EE194/BIO196: Modeling Biological Systems
HW 2: Population growth with arrays

Overview

In this homework, we will take HW #1 a bit further. Our basic biology will not change. However, instead of using 8 individual variables to model a population's vital rates, we will use two arrays. By doing this, we will gain two big advantages:

- The code will be simpler. Going from 8 variables down to only two arrays will make the code shorter and (hopefully) easier to read.
- The code will be more easily extensible. We can now switch from 4 age-based stages to as many as we like, with minimal change to our software.

HW #2 will have two parts. For the first part, we will keep the same vital rates from HW #1, and merely change our code from using individual variables to using arrays. In principle, we should get the same answers as we did in HW #1, which will let us easily validate that our new code works.

The second part will move to a more complex animal: desert tortoises. They can live to over 75 years old; we will model 75 different $l_x m_x$ pairs, grouped into six different life stages. A model this complex is only feasible with arrays – but since our code now uses arrays, it will be easy ☺.

In this homework, as in homework #1, fractional individuals are fine; no need to round anything.

Population growth with arrays

We discussed this in class. However, some quick reminders may still be useful. Consider our population from HW #1:

- They can be reasonably grouped into four categories: 0-1 years old, 1-2 years old, 2-3 and 3-4 years old. Any animal that reaches their fourth birthday immediately dies.
- We will model only the females.

How can we store our vital rates and population numbers with arrays?

- We will describe birth rate with one array *m*. It will have four elements: *m*[0], *m*[1], *m*[2] and *m*[3]. These correspond to m_0, m_1, m_2 and m_3 from HW #1. As with HW #1, we model births as a pulse immediately upon entering a new age group, and just before counting the population. We will always have *m*[0]=0.
- We will describe survival rates with another four-element array *p*. Again, *p*[0], *p*[1], *p*[2] and *p*[3] will correspond to p_0, p_1, p_2 and p_3 from HW #1. *p*[3] will always be zero.
- You should also have a population array *n*. Like the other arrays, *n* will have four elements; e.g., *n*[0] will always hold the current number of females between 0 and 1 year old, and *n*[3] the current number between 3 and 4 years old.

All good so far. Our vital rates are stored in two arrays, and our population size is in one array. No matter how many age categories we use, this will not change – only the size of the arrays will change, which doesn't really affect our code. But it's one thing to store our data; we still have to simulate it. How can we do that with arrays?

- We can compute the flow of individuals from one age class to the next over the course of the year by using *element-by-element* products. So, instead of saying *n1_next=n0*p0*, *n2_next=n1*p1* and *n3_next=n2*p2*, we can simply say *n_next*[1:4] = *n*[0:3]**p*[0:3].
- We can compute births in one fell swoop also, using a *dot product*. Instead of saying *n0_next = n0*p0*m1 + n1*p1*m2 + n2*p2*m3*, we can say *n_next*[0]=(*n*[0:3]**p*[0:3]).dot (*m*[1:4])

Problem #1

Repeat problems #1, #2 and #3 from HW #1 (but this time using arrays) and check that you get the same answers as you did in HW #1. Do this in a file HW2_problem1.py.

Problem #2.

Desert turtles live much longer than 4 years. We can break their life span into several periods:

- young juveniles, from ages 0-6. Birth rate=0, survival rate=.76
- older juveniles, from ages 7-14. Birth rate=.42, survival rate=.84
- young adults, ages 15-27. Birth rate=3.5, survival rate=.92
- mid-age adults, ages 28-52. Birth rate=4.3, survival rate=.95
- older adults, ages 53 and up. Birth rate=4.8, survival rate=.96

Even though there are five age groups, your simulation should work with full detail, and track all 75 different ages individually. That is, your arrays *m*, *p* and *n* should all be 75 entries. You must thus take the data above and distribute it; i.e., *m*[0] through *m*[6] should all be 0, *m*[7] through *m*[14] should all be .42, and so on until you set *m*[53] through *m*[74] all to 4.8. The same goes for your *p* array. You should do this efficiently using just a few lines of code, rather than assigning all 75 elements one by one.

Start with an initial population of no young juveniles, 100 older juveniles, 200 young adults, 400 mid-age adults and 500 older adults. (How did the population get so skewed? Perhaps the population was visited by a disease for which individuals build up more immunity with age). Just as with *m* and *p*, your *n* array should have 75 entries, and you should spread the initial population evenly. So for, e.g., older juveniles, you should set the eight entries *n*[7] through *n*[14] all to be 100/8=12.5.

After every year's simulation, you should report the population in each of the five categories, as well as the overall total population. You might consider using the *sum* function on array slices to make this easy. To help you check your code, after the first year you should have roughly 4564, 73, 180. 379 and 473 individuals in the five age categories.

For problem #2, you should turn in a file HW2_problem2.py.

Discussion questions:

1. Check your answer for problem #1 manually. Did the computer get the right answers?
2. For problem #2, we are keeping track of every 1-year age group. Why might we want to do this, even though we only have the vital rates for 5 age groups?
3. After the first five years, a biologist may want to validate the model by counting turtles in the wild and comparing the measured data to the model predictions. According to your model, after 5 years has the total turtle population increased or decreased?

4. In the wild, it is quite difficult to capture and count young juveniles. Thus, any data collected in the wild will have separate numbers for each of the five age categories but will not have any data for young juveniles. Assuming that the model is correct, and that the wild population exactly matches the model, then will the year-0 vs. year-5 counts in the wild (neither of which include young juveniles) show an increase or decrease in the total counted population?
5. If the modeled total population is increasing, but the counted population in the wild is decreasing, this might reasonably cast doubt on our model's validity. How might you argue the model's validity to a doubter, based on the collected data?

Extra credit

- One common goal of a population biologist is to try and increase the growth of an endangered population. Given that we always have limited financial resources, we often try to decide which vital rate, if increased, would most help the population. We call this *sensitivity analysis*. One simple way to find our model's sensitivity to a given vital rate is by running two simulations: once with the standard vital rates, and again by increasing the given vital rate by, say, 5%. Write a loop to do this for every one of the desert-turtle survival rates, and then see which is the most effective.
- Note that with the limited programming tools we've learned so far, this can get unwieldy quite quickly. Later on we will learn about *functions*, which will simplify the task.

Logistics:

- Use any lab PC system to write your code. You may use your own laptop if you prefer.
- The due date for this assignment is on the class calendar
- Submit your project at https://www.ece.tufts.edu/ee/194MSO/provide.cgi, which is also accessible from the course web page. You should turn in HW2_problem1.py, HW2_problem2.py and discussion.pdf (or whatever other format you use). If you did the extra-credit problem, turn that in as HW2_EC.py.