Modeling Tree Growth with Vector Calculus

Linnea Laux and Louise Nielsen

December 2016

1 Introduction

In this project, we sought to model tree growth in Mathematica by imagining a tree as a series of progressively larger circles starting at a certain point. The area of the circle in a given year represents a top-down view of the crown of the tree. Each new area encompasses all of the old area, though the center point of each circle is not the same. The tree exists on the backdrop of a scalar function, the value of which at a specific point represents the average amount of sun that that spot in the tree's habitat receives. The tree uses vector calculus to find out which direction to go for maximum sunny-shady contrast.

2 Our Model

The first pass of our model describes the tree as a circle of leaf coverage area that grows by a constant area multiplier in the direction of the point on the current circle with the highest gradient value from the scalar function of sunniness and shadiness. This is obviously an imperfect model, as it will often cause the tree to grow in the direction of the area with the greatest change rather than the area with the most sun. However, this often leads to a tree with a healthy balance of sun and shade and is not outside of the realm of realistic growth possibilities for many tree species.

Mathematically, our model for tree growth is very simple. It took in an initial circle and a sunniness function and used them to find the center of a second circle encompassing the first that would represent the new footprint of the tree.

Where:

$$
\begin{aligned}
r_i &= \text{radius of initial circle} \\
(x_i, y_i) &= \text{initial center point} \\
k_g &= \text{growth constant} \\
m &= \text{minimum margin between initial and new circle} \\
(x_g, y_g) &= \text{point on initial circle with maximum gradient} \\
r_n &= \text{radius of new circle} \\
d &= \text{distance between centers of initial and new circles} \\
(x_n, y_n) &= \text{new center point}
\end{aligned}
$$

The center of the new circle is determined by:

$$\begin{aligned} r_n &= r_i * k_g \\ d &= r_n - (r_i + m) \\ (x_n, y_n) &= \left(d\frac{(x_i - x_g)}{r_i}, d\frac{(y_i - y_g)}{r_i} \right) \end{aligned}$$

The values for r_i, k_g, m, and the initial center point are determined by us and can be varied. All other values are calculated. The calculations for r_n, d, and the new center point are fairly simple and are shown.

The calculation for the point with the highest gradient value is more complicated. We used a combination of several different Mathematica functions to take in a certain number of points on a circle, calculate the gradient at all of those points, and then find the maximum. The function findGradPoints takes in a scalar function (the sunniness function), a list of points (the points on the circle), and the two variables to be used in the scalar function. The function then finds the gradient at all of the points and arranges that into a list that is easy to work with. The next function, findMaxPoint, takes in that list and also the list of points on the circle and yields the position of the point at which the gradient is highest. The following code shows these functions:

```
findGradPoints[scalarFunc_, circlePoints_, x_, y_] :=
 Module[{gradFunc, listGradPoints},
  gradFunc = Grad[scalarFunc[x, y], {x, y}];
  listGradPoints = gradFunc /. {x -> circlePoints[[All, 1]],
      y -> circlePoints[[All, 2]]};
  Transpose[listGradPoints]
  ]

findMaxPoint[listCirclePoints_,listGradPoints_] :=
 Module[{listNorms, maxPosition, listMaxPoints, listMaxGradPoints,
    betterListMaxGradPoints},
  listNorms = Norm /@ listGradPoints;
  maxPosition = Position[listNorms, Max[listNorms]];
  listMaxGradPoints = listCirclePoints[[#]] & /@ maxPosition;
  Flatten[RandomChoice[listMaxGradPoints]]
  ]
```

The next function takes in the current center, the new point, and the distance between them, and gives the new center. This gets done quite cleverly by finding the vector between the two, as shown below.

```
findNewCenter[ptc_, ptn_, d_] := Module[{v, vhat},
  v = ptn - ptc;
  vhat = Norm[v];
  vhat d + ptc
  ]
```

To simplify implementation of the Mathematica ListAnimation, we created a function that takes in the initial center, the initial radius, and a set of less frequently changed parameters and returns the center coordinates and radius of the new circle. The parameters list contains the scalar function representing sunny and shadiness, the two variables the scalar functions takes, the growth constant, the minimum distance, and the number of points to check the gradient at in the

circle. The doEverything function calculates the new radius and the distance between the old and new center point and implements the findGradPoints, findMaxPoint, and findNewCenter functions to return a list with elements new center and new radius. The doEverything function is shown below:

```
doEverything[ptc_, ri_, {scalarFunc_, x_, y_, kg_, minDist_, npts_}] :=
  Module[{rg, d, initialCirclePoints, listGradPoints, p, ptnew},
  rg = ri*kg;
  d = rg - (ri + minDist);
  initialCirclePoints = CirclePoints[ptc, ri, npts];
  listGradPoints = findGradPoints[scalarFunc, initialCirclePoints, x, y];
  p = findMaxPoint[initialCirclePoints, listGradPoints];
  ptnew = findNewCenter[ptc, p, d];
  {ptnew, rg}
  ]
```

3 Results

We chose to start our example tree at (0,0) on the background of the function $f(x, y) = 2xy$. We grew it for 10 iterations.

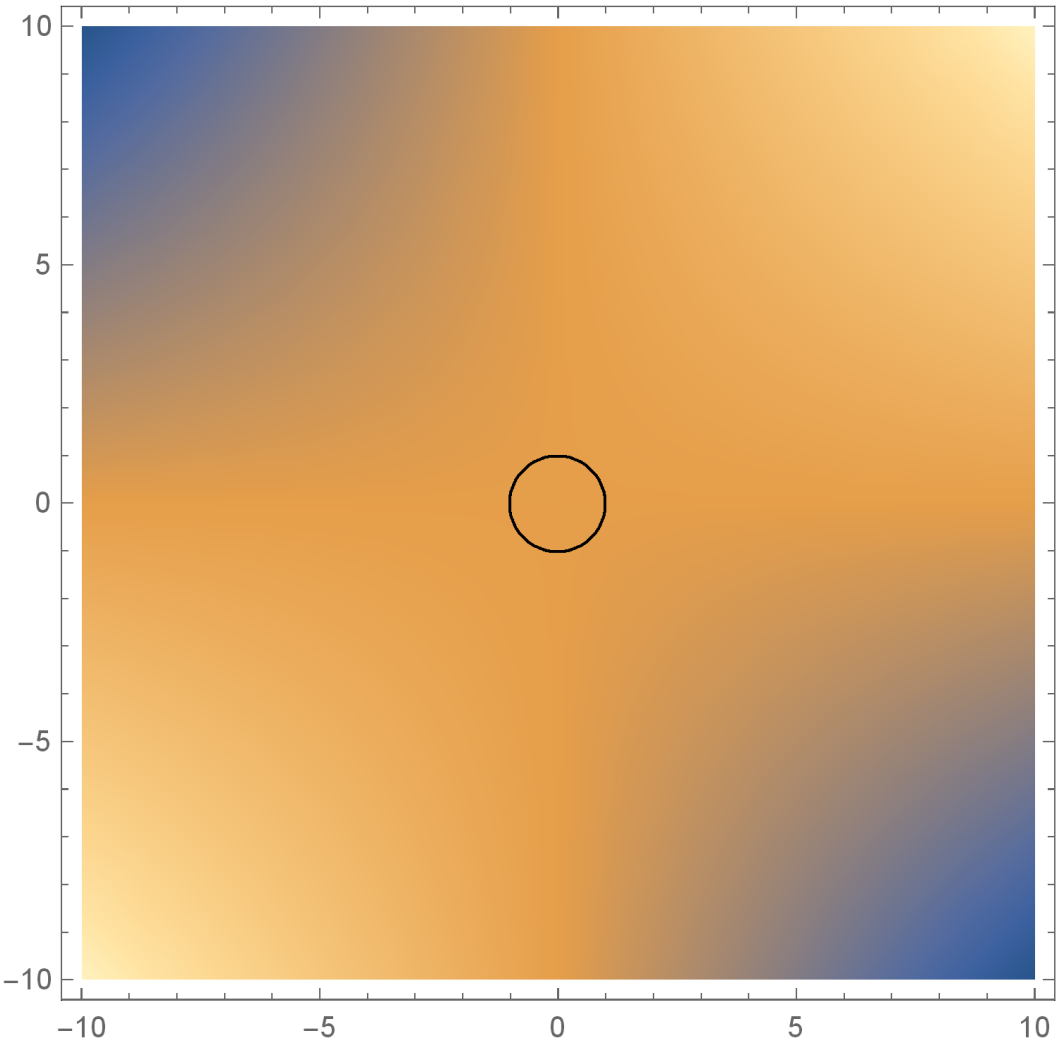

Figure 1: Our model after one year of growth.

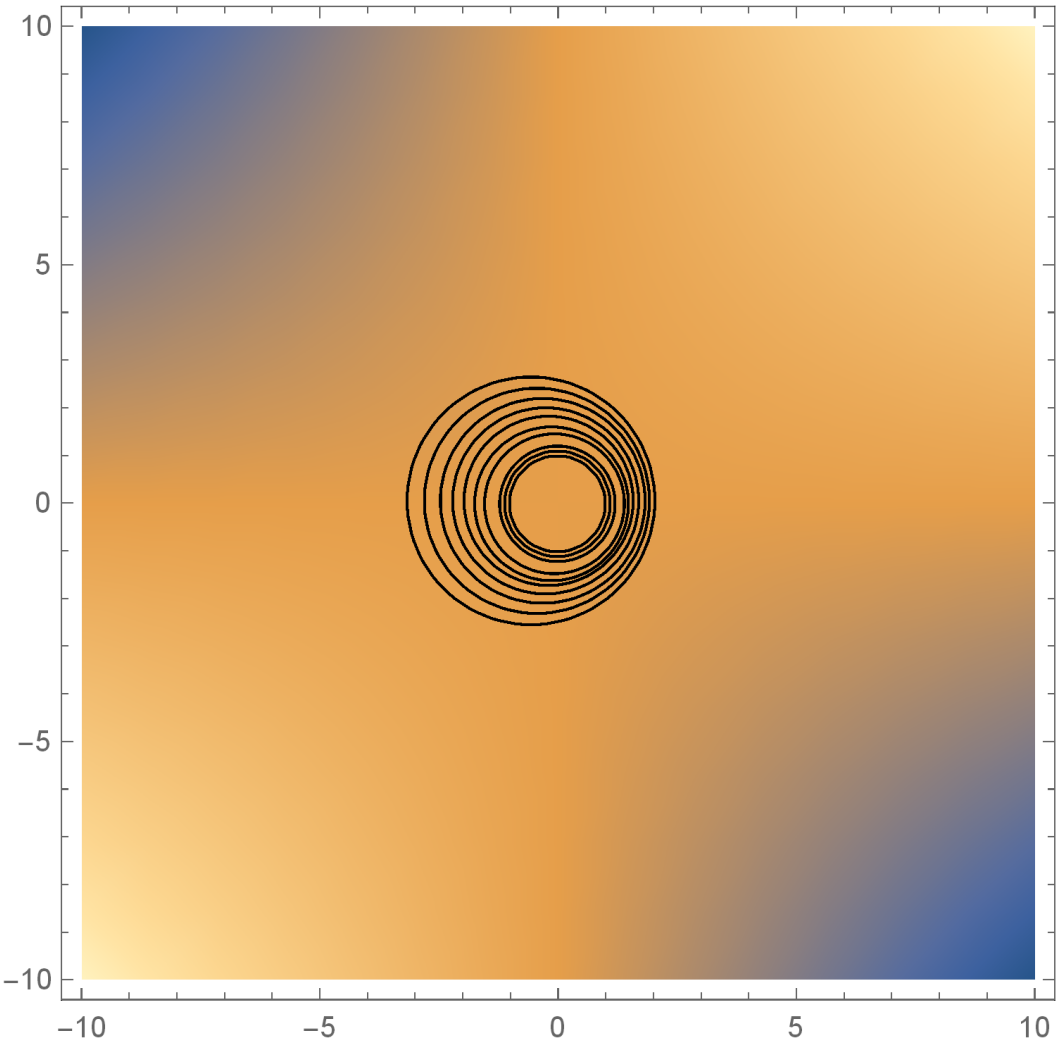

Figure 2: Our model after 10 iterations.