Quantitative Understanding in Biology Module III: Linear Difference Equations Lecture I: Introduction to Linear Difference Equations

Introductory Remarks

This section of the course introduces dynamic systems; i.e., those that evolve over time. Although dynamic systems are typically modeled using differential equations, there are other means of modeling them. While we will spend a good deal of time working with differential equations in the next section, we'll begin by considering a close cousin of the differential equation: the difference equation. We will note here that when we solve differential equations numerically using a computer, we often really solve their difference equation counterparts. So, having some facility with difference equations is important even if you think of your dynamic models in terms of differential equations.

We'll also spend some time in this section talking about techniques for developing and expressing models. While there is often more art than science involved, there are some good practices that we will share.

This section of the course will also begin to emphasize matrix formulations of models and linear algebra.

We will begin our exploration of these topics with two examples; one from molecular evolution and one from population dynamics.

An Molecular Evolution Example

(*Mathematical Models in Biology*; Allman and Rhodes)

A fundamental part of molecular evolution is the study of mutations of DNA sequences. As you know, a DNA molecule is a sequence made up from four nucleic acids: adenine, guanine, cytosine, and thymine. Adenine and guanine are chemically similar and are called purines; cytosine and thymine are also similar and are called pyrimidines.

Some simple mutations that a DNA sequence can undergo include deletions, insertions, and single nucleotide substitutions. DNA can also undergo bulk mutations such as the inversion (reversal) or duplication of a stretch of DNA. In fact, even entire genome duplications have occurred.

For our current purposes, however, we will consider only single nucleotide substitutions, and focus on the estimation of mutation rates. Within this family of mutations, substitutions that involve swapping a

purine with a pyrimidine are called transversions. Transitions are substitutions that leave the class of nucleotide unchanged.

Consider the ancestral sequence, S_0, an intermediate sequence S_1, and a descendant S_2.

S_0: AC**C**TGC**G**CTA
S_1: AC**G**TGC**A**CTA
S_2: AC**G**TGC**G**CTA

Seeing all three sequences, we count three mutation events. However, if we did not have knowledge of the intermediate sequence, we would only observe one mutation event, and might naïvely underestimate the mutation rate. This error arises from the back-mutation at the seventh position in the sequence. To understand how back-mutations can affect estimates of mutation rates, we'll build and explore a simple model of nucleotide substitution.

Imagine that at a single position in the genome we begin with a cytosine nucleotide at time t=0. After some discrete time period t=1, there is a probability that the nucleotide will change. In our simplest model, we will assume that all possible substitutions (transitions and transversions) are equally likely. We will denote the probability of each possible mutation as α. We then have

$P_1(A) = \alpha$
$P_1(G) = \alpha$
$P_1(C) = 1-3\alpha$
$P_1(T) = \alpha$

Note that our initial condition can also be expressed in terms of probabilities.

$P_0(A) = 0$
$P_0(G) = 0$
$P_0(C) = 1$
$P_0(T) = 0$

After a second time step, our system evolves again. The probability of observing a C at this position is now

$$\begin{aligned} P_2(C) &= P_1(A)\cdot\alpha + P_1(G)\cdot\alpha + P_1(C)\cdot(1-3\alpha) + P_1(T)\cdot\alpha \\ &= 3\cdot\alpha^2 + (1-3\alpha)^2 \end{aligned}$$

The probability of observing an A at this position is

$$\begin{aligned} P_2(A) &= P_1(A)\cdot(1-3\alpha) + P_1(G)\cdot\alpha + P_1(C)\cdot\alpha + P_1(T)\cdot\alpha \\ &= \alpha\cdot(1-3\alpha) + \alpha\cdot\alpha + \cdot(1-3\alpha)\cdot\alpha + \alpha\cdot\alpha \\ &= 2\cdot\alpha\cdot(1-3\alpha) + 2\cdot\alpha^2 \end{aligned}$$

The probability for G and T will be same, so we have

$P_2(A) = 2\cdot\alpha\cdot(1-3\alpha) + 2\cdot\alpha^2$
$P_2(G) = 2\cdot\alpha\cdot(1-3\alpha) + 2\cdot\alpha^2$
$P_2(C) = 3\cdot\alpha^2 + (1-3\alpha)^2$
$P_2(T) = 2\cdot\alpha\cdot(1-3\alpha) + 2\cdot\alpha^2$

We could carry on this way with t=3, 4, 5… There would be a fair amount of algebra (although we could probably simplify things along the way). An alternative is to formulate our problem in terms of matrices and linear algebra. The motivation for doing this is not simply to save on tedium (although that would probably be reason enough). More importantly, since there is an enormous body of theory already developed around matrices and linear algebra, if we can cast our problem in this context, we inherit a wealth of knowledge and tools for free.

We'll begin by defining a vector to hold the state variables of our system. The state variables in a model hold all of the information you need to fully specify the dynamic state of the system. In other words, anything that might change over time is considered a state variable. For our first pass at this, we would define our vector of state variables as

$$\boldsymbol{s}_i = \begin{pmatrix} P_i(A) \\ P_i(G) \\ P_i(C) \\ P_i(T) \end{pmatrix}$$

Note that technically we only need three of these values to fully specify the dynamic state of our system. There is an implied constitutive equation $P_i(A) + P_i(G) + P_i(C) + P_i(T) = 1$, and a three element vector would be sufficient. You have probably seen this line of reasoning before in your thermodynamics classes. You only need to specify two state variables (say, temperature and pressure) of a pure substance; everything else (enthalpy, density, etc.) can be computed from the equations of state for that substance. Admittedly, those equations are a good deal more complex, but the idea is the same. We'll have more to say about this concept later.

We can also define a state transition matrix, **M**, as

$$\boldsymbol{M} = \begin{pmatrix} 1-3\propto & \propto & \propto & \propto \\ \propto & 1-3\propto & \propto & \propto \\ \propto & \propto & 1-3\propto & \propto \\ \propto & \propto & \propto & 1-3\propto \end{pmatrix}$$

The system under investigation then works out quite cleanly. We can write…

$$\mathbf{s}_{i+1} = \mathbf{M}\cdot\mathbf{s}_i$$

…or, even more generally…

$$\mathbf{s}_n = \mathbf{M}^n\cdot\mathbf{s}_0$$

A quick review of matrix multiplication

In the above discussion, we made use of some prior knowledge you may have had regarding matrix multiplication. Here we'll review…

If you have two matrices, A and B, you can multiply them if the number of columns in A is equal to the number of rows in B. The resultant product will have the same number of rows as A and the same number of columns as B. In other words

$$\mathbf{A}_{m\times k} \cdot \mathbf{B}_{k\times n} = \mathbf{C}_{m\times n}$$

This implies one very important property of matrix multiplication: the order of the terms matters. We say that matrix multiplication is not commutative. In the above example, **B**·**A** would not necessarily even be defined. Even if it were (in which case m=n), the product would not necessarily be the same. In fact, if m=n≠k, then the products would not be the same size. For this reason, you'll often hear phrases as 'multiply **B** on the left by **A**'.

In general, if we have two compatible matrices $\mathbf{A}_{m\times k}$ and $\mathbf{B}_{k\times n}$, matrix multiplication is defined as follows:

$$\mathbf{C}_{m\times n} = \mathbf{A}_{m\times k} \cdot \mathbf{B}_{k\times n}$$

$$c_{i,j} = \sum_{l=1}^{k} a_{i,l} \cdot b_{l,j}$$

It is helpful to visualize this. Each element in the resultant product matrix C is a sum of products. You take elements in a row of the first matrix and multiply them by the corresponding elements in a column of the second matrix. Then sum the products.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,p} & \dots & a_{1,k} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,p} & \dots & a_{2,k} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,p} & \dots & a_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots \\ a_{i,1} & a_{i,2} & a_{i,3} & \cdots & a_{i,p} & \dots & a_{i,k} \\ \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,p} & \dots & a_{m,k} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & \cdots & b_{1,j} & \dots & b_{1,n} \\ b_{2,1} & b_{2,2} & b_{2,3} & \cdots & b_{2,j} & \dots & b_{2,n} \\ b_{3,1} & b_{3,2} & b_{3,3} & \cdots & b_{3,j} & \dots & b_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots \\ b_{q,1} & b_{q,2} & b_{q,3} & \cdots & b_{q,j} & \dots & b_{q,n} \\ \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots \\ b_{k,1} & b_{k,2} & b_{k,3} & \dots & b_{k,j} & \dots & a_{k,n} \end{pmatrix} =$$

$$\begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \cdots & c_{1,j} & \dots & c_{1,k} \\ c_{2,1} & c_{2,2} & c_{2,3} & \cdots & c_{2,j} & \dots & c_{2,k} \\ c_{3,1} & c_{3,2} & c_{3,3} & \cdots & c_{3,j} & \dots & c_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots \\ c_{i,1} & c_{i,2} & c_{i,3} & \cdots & c_{i,j} & \dots & c_{i,k} \\ \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & c_{m,3} & \dots & c_{m,j} & \dots & c_{m,k} \end{pmatrix}$$

While matrix multiplication is not commutative, it is associative. This means that the order of operations (within a string of multiplications) does not matter. In other words, you can prove (**A**·**B**)·**C** = **A**·(**B**·**C**). This implies that we can simply write **A**·**B**·**C**, and one can perform the operations in any arbitrary order.

A vector can be thought of as a special case of a matrix that happens to have one column. Often, you'll see vectors written lengthwise to save space; for example, we could have written our definition of the state variable vector like so:

$$\mathbf{s}_i = (\ P_i(A)\ \ P_i(G)\ \ P_i(C)\ \ P_i(T)\ )$$

However, for purposes of multiplication, the vector should be thought of as a column vector.

The transpose of a matrix is written as $\mathbf{A}^t$. All that this means is that we swap rows with columns. So if $\mathbf{B}_{nxm} = (\mathbf{A}_{mxn})^t$, then $b_{i,j} = a_{j,i}$. The transpose of a column vector is a row vector, which is just a 1xm matrix.

Example: If **v** and **w** are vectors, what are the dimensions of $\mathbf{v}^t \cdot \mathbf{w}$, and when is it defined?

$$(1 \quad 2 \quad 3)\begin{pmatrix}1\\2\\3\end{pmatrix} = 1^2 + 2^2 + 3^2 = 14$$

Note that the vectors need to be the same length for this computation to be defined. The result is a scalar value. Note that in the case of vectors, we could also just write **v**·**w**, which is the scalar product (or dot product) of the two vectors. The result is the same.

Example: If **v** and **w** are vectors, what are the dimensions of $\mathbf{v} \cdot \mathbf{w}^t$, and when is it defined?

$$\begin{pmatrix}1\\2\\3\end{pmatrix}(1 \quad 2 \quad 3 \quad 4) = \begin{pmatrix}1 & 2 & 3 & 4\\2 & 4 & 6 & 8\\3 & 6 & 9 & 12\end{pmatrix}$$

Here we get something reminiscent of a multiplication table. Note that the vectors do not need to be the same length. The table will have as many rows as the length of the first vector, and as many columns as the length of the second.

Example: Consider a matrix, **M**, that holds observed (x,y) data from an experiment, augmented with a column of ones. Compute $\mathbf{M}^t \cdot \mathbf{M}$.

$$\boldsymbol{M} = \begin{pmatrix}1 & x_1 & y_1\\1 & x_2 & y_2\\\vdots & \vdots & \vdots\\1 & x_n & y_n\end{pmatrix}$$

$$\boldsymbol{M}^t \cdot \boldsymbol{M} = \begin{pmatrix}1 & 1 & \cdots & 1\\x_1 & x_2 & \ldots & x_n\\y_1 & y_2 & \ldots & y_n\end{pmatrix}\begin{pmatrix}1 & x_1 & y_1\\1 & x_2 & y_2\\\vdots & \vdots & \vdots\\1 & x_n & y_n\end{pmatrix} = \begin{pmatrix}n & \sum x & \sum y\\\sum x & \sum x^2 & \sum xy\\\sum y & \sum xy & \sum y^2\end{pmatrix}$$

The quantities in the resultant matrix are exactly those that are needed when computing correlation or linear regression. For this reason, the augmented matrix **M** is sometimes referred to as the model matrix in the statistical literature.

Finally, we define a matrix raised to a power simply as multiplying the matrix by itself an appropriate number of times. This is only meaningful for square matrices.

A minimalist interpretation of a matrix is simply a tableau of numbers. You can also think of matrix multiplication as an arbitrarily defined procedure for manipulating those numbers. You will see over and over that appropriately defined matrices and formalisms do all sorts of really useful things. The examples above are just a few simple ones.

Back to Molecular Evolution

Now that we have formalized our definitions for matrix multiplication, we can reconsider our proposed model for molecular evolution:

$$\mathbf{s}_{i+1} = \mathbf{M}\cdot\mathbf{s}_i$$

Expanded, this can be written as

$$\boldsymbol{s}_{i+1} = \begin{pmatrix} P_{i+1}(A) \\ P_{i+1}(G) \\ P_{i+1}(C) \\ P_{i+1}(T) \end{pmatrix} = \begin{pmatrix} P_{A\to A} & P_{G\to A} & P_{C\to A} & P_{T\to A} \\ P_{A\to G} & P_{G\to G} & P_{C\to G} & P_{T\to G} \\ P_{A\to C} & P_{G\to C} & P_{C\to C} & P_{T\to C} \\ P_{A\to T} & P_{G\to T} & P_{C\to T} & P_{T\to T} \end{pmatrix} \begin{pmatrix} P_i(A) \\ P_i(G) \\ P_i(C) \\ P_i(T) \end{pmatrix}$$

For our model, which assumes that all transitions and transversions occur with equal probability, the governing equation is...

$$\boldsymbol{s}_{i+1} = \begin{pmatrix} P_{i+1}(A) \\ P_{i+1}(G) \\ P_{i+1}(C) \\ P_{i+1}(T) \end{pmatrix} = \begin{pmatrix} 1-3\propto & \propto & \propto & \propto \\ \propto & 1-3\propto & \propto & \propto \\ \propto & \propto & 1-3\propto & \propto \\ \propto & \propto & \propto & 1-3\propto \end{pmatrix} \begin{pmatrix} P_i(A) \\ P_i(G) \\ P_i(C) \\ P_i(T) \end{pmatrix}$$

Note that for this to be a valid matrix for this type of model, all of the columns in the transition matrix **M** must sum to exactly one, and all of the values must satisfy $0 \le m_{i,j} \le 1$, because the elements are probabilities. These models, where the state variables are probabilities of being in a particular state, and where the transition matrix gives the probabilities of transitioning from one state to the next, are called Markov models. Note that a transition matrix in a Markov model need not be symmetric.

A Word on Applied Mathematics

In our molecular evolution model, we have the relation…

$$\mathbf{s}_n = \mathbf{M}^n\cdot\mathbf{s}_0$$

…where M is the matrix above. Observed values are on the order of 10^{-8} mutations per base pair per year for mammalian mitochondrial DNA, 10^{-9} mutations per bp per year for chloroplast DNA in maize and barley, and a whopping 0.01 mutations per bp per year for the influenza A virus; i.e., $3\alpha \approx 10^{-8}$, 10^{-9} or 0.01.

If we wanted to compute $M^{40,000,000}$ we would not need to do 40,000,000 serial computation. A serial computation as follows takes about a minute on my computer.

```
>> a = 1e-11;
>> M = [1-3*a, a, a, a; a, 1-3*a, a, a; a, a, 1-3*a, a; a, a, a, 1-3*a];
>> M2 = M;
>> for (i=1:39999999)
   M2 = M2 * M;
end
>> M2

M2 =

    0.9988    0.0004    0.0004    0.0004
    0.0004    0.9988    0.0004    0.0004
    0.0004    0.0004    0.9988    0.0004
    0.0004    0.0004    0.0004    0.9988
```

Compare this to evaluating the matrix by raising it to a power; this is perceived as instantaneous on the same computer.

```
>> M^40000000

ans =

    0.9988    0.0004    0.0004    0.0004
    0.0004    0.9988    0.0004    0.0004
    0.0004    0.0004    0.9988    0.0004
    0.0004    0.0004    0.0004    0.9988
```

How might MATLAB be evaluating this so quickly? One possibility is that you can compute powers of two of a matrix quickly by successively squaring a matrix. For example, computing M^8 could be done by computing $M \cdot M \cdot M \cdot M \cdot M \cdot M \cdot M \cdot M$, which involves seven matrix multiplications. However, we could also compute $M^2 = M \cdot M$, and then compute $M^4 = M^2 \cdot M^2$, and then $M^8 = M^4 \cdot M^4$. This only involves three matrix multiplications. Arbitrary powers can use previous results, so M^{13} could be computed as $M^8 \cdot M^4 \cdot M$ which would require a total of five multiplications.

Mathematically, the results are the same. Practically, using one little trick, we can get results about a million times faster. Sometimes you can't just copy an equation out of a textbook.

Another issue that applied mathematicians face is the numerical precision of computers. However we compute $M^{40,000,000}$, it is our job to make sure that the result is still a valid Markov matrix. In this case, MATLAB does a pretty good job.

```
>> M40M = M^40000000;
>> sum(M40M)

ans =
    1.0000    1.0000    1.0000    1.0000

>> 1-sum(M40M)

ans =
  1.0e-009 *
    0.1048    0.1048    0.1046    0.1014
```

Here we take advantage of the behavior of the sum function in MATLAB. When you take the sum of a matrix, you get a vector where each element corresponds to the sum of elements in each column of the original matrix. To get an idea of how much rounding error we have incurred, we subtract each sum from one; here we see that the errors on the order of 10^{-9}.

Simple Population Growth

Consider a population of organisms that starts out at with P_0 members at time t=0. Assuming that in a given time step (say one year), some fraction, f_r, of the members will have reproduced and added a single member to the population. Another fraction, f_m, will have died.

We can write:

$P_{n+1} = P_n + f_r P_n - f_m P_n$

$P_{n+1} = P_n(1+f_r-f_m)$

We can see that

$P_{n+2} = P_{n+1}(1+f_r-f_m) = P_n(1+f_r-f_m)^2$

…and in general…

$P_n = P_0(1+f_r-f_m)^n$

The key parameter of our model is $1+f_r-f_m$, which we define as λ. Now our model is…

$P_n = \lambda^n P_0$

It is helpful to think about what the long-term fate of our population will be for different values of λ. Our population will eventually fizzle if $0 < \lambda < 1$, and (in our simple model) will grow in an unbounded manner if $\lambda > 1$. For $\lambda = 1$ (exactly), the population is stable. In this simple model, it is not possible for λ to take on

negative values, since its definition will not admit the possibility. Mathematically, we would expect to see oscillations between positive and negative population values; oscillations are OK, but a negative population doesn't make sense. We'll get back to this point in a while.

The exponential growth we see here is a critical result, and understanding how this growth behaves for different values of λ can't be overemphasized.

(Slightly) More Complex Population Growth

(*Mathematical Models in Biology*, Edelstein-Keshet)

Let us consider a slightly more complex example of population growth. We'll develop a model for the population of annual plants.

In our model, annual plants produce γ seeds per plant in the fall of each year. Seeds remain in the ground for up to two years. Some fraction of one-year-old seeds, α, germinate during the spring. Some fraction of two-year-old seeds, β, germinate during the spring. All plants die during the winter.

We'll begin by defining three state variables:

P_n	The number of plants that germinate from seeds during year n
$S1_n$	The number of one-year-old seeds in the ground at the beginning of the spring of year n
$S2_n$	The number of two-year-old seeds in the ground at the beginning of the spring of year n

It is always a very good idea to write down precise definitions of your model variables.

Our model, translated from words into equations, tells us that

$P_n = \alpha \cdot S1_n + \beta \cdot S2_n$
$S1_{n+1} = \gamma \cdot P_n$
$S2_{n+1} = (1-\alpha) \cdot S1_n$

Since we like to write future variables in terms of present variables, we will write…

$P_{n+1} = \alpha \cdot S1_{n+1} + \beta \cdot S2_{n+1}$

…and then substitute our 'definitions' of S1 and S2…

$P_{n+1} = \alpha \cdot \gamma \cdot P_n + \beta \cdot (1-\alpha) \cdot S1_n$

So it turns out that we can track the evolution of our system only by keeping track of P and S1.

$P_{n+1} = \alpha \cdot \gamma \cdot P_n + \beta \cdot (1-\alpha) \cdot S1_n$
$S1_{n+1} = \gamma \cdot P_n$

We have reduced our model to a system of linear difference equations. This begs for a matrix representation. In fact, we can write…

$$\begin{pmatrix} P_{n+1} \\ S1_{n+1} \end{pmatrix} = \begin{pmatrix} \alpha \cdot \gamma & \beta \cdot (1-\alpha) \\ \gamma & 0 \end{pmatrix} \begin{pmatrix} P_n \\ S1_n \end{pmatrix}$$

We can use this formulation to predict how the model will evolve over time by raising the matrix to any given power. Ideally, it would be nice to be able to treat the system analytically, and see if we can't learn anything about the trends in behavior of the system, much as we did in the simple growth model.

Introduction to Eigenvalues

We would like to formulate a solution of an arbitrary 2x2 system of linear difference equations. Our arbitrary system is…

$x_{n+1} = a_{11} \cdot x_n + a_{12} \cdot y_n$
$y_{n+1} = a_{21} \cdot x_n + a_{22} \cdot y_n$

We can turn this into a single equation that incorporates two time steps…

$$\begin{aligned} x_{n+2} &= a_{11} \cdot x_{n+1} + a_{12} \cdot y_{n+1} \\ &= a_{11} \cdot x_{n+1} + a_{12} \cdot (a_{21} \cdot x_n + a_{22} \cdot y_n) \\ &= a_{11} \cdot x_{n+1} + a_{12} \cdot a_{21} \cdot x_n + a_{22} \cdot a_{12} \cdot y_n \\ &= a_{11} \cdot x_{n+1} + a_{12} \cdot a_{21} \cdot x_n + a_{22} \cdot (x_{n+1} - a_{11} \cdot x_n) \end{aligned}$$

or

$x_{n+2} - (a_{11} + a_{22})x_{n+1} + (a_{22} \cdot a_{11} - a_{12} \cdot a_{21})x_n = 0$

We saw that for a 1x1 system, the solution is $x_n = x_0 \cdot \lambda^n$. We can try this solution in our case here…

$x_0 \cdot \lambda^{n+2} - (a_{11} + a_{22})\, x_0 \cdot \lambda^{n+1} + (a_{22} \cdot a_{11} - a_{12} \cdot a_{21})\, x_0 \cdot \lambda^n = 0$

Cancelling out a factor of $x_0 \cdot \lambda$, we get…

$\lambda^2 - (a_{11} + a_{22})\, \lambda + (a_{22} \cdot a_{11} - a_{12} \cdot a_{21}) = 0$

…which is a quadratic equation. If we define $b = (a_{11} + a_{22})$, and $c=(a_{22} \cdot a_{11} - a_{12} \cdot a_{21})$, we have

$$\lambda = \frac{b \pm \sqrt{b^2 - 4c}}{2}$$

So there are two possible values for λ. For now, we'll consider only the case of two distinct real values for λ. Other cases (repeated and complex roots) will be considered later.

Since our original system of equations is linear, the principle of superposition applies. This means that any linear combination of solutions to our system will also be a solution to our system. So our general solution is…

$x_n = A_1 \cdot \lambda_1^n + A_2 \cdot \lambda_2^n$

As this solution is only a guess, you can go back and substitute this result into the original equation and prove that it works.

Had we chosen to eliminate x instead of y, our solution would have been…

$y_n = B_1 \cdot \lambda_1^n + B_2 \cdot \lambda_2^n$

The A and B constants are different, but the values of the λs are the same. You can see this in the definitions for b and c; they are symmetric (substitute a_{ij} with a_{ji}, and you get the same values of b and c, and thus the same λ).

The values of λ are called the eigenvalues of the matrix. You can learn an incredible amount about the long-term behavior of the system simply by knowing the eigenvalues and inspecting these equations.

$x_n = A_1 \cdot \lambda_1^n + A_2 \cdot \lambda_2^n$
$y_n = B_1 \cdot \lambda_1^n + B_2 \cdot \lambda_2^n$

Firstly, the eigenvalue with the larger absolute value will dominate the long-term behavior of the system. For arbitrarily large n, the terms with the smaller eigenvalues will be negligible compared to those with the larger. Some cases arise:

$\lambda > 1$	System grows exponentially
$\lambda = 1$	System reaches an equilibrium
$0 < \lambda < 1$	System decays
$-1 < \lambda < 0$	System decays in an oscillatory manner
$\lambda = -1$	System reaches a stationary oscillation
$\lambda < -1$	System grows exponentially in an oscillatory manner

All of this applies to larger systems as well. A system with four state variables (such as in our molecular evolution example), will have four eigenvalues. Its long-term behavior will be governed by the largest. Eigenvalues for large systems (i.e., large matrices) are usually determined by a computer.

Back to Plants and Seeds

Back to our plant example, let us suppose that each plant produces 25 seeds. Let us also suppose that 7% of one-year-old seeds germinate, and that 5% of two-year-old seeds germinate. We can compute the eigenvalues of this system:

```
>> alpha=0.07; beta=0.05; gamma=25; M = [alpha * gamma, beta*(1-
alpha); gamma, 0]; eig(M)

ans =

    2.2636
   -0.5136
```


In this case, our population should grow quite healthily.

Intuitively, in this model we expect that if the number of seeds that ultimately germinates per plant is greater than one, we would have a sustainable population. We get the intuitive result we expect in the case where the all of the seeds germinate in the first year and there are just enough of them…

```
>> alpha=0.04; beta=0.0; gamma=25; M = [alpha * gamma, beta*(1-alpha);
gamma, 0]; eig(M)

ans =

     0
     1
```

And also when the same happens in the second year…

```
>> alpha=0.0; beta=0.04; gamma=25; M = [alpha * gamma, beta*(1-alpha);
gamma, 0]; eig(M)

ans =

     1
    -1
```

Note that in this case we have a stable equilibrium and an oscillating one, with neither being dominant. What does this tell you about the behavior of the model?

However, note the following…

```
>> alpha=0.02; beta=0.02; gamma=25; M = [alpha * gamma, beta*(1-
alpha); gamma, 0]; eig(M)

ans =

    0.9933
   -0.4933
```

Can you explain this result?

Exercises

What is the condition necessary for a sustainable population? [Ans: $\alpha + \beta - \alpha\beta \geq 1/\gamma$]. Derive this from both our reasoning above, and our knowledge of eigenvalues? Simulate such a system and observe its transient as well as its dynamic behavior. Simulate 'nearby'; systems with a dominant eigenvalue just above and just below the cutoff for sustainability. In all of these simulations, begin with 100 plants and no seeds.


```
>> alpha = 0.02; beta=0.02; gamma=25;
>> M = [alpha * gamma, beta*(1-alpha); gamma, 0];
>> x = [100, 0];
>> for(i=1:30)
x(i+1,:) = M * x(i,:)';
end
>> x

x =

  1.0e+003 *

    0.1000         0
    0.0500    2.5000
    0.0740    1.2500
    0.0615    1.8500
    0.0670    1.5375
    0.0636    1.6752
    0.0647    1.5910
    0.0635    1.6164
    0.0634    1.5878
    0.0628    1.5859
    0.0625    1.5710
    0.0620    1.5626
    0.0616    1.5511
    0.0612    1.5412
    0.0608    1.5306
    0.0604    1.5205
    0.0600    1.5103
    0.0596    1.5002
    0.0592    1.4901
    0.0588    1.4801
    0.0584    1.4702
    0.0580    1.4604
    0.0576    1.4506
    0.0572    1.4409
    0.0569    1.4312
    0.0565    1.4216
    0.0561    1.4121
    0.0557    1.4027
    0.0554    1.3933
    0.0550    1.3839
    0.0546    1.3747
>> plot(x(:,1))
```

Other interesting parameter values to plot are

```
>> alpha=0.00; beta=0.035; gamma=25;

>> alpha=0.001; beta=0.035; gamma=25;
```


Compute the eigenvalues for our model of molecular evolution. Choose your favorite value for a mutation rate.

```
a =

    0.0100

>> M = [1-3*a, a, a, a; a, 1-3*a, a, a; a, a, 1-3*a, a; a, a, a, 1-
3*a]

M =

    0.9700    0.0100    0.0100    0.0100
    0.0100    0.9700    0.0100    0.0100
    0.0100    0.0100    0.9700    0.0100
    0.0100    0.0100    0.0100    0.9700

>> eig(M)

ans =

    0.9600
    0.9600
    0.9600
    1.0000
```

Develop a model where the probability of transversion is lower than the probability of transition. What effect does this have on the long term distribution of nucleotide frequencies?

It turns out that two theorems can be proven for Markov matrices. The first states that any Markov matrix has $\lambda_1 = 1$, and that $|\lambda| \leq 1$ for all eigenvalues. Furthermore, a second theorem states that for a Markov matrix with all positive elements, $\lambda_1 = 1$ will be a strictly dominant eigenvalue (in other words, you can't have another eigenvalue of $\lambda_n = -1$. What do these theorems imply about the behavior of a system that can be described by a Markov model?

