Homework 1
Math 128A
Due Friday, 10/11/19, 11:59 PM

1. Consider the function
$$f(x) = \frac{1 - \cos(x)}{x^2}.$$
   (a) Evaluate $\lim_{x \to 0} f(x) = L$.
   (b) As $x \to 0$, at what rate does $f(x) \to L$?
   (c) Compute $f(x)$ as written on a computer for values of $x = 10^{-1}, 10^{-2}, \ldots, 10^{-10}$. Comment on your results.
   (d) Suppose that we are able to represent floating point numbers with N decimal digits of accuracy. Around what value of r will the evaluation of $f(x)$ produce very large relative errors when $|x| < r$?
   (e) Rearrange the expression for $f(x)$ to a mathematically equivalent expression so that the this new function evaluates accurately for very small values of x. Verify the success of your rearrangement computationally. Are there values of x where you expect accuracy problems with your rearrangement?

2. The Taylor series about $x = 0$ for the arctangent function converges for $-1 < x \leq 1$ and is given by
$$\tan^{-1}(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}.$$
   Write a computer program to evaluate $\tan^{-1}(x)$ by truncating the series to $N + 1$ terms.
   (a) Use your program to approximate π by evaluating $\tan^{-1}(1)$ for different values of N. Give the approximations and errors in a table. To compute the errors, use that $\pi \approx 3.141592653589793$.
   (b) Repeat part (a) using $x = 3^{-1/2}$.
   (c) Plot the errors in the approximations from parts (a) and (b) vs. the number of terms in the truncated sum on a log-log graph and a semilog (log-linear) graph (MATLAB commands are `loglog` and `semilogy`, respectively). What do these graphs indicate about the rate of convergence?
   (d) Analytically derive bounds for the errors in using the truncated series to approximate π using the series from parts (a) and (b).

3. Suppose we want to approximate the function $f(x) = e^x$ on the interval $[0, 1]$ using a second degree polynomial.
   (a) Let q_2 denote the second degree Taylor polynomial of f about $x = 0$. Derive an upper bound for the magnitude of error in using q_2 to approximate f on $[0, 1]$, i.e. bound the maximum of $|q_2(x) - e^x|$.

(b) Find the second degree polynomial, p_2, that interpolates f at the points $x = 0$, $x = 1/2$, and $x = 1$.

(c) Derive an upper bound for the magnitude of error in using p_2 to approximate f on $[0, 1]$.

(d) Plot f, q_2, and p_2 for $x \in [0, 1]$ on the same figure.

(e) Plot $|q_2(x) - e^x|$ and $|p_2(x) - e^x|$ on the same figure using a log scale for error (in MATLAB use command `semilogy`). Verify by inspection that the error bounds you derived hold, and comment on the quality of the two different approximations over $[0, 1]$.

4. The Taylor series for the function $f(x) = e^x$ converges for all x. This idea can be used to approximate e^x using only addition, multiplication, and division. Below is an example of such a code.

```
% myexp.m -- function for computing y=exp(x) using a Taylor series
%
function [y,Nterms]=myexp(x);
  oldsum = 0;
  newsum = 1;
  term   = 1;

  n      = 0;
  while newsum~=oldsum
    n = n+1;
    term = term*x/n;
    oldsum = newsum;
    newsum = newsum + term;
  end

  Nterms = n + 1;
  y = newsum;
```

(a) Assess the accuracy of the algorithm by using it to approximate $y = e^x$ on the interval $x \in [-20, 20]$ by comparing with the built-in library function for the exponential. Compute the absolute and relative errors as a function of x and plot the results (use log scale for the error; i.e. in MATLAB use command `semilogy` for plotting).

(b) For what values of x do you see poor performance from the algorithm? Explain the reason for the poor performance.

(c) Based on your answer from the previous part, modify the algorithm to eliminate the poor performance. Discuss the changes and demonstrate the performance of the modified code by plotting the errors as a function of x.

5. Given a set of distinct points x_k for $k = 0 \ldots n$, the j^{th} Lagrange interpolating polynomial is the unique degree n polynomial which satisfies

$$L_j(x_i) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Write a program to evaluate L_j.

(a) Let $x_k = -1 + 2k/n$ for some integer n for $k = 0 \ldots n$; these are $n + 1$ equally spaced points on $[-1, 1]$. For $n = 5$ plot all six Lagrange interpolating polynomials on the same figure. Repeat for $n = 10$ and $n = 15$.

(b) Repeat the previous part for the Chebyshev points:

$$x_k = \cos\left(\frac{2k+1}{2(n+1)}\pi\right), \quad k = 0 \ldots n.$$

(c) Discuss the difference in the plots from part (a) and part (b). In particular, note that high degree polynomial interpolants are known to exhibit large oscillations even for non oscillatory data unless one is careful about points are used for interpolation. Do your plots give some insight into this phenomenon? Which set of point locations is more likely to result in oscillatory interpolants?