Hans Guthrie
2/3/2014
ECE 540
Dr. Day

Homework 2: Basic Memory Allocation and Data Display

1. For the first problem we had to write the following functions:
    a. A function that allocates an array of doubles and fills them with values from Start to End. The function was to return a pointer to a double.

```
//This function allocates an array of doubles of length "Elements" on the heap.
//It then fills them with values from Start to End.
double *BuildArray( int Elements, double Start, double End )
{
       double * Array = new double[ Elements ]();      //Allocate the array we will
                                                       return
       double Diff = fabs( End - Start );              //Finds the difference
                                                       between the start and end
       int k;        //The for loop counter
       // Fill the array with numbers
       for ( k = 0; k < Elements; k++ )
       {
       Array[ k ] = Start + ( Diff / Elements )*( k + 1 );  // Calculate what
                                   value goes at each spot in teh array
       }
       return Array;
} //End of BuildArray function
```

    b. Next we were to write a function that filled a given double array pointer with the values of sin(x) for the given number of elements. It then returns a double pointer to the array.

```
//Allocates an array of doubles of length "Elements" and fills it with sin(x).
//Where x is the elements of the array (x) passed in. Returns a pointer to the
//allocated array
double *SineOfArray( double *x, int Elements )
{
       double * SineArray = new double[ Elements ]();         //Allocate space for
                                                              the output array
       int k;        //The for loop counter
       //Fill the array with numbers
       for ( k = 0; k < Elements; k++ )
       {
              SineArray[ k ] = sin( x[ k ] ); //Calculate sin(x) and put it in the
                                                   array
       }

       return SineArray;
}      //End SineOfArray
```

You need to check for valid
allocation of each new array for the
functions that you created
-3

c. The third function was to allocate an array of doubles and fill it with the Absolute Relative Error between two arrays. It then returned a pointer to this array.

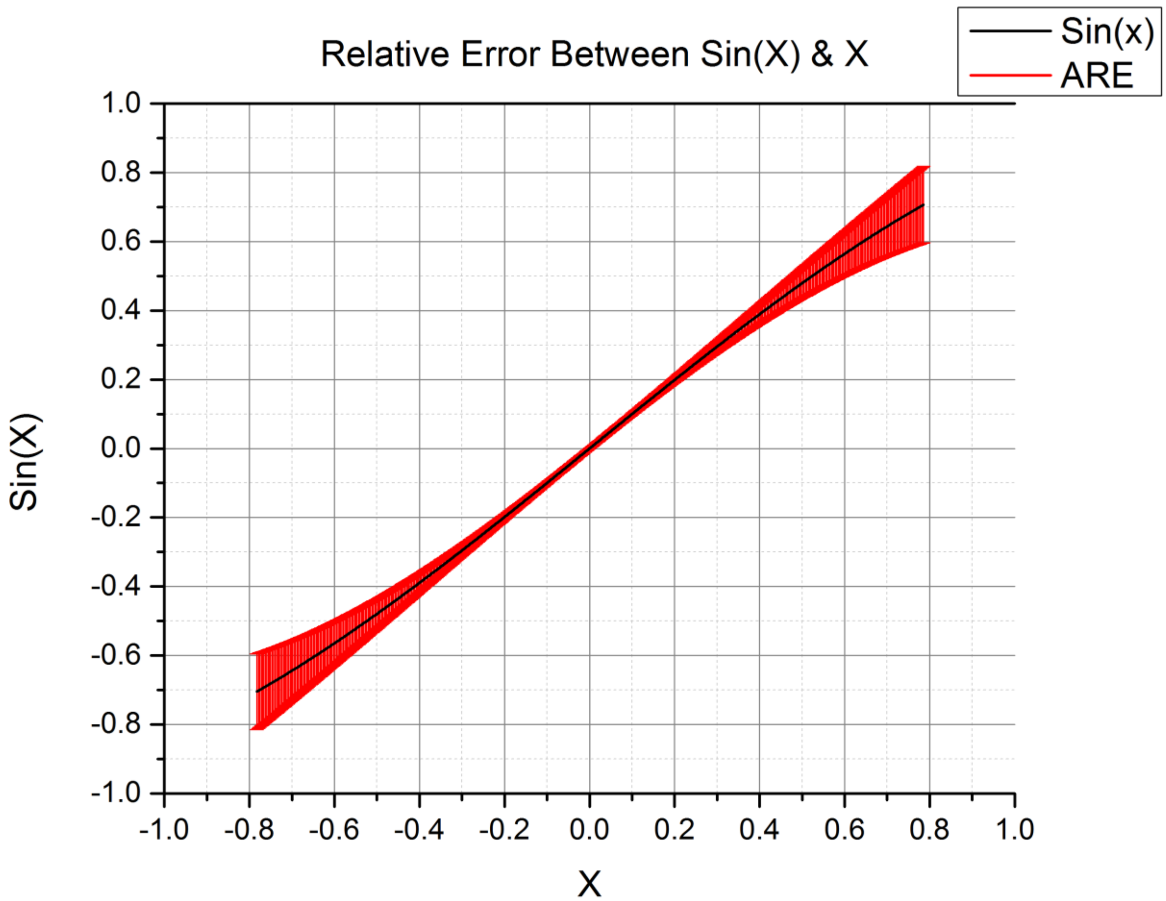
```
//Calculates the relative error between the given two arrays,
//and computers the error between them
double *AbsRelativErrorOfArrays( double *In1, double *In2, int Elements )
{
	double * ErrorArray = new double[ Elements ]();		//Allocate space for
								the output array
	int k;		//The for loop counter
	//Fill the array with numbers
	for ( k = 0; k < Elements; k++ )
	{
		//Calculates percent error for each spot in arrays, and saves to
		ErrorArray
		ErrorArray[ k ] = (fabs( In1[ k ] - In2[ k ] ) / fabs( In2[ k ] ));

	}
	return ErrorArray;
}	//End AbsRelativErrorOfArrays
```

2. Next we were to use the functions we just wrote to calculate the error between sin(x) and x for $|x| < \frac{\pi}{4}$. We computed sin(x) and then the ARE of sin(x) vs x over 500 poits from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$. It then searched through this data to determine the range of x values where the ARE was less than 0.1 %. Note the formula for ARE is $\frac{|\sin(x)-x|}{|\sin(x)|}$.

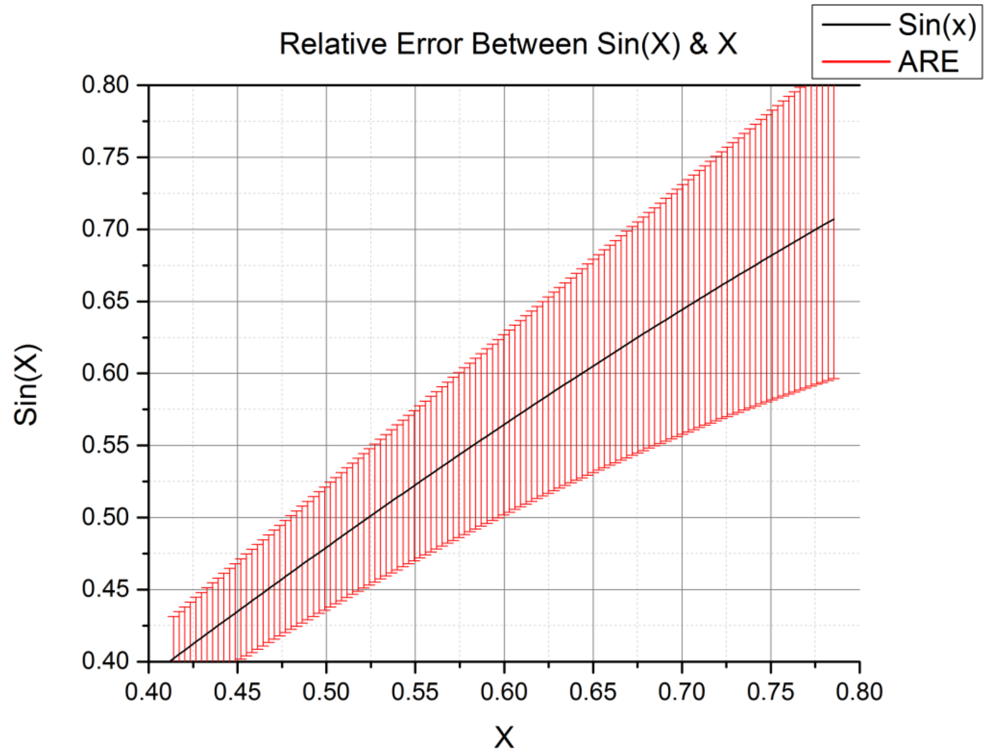
```
//Creates array via BuildArray function
double * Array = BuildArray( LENGTH_OF_TEST, -PI_4, PI_4 );
//Creates the sine array via 'Arr' and the SineArray function
double * SineArray = SineOfArray( Array, LENGTH_OF_TEST );
//Creates the error array via 'Arr' and 'Sine' arrays
double * ErrorArray = AbsRelativErrorOfArrays( Array, SineArray, LENGTH_OF_TEST );

// Calls function to find the lower bound of .001 in the 'Error' array
LowerBound = FindLowerBound( LENGTH_OF_TEST, ErrorArray );
// Calls function to find the upper bound of .001 in the 'Error' array
UpperBound = FindUpperBound( LENGTH_OF_TEST, ErrorArray );
```

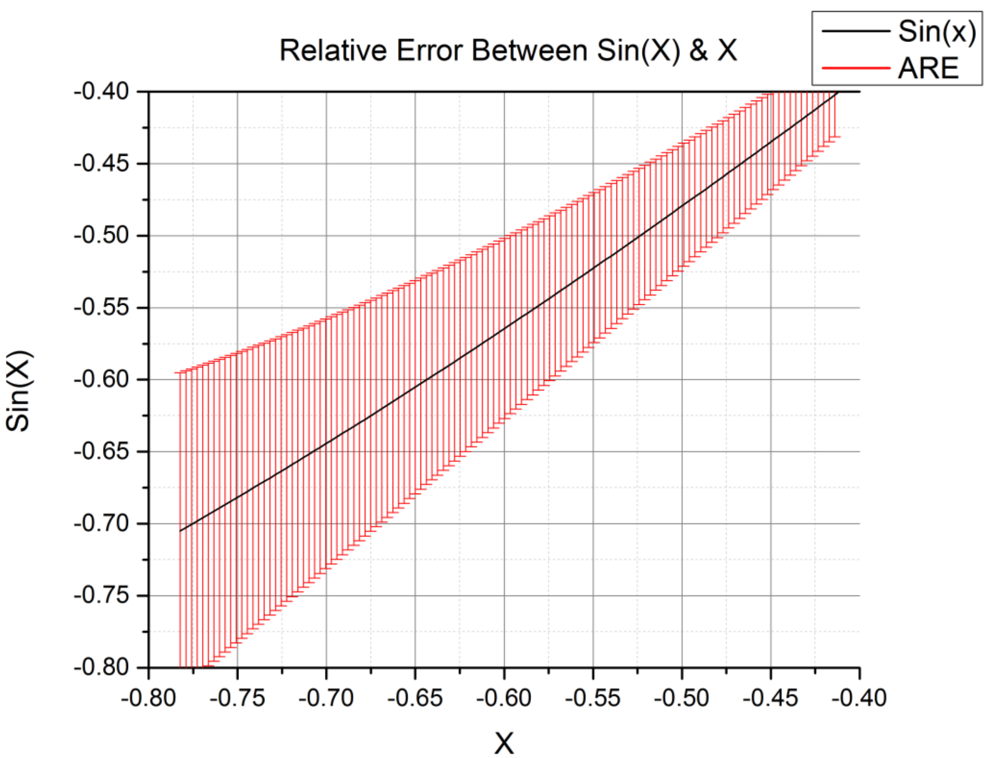
Need to document the lowerbound
and upperbound of your data.
-2

3. Last we were supposed to save the values we generated to a file and then plot them for different scales. I decided to use a program called Origin to plot these instead of Matlab. Here are the plots I came up with.

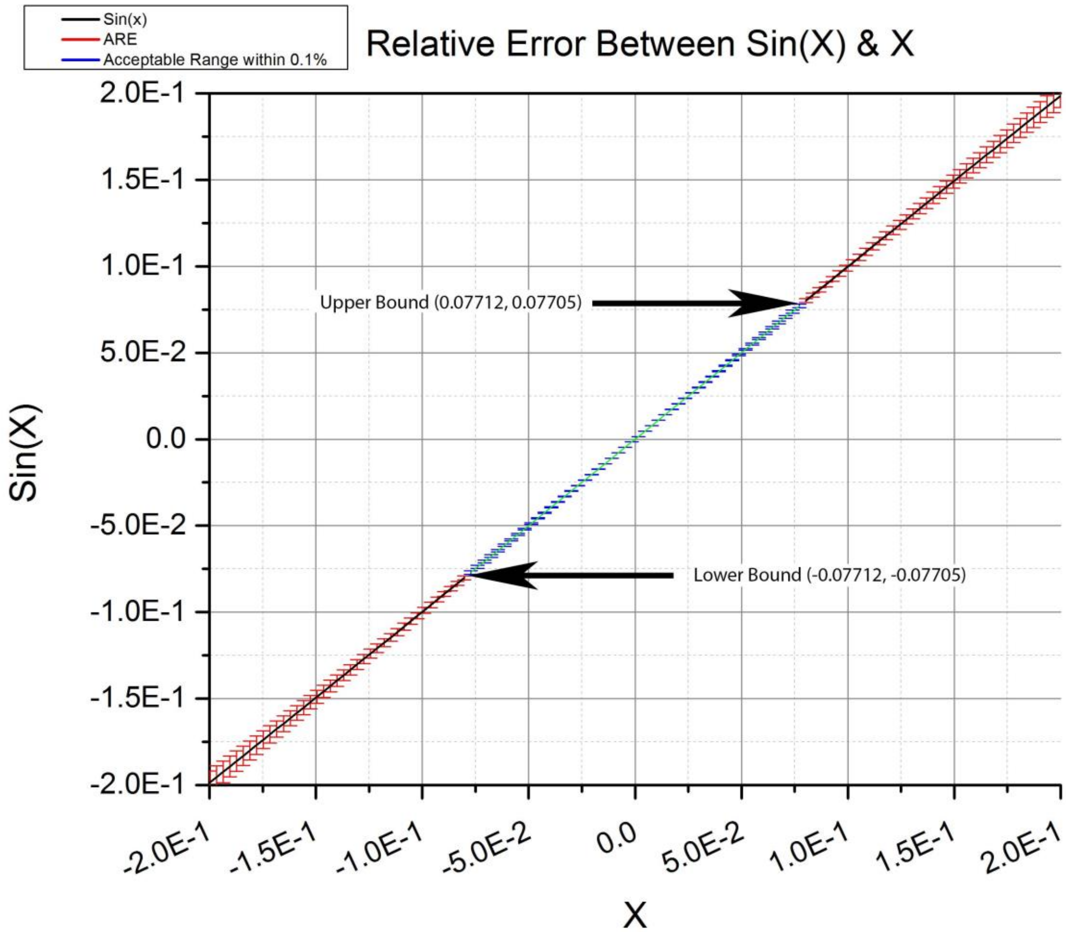


For values close to 0, x is a good approximation for sin(x), however for larger values there is an increasing margin of error. Here are some closeups showing the error at the extreme values.





Here is a graph showing where the ARE is greater than 0.1 percent.



```
0) Coding
      Comments ..............................  2 /2
      Prototypes ............................  0 /1
      Variable Names ........................  1 /1
      Structure .............................  0 /1
1) Functions
      BuildArray Error Check ................  0 /1
                  Formula ...................  1 /1
      SineOfArray Error Check ...............  0 /1
                  Formula ...................  1 /1
      AbsoluteRelativeErrorOfArrays
                  Error Check ...............  0 /1
                  Formula ...................  1 /1
2) Error Search
      Search for error range ................  2 /2
      Documentation of output ...............  0 /2
      Other .................................  1 /1

3) Plotting of Results.
      Proper labels and legend ..............  2 /2
      ARE bound displayed ...................  1 /1
      Proper view ...........................  1 /1

Total .......................................  13/20
```

Need to include your main so I can see the prototypes and structure -2