

MATH 215 – MATLAB Project

Project developed by: Jessica Bernier and William Yslas Vélez

In a separate attachment we give a proof of the following theorem: $\sum_{k=1}^n k^r$ (r a natural number) is

a polynomial in the variable n of degree $r+1$. Further, the theorem goes on to state that

$(r+1)! \sum_{k=1}^n k^r$ is a polynomial in n with integer coefficients. This seems like a reasonable

theorem, since you may already be familiar with the formulas:

$$2 \sum_{k=1}^n k = n(n+1) \text{ and } 6 \sum_{k=1}^n k^2 = n(n+1)(2n+1).$$

However, finding the polynomials for larger r is a more difficult task. If we were to apply the results of the previously mentioned theorem, we would have to use recursion. Recursion is a mathematical technique that allows us to calculate a sequence of objects, where knowledge of the objects up to k is then used to calculate the value of the next object.

The Fibonacci sequence, where

$$a_1=1, a_2=1 \text{ and } a_{n+2}=a_{n+1}+a_n$$

is a famous example of a sequence defined by recursion. Notice that the first two elements in the Fibonacci sequence are given and you use these, and the formula, to find the third element, then you use the second and the third to find the fourth. So, the Fibonacci sequence is

$$a_1=1, a_2=1, a_3=a_2+a_1=1+1=2, a_4=a_3+a_2=2+1=3, \text{ etc.}$$

The drawback to this method is that if we wanted to know the thousandth Fibonacci number, we would have to calculate all preceding Fibonacci numbers.

The theorem mentioned above is an example of recursion. If we were to use recursion, we would have to find the polynomial $\sum_{k=1}^n k^1$, and use that result, plus the recursive formula, to find

$\sum_{k=1}^n k^2$. We would then use these two polynomials, plus the recursive formula, to find $\sum_{k=1}^n k^3$. In

order to find $(r+1)! \sum_{k=1}^n k^r$, using this recursive technique, we would have to find the $r-1$

polynomials, $\sum_{k=1}^n k^1$, $\sum_{k=1}^n k^2$, ..., $\sum_{k=1}^n k^{r-1}$, plus the recursive formula developed in the proof of the theorem. This is a considerable amount of work.

Linear algebra provides a method for finding the polynomial for a fixed number r , without having to compute all previous polynomials. The idea behind this technique is very simple and we illustrate it in the following example. Suppose we wanted to know the coefficients of a quadratic equation, $f(x) = ax^2 + bx + c$. If we evaluated this polynomial at $x = 0$, $x = 1$, and $x = 2$, we would produce three linear equations, in the variables a , b , c . We could then use the techniques of linear algebra to solve for these coefficients. We would then write a system of linear equations, convert it into a matrix, and use familiar linear techniques to solve the matrix equation!

Problem 1:

Use MATLAB to answer the following question. Find a quadratic polynomial $f(x) = ax^2 + bx + c$ satisfying the following conditions: $f(0) = 2$, $f(1) = 3$, $f(2) = 7$.

Problem 2: Suppose $f(x) = ax^3 + bx^2 + cx + d$ is a cubic polynomial. Select four distinct numbers, p , q , r , s and set $f(p) = 1$, $f(q) = 3$, $f(r) = -1$, $f(s) = 1$. Using MATLAB, determine a , b , c , d .

Let's apply these ideas to the family of polynomials that we are studying. Suppose we want a polynomial the degree $i + 1$. We could start with an equation

$$f(x) = \sum_{k=1}^n k^i = a_{i+1}x^{i+1} + a_i x^i + a_{i-1}x^{i-1} + \dots + a_1 x + a_0, \quad 1 \leq x \leq n,$$

which is just a general form for polynomials (a_0, a_1, \dots, a_{i+1} are simply real coefficients.) We could write down similar equations for all x between 1 and n to get our system of equations. Returning to the quadratic example mentioned above, $r = 2$, our system of equations is:

$$\begin{cases} f(1) = a_2(1)^2 + a_1(1)^1 + a_0 1^0 = 1 \\ f(2) = a_2(2)^2 + a_1(2)^1 + a_0 2^0 = 3 \\ f(3) = a_2(3)^2 + a_1(3)^1 + a_0 3^0 = 6 \end{cases}$$

This system corresponds to: $A\vec{x} = \vec{b}$

$$\text{where } A = \begin{bmatrix} 1 & 1 & 1 \\ 2^2 & 2 & 1 \\ 3^2 & 3 & 1 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 3 \\ 6 \end{bmatrix}.$$

The matrix A has a special form, it is called a Vandermonde matrix. In the MATLAB help section, look up "Vandermonde".

We can generate an arbitrarily large matrix of this form, called a Vandermonde matrix, to find the polynomial corresponding to r . This project will require you to use MATLAB to acquire the polynomials and verify the theorem (a script has been provided that automatically generates the Vandermonde matrix and solves the system of equations.) Further, you will test the limits of MATLAB and to draw conclusions about the behavior of the polynomials you obtain.

Problem 3:

Use the MATLAB function `findPolyForSeries` to calculate the polynomials for $r = 1, 2, 3, \dots$. To do this, you will have to save the enclosed file “`findPolyForSeries.m`” in your MATLAB workspace directory. When MATLAB is running, type “`findPolyForSeries(r)`” in the command line. The program will prompt you for a value of r . For example, to find the polynomial

corresponding to $\sum_{k=1}^n k^2$ you would select the $r = 2$. Write down the polynomials that you

obtain in this way. (Use your calculator to convert the decimal values to fractional representations.) Is it true that, if you multiply each polynomial by $(r + 1)!$, the polynomials have integer coefficients?

Problem 4:

How many polynomials can MATLAB calculate before you get an error? Can you explain where this error might come from? (Hint: type in “`inverse = inv(ans)`” to display the inverse of matrix A. Are any of the rows similar for small values of r ? How about for larger values of r ? In investigating the behavior of the matrix, inverse, we could look at its individual rows. For a given matrix, plotting each row on the same graph might help to visualize this. Notice that there appears to be very regular sign changes in the rows. If we ignore the signs, by taking the absolute value of the elements, we could graph all of the rows together and investigate the pattern.

To accomplish this, first set: `inverse = inv(ans)`, then type in `r` equal to the value that you used. Next copy the following script and paste it onto MATLAB.

```
figure; hold on;
for i = 1 : r+2
plot(1 : r+2, abs(inverse(i, :)));
end
```

to get the graph for the matrix corresponding to `findPolyForSeries(r)`.

Problem 5:

Factor as many polynomials as you can. MATLAB will probably not be useful for this, so either do it by hand or, if you are familiar with it, use Mathematica. Refer to the document that contains a proof that the sum is a polynomial. The corollaries give you some information as to some of the factors. Do you notice any interesting behavior? Any conjectures?