MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.01—Introduction to EECS I
Fall Semester, 2007

Lecture 4 Notes

Introduction to Difference Equations

Note: There are some questions interspersed with these notes. You don't have to do or turn them in; we just thought they might help you think about the material more deeply.

This week and next, we're going to explore difference equations in some depth, looking at how to use them to model real-world situations and learning how to understand the systems they describe with greater depth and insight. In what follows, we remind you of the notion of a sequence, primarily to introduce notation, and then discuss the solution of a particular subset of difference equations, referred to as homogenous difference equations.

Sequences

Sequences are a set of numbers indexed by an integer. Sequences can be good ways of representing data in variety of applications, including digital audio, sampled-data control, stock prices, bank account balances, even computational complexity. The robot we are using in lab naturally generates sequences. Since the step function for the robot is executed at regular intervals (roughly every tenth of a second), a sequence can be generated by recording the robot position during every call of a step function.

Sequences are frequently graphed using "lollypop" or stem plots. For example, below is a stem plot (created using MATLAB) for 200 samples of a sequence generated by roughly twenty milliseconds of a sound sampled eight thousand times a second. Sequences can be specified explicitly, such as in the well-know power sequence:

$$y(n) = \alpha^n.$$

In the above representation of the power sequence, we have used one of the standard notations for indicating members of a sequence, though the notation is far from universal. Throughout this course we use the notation that the indices for elements of a sequence appear inside a pair of parentheses. We use this notation partly as a reminder that a sequence can also be thought of as a function which takes integer arguments. As a more specific example, for the power sequence with $\alpha = \frac{1}{2}$, members of this sequence would be

$$\{..., y(-3) = 8, y(-2) = 4, y(-1) = 2, y(0) = 1, y(1) = \frac{1}{2}, y(2) = \frac{1}{4}, ...\}$$

In the above simple example, the sequence was indexed with both positive and negative integers. For applications where the sequence is generated by time samples, then one is usually interested in one-sided sequences, indexed by non-negative integers.

Sequences can often be represented implicitly, where an equation is given which relates one element of a sequence to other elements. For example, the elements of a power sequence satisfy

$$y(n + 1) = \alpha y(n)$$

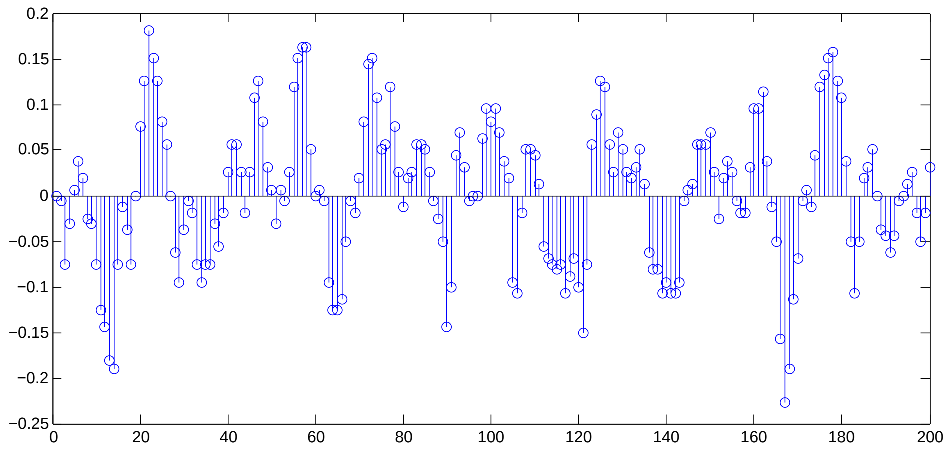

Figure 1: Stem plot of 200 samples of a sound (roughly 20 milliseconds)

and this form is usually referred to as a difference equation and requires addition information to completely specify $y(n)$. For this case, one must a specific value for $y(n)$ for some value of n.

As an example, if $y(0) = 1$ and $\alpha = \frac{1}{2}$ then the solution to the above difference equation is

$$y(n) = \left(\frac{1}{2}\right)^n$$

Another common sequence which has a very natural difference equation representation is the well-known Fibonacci sequence, in which the next element is the sequence is the sum of the previous two. More specifically

$$y(n) = y(n-1) + y(n-2).$$

Since the next entry in the Fibonacci sequence depends on two prior values of the sequence, in order to precisely determine the Fibonacci sequence from the above difference equation, it is necessary to specify two values of in the sequence. In the case of the classical Fibonacci numbers, $y(0) = 0$ and $y(1) = 1$. Then, using the difference equation it is easily determined that

$$\{y(2) = 1, y(3) = 2, y(4) = 3, ...\}.$$

First-Order Homogenous Linear Difference Equations

In general, a difference equation, together with some initial conditions, defines an infinite sequence of values, $y(0), \ldots$. A difference equation is a k*th order* difference equation when $y(n)$ depends only on the k previous values of the sequence y. That is,

$$y(n) = g(y(n-1), \ldots, y(n-k)) \ \ ,$$

for some function g. This class of equations is very big (g could involve products of the previous values, or exponentials or trig functions), and difficult to analyze.

If we restrict our attention to homogenous *linear* difference equations, which have the form

$$y(n) = \alpha_1 y(n-1) + \ldots + \alpha_k y(n-k)) \ \ ,$$

for real constant values $\alpha_1, \ldots, \alpha_k$, then we limit the class considerably, and also make them relatively easy to analyze. The above equation is referred to as homogenous because $y(n)$ is a weighted sum of other values of the sequence y, and will be zero if all the other values of y are zero. As you have probably already noticed, linear problems are fundamentally simpler and easier to work with than most others. Because of this, we would often prefer to make an easily analyzed linear model of a real-world situation, even if the model doesn't fit perfectly, than to make an accurate non-linear model that is difficult to analyze.

The simplest homogenous linear difference equations are *first order*; they have the form

$$y(n) = \alpha_1 y(n-1) \ \ ,$$

and require stipulation of $y(0)$ (which we'll also call v_0, for short).

There are many ways to calculate the values of a difference equation, so if we wanted to know $y(n)$, we could compute it relatively easily by starting with the initial values and propagating forward. Such a method is referred to as "plug and chug". Now we'll see how to derive a *closed form* definition of $y(n)$ that can be computed more directly. In the first-order case, we can do it pretty much by inspection. Let's start expanding the definition:

$$\begin{aligned} y(n) &= \alpha_1 y(n-1) \\ &= \alpha_1(\alpha_1 y(n-2)) \\ &= \alpha_1(\alpha_1(\alpha_1 y(n-3))) \\ &= \alpha_1(\alpha_1(\alpha_1 \ldots y(0))) \\ &= \nu_0 \alpha_1^n \end{aligned}$$

Question 1. If your computer didn't have exponentiation defined as a primitive, what would be the running time of the fastest algorithm you know for computing $y(n)$, in Θ notation?

If all we got out of this maneuver was a somewhat faster algorithm, we'd be happy; but in fact, we'll also get a considerably deeper understanding of the sequence that's being defined by the difference equation.

Example 1 *Your great aunt Zelda put $100 into an interest-bearing bank account 30 years ago and forgot about it. The account made 5% interest, compounded annually. She just died, and you inherited the account. How much did you get?*

We can model the balance in Aunt Zelda's bank account as a FOLDE (first-order linear difference equation), where n indexes the number of years since Zelda opened the account:

$$y(n) = y(n-1) + 0.05y(n-1) = 1.05y(n-1) \ ,$$

with $\nu_0 = 100$. So, from the analysis above, we know that $y(30) = 100 \cdot 1.05^{30} \approx 432$. So, you inherited about $432. Not too bad. Sadly, Zelda died tragically young. How much would you have gotten if Zelda had lived another 30 years? $y(60) = 100 \cdot 1.05^{60} \approx 1868$. We can see the exponential growth here, even though the exponent is pretty small.

Example 2 *My great uncle Oswald put $100 into a management-fee-bearing bank account 30 years ago and forgot about it. The account paid a 5% fee, deducted annually. He just died, and I inherited the account. How much did I get?*

Uncle Oswald's bank account is also modelable as a FOLDE in the same way, but with an exponent of 0.95:

$$y(n) = y(n-1) - 0.05y(n-1) = 0.95y(n-1) \ .$$

So, we know that the amount I stand to inherit is $y(30) = 100 \cdot 0.95^{30} \approx 21$. Sigh.

We can see that the value of α_1 governs the *qualitative* behavior of these systems:

- If $\alpha_1 > 1$, the values increase without bound, and we say that the system *diverges.*
- If $0 < \alpha_1 < 1$, the values decrease to 0, and we say that that the system *converges monotonically.*
- If $\alpha_1 = 1$, then the system just maintains its initial value.

What if α is negative? It's a little bit harder to come up with natural first-order systems that behave this way, but it is still useful to study the math. Let's consider two systems:

$$\begin{aligned} y(n) &= \frac{1}{2}y(n-1) \\ g(n) &= -\frac{1}{2}g(n-1) \end{aligned}$$

When the root, α_1, is negative, as it is in $g(n)$, then the value oscillates between negative and positive values.

- If $\alpha_1 < -1$, the absolute values increase without bound, and we say that the system diverges and oscillates.
- If $-1 < \alpha_1 < 0$, the values decrease to 0, and we say that that the system converges and oscillates.
- If $\alpha_1 = -1$, then the system maintains its initial absolute value, but oscillates between v_0 and $-v_0$.

Example 3 *Consider a robot that is driving toward a wall, and should stop when it is at a distance 0 from the wall. If it is programmed so that, on step* n *of its program, it moves at a velocity proportional to* $d(n-1)$*, the distance to the wall at the previous step, then we can model the behavior of the system as*

$$\begin{aligned} d(n) &= d(n-1) + \delta_t v(n-1) \\ &= d(n-1) + \delta_t \gamma d(n-1) \\ &= (1 + \delta_t \gamma) d(n-1) \ , \end{aligned}$$

where δ_t *is the amount of time that passes between steps. The constant of proportionality,* γ*, is under our control. How should we choose it to ensure that the system is stable?*

In order for it to converge, we need to be sure that the coefficient $\alpha_1 = 1 + \delta_t\gamma$ has magnitude less than 1. Furthermore, since we want it to converge monotonically (oscillating would mean that it would bang into the wall), we need α_1 to be positive. So, that means we need to choose

$$\begin{array}{rcl} 0 < & 1 + \delta_t\gamma & < 1 \\ -1 < & \delta_t\gamma & < 0 \\ -\frac{1}{\delta_t} < & \gamma & < 0 \end{array}$$

Question 2. What happens if we set γ very near but < 0?

Question 3. What happens if we set γ very near but $> -1/\delta_t$?

Second-Order Linear Difference Equations

Now, let's consider *second order* equations, in which the value at time step n depends on two previous values: they have the form

$$y(n) = \alpha_1 y(n-1) + \alpha_2 y(n-2) \ ,$$

and we'll find that in order to completely specify their behavior, we'll need to specify two initial conditions, typically $y(0)$ and $y(1)$ (called v_0 and v_1, for short).

It's hard to see what the form of the closed-form solution might be, just by inspecting the difference equation. We do know that first-order difference equations have a solution of the form

$$y(n) = c\lambda^n$$

and we are going to guess that second-order difference equations also have solutions of the same form, though we will allow c and λ to be complex.[1] Have patience, and let's see what the consequences of this guess are.

If we plug that form into the original difference equation, we get:

$$c\lambda^n = \alpha_1 c\lambda^{n-1} + \alpha_2 c\lambda^{n-2} \ .$$

We can simplify this by dividing through by $c\lambda^{n-2}$, to get the quadratic equation

$$\lambda^2 = \alpha_1\lambda + \alpha_2 \ .$$

That seems pretty simple.

Now, we can solve that quadratic equation, which is called the *characteristic equation*, finding that there are generally two roots,

$$(\lambda_1, \lambda_2) = \frac{\alpha_1 \pm \sqrt{\alpha_1^2 + 4\alpha_2}}{2} \ .$$

Question 4. Convince yourself we haven't made a sign error above.

So, what does this mean? Well, by construction, it means that for any constant c, the sequence specified by $y(n) = c\lambda_1^n$ will satisfy the difference equation at every value of n. And, so, too, will $y(n) = c\lambda_2^n$. We seem to have gotten more than we bargained for: we have two whole *families* of sequences that satisfy the equation we were given. Furthermore, if $y(n) = c\lambda_1^n$ and $y(n) = c\lambda_2^n$ both satisfy the equation, then so does $y(n) = c_1\lambda_1^n + c_2\lambda_2^n$.

That any linear combination of solutions is also a solution is a very property of some systems, and is called the *principle of superposition*. In this case, superposition holds due to the linearity of the homogenous difference equation.[2] We can test superposition by plugging it into the original equation:

$$\begin{aligned} y(n) &= \alpha_1 y(n-1) + \alpha_2 y(n-1) \\ c_1\lambda_1^n + c_2\lambda_2^n &= \alpha_1(c_1\lambda_1^{n-1} + c_2\lambda_2^{n-1}) + \alpha_2(c_1\lambda_1^{n-2} + c_2\lambda_2^{n-2}) \ . \end{aligned}$$

But we know that the term involving λ_1 on the left side is equal to the sum of the two λ_1 terms on the right side, and the same for the λ_2 terms; so this equation is satisfied.

Just to get an idea how this goes, let's try it on the difference equation for the Fibonacci numbers:

$$y(n) = y(n-1) + y(n-2) \ .$$

[1] "Complex!" we hear you say. "That seems like a crazy guess." Yes; we'll see how it happens in just a bit.

[2] It is important to note that this property only holds for the solution families, and not after we incorporate the initial conditions.

In this case $\alpha_1 = \alpha_2 = 1$. So, when we solve for the roots, we get

$$(\lambda_1, \lambda_2) = \frac{1 \pm \sqrt{5}}{2} \quad ,$$

and so we know that any sequence of the form

$$y(n) = c_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^n \quad ,$$

will satisfy the difference equation.

In order to get the Fibonacci numbers, we need to find the particular values of c_1 and c_2 that cause $y(0) = 0$ and $y(1) = 1$. We start by seeing what constraint we can get from the first initial value:

$$\begin{aligned} y(0) &= 0 \\ c_1 \left(\frac{1+\sqrt{5}}{2}\right)^0 + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^0 &= 0 \\ c_1 + c_2 &= 0 \\ c_1 &= -c_2 \ . \end{aligned}$$

Now, for the second constraint:

$$\begin{aligned} y(1) &= 1 \\ c_1 \left(\frac{1+\sqrt{5}}{2}\right)^1 + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^1 &= 1 \\ c_1 \frac{1+\sqrt{5}}{2} - c_1 \frac{1-\sqrt{5}}{2} &= 1 \\ c_1((1+\sqrt{5}) - (1-\sqrt{5})) &= 2 \\ c_1 2\sqrt{5} &= 2 \\ c_1 &= \frac{1}{\sqrt{5}} \end{aligned}$$

So, the final equation for the Fibonacci numbers is:

$$y(n) = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n \quad .$$

Evaluating the numeric constants, this comes out to approximately

$$y(n) = 0.44721 \cdot 1.618^n - 0.44721 \cdot (-0.618)^n \quad .$$

Question 5. Are there values of $y(0)$ and $y(1)$ such that

$$y(n) = c_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

converges to 0?

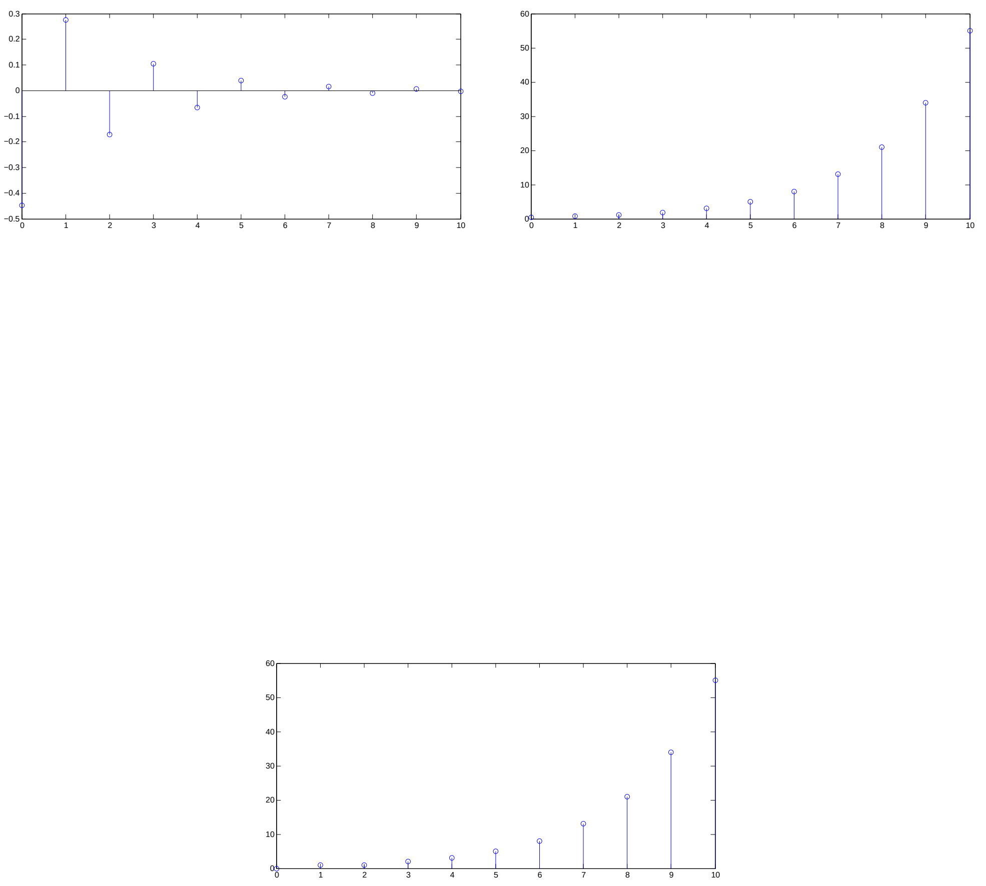

Figure 2: Plots of the two components of the Fibonacci sequence, and then their sum. The component in the leftmost graph is due to the negative root, and we can clearly see its oscillation, but note that the magnitude is very small. The third sequence is the sum of the first two, and we can see that the large exponential sequence due to the positive root overwhelms the other one, and yields the familiar fibs.

Imaginary Roots

If we examine this difference equation

$$y(n) = y(n-1) - y(n-2) \quad ,$$

which is only one minus-sign different from the equation for the Fibonacci numbers, things turn out quite a bit differently. The roots are

$$(\lambda_1, \lambda_2) = \frac{1 \pm \sqrt{-3}}{2} \quad .$$

You might worry that the fact that the roots are complex will cause us trouble. In fact, things go surprisingly smoothly.

Our initial values are real, and all subsequent values are sums of previous values, so they have to be real, too. So, the only concern might be that we won't be able to find c_1 and c_2 to make the equation have the right initial values. But that goes okay, too. If our initial conditions are $y(0) = 0$ and $y(1) = 1$ as before, then we still have that $c_1 = -c_2$. Now, for the second constraint:

$$\begin{aligned} y(1) &= 1 \\ c_1\left(\frac{1+\sqrt{-3}}{2}\right)^1 + c_2\left(\frac{1-\sqrt{-3}}{2}\right)^1 &= 1 \\ c_1 &= \frac{1}{\sqrt{-3}} \end{aligned}$$

So, the final equation for this sequence is:

$$y(n) = \frac{1}{\sqrt{-3}}\left(\frac{1+\sqrt{-3}}{2}\right)^n - \frac{1}{\sqrt{-3}}\left(\frac{1-\sqrt{-3}}{2}\right)^n \quad ,$$

which turns out always to be real valued.[3]

To get some idea for what's going on there, start by observing that either λ_1 and λ_2 are both real, or λ_1 and λ_2 are complex conjugates. And you can probably see that if the λ's are complex conjugates, c_1 and c_2 will also be complex conjugates of one another (clearly true in our example above, but it's true even when $y(0)$ is not 0). The result from multiplying and summing is that all the imaginary terms end up cancelling, and only real terms remain.

Representing complex numbers

It will turn out that sometimes it's easier to think about a complex number $a + bj$ instead as $Me^{j\theta}$, where M is the *magnitude* of the number (sometimes written as $|a + bj|$), $\sqrt{a^2 + b^2}$, and θ is its *angle*, arctan(b/a). So, if we think of (a, b) as a point in the complex plane, then M, θ is its representation in polar coordinates.

This representation is justified by Euler's equation

$$e^{xi} = \cos x + j \sin x \quad ,$$

[3]Of course, if you compute this numerically, you'll end up with little residual imaginary parts; you might need to do `val.real` in Python to just get the real part after the computation.

which can be directly derived from series expansions of e^z, $\sin z$ and $\cos z$. To see that this is reasonable, let's take our number, represent it as a complex exponential, and then apply Euler's equation:

$$\begin{aligned} a + bj &= Me^{j\theta} \\ &= M\cos\theta + jM\sin\theta \\ &= a + bj \end{aligned}$$

So what? There are some operations on complex numbers that are much more straightforwardly done in the exponential representation. In particular, let's consider raising a complex number to a power. In the cartesian representation, we get big hairy polynomials. In the exponential representation, we get, in the quadratic case,

$$\begin{aligned} \left(Me^{j\theta}\right)^2 &= Me^{j\theta}Me^{j\theta} \\ &= M^2e^{j2\theta} \end{aligned}$$

More generally, we have that

$$\left(Me^{j\theta}\right)^n = M^n e^{jn\theta} \quad ,$$

which is much tidier. This is an instance of an important trick in math and engineering: changing representations. We will often find that representing something in a different way will allow us to do some kinds of manipulations more easily. There is no *one* best representation; they are better in different circumstances.

So, what happens as n tends to infinity? The first factor, M^n, will grow or shrink depending on the magnitude of M. And the second factor will just cause the vector to rotate around and around the origin; but because we're always taking the sin and cos of $n\theta$, the contribution of that factor won't increase or decrease substantially as n increases. This means that we can just think about the magnitude when we want to understand the qualitative behavior of the sequence.

Qualitative Behavior

So, finally, what happens when we have two roots λ_1 and λ_2. No matter whether they're real or complex, positive or negative, the main issue is whether their magnitude is greater than 1. (For complex roots, this means that they are outside the unit circle in the complex plane.) If so, the sequence will diverge. If it is less than 1, it will converge to 0. If it is equal to 1, it will converge to some non-zero value (in the case of real roots), or converge to a sinusoidal oscillation (in the case of complex roots).

General case

The same basic story holds for linear difference equations of any order k:

$$y(n) = \alpha_1 y(n-1) + \ldots + \alpha_k y(n-k) \quad .$$

You derive the characteristic equation:

$$\lambda^n = \alpha_1\lambda^{n-1} + \ldots + \alpha_k\lambda^{n-k} \quad ,$$

and solve it to get roots $\lambda_1, \dots, \lambda_k$. Then, using k initial conditions, you solve the set of k linear equations in k unknowns:

$$\begin{aligned} y(0) &= c_1 + \dots + c_k \\ y(1) &= c_1\lambda_1 + \dots + c_k\lambda_k \\ &\dots \\ y(k) &= c_1\lambda_1^k + \dots + c_k\lambda_k^k \end{aligned}$$

to find values for $c_1, \dots, c_k$. And then, finally, you have a closed form of

$$y(n) = c_1\lambda_1^n + \dots + c_k\lambda_k^n \ .$$

Well, it's not always so smooth. First of all, there's the issue of finding all the roots. For third and maybe fourth order polynomials it's not too hard, but after that you need to do it numerically and it can be difficult to find them all. Second, it can happen that the same root will be repeated (for instance, $y(n) = 4y(n-1) - 4y(n-2)$ will generate the root 2, twice). This doesn't happen very much, and it gets a little tricky when it does. You can learn more about this in 6.003!

Coyotes and Roadrunners

Imagine a world populated with coyotes and roadrunners. Roadrunners eat grass and coyotes eat roadrunners (when they can catch them). The roadrunner population naturally increases, but when there are coyotes around, they eat the roadrunners and decrease the coyote population. The coyote population, in the absence of roadrunners, tends to decrease, but when there are roadrunners around to eat, they increase.

We can model this system as a set of coupled linear difference equations (though a real population biologist would probably feed us to the coyotes for the naivete of the model). Let $c(n)$ be the number of coyotes in Acme valley in year n, and let $r(n)$ be the number of roadrunners. Then, we might say that

$$\begin{aligned} c(n) &= \gamma_1 c(n-1) + \gamma_2 r(n-1) \quad (1) \\ r(n) &= \gamma_3 r(n-1) + \gamma_4 c(n-1) \quad (2) \end{aligned}$$

Imagine that Acme valley was stocked, in year 0, with 100 roadrunners and 10 coyotes. We'd like to know how many roadrunners and coyotes it will have in years to come.

In order to do this, we need to arrange things so that we have one equation that's about either coyotes or roadrunners only. We can do that through some fairly straightforward algebraic manipulations. Let's start with equation 1, and see if we can make it define the roadrunner population in some year, depending on the coyote populations in other years.

$$\begin{aligned} c(n) &= \gamma_1 c(n-1) + \gamma_2 r(n-1) \\ c(n) - \gamma_1 c(n-1) &= \gamma_2 r(n-1) \\ c(n) - \gamma_1 c(n-1) &= \gamma_2 r(n-1) \\ \frac{1}{\gamma_2}c(n) - \frac{\gamma_1}{\gamma_2}c(n-1) &= r(n-1) \ . \end{aligned}$$

Now we know an expression for $r(n-1)$ in terms of values of c; and we can also derive an expression for $r(n)$, just by taking the one for $r(n-1)$ and moving all the temporal indices forward by one:

$$r(n) = \frac{1}{\gamma_2}c(n+1) - \frac{\gamma_1}{\gamma_2}c(n) \ .$$

So, we can take equation 2 and substitute these expressions in for $r(n)$ and $r(n-1)$, getting:

$$\begin{aligned}
r(n) &= \gamma_3 r(n-1) + \gamma_4 c(n-1) \\
\frac{1}{\gamma_2}c(n+1) - \frac{\gamma_1}{\gamma_2}c(n) &= \frac{\gamma_3}{\gamma_2}c(n) - \frac{\gamma_1\gamma_3}{\gamma_2}c(n-1) + \gamma_4 c(n-1) \\
c(n+1) &= (\gamma_1+\gamma_3)c(n) + (\gamma_2\gamma_4 - \gamma_1\gamma_3)c(n-1) \\
c(n) &= (\gamma_1+\gamma_3)c(n-1) + (\gamma_2\gamma_4 - \gamma_1\gamma_3)c(n-2)
\end{aligned}$$

Woo hoo! Now we have a second order difference equation, and we know what to do.

Question 6. What is the difference equation for $r(n+1)$?

Will the coyotes eat all the roadrunners and die out themselves? Will the roadrunner population overrun the Acme valley? Only the γs and the initial conditions will tell us for sure.

Beep beep!