf mass of
tree
end

% Create plot
figure
plot(height,lm,'r')
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Pyramidal Tree VS. Height of Tree')

```

1.d.vii.

```

function plotFountainTree
% Modeling the relationship between the leaf mass of a fountain tree and
% the circumference of trunk
% Plot leaf mass Vs perimeter of trunk

% Initialize 200 regularly spaced circumferences of trunk between 0 and 3
c= linspace(0,3,200);

% Initialize lm, a vector bin to store 200 values of the leaf mass of the tree
% corresponding to the respective circumference c
lm= zeros(200);

% Initialize r and t
r= 1/6;          % Ratio of the arc length of the leaf base to the
circumference of trunk
t= 1/10;         % Ratio of the thickness of the leaf to the arc length of
the leaf base

for k=1:200
    perimeterOfTrunk= c(k);
    arcLengthLeafBase= r*c(k);
    thicknessLeafBase= t*arcLengthLeafBase;
    n= subTree2(perimeterOfTrunk,arcLengthLeafBase,thicknessLeafBase);
    m= 15*arcLengthLeafBase;           % Mass of 1 leaf, justified in report
    lm(k)= m*n;
end

% Create plot
figure
plot(c,lm,'r')
xlabel('Circumference of Trunk')

```

```

ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Fountain Tree VS. Circumference of Trunk')

```

1.d.viii.

```

function plotAllSLVTree
% Modeling the relationship between the leaf mass of a Spreading, Layered,
% Vase tree and the height of the tree
% Plot leaf mass Vs height of tree

% Initialize 30 regularly spaced heights of tree between 0 and 30
height= linspace(1,30,30);

% Initialize lm, a vector bin to store 30 values of the leaf mass of the tree
% corresponding to the respective height for all combinations of region and
% season
lm1= zeros(30);      % Alpine & Spring/Summer
lm2= zeros(30);      % Alpine & Autumn
lm3= zeros(30);      % Temperate & Spring/Summer
lm4= zeros(30);      % Temperate & Autumn
lm5= zeros(30);      % Tropics

for j=1:30
    h= height(j);
    stemAngle= pi/2;
    stemLength= 1/5*j;
    branchLengthRatio= 0.6;
    avgBranchAngleL= 41.4/360*2*pi;
    avgBranchAngleR= 24.6/360*2*pi;
    meanLeaf= 60;
    leafRatio= 1.1;
    n=0;
    k=15;
    n=
    subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
    lm1(j)= 1.67/1000*n;           % Store in counter lm1(j) the leaf mass
    of tree
    lm2(j)= 1.14/1000*n;           % Store in counter lm2(j) the leaf mass
    of tree
    lm3(j)= 2.3/1000*n;           % Store in counter lm3(j) the leaf mass
    of tree
    lm4(j)= 1.55/1000*n;           % Store in counter lm4(j) the leaf mass
    of tree
    lm5(j)= 4.72/1000*n;           % Store in counter lm5(j) the leaf mass
    of tree
end

% Create plot
figure
plot(height,lm1,'r',height,lm2,'b',height,lm3,'g',height,lm4,'c',height,lm5,'k')
leg= legend('Alpine Spring/Summer','Alpine Autumn','Temperate
Spring/Summer',...
    'Temperate Autumn','Tropics');
set(leg,'Location','NorthWest');
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Spreading/Layered/Vase Tree VS. Height of Tree')

```

1.d.ix.

```

function plotAllWeepingTree
% Modeling the relationship between the leaf mass of a Weeping tree and
% the height of the tree
% Plot leaf mass Vs height of tree

% Initialize 30 regularly spaced heights of tree between 0 and 20
height= linspace(1,20,30);

% Initialize lm, a vector bin to store 30 values of the leaf mass of the tree
% corresponding to the respective height for all combinations of region and
% season
lm1= zeros(30);      % Alpine & Spring/Summer
lm2= zeros(30);      % Alpine & Autumn
lm3= zeros(30);      % Temperate & Spring/Summer
lm4= zeros(30);      % Temperate & Autumn
lm5= zeros(30);      % Tropics

for j=1:30
    h= height(j);
    stemAngle= pi/2;
    stemLength= 1/3*j;
    branchLengthRatio= 0.55;
    avgBranchAngleL= 41.4/360*2*pi;
    avgBranchAngleR= 24.6/360*2*pi;
    meanLeaf= 150;
    leafRatio= 1.1;
    n=0;
    k=15;
    n=
    subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
        lm1(j)= 1.67/1000*n;           % Store in counter lm1(j) the leaf mass
    of tree
        lm2(j)= 1.14/1000*n;           % Store in counter lm2(j) the leaf mass
    of tree
        lm3(j)= 2.3/1000*n;           % Store in counter lm3(j) the leaf mass
    of tree
        lm4(j)= 1.55/1000*n;           % Store in counter lm4(j) the leaf mass
    of tree
        lm5(j)= 4.72/1000*n;           % Store in counter lm5(j) the leaf mass
    of tree
end

% Create plot
figure
plot(height,lm1,'r',height,lm2,'b',height,lm3,'g',height,lm4,'c',height,lm5,'k')
leg= legend('Alpine Spring/Summer','Alpine Autumn','Temperate
Spring/Summer',...
    'Temperate Autumn','Tropics');
set(leg,'Location','NorthWest');
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Weeping Tree VS. Height of Tree')

```

1.d.x.

```

function plotAllFullCrownedTree
% Modeling the relationship between the leaf mass of a Full-crowned
% tree and the height of the tree
% Plot leaf mass Vs height of tree

% Initialize 30 regularly spaced heights of tree between 0 and 30

```

```

height= linspace(1,30,30);

% Initialize lm, a vector bin to store 30 values of the leaf mass of the tree
% corresponding to the respective height for all combinations of region and
% season
lm1= zeros(30);          % Alpine & Spring/Summer
lm2= zeros(30);          % Alpine & Autumn
lm3= zeros(30);          % Temperate & Spring/Summer
lm4= zeros(30);          % Temperate & Autumn
lm5= zeros(30);          % Tropics

for j=1:30
    h= height(j);
    stemAngle= pi/2;
    stemLength= 1/4*j;
    branchLengthRatio= 0.58;
    avgBranchAngleL= 41.4/360*2*pi;
    avgBranchAngleR= 24.6/360*2*pi;
    meanLeaf= 75;
    leafRatio= 1.1;
    n=0;
    k=15;
    n=
    subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngl
eR,meanLeaf,leafRatio,n,k);
    lm1(j)= 1.67/1000*n;           % Store in counter lm1(j) the leaf mass
    of tree
    lm2(j)= 1.14/1000*n;           % Store in counter lm2(j) the leaf mass
    of tree
    lm3(j)= 2.3/1000*n;           % Store in counter lm3(j) the leaf mass
    of tree
    lm4(j)= 1.55/1000*n;           % Store in counter lm4(j) the leaf mass
    of tree
    lm5(j)= 4.72/1000*n;           % Store in counter lm5(j) the leaf mass
    of tree
end

% Create plot
figure
plot(height,lm1,'r',height,lm2,'b',height,lm3,'g',height,lm4,'c',height,lm5,'k
')
leg= legend('Alpine Spring/Summer','Alpine Autumn','Temperate
Spring/Summer',...
    'Temperate Autumn','Tropics');
set(leg,'Location','NorthWest');
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of Full-crowned Tree VS. Height of Tree')

```

1.d.xi.

```

function plotAllColumnarTree
% Modeling the relationship between the leaf mass of a columnar tree and
% the height of the tree for all combinations of region and season
% Plot leaf mass Vs height of tree
% Assumptions
% - Mass of leaf (in kg) only depends on its region and season; it is the
% same for all types of trees
% - Values for mass is explained in report

% Initialize 10 regularly spaced heights of tree between 0 and 30
height= linspace(1,30,10);

```

```

% Initialize lm, a vector bin to store 10 values of the leaf mass of the tree
% corresponding to the respective height for all combinations of region and
% season
lm1= zeros(10);      % Alpine & Spring/Summer
lm2= zeros(10);      % Alpine & Autumn
lm3= zeros(10);      % Temperate & Spring/Summer
lm4= zeros(10);      % Temperate & Autumn
lm5= zeros(10);      % Tropics

for j=1:10
  % Use subTree to obtain the total number of leaves on a specified tree.
  % In this model, subTree represents branches instead of a tree.
  % This model is used to model the following types of tree:
  % - Columnar Tree
  % Assumptions:
  %   - At each level that branching occurs, there is an equal probability
that
  %   there are 3 to 6 main branches.
  %   - There is a deterministic ratio (<0) of the gaps in the branching
levels from
  %   the second branching level all the way to the maximum branching level
  %   - Since it is a columnar tree, the lengths of the main branches remain
  %   constant throughout all the branching levels

  % Initialize variables
  heightTree= height(j);                      % Height of tree
  heightFirstBranchLevel= 1/9*heightTree;        % Height of the first
branching level from the ground
  heightSecondBranchLevel= 0.4*heightFirstBranchLevel;% Height of the second
branching level from the first branching level
  h= 1/5*heightTree;                          % Length of main branch
  stemLength= 1/5*heightTree/12;                % Length of main stem of the
main branch of the first branching level
  stemAngle= pi/2;                            % Angle (radians) of main
branch from the trunk
  branchLengthRatio= 0.5;                      % Ratio of length of newly
bifurcated branch to length of parent branch (<0)
  avgBranchAngleL= 41.4/360*2*pi;             % Average anticlockwise angle
(radians) of newly bifurcated branch from main branch
  avgBranchAngleR= 24.6/360*2*pi;              % Average clockwise angle
(radians) of newly bifurcated branch from main branch
  meanLeaf= 60;                               % Average number of leaves per
unit length of main branch
  leafRatio= 1.1;                            % Ratio of the number of
leaves on a newly bifurcated branch to its parent branch (>0)

  % Calculate number of levels that branching occurs. Determination of
  % equation is detailed in report.
  levels= round((heightTree-
heightFirstBranchLevel)/(heightSecondBranchLevel));

  % Initialize totaln to keep count of total number of leaves on tree
  totaln= 0;

  % For each level of branching
  for l=1:levels
    r= round(3*rand+3);                      % Equal probability that there
are 3 to 6 main branches
    % For each main branch
    for b=1:r
      % Initialize n, the initial number of leaves, and k, the maximum
number of
      % bifurcations of a series of branches
      n= 0;

```

```

k= 10;
% Call function subTree to determine number of leaves on main
branch, b
n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
n= n-meanLeaf; % Subtract the number of leaves on the
main branch, since physically, leaves do not grow on the main branch
totaln= totaln+n; % Update total number of leaves on j-th
tree
end
lm1(j)= 1.67/1000*totaln; % Store in counter lm1(j) the leaf
mass of tree
lm2(j)= 1.14/1000*totaln; % Store in counter lm2(j) the leaf
mass of tree
lm3(j)= 2.3/1000*totaln; % Store in counter lm3(j) the leaf
mass of tree
lm4(j)= 1.55/1000*totaln; % Store in counter lm4(j) the leaf
mass of tree
lm5(j)= 4.72/1000*totaln; % Store in counter lm5(j) the leaf
mass of tree
end

% Create plot
figure
plot(height,lm1,'r',height,lm2,'b',height,lm3,'g',height,lm4,'c',height,lm5,'k')
leg= legend('Alpine Spring/Summer','Alpine Autumn','Temperate
Spring/Summer',...
'Temperate Autumn','Tropics');
set(leg,'Location','NorthWest');
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Columnar Tree VS. Height of Tree')

```

1.d.xii.

```

function plotAllPyramidalTree
% Modeling the relationship between the leaf mass of a pyramidal tree and
% the height of the tree
% Plot leaf mass Vs height of tree
% Assumptions
% - Mass of leaf (in kg) only depends on its region and season; it is the
% same for all types of trees

% Initialize 10 regularly spaced heights of tree between 0 and 30
height= linspace(1,30,10);

% Initialize lm, a vector bin to store 10 values of the leaf mass of the tree
% corresponding to the respective height for all combinations of region and
% season
lm1= zeros(10); % Alpine & Spring/Summer
lm2= zeros(10); % Alpine & Autumn
lm3= zeros(10); % Temperate & Spring/Summer
lm4= zeros(10); % Temperate & Autumn
lm5= zeros(10); % Tropics

for j=1:10
% Use subTree to obtain the total number of leaves on a specified tree.
% In this model, subTree represents branches instead of a tree.
% This model is used to model the following types of tree:
% - Pyramidal Tree

```

```

% Assumptions:
% - At each level that branching occurs, there is an equal probability
that
%   there are 3 to 6 main branches.
% - There is a deterministic ratio (<0) of the gaps in the branching
levels from
%   the second branching level all the way to the maximum branching level.
% - Since it is a pyramidal tree, the lengths of the main branches vary
by
%   a deterministic ratio (<0) all the way from the first branching level
to the
%   maximum branching level.

% Initialize variables
heightTree= height(j);                                % Height of tree
heightFirstBranchLevel= 1/7*heightTree;                % Height of the first
branching level from the ground
heightSecondBranchLevel= 0.6*heightTree;              % Height of the second
branching level from the first branching level
h= 1/4*heightTree;                                    % Length of main branch of the
first branching level
stemLength= 1/4*heightTree/10;                         % Length of main stem of the
main branch of the first branching level
stemAngle= pi/2;                                      % Angle (radians) of main
branch from the trunk
branchLengthRatio= 0.6;                                % Ratio of length of newly
bifurcated branch to length of parent branch (<0)
avgBranchAngleL= 41.4/360*2*pi;                      % Average anticlockwise angle
(radians) of newly bifurcated branch from main branch
avgBranchAngleR= 24.6/360*2*pi;                      % Average clockwise angle
(radians) of newly bifurcated branch from main branch
meanLeaf= 200;                                         % Average number of leaves per
unit length of main branch
leafRatio= 1.2;                                         % Ratio of the number of
leaves on a newly bifurcated branch to its parent branch (>0)

% Calculate number of levels that branching occurs. Determination of
% equation is detailed in report.
% m is the difference between the height of the first and second branching
% levels divided by the height of the first branching level from the
ground.
% Its derivation is detailed in the report.
m= heightSecondBranchLevel/heightFirstBranchLevel;
levels= round((heightTree-
heightFirstBranchLevel)/(heightSecondBranchLevel));

% Initialize totaln to keep count of total number of leaves on tree
totaln= 0;

% For each level of branching
for l=1:levels
  r= round(3*rand+3);                                  % Equal probability that there
are 3 to 6 main branches
  % Below is a mathematical method to calculate the subsequent
  % lengths of the branches at each higher branching level. The
  % detailed derivation is done in the report.
  sum= 0;                                              % Initialize sum
  for i=1:(l-1)
    s= (heightSecondBranchLevel*(1-l*m+m))/3;
    sum= sum+s;                                         % Update sum
  end
  stemLength= stemLength-sum;                          % Update stemLength
  % For each main branch
  for b=1:r

```

```

        % Initialize n, the initial number of leaves, and k, the maximum
number of                                % bifurcations of a series of branches
n= 0;                                     k= 10;
                                         % Call function subTree to determine number of leaves on main
branch, b                                 n=
subTree(h,stemAngle,stemLength,branchLengthRatio,avgBranchAngleL,avgBranchAngleR,meanLeaf,leafRatio,n,k);
                                         n= n-meanLeaf;           % Subtract the number of leaves on the
main branch, since physically, leaves do not grow on the main branch
                                         totaln= totaln+n;       % Update total number of leaves on j-th
tree
end
lm1(j)= 1.67/1000*totaln;                  % Store in counter lm1(j) the leaf
mass of tree
lm2(j)= 1.14/1000*totaln;                  % Store in counter lm2(j) the leaf
mass of tree
lm3(j)= 2.3/1000*totaln;                  % Store in counter lm3(j) the leaf
mass of tree
lm4(j)= 1.55/1000*totaln;                  % Store in counter lm4(j) the leaf
mass of tree
lm5(j)= 4.72/1000*totaln;                  % Store in counter lm5(j) the leaf
mass of tree
end

% Create plot
figure
plot(height,lm1,'r',height,lm2,'b',height,lm3,'g',height,lm4,'c',height,lm5,'k')
leg= legend('Alpine Spring/Summer','Alpine Autumn','Temperate
Spring/Summer',...
    'Temperate Autumn','Tropics');
set(leg,'Location','NorthWest');
xlabel('Height of Tree')
ylabel('Leaf mass of Tree')
title('Plot of Leaf mass of a Pyramidal Tree VS. Height of Tree')

```

1.d.xiii.

```
% Script
% Modeling the relationship between the leaf mass of tree and
% the height of tree or circumference of trunk (fountain)
% This model is used to model the following types of tree:
% - Spreading Tree
% - Weeping Tree
% - Layered Tree
% - Vase Tree
% - Full-crowned Tree
% - Fountain Tree
% - Columnar Tree
% - Pyramidal Tree
% Assumptions:
%   - The tree grows perpendicularly out of the ground
%   - Angles and ratios used are justified in the report

% Request user input for type of tree
% Create cell array of the type of trees
T= {'Full-crowned'; 'Vase'; 'Spreading'; 'Layered'; 'Weeping'; 'Pyramidal';
'Columnar'; 'Fountain'};
disp('List of tree types:')
disp(T)
type= input('Type of tree: ','s');

% If it is a fountain tree, regions and seasons are irrelevant
if strcmp(type, 'Fountain')
    plotFountainTree
    return
end

% If it is not a fountain tree, request user input for region and season
% Create cell array of the regions and seasons
% Assume that summer and spring have the same effect on leaf mass and in
% winter, leaf mass becomes 0 for these trees
R= {'Alpine'; 'Temperate'; 'Tropics'};
disp('List of regions:')
disp(R)
region= input('Region: ','s');
S= {'Spring'; 'Summer'; 'Autumn'};
disp('List of seasons (except winter, where leaf mass is erratic, or zero):')
disp(S)
season= input ('Season: ','s');
% Since Summer and Spring have the same effect on leaf mass
if strcmp(season, 'Summer')
    season='Spring';
end

if strcmp(type, 'Spreading') || strcmp(type, 'Layered') || strcmp(type, 'Vase')
    plotSLVTree(region, season)
    return
elseif strcmp(type, 'Weeping')
    plotWeepingTree(region, season)
    return
elseif strcmp(type, 'Full-crowned')
    plotFullCrownedTree(region, season)
    return
elseif strcmp(type, 'Columnar')
    plotColumnarTree(region, season)
    return
elseif strcmp(type, 'Pyramidal')
    plotPyramidalTree(region, season)
```

```

    return
end

```

2. Models for Leaf Shape

2.a. Model 1

2.a.i.

```

function SA= leafSA1(S,r,p)
% Returns the total surface area, SA, of leaves on one particular level on the
% tree
% S is a cell array of all the leaf shapes in 1-0 form. 1 if there is a
% part of leaf there, 0 if there isn't.
% r is the radius of the circular level (we will model a square though)
% p is the probability that a leaf will grow at a point in this level
% All formulas used will be detailed in the report
% Assumptions
%   - All leaves of the same shape orientate in the same way
%   - All leaves of the same shape are exactly the same
%   - All levels are modeled as a square
%   - In an attempt to model a 3D level, if the conditions are right for a
%     leaf to grow at a point, 3 leaves grow: 1 in the plane, 1 upwards out
%     of the plane, and 1 downwards out of the plane
%   - Leaves can only grow where there are no leaves or branches in its
%     growth area

% Initialize and calculate relevant variables
n= round(2*r);                                     % length of the side of the level
(assumed to be a square)
level= zeros(n,n);                                % create a level of the tree. 0
represents there being no leaf, 1 being is part of a leaf. all points (1cm^2)
are initialized as 0.
[a,b]= size(S);                                    % determine size of S (S is always
an odd-by-odd matrix)
horizontalbound= floor(b/2);
verticalbound= floor(a/2);
SA= 0;                                              % total surface area of all the
leaves on the level

% Create the branches on the level (justified in report)
for x=1:n
    y=round((n/2)*(sin(0.5*x^2/n^2)+cos(0.15*x^2/n^2)));
    level(y,x)=2;
end
for y=1:n
    x=round((n/2)*(sin(0.5*x^2/n^2)+cos(0.15*x^2/n^2)));
    level(y,x)=2;
end

% Calculate the surface area of the leaf
sum= 0;                                            % initialize sum to count surface
area of leaf
for q=1:a
    for w=1:b
        if S(q,w)==1                            % if there is a part of the leaf
here
            sum= sum+1;                         % update surface area counter
        end
    end
end

```

```
% Create the leaves on the level (planar)
for i=1:n
    for j=1:n
        % if probability is hit, and level(i,j) is within boundaries for a
        % leaf to grow
        if rand < p && ...
            j-horizontalbound >= 1 && j+horizontalbound <= n ...
            && i-verticalbound >= 1 && i+verticalbound <= n
            % and if there are no parts of a leaf or branch in the boundary
            allZeros= 1; % initialize counter allZeros: 1 if there
are no leafs or branches in boundary, 0 otherwise
            f= 0; % initialize counter f for rows
            while allZeros==1 && f<a
                g= 0; % initialize counter g for columns
                while allZeros==1 && g<b
                    if level(i-verticalbound+f,j-horizontalbound+g) ~=0
                        allZeros= 0;
                    end
                    g= g+1; % Update g
                end
                f= f+1; % Update f
            end
            if allZeros==1
                % if there are really no leaves or branches in the boundary, a
leaf grows
                level(i-verticalbound:i+verticalbound,j-
horizontalbound:j+horizontalbound)= S;
                SA= SA+sum; % add to the total surface area the
surface area of 1 leaf
            end
        end
    end
end

SA= SA*3; % 3D level, where leaves grow in 3 directions
```

2.a.ii.

```
function [LS,Shapes,trans,pc]= leafShapes
% Create Shapes, a cell array that stores names of shapes of leaves
% Create LS, a cell array that contains 1-0 matrices representing these
% leaves. The k-th element of Shapes corresponds to the respective shape
% in LS.
% Each element in the matrix represents a 1cm^2 area
% Create trans, the transmittance of the leaf

% Create a list of all leaf shapes
Shapes= {'Sword';'Elliptic';'Maple';'Round';'Bipinnate'};

% Create a list of all the transmittance values of each leaf shape
% (explained in report)
trans= [0.65;0.6;0.55;0.5;0.7];

% pc is the proportionality constant that affects intensity indirectly for
% each leaf shape
pc= [5;9;7;10;14];

% Shapes of different types of leaves
% Sword-shaped (11x3)
sword= [0,1,0;
         0,1,0;
         1,1,1;
         1,1,1;
```

```

1,1,1;
1,1,1;
1,1,1;
1,1,1;
1,1,1;
0,1,0;
0,1,0;];

% Elliptic-shaped (11x5)
elliptic= [0 0 1 0 0;
            0 1 1 1 0;
            0 1 1 1 0;
            1 1 1 1 1;
            1 1 1 1 1;
            1 1 1 1 1;
            1 1 1 1 1;
            1 1 1 1 1;
            0 1 1 1 0;
            0 1 1 1 0;
            0 0 1 0 0];

% Maple (Lobed)-shaped (11*9)
maple= [0 0 0 0 1 0 0 0 0;
            1 0 0 1 1 1 0 0 1;
            1 1 0 1 1 1 0 1 1;
            0 1 1 1 1 1 1 1 0;
            0 1 1 1 1 1 1 1 0;
            0 1 1 1 1 1 1 1 0;
            0 1 1 1 1 1 1 1 0;
            0 0 1 1 1 1 1 0 0;
            0 0 1 1 1 1 1 0 0;
            0 0 1 1 1 1 1 0 0;
            0 0 0 1 1 1 0 0 0;
            0 0 0 1 1 1 0 0 0];
            0 0 0 1 1 1 0 0 0];

% Round-shaped (9*9)
round= [0 0 0 0 1 0 0 0 0;
            0 0 0 1 1 1 0 0 0;
            0 0 1 1 1 1 1 0 0;
            0 1 1 1 1 1 1 1 0;
            1 1 1 1 1 1 1 1 1;
            0 1 1 1 1 1 1 1 0;
            0 0 1 1 1 1 1 0 0;
            0 0 0 1 1 1 0 0 0;
            0 0 0 0 1 0 0 0 0];
            0 0 0 0 1 0 0 0 0];

% Bipinnate (11*7)
bipinnate=[ 1 0 0 1 0 0 1 ;
            0 1 1 1 1 1 0 ;
            1 0 0 1 0 0 1 ;
            0 1 1 1 1 1 0 ;
            0 1 0 1 0 1 0 ;
            1 0 1 1 1 0 1 ;
            0 1 0 1 0 1 0 ;
            1 0 1 1 1 0 1 ;
            0 1 0 1 0 1 0 ;
            0 0 1 1 1 0 0 ;
            0 0 0 1 0 0 0 ];

% Create LS
LS= cell(5);
LS{1}= sword;
LS{2}= elliptic;
LS{3}= maple;
LS{4}= round;

```

```
LS{5}= bipinnate;
```

2.b. Model 2

2.b.i.

```

function
[levels,lengthFirstBranch,heightFirstBranchLevel,heightSecondBranchLevel]=
treeData(type,height)
% Returns the following outputs:
% m is the difference between the height of the first and second branching
% levels divided by the height of the first branching level from the ground.
% levels refer to the number of levels that tree has
% lengthFirstBranch is the length of branches in the first branching level
% The following are the input parameters:
% height is a double value that is the height of the tree
% type is a string and it is one of the tree types: 'Spreading', 'Layered',
% 'Vase', 'Weeping','FullCrowned', 'Columnar', 'Pyramidal'
% Values for all the characteristics of trees below are justified in the
% report
% heightFirstBranchLevel is the height of the first branching level from
% the ground
% heightSecondBranchLevel is the height of the second branching level from
% the first branching level

% Create level 1 fields in T field array
T= {'Spreading','Layered','Vase','Weeping',...
      'FullCrowned','Columnar','Pyramidal'};

% Find out which tree type the user input
for a=1:length(T)
    if strcmp(type,T(a))
        i=a;                                % i is the rank of tree type
    end
end

if i==1 || i==2 || i==3          % Spreading, Layered, Vase
    heightFirstBranchLevel= height/5;
    heightSecondBranchLevel= height/7;
    levels= round((height-heightFirstBranchLevel)/(heightSecondBranchLevel));
    lengthFirstBranch= height/5;
elseif i==4                      % Weeping
    heightFirstBranchLevel= height/3;
    heightSecondBranchLevel= height/5;
    levels= round((height-heightFirstBranchLevel)/(heightSecondBranchLevel));
    lengthFirstBranch= height/7;
elseif i==5                      % FullCrowned
    heightFirstBranchLevel= height/4;
    heightSecondBranchLevel= height/5;
    levels= round((height-heightFirstBranchLevel)/(heightSecondBranchLevel));
    lengthFirstBranch= height/6;
elseif i==6                      % Columnar
    heightFirstBranchLevel= height/9;
    heightSecondBranchLevel= height*4/10;
    levels= round((height-heightFirstBranchLevel)/(heightSecondBranchLevel));
    lengthFirstBranch= height/10;
elseif i==7                      % Pyramidal
    heightFirstBranchLevel= height/7;
    heightSecondBranchLevel= height*6/10;
    levels= round((height-heightFirstBranchLevel)/(heightSecondBranchLevel));
    lengthFirstBranch= height/6;
end

```

2.b.ii.

```

function I= intensitySunlight(region,season)
% Returns I, the intensity of sunlight (Watts per m^2)
% region is a string and it is the type of region that the leaf is in:
% Alpine, Temperate, Tropics
% season is a string and it is the type of season that the leaf is in:
% Spring/Summer, Autumn

% Create a list of all regions, seasons
Regions= {'Alpine','Temperate','Tropics'};
Seasons= {'Spring','Summer','Autumn'};

% Find out which region the user input
for a=1:length(Regions)
    if strcmp(region,Regions(a))
        i=a; % i is the rank of region
    end
end

% Find out which season the user input
for b=1:length(Seasons)
    if strcmp(season,Seasons(b))
        j=b; % j is the rank of season
    end
end

if i==1 % Alpine
    if j==1 || j==2 % Spring or Summer
        I= 112;
    elseif j==3 % Autumn
        I= 81.75;
    end
elseif i==2 % Temperate
    if j==1 || j==2 % Spring or Summer
        I= 133;
    elseif j==3 % Autumn
        I= 108;
    end
elseif i==3 % Tropics
    I= 157;
end

```

2.b.iii.

```

function TSA= leafTSAColumnar(shape,type,height,I)
% Returns TSA, the total surface area of all the leaves on the Columnar tree
% shape is the shape of the leaf
% type is the type of tree
% height is the height of tree (in meters)
% I is the intensity of sunlight (Watts/m^2)

% Extract 1-0 array for leaf shapes
[LS,Shapes,trans,pc]= leafShapes; % Call function leafShapes
for k=1:length(Shapes)
    if strcmp(shape,Shapes(k))
        S= LS{k}; % S is a 1-0 matrix that represents the
leaf
        t= trans(k); % t is the transmittance of the leaf
        A= pc(k); % A is the proportionality constant
associated with the leaf shape that affects intensity
    end

```

```

end

% Calculate the surface area of the leaf
[a,b]= size(S);
% an odd-by-odd matrix)
sum= 0;
area of leaf
for q=1:a
  for w=1:b
    if S(q,w)==1
here
      sum= sum+1;
    end
  end
end
sum=sum/10^4;                                % Convert to m^2

% Extract data for specified type of tree with its height
[levels,lengthFirstBranch,heightFirstBranchLevel,heightSecondBranchLevel]=
treeData(type,height);

% Initialize TSA
TSA= 0;

% Store intensity of sunlight (because I will change down the levels of
% tree)
store= I;

% I1 is a scaled value of intensity (explained in report)
I1=I/A;

% Calculate fixed radius of level (explained in report) (multiplied by 100
% to convert to cm)
r= height/5*100;

% Create all the levels on the tree (top to base)
for k=1:levels
  p= max(0,(I-3)/I);                         % p is the probability that a leaf
  will appear at a point on the k-th level of the tree (when intensity is too
  low, probability hits 0)
  SA= leafSA1(S,r,p);                        % Call function leafSA1 to obtain the
  surface area of the leafs on the k-th level
  I= store+p*(-store+t*I);                    % Update the intensity of sunlight
  reaching the next lower level
  TSA= TSA+SA;                                % Update total surface of the leafs on
  the tree
end

```

2.b.iv.

```

function TSA= leafTSAPyramidal(shape,type,height,I)
% Returns TSA, the total surface area of all the leaves on the Pyramidal tree
% shape is the shape of the leaf
% type is the type of tree
% height is the height of tree (in meters)
% I is the intensity of sunlight (Watts/m^2)

% Extract 1-0 array for leaf shapes
[LS,Shapes,trans,pc]= leafShapes;           % Call function leafShapes
for k=1:length(Shapes)
  if strcmp(shape,Shapes(k))

```

```

S= LS{k};                                % S is a 1-0 matrix that represents the
leaf
t= trans(k);                            % t is the transmittance of the leaf
A= pc(k);                               % A is the proportionality constant
associated with the leaf shape that affects intensity
end
end

% Calculate the surface area of the leaf
[a,b]= size(S);                         % determine size of S (S is
always an odd-by-odd matrix)
sum= 0;                                   % initialize sum to count surface
area of leaf
for q=1:a
  for w=1:b
    if S(q,w)==1                         % if there is a part of the leaf
here
      sum= sum+1;                        % update surface area counter
    end
  end
end
sum=sum/10^4;                           % Convert to m^2

% Extract data for specified type of tree with its height
[levels,lengthFirstBranch,heightFirstBranchLevel,heightSecondBranchLevel]=
treeData(type,height);

% Initialize TSA
TSA= 0;

% I1 is a scaled value of intensity (explanation in report)
I1=I/A;

% Create all the levels on the tree (top to base)
for k=1:levels
  r= lengthFirstBranch*(height-heightFirstBranchLevel-(k-1)*...
    heightSecondBranchLevel)/(height-heightFirstBranchLevel)*100;   %
Calculate the radius of the k-th level (multiply by 100 to obtain cm)
  p= max(0,(I-3)/I);                  % p is the probability that a leaf
will appear at a point on the k-th level of the tree (when intensity is too
low, probability hits 0)
  SA= leafSA1(S,r,p);                % Call function leafSA1 to obtain the
surface area of the leafs on the k-th level
  I= I1+((height-heightFirstBranchLevel-(k-2)*heightSecondBranchLevel)/...
    (height-heightFirstBranchLevel-(k-1)*heightSecondBranchLevel))^2*...
    p*(-I1+t*I);                     % Update the intensity of sunlight
reaching the next lower level
  TSA= TSA+SA;                      % Update total surface of the leafs
on the tree
end

2.b.v.

function TSA= leafTSAOthers(shape,type,height,I)
% Returns TSA, the total surface area of all the leaves on all other tree

```

```

% shapes is the shape of the leaf
% type is the type of tree
% height is the height of tree (in meters)
% I is the intensity of sunlight (Watts/m^2)
% Assumptions:
%   - The tree has a biggest radius in the middle level (remains constant
%   for a few levels), and has the smallest radii at the top and bottom
%   levels

% Extract 1-0 array for leaf shapes
[LS,Shapes,trans,pc]= leafShapes;           % Call function leafShapes
for k=1:length(Shapes)
    if strcmp(shape,Shapes(k))
        S= LS{k};                           % S is a 1-0 matrix that represents the
leaf
        t= trans(k);                      % t is the transmittance of the leaf
        A= pc(k);                         % A is the proportionality constant
associated with the leaf shape that affects intensity
    end
end

% Calculate the surface area of the leaf
[a,b]= size(S);
% determine size of S (S is always an odd-by-odd matrix)
sum= 0;                                     % initialize sum to count surface
area of leaf
for q=1:a
    for w=1:b
        if S(q,w)==1                     % if there is a part of the leaf
here
            sum= sum+1;                  % update surface area counter
        end
    end
end
sum=sum/10^4;                                % Convert to m^2

% Extract data for specified type of tree with its height
[levels,lengthFirstBranch,heightFirstBranchLevel,heightSecondBranchLevel]=
treeData(type,height);

% Initialize TSA
TSA= 0;

% Store intensity of sunlight (because I will change down the levels of
% tree)
store= I;

% I1 is a scaled value of intensity (explained in report)
I1=I/A;

% Create a third of all the levels on the tree (top to middle)
for k=1:floor(levels/3)
    r= 2*lengthFirstBranch*(height-heightFirstBranchLevel-(levels-k-1)*...
        heightSecondBranchLevel)/(height-heightFirstBranchLevel)*100;   %
Calculate the radius of the k-th level (multiply by 100 to obtain cm)

```

```

p= max(0,(I-3)/I); % p is the probability that a leaf
will appear at a point on the k-th level of the tree (when intensity is too
low, probability hits 0)
SA= leafSA1(S,r,p); % Call function leafSA1 to obtain the
surface area of the leafs on the k-th level
I= I1+((height-heightFirstBranchLevel-(k-2)*heightSecondBranchLevel)/...
(height-heightFirstBranchLevel-(k-1)*heightSecondBranchLevel))^2* ...
p*(-I1+t*I); % Update the intensity of sunlight
reaching the next lower level
TSA= TSA+SA; % Update total surface of the leafs
on the tree
end

% Create the middle third of the levels (constant radius) on the tree
% (middle)
for k=1:floor(levels/3)
    p= max(0,(I-3)/I); % p is the probability that a leaf
will appear at a point on the k-th level of the tree (when intensity is too
low, probability hits 0)
    SA= leafSA1(S,r,p); % Call function leafSA1 to obtain the
surface area of the leafs on the k-th level
    I= store+p*(-store+t*I); % Update the intensity of sunlight
reaching the next lower level
    TSA= TSA+SA; % Update total surface of the leafs
on the tree
end

% Create the final third of the remaining levels on the tree (middle to
bottom)
for k=floor(levels/3):-1:1
    r= 1.5*lengthFirstBranch*(height-heightFirstBranchLevel-(k-1)*...
heightSecondBranchLevel)/(height-heightFirstBranchLevel)*100; % 
Calculate the radius of the k-th level (multiply by 100 to obtain cm)
    p= max(0,(I-3)/I); % p is the probability that a leaf
will appear at a point on the k-th level of the tree (when intensity is too
low, probability hits 0)
    SA= leafSA1(S,r,p); % Call function leafSA1 to obtain the
surface area of the leafs on the k-th level
    I= I1+((height-heightFirstBranchLevel-(k-2)*heightSecondBranchLevel)/...
(height-heightFirstBranchLevel-(k-1)*heightSecondBranchLevel))^2* ...
p*(-I1+t*I); % Update the intensity of sunlight
reaching the next lower level
    TSA= TSA+SA; % Update total surface of the leafs
on the tree
End

```

2.c. Integration

2.c.i.

```

function rP= ratePhotosynthesis(region,season)
% Return rP, the rate of photosynthesis, which is in nanomol per gram of
% leaf per sec
% region is a string and it is the type of region that the leaf is in:
% Alpine, Temperate, Tropics
% season is a string and it is the type of season that the leaf is in:

```

```
% Spring/Summer, Autumn
% Assumptions:
% - We only look at spring, summer and autumn. Data for spring and summer
% are similar

% Create a list of all regions, seasons
Regions= {'Alpine','Temperate','Tropics'};
Seasons= {'Spring','Summer','Autumn'};

% Find out which region the user input
for a=1:length(Regions)
    if strcmp(region,Regions(a))
        i=a; % i is the rank of region
    end
end

% Find out which season the user input
for b=1:length(Seasons)
    if strcmp(season,Seasons(b))
        j=b; % j is the rank of season
    end
end

if i==1 % Alpine
    if j==1 || j==2 % Spring or Summer
        rP= 184.5;
    elseif j==3 % Autumn
        rP= 143.1;
    end
elseif i==2 % Temperate
    if j==1 || j==2 % Spring or Summer
        rP= 210.7;
    elseif j==3 % Autumn
        rP= 179.3;
    end
elseif i==3 % Tropics
    rP= 85.9;
end
```

2.c.ii.

```
function rT= rateTranspiration(region,season,shape)
% returns rT, which is the rate of transpiration in grams per hour
% per 100 cm^2
% region is a string and it is the type of region that the leaf is in:
% Alpine, Temperate, Tropics
% season is a string and it is the type of season that the leaf is in:
% Spring/Summer, Autumn
% shape is a string and it is the shape of the leaf
% Assumptions:
% - We only look at spring, summer and autumn.
% - Temperature data for different regions and season are explained in
% the report (Temperature is in degree celsius)
% - Leaves do not transpire below 0 degrees celsius (e.g. Alpine spring
% and autumn)

% Temperature Ranges (degree celsius):
% Alpine:
% Spring = -15-5
% Summer = 0-5
% Autumn = -15-5
% Temperate:
% Spring = 18-25
```

```

% Summer = 26-30
% Autumn = 12-17
% Tropics
% Spring,Summer,Autumn = 22-32

% Function that relates temperature t to transpiration rate (in the order of
% regions: Alpine, Temperate, Tropics)
% {max(0,t/10), t*3/20, t/5};

% Transpiration rate of leafs (in the order of shapes: Sword, Elliptic,
% Maple, Round, Bipinnate
% {0.19*f, 0.44*f, 0.21*f, 0.3*f, 0.19*f};

% Create a list of all regions, seasons, leaf shapes
Regions= {'Alpine','Temperate','Tropics'};
Seasons= {'Spring','Summer','Autumn'};
Shapes= {'Sword','Elliptic','Maple','Round','Bipinnate'};

% Find out which region the user input
for a=1:length(Regions)
    if strcmp(region,Regions(a))
        i=a;                                % i is the rank of region
    end
end

% Find out which season the user input
for b=1:length(Seasons)
    if strcmp(season,Seasons(b))
        j=b;                                % j is the rank of season
    end
end

% Find out which leaf the user input
for c=1:length(Shapes)
    if strcmp(shape,Shapes(c))
        k=c;                                % k is the rank of leaf shape
    end
end

if i==1
    if j==1 || j==3
        t= 0;                                % Alpine
        f= max(0,t/10);                      % Spring or Autumn
                                                % t is always below zero
                                                % function relating t and transpiration rate
                                                % Sword
        if k==1
            rT= 0.19*f;
        elseif k==2
            rT= 0.44*f;
        elseif k==3
            rT= 0.21*f;
        elseif k==4
            rT= 0.3*f;
        elseif k==(5)
            rT= 0.19*f;
        end
    elseif j==2
        t= 5*rand;                            % Summer
                                                % Randomize a temperature betw the specified
                                                % function relating t and transpiration rate
                                                % Sword
        range for this region
        f= max(0,t/10);
        if k==1
            rT= 0.19*f;
        elseif k==2
            rT= 0.44*f;
        elseif k==3
            rT= 0.21*f;
    end
end

```

```

        elseif k==4                      % Round
            rT= 0.3*f;
        elseif k==(5)                    % Bipinnate
            rT= 0.19*f;
        end
    end
elseif i==2                         % Temperate
if j==1                            % Spring
    t= (25-18)*rand+18;             % Randomize a temperature betw the specified
range for this region
    f= t*3/20;                     % function relating t and transpiration rate
    if k==1                          % Sword
        rT= 0.19*f;
    elseif k==2                      % Elliptic
        rT= 0.44*f;
    elseif k==3                      % Maple
        rT= 0.21*f;
    elseif k==4                      % Round
        rT= 0.3*f;
    elseif k==(5)                  % Bipinnate
        rT= 0.19*f;
    end
elseif j==2                         % Summer
    t= (30-26)*rand+26;             % Randomize a temperature betw the specified
range for this region
    f= t*3/20;                     % function relating t and transpiration rate
    if k==1                          % Sword
        rT= 0.19*f;
    elseif k==2                      % Elliptic
        rT= 0.44*f;
    elseif k==3                      % Maple
        rT= 0.21*f;
    elseif k==4                      % Round
        rT= 0.3*f;
    elseif k==(5)                  % Bipinnate
        rT= 0.19*f;
    end
elseif j==3                         % Autumn
    t= (17-12)*rand+12;             % Randomize a temperature betw the specified
range for this region
    f= t*3/20;                     % function relating t and transpiration rate
    if k==1                          % Sword
        rT= 0.19*f;
    elseif k==2                      % Elliptic
        rT= 0.44*f;
    elseif k==3                      % Maple
        rT= 0.21*f;
    elseif k==4                      % Round
        rT= 0.3*f;
    elseif k==(5)                  % Bipinnate
        rT= 0.19*f;
    end
end
elseif i==3                         % Tropics (seasons do not matter)
t= (32-22)*rand+22;                 % Randomize a temperature betw the specified range
for this region
    f= t/5;                        % function relating t and transpiration rate
    if k==1                          % Sword
        rT= 0.19*f;
    elseif k==2                      % Elliptic
        rT= 0.44*f;
    elseif k==3                      % Maple
        rT= 0.21*f;
    elseif k==4                      % Round

```

```

    rT= 0.3*f;
elseif k==(5)           % Bipinnate
    rT= 0.19*f;
end
end

```

2.c.iii.

```

function MpA= massPerArea(region,season)
% Returns MpA, the mass (g) per unit area (cm^2) of a leaf
% region is a string and it is the type of region that the leaf is in:
% Alpine, Temperate, Tropics
% season is a string and it is the type of season that the leaf is in:
% Spring/Summer, Autumn

% Create a list of all regions, seasons
Regions= {'Alpine','Temperate','Tropics'};
Seasons= {'Spring','Summer','Autumn'};

% Find out which region the user input
for a=1:length(Regions)
    if strcmp(region,Regions(a))
        i=a;                         % i is the rank of region
    end
end

% Find out which season the user input
for b=1:length(Seasons)
    if strcmp(season,Seasons(b))
        j=b;                         % j is the rank of season
    end
end

if i==1                  % Alpine
    if j==1 || j==2          % Spring or Summer
        MpA= 0.013916;
    elseif j==3              % Autumn
        MpA= 0.0095;
    end
elseif i==2                % Temperate
    if j==1 || j==2          % Spring or Summer
        MpA= 0.019167;
    elseif j==3              % Autumn
        MpA= 0.012916;
    end
elseif i==3                % Tropics
    MpA= 0.039333;
end

```

2.c.iv.

```

% Script
% An attempt to qualitatively decide which shape of leaf is best for the
% given conditions
% Plots 2 graphs comparing all the shapes of leaves with respect to:
% Graph 1: Total rate of photosynthesis (nanomols per sec)
% Graph 2: Total rate of transpiration (grams per hour)

% Request user input
% Create cell array of the type of trees
T= {'FullCrowned'; 'Vase'; 'Spreading'; 'Layered'; 'Weeping'; 'Pyramidal';
'Columnar'};

```

```

disp('List of tree types:')
disp(T)
type= input('Type of tree: ','s');
height= input('Height of Tree (meters): ');
Regions= {'Alpine';'Temperate';'Tropics'};
disp('List of regions:')
disp(Regions)
region= input('Region: ','s');
Seasons= {'Spring';'Summer';'Autumn'};
disp('List of seasons:')
disp(Seasons)
season= input('Season: ','s');

% Determine Intensity of sunlight at this region and season
I= intensitySunlight(region,season);

% Create a list of all leaf shapes
Shapes= {'Sword';'Elliptic';'Maple';'Round';'Bipinnate'};

% Find out which tree type the user input
for a=1:length(T)
    if strcmp(type,T(a))
        i=a;                                % i is the rank of tree type
    end
end

% Create vector bins to count the total rate of photosynthesis and total rate
% of transpiration for each leaf shape
rp=zeros(5,1);           % total rate of photosynthesis
rt=zeros(5,1);           % total rate of transpiration

% Grow the tree with varying leaf shapes
for k=1:5
    shape= Shapes{k};
    if i==1 || i==2 || i==3 || i==4 || i==5          % Full-crowned, Vase,
Spreading, Layered, Weeping
        TSA= leafTSAOthers(shape,type,height,I);
    elseif i==6                                     % Pyramidal
        TSA= leafTSAPyramidal(shape,type,height,I);
    elseif i==7                                     % Columnar
        TSA= leafTSAColumnar(shape,type,height,I);
    end
    rP= ratePhotosynthesis(region,season);
    rT= rateTranspiration(region,season,shape);
    MPA= massPerArea(region,season);
    M= MPA*TSA;                                    % Total mass of leaves (g)
    rp(k)= rP*M;
    rt(k)= rT*M;
end

% Draw bar graphs for photosynthesis
subplot(2,1,1)
bar(rp)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total rate of photosynthesis (nanomols per second)')
title('Total rate of photosynthesis of tree (with different leaf shapes)')

% Draw bar graphs for transpiration
subplot(2,1,2)
bar(rt)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total rate of transpiration (grams per hour)')
title('Total rate of transpiration of tree (with different leaf shapes)')

```

2.c.v.

```
% Script
% An attempt to qualitatively decide which shape of leaf is best for the
% given conditions
% Plots bar graphs comparing all the shapes of leaves with respect to:
% Total leaf mass of tree

% Request user input
% Create cell array of the type of trees
T= {'FullCrowned'; 'Vase'; 'Spreading'; 'Layered'; 'Weeping'; 'Pyramidal';
'Columnnar'};
disp('List of tree types:')
disp(T)
type= input('Type of tree: ','s');
height= input('Height of Tree (meters): ');
Regions= {'Alpine'; 'Temperate'; 'Tropics'};
disp('List of regions:')
disp(Regions)
region= input('Region: ','s');
Seasons= {'Spring'; 'Summer'; 'Autumn'};
disp('List of seasons:')
disp(Seasons)
season= input('Season: ','s');

% Determine Intensity of sunlight at this region and season
I= intensitySunlight(region,season);

% Create a list of all leaf shapes
Shapes= {'Sword'; 'Elliptic'; 'Maple'; 'Round'; 'Bipinnate'};

% Find out which tree type the user input
for a=1:length(T)
    if strcmp(type,T(a))
        i=a;                                % i is the rank of tree type
    end
end

% Create vector bin to count total mass (kg) for each leaf shape
mass=zeros(5,1);

% Grow the tree with varying leaf shapes
for k=1:5
    shape= Shapes{k};
    if i==1 || i==2 || i==3 || i==4 || i==5          % Full-crowned, Vase,
    Spreading, Layered, Weeping
        TSA= leafTSAOthers(shape,type,height,I);
    elseif i==6                                     % Pyramidal
        TSA= leafTSAPyramidal(shape,type,height,I);
    elseif i==7                                     % Columnar
        TSA= leafTSAColumnar(shape,type,height,I);
    end
    MpA= massPerArea(region,season);                % Total mass of leaves (g)
    M= MpA*TSA;
    mass(k)= M/1000;
end

% Draw bar graphs for mass
bar(mass)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total leaf mass (kg)')
title('Total leaf mass (with different leaf shapes)')
```

2.c.vi.

```
% Script
% Examination of Alpine Tree
% Characteristics: Pyramidal type tree, average height of 15 meters
% Usually found in Alpine region.
% We examine Summer season.
% Plot bar graphs comparing all the shapes of leaves with respect to:
% Graph 1: Total rate of photosynthesis (nanomols per sec)
% Graph 2: Total rate of transpiration (grams per hour)
% Graph 3: Total leaf mass (kg)

height= 15;
type= 'Pyramidal';
region= 'Alpine';
season= 'Summer';

% Determine Intensity of sunlight at this region and season
I= intensitySunlight(region,season);

% Create a list of all leaf shapes
Shapes= {'Sword';'Elliptic';'Maple';'Round';'Bipinnate'};

% Create vector bins to count the total rate of photosynthesis and total rate
% of transpiration for each leaf shape
rp=zeros(5,1);           % total rate of photosynthesis
rt=zeros(5,1);           % total rate of transpiration

% Create vector bin to count total mass (kg) for each leaf shape
mass=zeros(5,1);

% Grow the tree with varying leaf shapes
for k=1:5
    shape= Shapes{k};
    TSA= leafTSAPyramidal(shape,type,height,I);
    rP= ratePhotosynthesis(region,season);
    rT= rateTranspiration(region,season,shape);
    MpA= massPerArea(region,season);
    M= MpA*TSA;                  % Total mass of leaves (g)
    mass(k)= M/1000;
    rp(k)= rP*M;
    rt(k)= rT*M;
end

% Draw bar graphs for photosynthesis
subplot(3,1,1)
bar(rp)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total rate of photosynthesis (nanomols per second)')
title('Total rate of photosynthesis of tree (with different leaf shapes)')

% Draw bar graphs for transpiration
subplot(3,1,2)
bar(rt)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total rate of transpiration (grams per hour)')
title('Total rate of transpiration of tree (with different leaf shapes)')

% Draw bar graphs for mass
subplot(3,1,3)
bar(mass)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
```

```
ylabel('Total leaf mass (kg)')
title('Total leaf mass (with different leaf shapes)')
```

2.c.vii.

```
% Script
% Examination of Ficus Elastica (aka Rubber Tree)
% Characteristics: Spreading type tree, average height of 15 meters
% Usually found in Tropics region.
% We examine Summer season.
% Plot bar graphs comparing all the shapes of leaves with respect to:
% Graph 1: Total rate of photosynthesis (nanomols per sec)
% Graph 2: Total rate of transpiration (grams per hour)
% Graph 3: Total leaf mass (kg)

height= 15;
type= 'Spreading';
region= 'Tropics';
season= 'Summer';

% Determine Intensity of sunlight at this region and season
I= intensitySunlight(region,season);

% Create a list of all leaf shapes
Shapes= {'Sword';'Elliptic';'Maple';'Round';'Bipinnate'};

% Create vector bins to count the total rate of photosynthesis and total rate
% of transpiration for each leaf shape
rp=zeros(5,1); % total rate of photosynthesis
rt=zeros(5,1); % total rate of transpiration

% Create vector bin to count total mass (kg) for each leaf shape
mass=zeros(5,1);

% Grow the tree with varying leaf shapes
for k=1:5
    shape= Shapes{k};
    TSA= leafTSAPyramidal(shape,type,height,I);
    rP= ratePhotosynthesis(region,season);
    rT= rateTranspiration(region,season,shape);
    MpA= massPerArea(region,season);
    M= MpA*TSA; % Total mass of leaves (g)
    mass(k)= M/1000;
    rp(k)= rP*M;
    rt(k)= rT*M;
end

% Draw bar graphs for photosynthesis
subplot(3,1,1)
bar(rp)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total rate of photosynthesis (nanomols per second)')
title('Total rate of photosynthesis of tree (with different leaf shapes)')

% Draw bar graphs for transpiration
subplot(3,1,2)
bar(rt)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total rate of transpiration (grams per hour)')
title('Total rate of transpiration of tree (with different leaf shapes)')

% Draw bar graphs for mass
subplot(3,1,3)
```

```
bar(mass)
set(gca,'XTickLabel',{'Sword','Elliptic','Maple','Round','Bipinnate'})
ylabel('Total leaf mass (kg)')
title('Total leaf mass (with different leaf shapes)')
```

10.2 Leaf Shapes

```
[ 1 0 0 1 0 0 1 ;
 0 1 1 1 1 1 0 ; Bipinnate Leaf
 1 0 0 1 0 0 1 ;
 0 1 1 1 1 1 0 ;
 0 1 0 1 0 1 0 ;
 1 0 1 1 1 0 1 ;
 0 1 0 1 0 1 0 ;
 1 0 1 1 1 0 1 ;
 0 1 0 1 0 1 0 ;
 0 0 1 1 1 0 0 ;
 0 0 0 1 0 0 0 ];

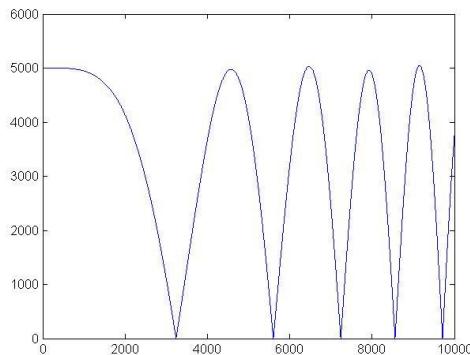
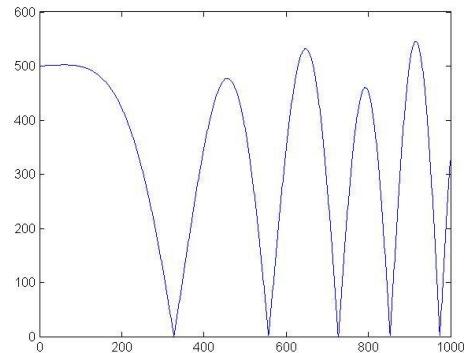
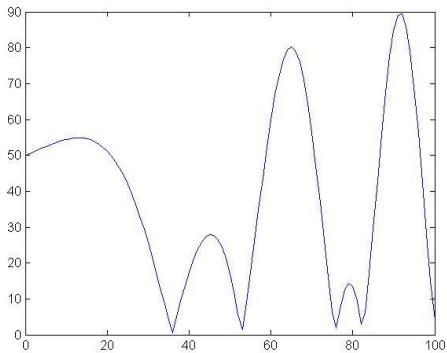
[0 0 0 0 1 0 0 0 0;
 0 0 0 1 1 1 0 0 0; Round-shaped Leaf
 0 0 1 1 1 1 1 0 0;
 0 1 1 1 1 1 1 1 0;
 1 1 1 1 1 1 1 1 1;
 0 1 1 1 1 1 1 1 0;
 0 0 1 1 1 1 1 0 0;
 0 0 0 1 1 1 0 0 0;
 0 0 0 0 1 0 0 0 0];

[0 0 0 0 1 0 0 0 0;
 1 0 0 1 1 1 0 0 1;
 1 1 0 1 1 1 0 1 1; Maple Leaf
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 0 1 1 1 1 1 0 0;
 0 0 1 1 1 1 1 0 0;
 0 0 1 1 1 1 1 0 0;
 0 0 0 1 1 1 0 0 0;
 0 0 0 1 1 1 0 0 0];

[0 0 1 0 0;
 0 1 1 1 0;
 0 1 1 1 0;
 1 1 1 1 1; Elliptic Leaf
 1 1 1 1 1;
 1 1 1 1 1;
 1 1 1 1 1;
 0 1 1 1 0;
 0 1 1 1 0;
 0 0 1 0 0];
```

```
[0,1,0;  
0,1,0;  
1,1,1;  
1,1,1;      Sword-shaped Leaf  
1,1,1;  
1,1,1;  
1,1,1;  
1,1,1;  
1,1,1;  
0,1,0;  
0,1,0;];
```

10.3 Branching Structures for LeafSA2 Model



10.4 Letter to a Scientific Journal Editor

Dear Sir or Madam,

As participants of the International Mathematical Contest in Modeling 2012, we have conducted research and qualitative analysis on leaves and trees to enable us to model the total leaf mass per tree, provided specific parameters such as geographical region, season, and overall shape of the tree. In this model, we also employ mathematical analysis from currently available research papers to determine the relationship between leaf mass per area (LMA) and amount of solar irradiation received by the plant. Hence, given a general geographical region (alpine, temperate, or tropics) and a specific season, our model would be able to deduce the approximate total mass of leaves on a tree.

From this model, we found that tropical trees, at a given height, always have the greatest mass of leaves as compared to the trees in other seasons. This finding in specific and our model in general enable biologist and ecology researchers to better predict the mass of leaves in a given tree. Moreover, this model predicts that in the spring, trees generally have higher mass than in autumn, based on the amount of light irradiation received and the water content in the leaves. This is interesting to note as usually, when one thinks of leaf mass, one does not take the varying seasonal light irradiation into account.

Moreover, from our mathematical analysis and modeling of currently available research papers, we deduce that the leaves of trees that grow in the alpine region are more efficient in performing photosynthesis. Efficient here means that per given mass, the leaves of alpine trees are able to have a higher rate of photosynthesis as compared to the leaves of trees that grow in other regions. In this regard, the leaves of tropical trees came last. We try to explain this phenomenon by the varying amount of sunlight and water that are available in different regions.

Moreover, we have also come up with a probabilistic model to analyze the effects of having different shapes of leaves in a tree. This model is useful as it tries to analyze the efficiency of different leaf shapes in obtaining sunlight for photosynthesis. According to our simulations, trees with sword-shaped leaves are the most efficient in getting the most leaf surface area that are exposed to solar irradiation. This explains that trees in Alpine region have needle-like leaves that, together in a bunch, are similar to the sword-shaped leaves. However, our prediction falls short in explaining the shapes of leaves in the tropical region. After doing further research, we discovered that there are many other environmental factors that affect the shape of leaves in a region, sunlight being only one of the major factors.

In conclusion, our findings are both useful and helpful for ongoing and further research in the attempt to model the shape and mass of leaves in a tree. Our model can further be calibrated to better represent real-life conditions to produce more accurate data. We hope that you are as excited as we are with regards to our research and model findings, and we would be glad if you are able to consider including our research findings on your journal.