proj2_a

October 5, 2016

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: import matplotlib.pyplot as plt
```

1 Vandermonde Matrix

To begin part 1 of creating a vandermonde matrix I created a C++ program called vandermonde.cpp. In this program I defined five square matrices with the size $n = [5, 9, 17, 33, 65]$. The vandermonde matrix was then computed as the vector

$$v = linspace(0, 1, n[i])$$

and the matrix

$$A = [(v^T)^0, (v^T)^1, (v^T)^2, ..., (v^T)^{n-1}]$$

. Next, to prove the ill-conditioned property of this matrix. I created a random vector $x = Random(n)$ to use as the known values of x. From the know x values I created the correct b values with the equation $Ax = b$. To prove the ill-conditioned property, I reverse soled the equation $Ax = b$ to find x'. This method used $Solve(A, b)$ and preformed a linear solve to find x'. With the approximated value of x found I used the original equation $Ax = b$ with x' to calculate the approximate value of b'. With the values of x and x' found and the values of b and b' found I could calculate the error and the residual for the vandermonde matrix.

```
In [3]: %cat vandermonde.cpp
```

```
/* Project - Project 2_Part a
 * Prof - Dr Xu
 * Name - Jake Rowland
 * Date - 10/6/16
 * Purpuse - Create vandermonde matrix to show ill-conditioning
*/

#include <iostream>
```

```
#include <fstream>
#include <vector>
#include <cmath>
#include <cstdlib>
#include "matrix.hpp"

int main() {

        //Defining the different n's for the project and the files to write to
        std::vector<int> n = {5, 9, 17, 33, 65};
        std::ofstream residOut("residual.txt", std::ios::out);
        std::ofstream errorOut("error.txt", std::ios::out);
        std::ofstream nOut("n.txt", std::ios::out);

        //For all values of n
        for(int i = 0; i < n.size(); i++)
        {
                //Create a vector of equally spaced n entries between 0 and 1
                Matrix v = Linspace(0, 1, n[i]);

                //Create a nXn matrix
                Matrix A(n[i],n[i]);

                //Define the power
                double power = 0;

                //A(i,j) = v(i)^(i-1)
                for(int x = 0; x < n[i]; x++)
                {
                        for(int y = 0; y < n[i]; y++)
                        {
                                A(y,x) = pow(v(y), power);
                        }
                        power ++;
                }

                //True value of x
                Matrix xMat = Random(n[i]);

                //True value of b with value of x
                Matrix B(n[i]);

                //Find the value of b
                for(int x = 0; x < n[i]; x++)
                {
                        double bi = 0;
                        for(int y = 0; y < n[i]; y++)
                        {
                                double tempA = A(x,y);
                                double tempX = xMat(y);
                                bi += tempA * tempX;
                        }
                        B(x) = bi;
                }

                //Copy matrix because they are modified
                Matrix Acopy1 = A;
                Matrix Bcopy1 = B;

                //Approximate value of x
                Matrix xHat = LinearSolve(Acopy1,Bcopy1);

                //Approximate value of b
                Matrix bHat = A*xHat;

                //Find the residual vector
                Matrix residual = B - bHat;

                //Find the error vector
                Matrix error = xMat - xHat;

                //Calculate the norm of each vector
                double residualNorm = Norm(residual);
                double errorNorm = Norm(error);

                //Print to text file
                residOut <<  residualNorm << "\n";
                errorOut << errorNorm << "\n";
                nOut << n[i] << "\n";
        }
}
```

Below is the result of vandermonde.cpp. Each entry in **n**, **residual**, & **error** are values for the same vandermonde matrix.

```
In [4]: n = open("n.txt").read()
        n = n.split("\n")
        n.remove("")

        residual = open("residual.txt").read()
        residual = residual.split("\n")
        residual.remove("")

        error = open("error.txt").read()
        error = error.split("\n")
        error.remove("")
        print ("n = ", n)
        print ("residual = ", residual)
        print ("error = ", error)

n =  ['5', '9', '17', '33', '65']
residual =  ['0', '1.08921e-15', '1.55431e-15', '25.394', '50.2465']
error =  ['2.82112e-14', '1.61079e-11', '0.00140697', '3.54709', '5.01578']
```

When $ n = 5$ you can see that there is no residual effect of the ill-conditioned matrix and very little error associated with that small of a matrix. However the residual and error only increase as n increases until there is a very high residual and error when $n = 65$ The huge amount or error is caused because the tiny changes in A(i,j) when solving the linear system result in very drastic changes in the result. These small changes only effect the final result more as smaller and smaller values are used in larger and larger matrices.