Math 42, Homework 3

Evan Coons

due April 22 11:59pm

Question 1

I found the difference between state diagrams vs compartmental diagrams interesting, which puts the states in boxes instead of the just the population. Also, the assumption that the state completely clears and refills between each step is useful. Finally, it is always cool to see how useful eigenvalues and eigenvectors are.

Question 2

I found the ideas of steady state and fixed point of a recurrence relation interesting. To me, this concept is easier to understand with differential equations, but I am beginning to grasp similar ideas for recurrence relations. Taylor's' theory and the cobweb method were also cool to check out, as we did not go over them in class. Another thought is about inhomogenous equations... they were briefly mentioned, and I am wondering how you would approach it. Finally, I want to try to build a predator/prey model.

Question 3

The lack of graphical sophistication in the US compared to other countries (at least at the time of publication) was surprising to me. For some reason, graphics had a negative perception for a time. Graphical competence was low for both producers and consumers. In chapter 4, I like the idea of maximizing data-ink, but I also think creative illustrations and styles can make the data more pleasing to look at.

Question 4 - Exercise 2 in Section 1.8 of Mooney and Swift

Find a closed form solution.
We have:

$$x(n) = Rx(n-1) + a \tag{1}$$
$$\tag{2}$$

Starting with $x(1)$ to find a pattern:

$$x(1) = Rx(0) + a \tag{3}$$
$$x(2) = Rx(1) + a = R(Rx(0) + a) + a = R^2x(0) + Ra + a \tag{4}$$
$$x(3) = R^3x(0) + R^2a + Ra + a \tag{5}$$

So we find the closed form solution to be:

$$x(n) = R^n x(0) + \sum_{n=0}^{n-1} aR^n \tag{6}$$

Question 5 - Exercise 1 in Section 3.13 of Mooney and Swift

(a) First, we can make a state diagram:

$$\begin{tikzcd}
x_1 \arrow[r, "0.8"'] & x_2 \arrow[l, bend right, "0.8"'] \arrow[r, "0.9"'] & x_3 \arrow[ll, bend right=50, "3.5"']
\end{tikzcd}$$

* different from compartment diagram

(b) We can write the equations:

$$x_1(n+1) = 0.8x_2(n) + 3.5x_3(n) \tag{7}$$
$$x_2(n+1) = 0.8x_1(n) \tag{8}$$
$$x_3(n+1) = 0.9x_2(n) \tag{9}$$

As a Leslie matrix:

$$\begin{pmatrix} 0 & 0.8 & 3.5 \\ 0.8 & 0 & 0 \\ 0 & 0.9 & 0 \end{pmatrix}$$

(c) Compute eigenvalues with numpy eig() function:

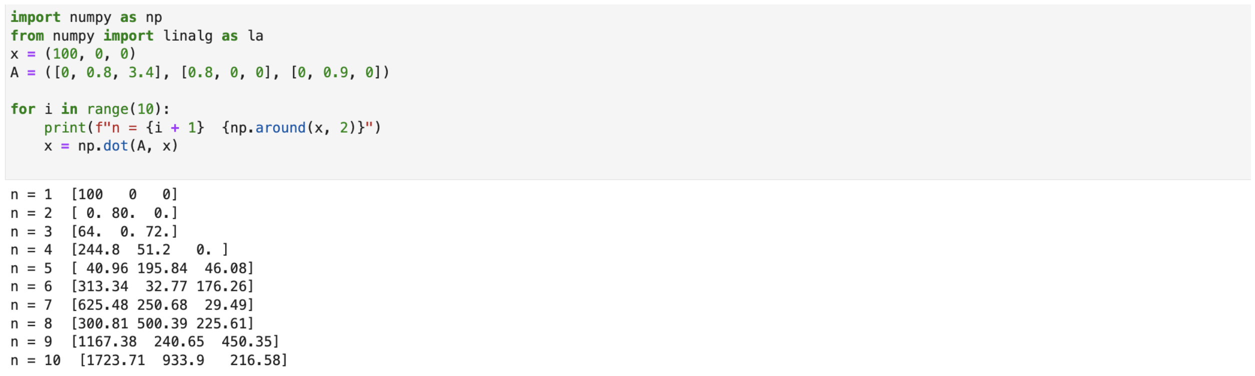
$$(1.51697, -0.758486 + 1.042066i, -0.758486 - 1.042067i)$$

The moduli of the eigenvalues are

$$1.51697, 1.288, 1.288$$

Based on the largest eigenvalue, the population will increase. It will increase at a rate of 0.51697, or 51.697%

(d) Simulation



```python
import numpy as np
from numpy import linalg as la
x = (100, 0, 0)
A = ([0, 0.8, 3.4], [0.8, 0, 0], [0, 0.9, 0])

for i in range(10):
    print(f"n = {i + 1}  {np.around(x, 2)}")
    x = np.dot(A, x)
```

```
n = 1  [100   0   0]
n = 2  [ 0. 80.  0.]
n = 3  [64.  0. 72.]
n = 4  [244.8  51.2   0. ]
n = 5  [ 40.96 195.84  46.08]
n = 6  [313.34  32.77 176.26]
n = 7  [625.48 250.68  29.49]
n = 8  [300.81 500.39 225.61]
n = 9  [1167.38  240.65  450.35]
n = 10  [1723.71  933.9   216.58]
```

(e) Finding an associated eigenvector with dominant eigenvalue with python:

$$(0.8502, 0.4518, 0.2701)$$

Normalizing eigenvector to find the population proportions:

$$(0.54080, 0.2874, 0.17181)$$

Proportion from the simulation after 10 years:

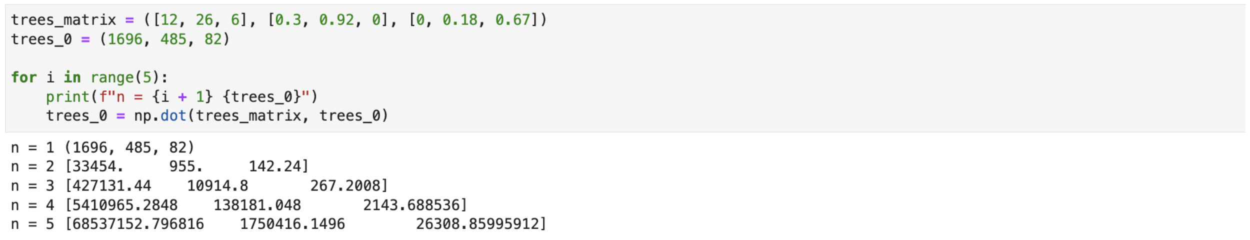
$$(0.4006, 0.37239, 0.22698)$$

Although this isn't very close, I found that the simulation was much more accurate after 20 years it just took some more iterations:

$$(0.56451, 0.27348, 0.16202)$$

Question 6 - Exercise 2 in Section 3.13 of Mooney and Swift

This model does not make sense because the trees grow way too fast and there is no upper bound. This model does not account for limits in the environment. 70 million trees in 250 years is unreasonable.

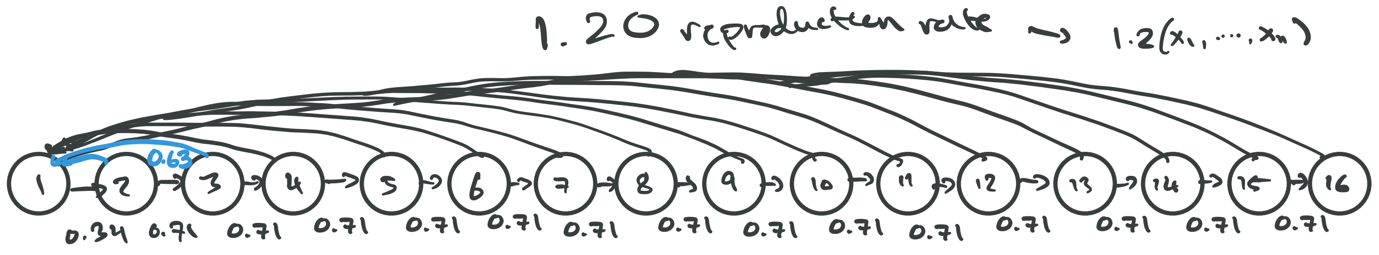


```
trees_matrix = ([12, 26, 6], [0.3, 0.92, 0], [0, 0.18, 0.67])
trees_0 = (1696, 485, 82)

for i in range(5):
    print(f"n = {i + 1} {trees_0}")
    trees_0 = np.dot(trees_matrix, trees_0)
```

```
n = 1 (1696, 485, 82)
n = 2 [33454.      955.      142.24]
n = 3 [427131.44    10914.8       267.2008]
n = 4 [5410965.2848    138181.048       2143.688536]
n = 5 [68537152.796816    1750416.1496        26308.85995912]
```

Question 7 - Project 3.1: Age Structured Bobcats in section 3.14

1. State Diagram



Leslie matrix from table 3.6:

$$\begin{pmatrix} 0.63 & 0.63 & 1.20 & 1.20 & 1.20 & \dots & 1.20 \\ 0.34 & 0 & & \dots & & & 0 \\ 0 & 0.71 & 0 & & \dots & & 0 \\ 0 & 0 & 0.71 & 0 & \dots & & 0 \\ \vdots & & & \ddots & & & \\ & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0.71 & 0 \end{pmatrix}$$

2. Dominant eigenvalue
   Eigenvalues are: (1.241, 0.639+0.291j, 0.639-0.291j, 0.468+0.512j, 0.468-0.512j, 0.24 +0.646j, 0.24 -0.646j, -0.013+0.681j, -0.013-0.681j, -0.258+0.617j, -0.258-0.617j, -0.462+0.465j, -0.462-0.465j, -0.645+0.j)

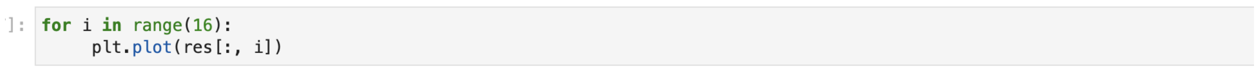
   Normalized, these are: (1.241, 0.702, 0.702, 0.694, 0.694, 0.689, 0.689, 0.681, 0.681, 0.669, 0.669, 0.655, 0.655, 0.645, 0.648, 0.648)

   So the first eigenvalue is dominant. **The dominant eigenvalue is 1.241. The population will be growing at 24.1%.**

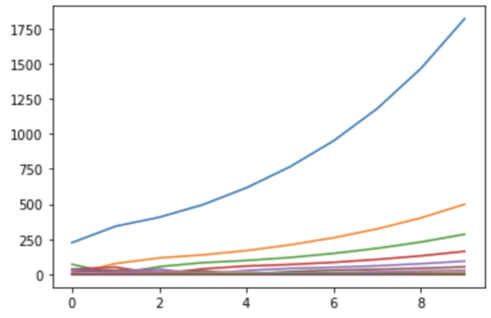
   The corresponding eigenvector is: (0.94853, 0.25978, 0.14857, 0.08497, 0.0486 , 0.02779, 0.0159, 0.00909, 0.0052 , 0.00297, 0.0017 , 0.00097, 0.00056, 0.00032, 0.00018, 0.0001)

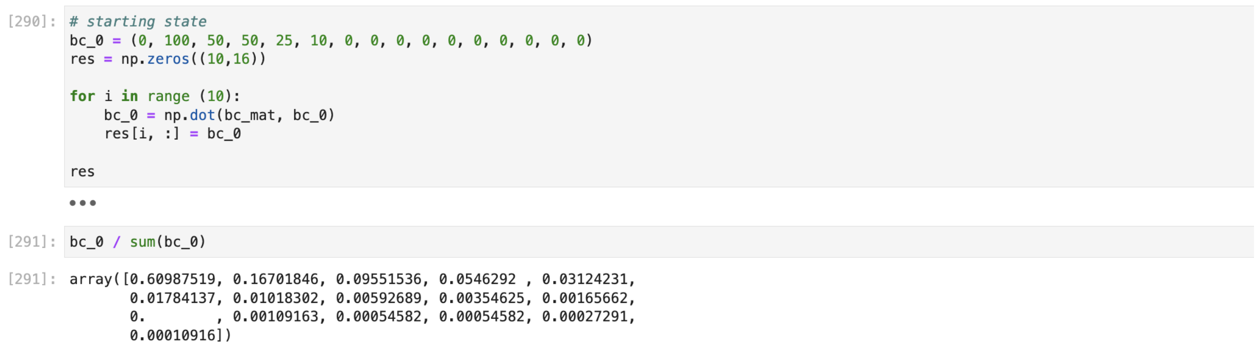
   Normalized to add up to 1: (These are the proportion of population in each state 1 to 16)
   (0.60981098, 0.16703099, 0.09553813, 0.05464832, 0.03124598, 0.01787322, 0.01022245, 0.00585059, 0.00334319, 0.00192876, 0.00109297, 0.00064292, 0.00038575, 0.00019288, 0.00012858,0.00006429)

3. Actually modeling over 10 years with starting state $(0, 100, 50, 50, 25, 10, 0, 0, \ldots 0)$
   Using python, I found:



```
]: for i in range(16):
       plt.plot(res[:, i])
```





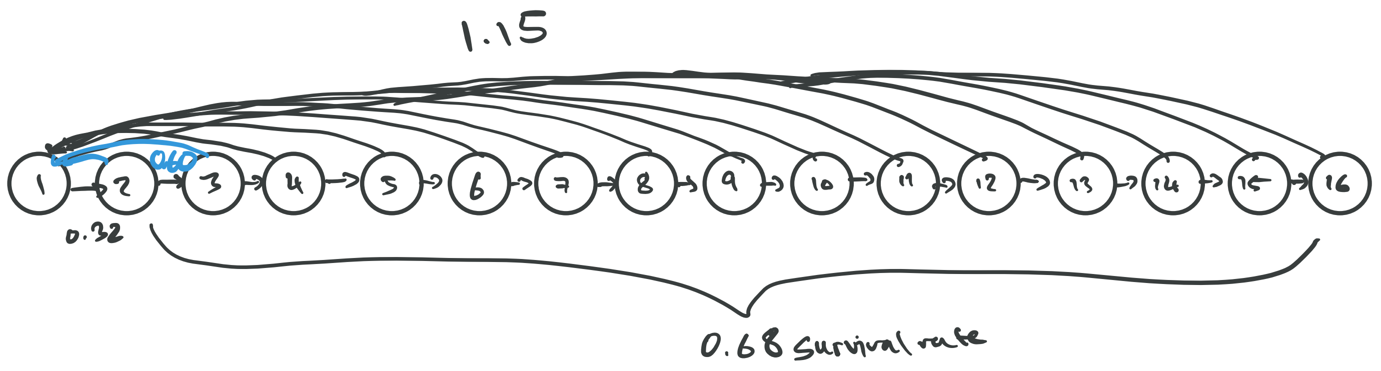
```
[290]: # starting state
       bc_0 = (0, 100, 50, 50, 25, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
       res = np.zeros((10,16))

       for i in range (10):
           bc_0 = np.dot(bc_mat, bc_0)
           res[i, :] = bc_0

       res
       •••

[291]: bc_0 / sum(bc_0)

[291]: array([0.60987519, 0.16701846, 0.09551536, 0.0546292 , 0.03124231,
              0.01784137, 0.01018302, 0.00592689, 0.00354625, 0.00165662,
              0.        , 0.00109163, 0.00054582, 0.00054582, 0.00027291,
              0.00010916])
```

   Finding the distribution in each age class with simulation after 10 years: (0.60987519, 0.16701846, 0.09551536, 0.0546292 , 0.03124231, 0.01784137, 0.01018302, 0.00592689, 0.00354625, 0.00165662, 0, 0.00109163, 0.00054582, 0.00054582, 0.00027291, 0.00010916)

   This is almost exactly the same as the eigenvector

4. Now, doing all of this for the worst case data:

   State Diagram:



   Leslie Matrix:

$$\begin{pmatrix} 0.6 & 0.6 & 1.15 & 1.15 & 1.15 & \dots & 1.15 \\ 0.32 & 0 & & \dots & & & 0 \\ 0 & 0.68 & 0 & & \dots & & 0 \\ 0 & 0 & 0.68 & 0 & \dots & & 0 \\ \vdots & & & \ddots & & & \\ & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0.68 & 0 \end{pmatrix}$$

   Using numpy eig() function, I found:

   Eigenvalues:
   (1.183, 0.612+0.279j, 0.612-0.279j, 0.448+0.491j, 0.448-0.491j, 0.23 +0.618j, 0.23 -0.618j, -0.012+0.652j, -0.012-0.652j, -0.246+0.59j , -0.246-0.59j , -0.442+0.445j, -0.442-0.445j, -0.617+0.j, -0.572+0.239j, -0.572-0.239j)

   Normalized eigenvalues (moduli):
   (1.183, 0.672, 0.672, 0.664, 0.664, 0.66 , 0.66 , 0.652, 0.652, 0.639, 0.639, 0.627, 0.627, 0.617, 0.62 , 0.62)

   The dominant eigenvalue is 1.183. The population will be growing at 18.3%

   A corresponding Eigenvector is:
   (0.9494508, 0.25685741, 0.14766266, 0.08488858, 0.0488009, 0.02805475, 0.01612816, 0.00927179, 0.00533018, 0.00306423, 0.00176157, 0.00101269, 0.00058218, 0.00033468, 0.0001924, 0.00011061)

   Normalizing to add up to 1:
   (0.61116743, 0.16534073, 0.09505138, 0.05464331, 0.03141344, 0.01805902, 0.0103818 , 0.00596831, 0.00343107, 0.00197246, 0.00113393, 0.00065188, 0.00037475, 0.00021544, 0.00012385, 0.0000712)
   This should represent the population proportion in each class 1 through 16:

   Doing the simulation for ten years, starting with the same conditions, the ending population proportions are:
   (0.61114526, 0.16532268, 0.09503549, 0.05463641, 0.03140787, 0.01802827, 0.01034418, 0.00605416, 0.00363864, 0.001702 , 0. , 0.00114257, 0.00057129, 0.00057129, 0.00028564, 0.00011426)

This ten year simulation results in a population distribution that is almost exactly the same as the normalized eigenvector.

Question 8 - Project 1.3: Blood Alcohol Model in Section 1.9

In California, the legal limit for BAC is 0.08 or 0.8g/100mL. I will choose sex and weight to be male and 150 pounds (68 kg) and time step to be 1 min.

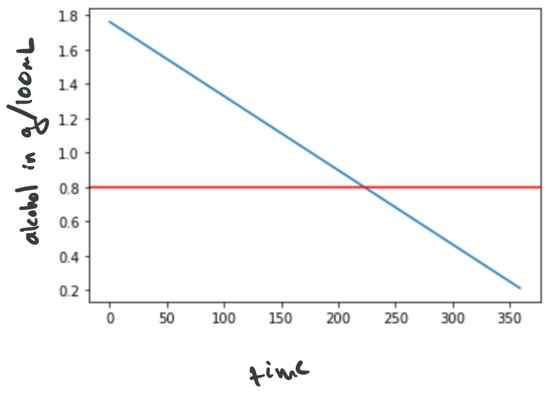
I will also assume

- There are 0.68(68.04) = 46.3 Liters of Blood in 150lb male.
- The human body eliminates 12g alcohol per hour = 0.2g per minute.

1. Plotting alcohol vs time after 6 beers. This hypothetical can legally drive home after 223 minutes. Almost 4 hours! The red line represents the legal driving limit.

```python
x_0 = 13.6 * 6 / 46.3
X = []
for i in range(360):
    X.append(x_0)
    total_grams = x_0 * 46.3
    total_grams -= 0.2
    x_0 = total_grams/46.3
```

```
124]:
```
```python
import matplotlib.pyplot as plt
plt.plot(X)
plt.axhline(y = 0.8, color = "red")
plt.show()
```



2. Alcohol concentration vs time, but with a more reasonable drinking rate. assuming:

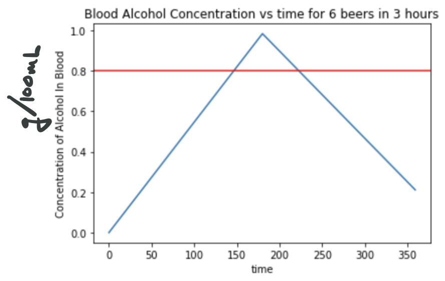
    - 2 beers are consumed per hour for 3 hours.

```
# one beer every 30 minutes
# rate is in minutes per beer (13.6 grams)
```

```
X1 = []
BAC = 0
for i in range(360):
    X1.append(BAC)
    if(i < 180): # if less than three hours, assuming it will take 3 hours to drink 6 beers.
        BAC = BAC + 13.6 / blood_liters * (1 / 30)
    BAC = BAC - 0.2 / blood_liters
```

```
plt.plot(X1)
plt.axhline(y = 0.8, color = "red")
plt.ylabel("Concentration of Alcohol In Blood")
plt.xlabel("time")
plt.title("Blood Alcohol Concentration vs time for 6 beers in 3 hours")
```

```
Text(0.5, 1.0, 'Blood Alcohol Concentration vs time for 6 beers in 3 hours')
```



(The rest is on next page)

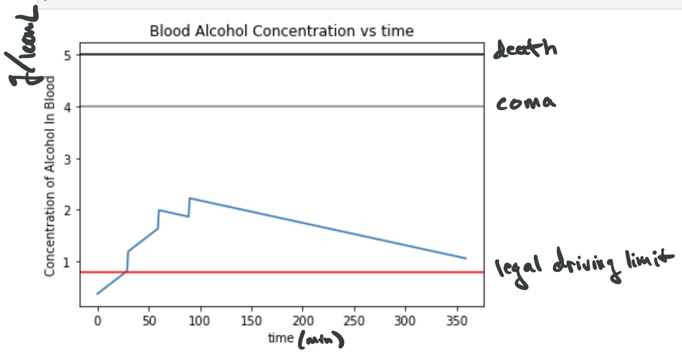
3. For the third model, I made a function that takes the amount of shots and beers and their frequency (in minutes per beer or shot) and plots blood alcohol content. I assumed:

   - a constant rate of consumption for beer
   - an instant consumption for shots

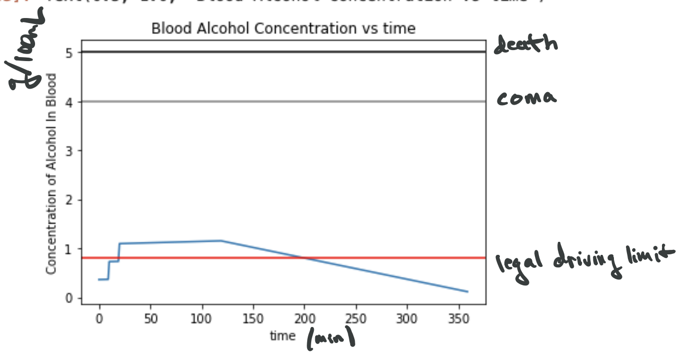
```
28]: def drinking(i, shots, shots_freq, beer, beer_freq):
         add_alcohol = 0
         if i % shots_freq == 0 and shots > (i / shots_freq):
             add_alcohol = float(16.7 / blood_liters)
         if beer > (i / beer_freq):
             add_alcohol += float(13.6 / blood_liters * (1 / beer_freq))
         return add_alcohol
```

```
[ ]: def simulate_drinking(shots, shots_freq, beer, beer_freq):
         X2 = []
         BAC = 0
         for i in range(360):
             alc = drinking(i, shots, shots_freq, beer, beer_freq)
             BAC += alc
             BAC -= 0.2 / blood_liters
             X2.append(BAC)
         return X2
```

```
[ ]: plt.plot(simulate_drinking(4, 30, 4, 15))
     plt.axhline(y = 0.8, color = "red")
     plt.axhline(y = 4.0, color = "gray")
     plt.axhline(y = 5.0, color = "black")
     plt.ylabel("Concentration of Alcohol In Blood")
     plt.xlabel("time")
     plt.title("Blood Alcohol Concentration vs time")
     plt.show()

     plt.plot(simulate_drinking(3, 10, 2, 60))
     plt.axhline(y = 0.8, color = "red")
     plt.axhline(y = 4.0, color = "gray")
     plt.axhline(y = 5.0, color = "black", label = "death")
     plt.ylabel("Concentration of Alcohol In Blood")
     plt.xlabel("time")
     plt.title("Blood Alcohol Concentration vs time")
```



```
53]: Text(0.5, 1.0, 'Blood Alcohol Concentration vs time')
```


Blood Alcohol Concentration vs time
g/100mL
Concentration of Alcohol In Blood
death
coma
legal driving limit
time (min)