Lecture 11: Powers of Matrices, Difference Equations

Difference Equations

A difference equation, also sometimes called a recurrence equation, is an equation that defines a sequence recursively, i.e. each term of the sequence is defined as a function of the previous terms of the sequence:

\[ x_n = f(x_{n-1}, x_{n-2}, \ldots, x_0) \]

We’ve already seen a type of difference equation in Project 3, called discrete Markov chains. The state variable, \( x_n \), can be evolved in time through the probability transition matrix \( A \), such that

\[ x_n = Ax_{n-1} \]

In this case, \( f(x_{n-1}, x_{n-2}, \ldots, x_0) = f(x_{n-1}) = Ax_{n-1} \). As a simple example, take \( A = 2 \) (a scalar) and \( x_0 \) is 1. Then, \( x_1 = Ax_0 = (2)(1) = 2 \). And, \( x_2 = Ax_1 = 2(2) = 4 \), etc. So, we end up getting the sequence

\[ 1, 2, 4, 8, 16, 32, \ldots \]

What do you notice about the behavior of the sequence? It gets larger. Thus, as we keep going along in the sequence, we’ll eventually approach infinity, i.e.

\[ \lim_{n \to \infty} x_n \to \infty \]

However, let’s take \( A = 1/2 \) instead. Now the sequence looks like:

\[ 1, 1/2, 1/4, 1/8, 1/16, \ldots \]

And the long term behavior? It’s going to zero. Completely different behavior and highly dependent on the choice of \( A \). Can you make a guess at the value of \( A \) which is the turning point between going to zero and going to infinity?

Another important feature in this example is you can iterate these things. So, if

\[ x_n = A x_{n-1} = A(Ax_{n-2}) = A^2 x_{n-2} = \ldots = A^n x_0 \]

So, this is where being able to compute powers of matrices becomes useful. However, it’s not always informative. For example, as we saw previously, what if \( A^n \) goes to infinity? It can become difficult to quantify solutions in that case.

Let’s try a special matrix we introduced last time from the drunkards walk. The transient matrix \( Q \) is

\[ Q = \begin{bmatrix} 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{bmatrix}. \]

We can find the power matrices of \( Q \) by using MATALB and see what happens. Please try \( Q^{100}, Q^{500}, Q^{1000}, Q^{2000}, Q^{2145}, Q^{2149} \).
Linear Difference Equations and Linear Algebra

In general, difference equations can be much more complex. But, let’s look at the case of linear difference equations. In this case,

\[ x_n = f(x_{n-1}, x_{n-2}, \ldots, x_0) = c_{n-1}x_{n-1} + c_{n-2}x_{n-2} + \ldots + c_0x_0 \]

So, the terms of the sequence are just linear combination of terms in the previous part of the sequence. A very famous example of this is the Fibonacci sequence, which is defined as the following:

\[ x_n = x_{n-1} + x_{n-2} \]

with \( x_0 = 1, x_1 = 1 \). Notice, this needs two initial conditions. Why’s that? You should be able to answer this. But, if we plug in starting with \( n = 2 \), we get the sequence:

1, 1, 2, 3, 5, 8, 13, ...

So, first you’ll notice that it’s also going to infinity. We can explain why later through looking at the magnitude of the eigenvalues. Secondly, this is a linear system. Since linear algebra is the study of linear things, we should be able to write this in a form we know - particularly in a matrix form.

To do so, we’re going to make use of a trick. It’s the idea that we can think about one dimensional objections as objects embedded in a higher dimensional space. As an example, we look at the Fibonacci sequence again:

\[ x_n = x_{n-1} + x_{n-2} \]

Rather than look directly at the equation, we’re going to define the vector \( y_n = (x_n, x_{n-1}) \). Now, we also know that \( y_{n-1} = (x_{n-1}, x_{n-2}) \). For example, the vectors \( y_n \) with respect to Fibonacci sequence

\[ x_1 = 1, x_2 = 1, x_3 = 2, x_4 = 3, x_5 = 5, x_6 = 8, x_7 = 13, x_8 = 21, x_9 = 34, \ldots \]

are

\[ y_2 = (1, 1), y_3 = (1, 2), y_4 = (2, 3), y_5 = (3, 5), \]
\[ y_6 = (5, 8), y_7 = (8, 13), y_8 = (13, 21), y_9 = (21, 35), \ldots . \]

You may find out there is no \( y_1 \) since we don’t have \( x_0 \) term in our Fibonacci sequence.

Then, we want to be able to find a matrix \( A \) such that

\[ y_n = Ay_{n-1}. \]

To do that, let’s expand it out:

\[
\begin{bmatrix}
  x_n \\
  x_{n-1}
\end{bmatrix} =
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
  x_{n-1} \\
  x_{n-2}
\end{bmatrix} =
\begin{bmatrix}
  a_{11}x_{n-1} + a_{12}x_{n-2} \\
  a_{21}x_{n-1} + a_{22}x_{n-2}
\end{bmatrix}
\]
Let’s start with the first row. So, we want to find the $a_{ij}$ constants so that $x_n = x_{n-1} + x_{n-2}$. Well, that’s easy. That means we want $a_{11} = 1$ and $a_{12} = 1$. How about the second row? We just want $x_{n-1} = x_{n-1}$, so $a_{21} = 1$ and $a_{22} = 0$. Thus, the Fibonacci sequence can be described as:

$$\begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{n-1} \\ x_{n-2} \end{bmatrix}$$

This should also make it clear why we needed two initial conditions. The vector has two elements, so the initial vector needs two initial conditions.

Let’s take another example. Suppose we have the equation:

$$x_n = x_{n-1} - 2x_{n-2} + x_{n-3}$$

Quick check: How many initial conditions are we going to need? Right, three. What’s our vector $y_n$ going to look like? It’s going to be $y_n = (x_n, x_{n-1}, x_{n-2})$. We need to build our matrix $A$, which is going to be:

$$\begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{n-1} \\ x_{n-2} \\ x_{n-3} \end{bmatrix}$$

The interesting thing is we can look at the eigenvalues to tell us what’s going to happen as $n$ goes to infinity, detailed below.

**Eigenvalues of Linear Difference Equations**

Let’s go back to our Fibonacci sequence example:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Now, if we solve for the eigenvalues, what do we find? Well, we find that they are:

$$\lambda = \frac{1 \pm \sqrt{5}}{2} \approx \{-0.6180, 1.618\}$$

In fact, the positive one is called the golden ratio - something that you may have heard before in geometry or other areas. But, the important part is that $|\lambda| > 1$ for at least one of the eigenvalues. This means that the terms are going to grow in time. So, as $n$ gets large, $x_n$ goes to infinity.

As a second example, consider a probability matrix $A$ defined as:

$$A = \begin{bmatrix} .5 & .5 \\ 1 & 0 \end{bmatrix}$$

Then, the eigenvalues are $\lambda = 1, -1/2$. Notice that the magnitude of neither is bigger than one, so the solution won’t grow. Remember this was a requirement in probability matrices - the state vectors never get bigger, only the percentages change.
Difference Equations in MATLAB

We’ll be breaking the MATLAB portion into two pieces. One of which is how to run these in time \( n = 1, 2, 3, \ldots \) with a for loop, and the other we’ll use matrix powers, similar to before.

- **for loops**
  
  Let’s talk about our Fibonacci sequence and how we would program it in MATLAB. Recall, it is:

  \[
  x_n = x_{n-1} + x_{n-2}, \quad x_1 = 1, \quad x_2 = 1
  \]

  Notice I start at index 1 rather than 0. This is so it’s easier to compare to MATLAB, which forces you to start at index 1. So, in MATLAB, we want to set the first two elements to be 1. To do so, we can type:

  \[
  >> x(1) = 1; \ x(2) = 1;
  \]

  Then, we want to run a for loop that computes the previous two indices and adds them together. So, suppose we want 10 points, we would type:

  \[
  >> \text{for } k = 3:10 \\
  \quad x(k) = x(k-1) + x(k-2);
  \text{end}
  \]

  Then, we could plot the result and see what happens: We see that it continues to grow.

  ![Plot of Fibonacci sequence](image)

  This matches up with the fact that one of the eigenvalues was bigger than one, so it keeps increasing. You may be asked in the homework to plot a difference equation.

- **matrix powers**

  The other command we’ve seen before is computing matrix powers. Remember this is
useful if we care about a certain point in time. So, for example, if we want to know \( x_{10} \) for our Fibonacci example, we can setup the matrix:

\[
y_{10} = A y_0 = A^2 y_8 = ... A^{10} y_0
\]

This is pretty straightforward in MATLAB. We just do:

\[
>> y0 = [1;1] \quad A = [1,1;1,0]; \quad y10 = A^{10}*y0
\]

This is sort of nice, because it tells us the \( n = 10 \) element, but also the \( n = 9 \) element, because remember, \( y_n = (x_n, x_{n-1}) \).

**Example: Nonlinear Difference Equation**

As a final comment, we can talk about non-linear difference equations also. We won’t really discuss too much on analyzing them, but we’ll talk about one fairly famous one, called the *logistic equation*. It looks like:

\[
x_n = r x_{n-1} (1 - x_n)
\]

where \( r \) is some constant parameter in the system. Clearly, this isn’t linear, as there’s an \( x_n^2 \) term if we were to expand the right hand side. You won’t be asked to study these types of equations, but you should be able to plot the trajectory of their solutions.
Exercises

1. Consider the difference equation:

\[ x_n = -2x_{n-1}, \quad x_0 = 1; \]

Write a for loop in MATLAB to solve for the first 20 values of \( n \) and plot them. What do you notice?

2. Consider the difference equation:

\[ x_n = -2x_{n-1} + 3x_{n-2} - 4x_{n-3}, \quad x_0 = 1; \]

Write the linear matrix equation \( y_n = Ay_n \) that describes this equation. Compute the eigenvalues. What do you predict the behavior of the solutions to be based on the eigenvalues?

3. For the matrix found in Ex 2., compute \( A^2 \), \( A^5 \), and \( A^{10} \). Given initial state vector \( x_0 \), what time steps \( x_j \) would these matrices correspond to if multiplied by \( x_0 \)?