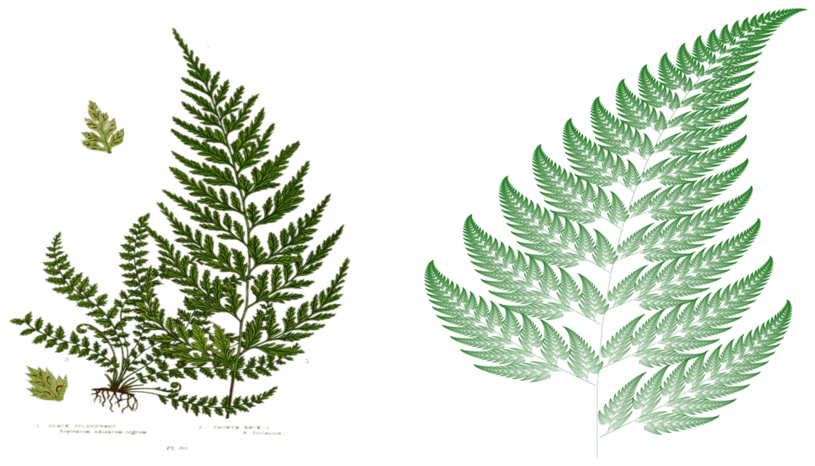
Fractal Ferns

1. INTRODUCTION What are fractals? Fractals have no consistent definition, but rather constant definable traits. These traits are the following: nowhere differentiable, but continuous everywhere; iterated, infinitely self-similar objects; detailed structure when magnified; irregularity locally and globally; and have a fractal dimension. Fractals provide models of many structures from the natural world; coastlines, crystals, DNA, heart rates, snowflakes, ocean waves, blood vessels, lightening bolts, and many others [**2**]. While there are non-natural fractal objects we will be focused particularly on fractal ferns, specifically recreating a fern found in nature. In this article we will heavily rely on the work of Michael Barnsley, building intuition for his fern-generation method and modifying it in order to model a western sword fern.



(a) Black Spleenwort [**5**] (b) Barnsley Fern [**3**]

Figure 1. Barnsley Fern Comparison

The Barnsley Fern is a fractal named for its creator Michael Barnsley. It was made to resemble the Black Spleenwort fern and was first showcased in his book *Fractals Everywhere* in 1988 [**1**]. We can see that Figure 1a is the fern that Barnsley was trying to replicate, and Figure 1b was what he came up with.

2. AFFINE TRANSFORMATIONS AND ITERATED FUNCTION SYSTEMS

An **affine transformation** is any transformation that preserves distances, ratios, and collinearity [**7**]. Here in Figure 2 we see a depiction of an affine transformation of the form

$$T\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos\alpha & -s\sin\beta \\ r\sin\alpha & s\cos\beta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} h \\ k \end{bmatrix} \tag{1}$$

where r and s are scaling values, α and β are rotation angles, and $\vec{v} = \begin{bmatrix} h \\ k \end{bmatrix}$ is a shift vector. In particular, the basis vector e_1 under this transformation is rotated counter-clockwise by an angle of α, scaled by the value r, and then translated by the shift vector $\vec{v}$. Likewise, e_2 is rotated clockwise by an angle β, scaled by s, and translated by the shift vector $\vec{v}$. $T(e_1)$ and $T(e_2)$ are the transformed basis vectors e_1 and e_2, showing the corresponding rotation, dilation, and shift.

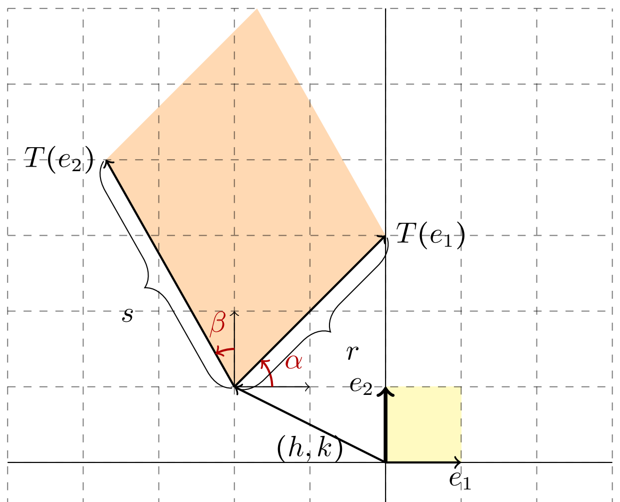


Figure 2. Affine Transformation

The method for generating the Barnsley Fern is an **iterated function system (IFS)**. The IFS takes a point and randomly transforms it using one of four affine transformations, each with a different probability. The collection of all locations the point visits under repeated transformations creates the image. The four affine maps used to

Matrix	a	b	c	d	h	k	p
F_1	0	0	0	0.16	0	0	0.01
F_2	0.85	0.04	-0.04	0.85	0	1.60	0.85
F_3	0.20	-0.26	0.23	0.22	0	1.60	0.07
F_4	-0.15	0.28	0.26	0.24	0	0.44	0.07

Table 1. Barnsley Fern Matrix Values

generate the Barnsley fern are of the form

$$F_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} h \\ k \end{bmatrix} \tag{2}$$

with values (and probabilities p) from Table 1 [**6**].

Notice the affine maps supplied do not specify rotation angles and scalars as in (1), but instead just the numerical entries in the matrix. In order to better visualize the geometry of the iterated function system, we convert each F_i from form (2) to form (1). That is, we want to have

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} r\cos\alpha & -s\sin\beta \\ r\sin\alpha & s\cos\beta \end{bmatrix}.$$

Which implies

$$\begin{bmatrix} a \\ c \end{bmatrix} = r \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \tag{3}$$

and

$$\begin{bmatrix} b \\ d \end{bmatrix} = s \begin{bmatrix} -\sin(\beta) \\ \cos(\beta) \end{bmatrix}, \tag{4}$$

resulting in formulas for α and β:

$$\frac{1}{r} \begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \implies \alpha = \cos^{-1}\left(\frac{a}{r}\right) = \sin^{-1}\left(\frac{c}{r}\right) \tag{5}$$

and

$$\frac{1}{s}\begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} -\sin(\beta) \\ \cos(\beta) \end{bmatrix} \implies \beta = -\sin^{-1}\left(\frac{b}{s}\right) = \cos^{-1}\left(\frac{d}{s}\right). \tag{6}$$

To solve for the scalars r and s we need to recall a fact from trigonometry, which is that for any angles α and β we have

$$(\cos\alpha)^2 + (\sin\alpha)^2 = (-\sin\beta)^2 + (\cos\beta)^2 = 1. \tag{7}$$

Calculating the lengths of the vectors in (3) and (4) we find

$$\sqrt{a^2 + c^2} = r\sqrt{(\cos\alpha)^2 + (\sin\alpha)^2} = r \tag{8}$$

and

$$\sqrt{b^2 + d^2} = s\sqrt{(-\sin\beta)^2 + (\cos\beta)^2} = s. \tag{9}$$

Now that we can determine the angles and scalars involved, we can convert the affine transformations into a format that allows for changing the angles and scalars individually.

3. UNPACKING THE BARNSLEY FERN TRANSFORMATIONS We begin by analyzing the matrix in F_1, noticing

$$\begin{bmatrix} 0 & 0 \\ 0 & 0.16 \end{bmatrix} = .16\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

which corresponds to $r = 0$, $s = .16$, and $\beta = 0$, with any angle possible for α.

Moving on, in F_2 we have $a = 0.85, b = 0.04, c = -0.04,$ and $d = 0.85$. So beginning with the left hand side and using Equation 8 we get the following:

$$\sqrt{0.85^2 + (-0.04)^2} \approx 0.8509 = r.$$

Now using Equation 5 we get that

$$\begin{bmatrix} \frac{0.85}{0.8509} \\ \frac{-0.04}{0.8509} \end{bmatrix} = \begin{bmatrix} 0.9989 \\ -0.0470 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix}.$$

Thus we have that $\cos(\alpha) = 0.9989$, implying $\alpha = \cos^{-1}(0.9989) \approx 0.0470$. Similarly $\sin\alpha = -0.0470$, implying $\alpha = \sin^{-1}(-0.0470) \approx -0.0470$

Then since cosine disregards the negative sign, we can say that $\alpha \approx -0.0470$. Thus our converted a and c are $a = 0.8509 \cdot \cos(-0.0470)$ and $c = 0.8509 \cdot \sin(-0.0470)$. Now since the right hand side is the same barring a negative sign, we can deduce that the converted matrix is

$$F_2 = \begin{bmatrix} 0.8509 \cdot \cos(-0.0470) & -0.8509 \cdot \sin(-0.0470) \\ 0.8509 \cdot \sin(-0.0470) & 0.8509 \cdot \cos(-0.0470) \end{bmatrix}.$$

Similarly, we convert the matrices for F_3

$$\begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} = \begin{bmatrix} 0.3048 \cdot \cos(0.8551) & -0.3406 \cdot \sin(0.8685) \\ 0.3048 \cdot \sin(0.8551) & 0.3406 \cdot \cos(0.8685) \end{bmatrix}$$

and for F_4

$$\begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} = \begin{bmatrix} -0.3002 \cdot \cos(1.0475) & -0.3668 \cdot \sin(-0.8622) \\ -0.3002 \cdot \sin(1.0475) & 0.3668 \cdot \cos(-0.8622) \end{bmatrix}$$

.

After the conversions are done, we modified a Python program ([**6**]) so each transformation F_i is of the form from Equation 1, depending on defined values $r, s, \alpha,$ and β. We make them separate from the matrix in the code in order to easily

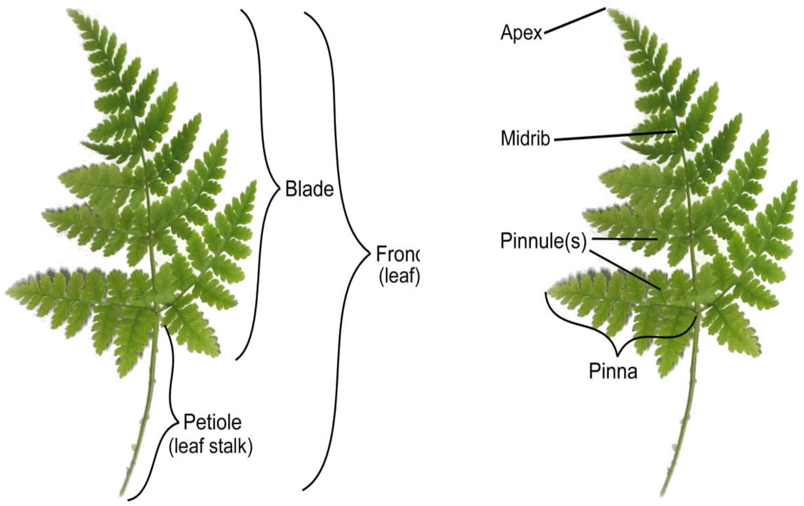


Figure 3. General Fern Structure [**4**]

manipulate them individually. Our code is included in Section 6.

4. VARYING THE PARAMETERS AND UNDERSTANDING THE CODE

Now that we have set up the code properly, it is time to figure out what everything does. That is, what do the scalars do specifically to the fern depending on what matrices they are in? Same goes for the rotations. For the first matrix, which deals with plotting the petiole of the fern, there is only one scalar that is not set to zero and that would be s. The variable s for F_1 adjusts with the height of the petiole.

Each affine map generates a different portion of the fern. We have that F_1 generates the petiole (see Figure 3), F_3 generates the bottom left pinna (see Figure 3), F_4 generated the bottom right pinna (see Figure 3), and F_2 generates successively smaller pinnae (plural of pinna [**4**]) until it reaches its apex [**1**].

The most complex of all the matrices is F_2, which makes copies of the initial pinnae (defined by F_3 and F_4) and petiole and moves up the y-axis while shrinking them. The scalars r and s widens and lengthens the entire fern, respectively. The reason it does the entire fern and not just the copies it makes is due to it needing to be a fractal and keeping the self similarity. The rotation angle α rotates the pinnae about the stem (when positive it rotates it clockwise and when negative it rotates it counterclockwise). The rotation angle β curves the tip of the fern from left to right (left when positive and right when negative). Additionally, β curves the individual pinnae up or down (down when positive and up when negative).

The next matrix F_3 creates the left side of the fern, meaning that it creates the left pinna that F_2 then copies to complete the left side of the fern. The scalar r makes the pinnules (See Figure 3) wider; that is, it makes these pinnules bigger on the y-axis as the value gets larger. Now the scalar s stretches the pinna out about the x-axis as the value gets larger. The angle α rotates the whole pinna about the x-axis. When the value is set to 0 the pinna is completely sideways and looks like a line. When the value is set to $\frac{\pi}{2}$ the pinna is facing forwards, and the cycle continues where at π the pinna is sideways again. The other angle β does almost the same thing, but instead of rotating the pinna about the x-axis, it rotates it about the y-axis. The matrix F_4 has all of its variables do the same things as F_3 except reflected on the right side.

The later part of the code, after all the variable, matrices, shift vectors, and reflection matrix are set up, does the probability calculating and plotting of the points that eventually make up the fern. It starts in a "for loop", which iterates from i = 1 to i = $90,000$. What this does is allows the fern do be detailed since each iteration is represented by a dot on the plot. Next it sets up the probability by letting z be a random number between 1 and 100. Then comes the "if statements". The first of these is for F_1, which as we can see from the Table 1 has a probability of 1%. Within the if statement is the calculation of what x and y are going to be using the Equation 1, given the current x and y values.

The next if statement does the same thing for F_2, except with its corresponding probability of 85%. Then the next "if statements" are for F_3 and F_4, both with the probability of 7%. Each of these "if statements" has an individual color connected to them, so if it is chosen then the dot that is plotted will be in its corresponding color. The last part of the "for loop" is adding 1 to i that way it can properly iterate. The last part of the code is plot all the points and limits the plot size.

5. CREATING THE WESTERN SWORD FERN After figuring out what each rotation and scalar does to the fern, it is time to test different values in order to get the Barnsley fern looking like a Western Sword fern. This process was a lot of making educated guesses based on what the different parameters do and then comparing to the

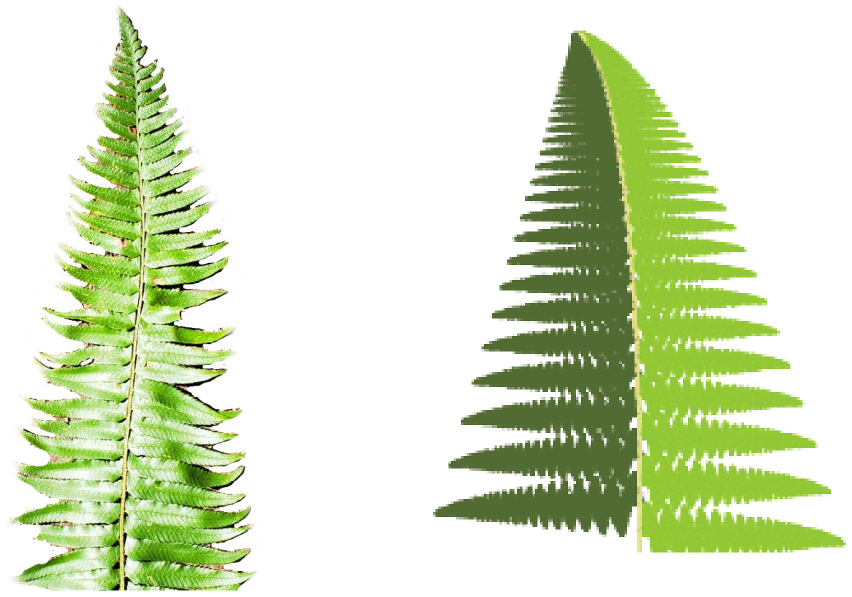


Figure 4. Western Sword Fern vs. Generated Fern

picture of the Western Sword fern taken by the first author in the McDonald National Forest. Finally after many weeks of adjusting we got an accurate representation, see Figure 4.

6. WESTERN SWORD FERN RECREATION PYTHON CODE

```
# modified from https://www.geeksforgeeks.org/barnsley-fern-in-
                                   python/

import math
import matplotlib.pyplot as plt
from random import randint

#Scalings (r and s) and Rotations (alpha and beta) for the Matrices
'''Matrix f_1''' #Stem
f_1r = 0.00 #initial value = 0.00

f_1s =  0.16 #initial value = 0.16

f_1alpha = 0.00 #initial value = 0.00

f_1beta = 0.00 #initial value = 0.00

'''Matrix f_2''' #Duplicates
f_2r = .93 #initial value = 0.8509 #widens the whole fern

f_2s = .9 #initial value = 0.8509 #lengthens the whole fern

f_2alpha = 0.003 #initial value = -0.0470 #rotates pinnas about the
                                   stem
```

```
f_2beta = 0.003 #initial value = -0.0470 #curves tip of fern and
                                   points pinnas up or down

'''Matrix f_3''' #Left side
f_3r = 0.5 #initial value = 0.3048 #widens pinna on the y-axis

f_3s = 0.12 #initial value = 0.3406 #makes pinna's length longer

f_3alpha = 1.57 #initial value = 0.8551 #rotates pinna about the x-
                                   axis

f_3beta = 1.57 #initial value = 0.8685 #rotates pinna about the y-
                                   axis

'''Matrix f_4''' #Right side
f_4r = 0.5 #initial value = 0.3002

f_4s = 0.12 #initial value = 0.3668

f_4alpha = 1.57 #initial value = 1.0475

f_4beta = 1.57 #initial value = -0.8622


#Matrices
f_1 = [[f_1r * math.cos(f_1alpha), f_1s * -math.sin(f_1beta)],
       [f_1r * math.sin(f_1alpha), f_1s *  math.cos(f_1beta)]]

f_2 = [[f_2r * math.cos(f_2alpha), f_2s * -math.sin(f_2beta)],
       [f_2r * math.sin(f_2alpha), f_2s *  math.cos(f_2beta)]]

f_3 = [[f_3r * math.cos(f_3alpha), f_3s * -math.sin(f_3beta)],
       [f_3r * math.sin(f_3alpha), f_3s *  math.cos(f_3beta)]]

f_4 = [[f_4r * math.cos(f_4alpha), f_4s * -math.sin(f_4beta)],
       [f_4r * math.sin(f_4alpha), f_4s *  math.cos(f_4beta)]]


#Shift Vectors
s_1 = [[0.00], #Initial Vector = [[0.00],
       [0.00]] #                  [0.00]]

s_2 = [[0.00], #Initial Vector = [[0.00],
       [1]] #                     [1.60]]

s_3 = [[0.0], #Initial Vector = [[0.00], #Shifts on the x-axis
       [1]] #                     [1.60]]

s_4 = [[0.0], #Initial Vector = [[0.00],#Shifts on the x-axis
       [0.44]] #                  [0.44]]

#Reflection Matrix
r_m = [[-1, 0],
       [ 0, 1]]

def create_fern():

    # initializing the list
    x = []
    y = []
    c = []
```

```
# setting first element to 0 and color to white
x.append(0)
y.append(0)
c.append('white')

current = 0

for i in range(1, 90000):

    # generates a random integer between 1 and 100
    z = randint(1, 100)

    # the x and y coordinates of the equations are appended in
                                  the lists respectively.

    # for the stem with the probability of 0.01
    if z == 1:
        x.append(f_1[0][0] * (x[current]) + f_1[0][1] * (y[
                                      current]) + s_1[0][0])
        y.append(f_1[1][0] * (x[current]) + f_1[1][1] * (y[
                                      current]) + s_1[1][0])
        c.append('khaki')

    # for the copies of the stem and bottom pinnas with the
                                  probability of 0.85
    if z>= 2 and z<= 86:
        x.append(f_2[0][0] * (x[current]) + f_2[0][1] * (y[
                                      current]) + s_2[0][0])
        y.append(f_2[1][0] * (x[current]) + f_2[1][1] * (y[
                                      current]) + s_2[1][0])
        c.append(c[current])

    # for the bottom pinna(left) with the probability of 0.07
    if z>= 87 and z<= 93:
        x.append(f_3[0][0] * (x[current]) + f_3[0][1] * (y[
                                      current]) + s_3[0][0])
        y.append(f_3[1][0] * (x[current]) + f_3[1][1] * (y[
                                      current]) + s_3[1][0])
        c.append('darkolivegreen')

    # for the bottom pinna(right) with the probability of 0.07
    if z>= 94 and z<= 100:
        x.append((f_4[0][0] * (x[current]) * r_m[0][0] + f_4[1][
                                      0] * (x[current]) *
                                      r_m[0][1])
            + (f_4[0][1] * (y[current]) * r_m[0][0] + f_4[1][
                                          1] * (y[current
                                          ]) * r_m[0][1])
                                           + s_4[0][0])
        y.append((f_4[0][0] * (x[current]) * r_m[1][0] + f_4[1][
                                      0] * (x[current]) *
                                      r_m[1][1])
            + (f_4[0][1] * (y[current]) * r_m[1][0] + f_4[1][
                                          1] * (y[current
                                          ]) * r_m[1][1])
                                           + s_4[1][0])
        c.append('yellowgreen')

    current = current + 1
    plt.scatter(x, y, s = 0.2, edgecolor = c)

    plt.xlim([-2.5, 2.5])
    plt.ylim([0, 10.5])

create_fern()
```

REFERENCES

1. Barnsley fern. `https://en.wikipedia.org/wiki/Barnsley_fern`, Mar 2021.
2. Fractal. `https://en.wikipedia.org/wiki/Fractal`, Apr 2021.
3. Laug. Barnsley fern. `https://en.wikipedia.org/wiki/File:Barnsley_fern_2000x2000.png`, Feb 2019.
4. U. D. of Agriculture. Fern structure. `https://www.fs.fed.us/wildflowers/beauty/ferns/structure.shtml`.
5. A. Pratt and E. Step. *The flowering plants, grasses, sedges, ferns of Great Britain and their allies, the club mosses, horsetails, etc*, volume v.1. London, F. Warne, 1905. https://www.biodiversitylibrary.org/bibliography/11595.
6. P. Singh. Barnsley fern in python. `https://www.geeksforgeeks.org/barnsley-fern-in-python/`.
7. E. W. Weisstein. Affine transformation. *`https://mathworld.wolfram.com/AffineTransformation.html`*, 2004.