Homework 1

Collaborators: Ryan Campbell,

Problem 1

(T) (10 points) Assume we have a floating point number system, represented in binary, which allocates 1 bit to sign, 4 bits to the mantissa (q) and 4 bits to the characteristic (c), and our numbers are of the form $\pm(1+q)(2c-7)$

(a) What is machine epsilon?

(b) Find the floating point representation of 71.

(c) Find the floating point representation of π.

(d) Comment on the relative errors of your representations from (b) and (c)

Problem 2

(T) (12 points) Near certain values of x each of the following functions cannot be accurately computed using the formula as given due to cancellation error. Identify the values of x and propose a reformulation to remedy the problem.

(a) $f(x) = 1 + \cos x$

(b) $f(x) = 1 - 2\sin 2x$

(c) $f(x) = ln(x) - 1$

Problem 3

(T) (10 points)

(a) For which positive integer(s) k can the number $5 + 2 - k$ be represented exactly in double precision floating point arithmetic?

(b) Find the largest integer k for which $fl(19 + 2 - k) > fl(19)$ in double precision floating point arithmetic.

Problem 4

(T) (13 points)

(a) What is the condition number of evaluation of the function $f(x) = e^{\cos x}$ at the point x. About how many digits of accuracy would you expect to lose when performing this operation at $x = 1000$?

(b) Suppose that f and g are continuously differentiable functions. Let $\kappa_f(x)$ denote the condition number of evaulation of the function f at x and similarly for $\kappa_g(x)$ Find a relationship between the condition number of evaluation of $h(x) = f(x)g(x)$ and κ_f and κ_g.

(c) Let $\kappa_f(x)$ denote the conditon number of evaluation of the function f at the point x. Find a function f which is infinitely differentiable on the interval $(0, 1)$ (but which may have singularities at $x = 0$) such that

$$\lim_{x \to 0^+} \kappa_f(x) = \infty$$

Problem 5

(23 points) Consider the function

$$f(x) = \frac{1 - cos(x)}{x^2}$$

(a) (T) Evaluate $\lim_{x \to 0} f(x) = L$.

(b) (T)As $x \to 0$, at what rate does $f(x) \to L$?

(c) Compute $f(x)$ as written on a computer for values of $x = 10 - 1, 10 - 2, ..., 10 - 10$. 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?

Problem 6

(19 points) Construct and implement an algorithm to approximate an integral of a function f using (i) midpoint rule and (ii) trapezoidal rule.

(a) Use these algorithms to approximate the integral $\int_0^{0.1} xe^{-(20x)^2} dx$ for N = 32, 64, ..., 4096 subintervals. Report your absolute and relative errors.

(b) Plot a refinement study comparing these two methods, as well as the right-hand Riemann sum. Comment on your convergence results.

```
%function: integrate from 0 to 0.1, xe^-(20x)^2 for N = 32, 64, 4096
%subintervals
mp = 0; %midpoint rule: detla x = (b-a)/n for n subintervals and mi is the midpoint of
tp = 0; %trapezoid rule: delta x = [f(x_0) + 2(fx1)+..+2f(x_n)]

%bounds upb = upper bound, lowb = lower bound
u = 0;
l = 0.1;

%interval
N1 = 32;
N2 = 64;
N3 = 4096;

delta_x = (u-l)/(N1); %delta x for both trapezoid and midpoint rule
x = [l:delta_x:u]; %x for both trapezoid and midpoint rule

%function
```

```
F_n = x*exp()
```

```
Error using exp
Not enough input arguments.
```

```
f = @(x) (x.*exp(-(20.*x).^2));

% Loop to sum Midpoint Rule and Trapezoid Rule
for i = 1:N1
    tp = tp+(f(i)+f(i+1))/2*delta_x
    mp = mp + f((2*i+1)/2)*delta_x
end

tp = tp
mp = mp
```

Problem 7

(22 points) The Taylor series about $x = 0$ for the arctangent function converges for $-1 < x \leq 1$ and is given by

$$\tan^{-1}(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$$

(a) Write a computer program to evaluate $\tan^{-1}(x)$ by truncating the series to $N + 1$ terms.

(b) 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 $\pi \approx 3.141592653589793$.

(c) Repeat part (b) using $x = 3^{-1/2}$.

(d) Plot the errors in the approximations from parts (b) and (c) vs. the number of terms in the truncated sum on a log-log graph and a semilog (log-linear) graph. What do these graphs indicate about the rate of convergence?

(e) (T) Analytically derive a bound for the error for the series in parts (b) and (c).

Problem 8

(20 points) You are running a simulation that updates time every 0.1 seconds. The time in the simulation is kept by incrementing a time variable. See the below code.

```
dt = 0.1;          % time step
t = 0;             % initialize time
Nsteps = 86400;  % 86,400 is the number of steps to take for 1 day

% loop in time
```

```
%
for j=1:Nsteps

    %
    % SOME SIMULATION
    %

    % update time
    %
      t = t + dt;
end
```

(a) Suppose you are simulating one day (86,400 s). Implement the above code, and compute the absolute and relative errors in the time at the end of the simulation.

```
dt = 0.1;          % time step
t = 0;             % initialize time
Nsteps = 86400;  % 86,400 is the number of steps to take for 1 day

% loop in time
%
for j=1:Nsteps

    %
    % SOME SIMULATION
    %

    % update time
    %
      t = t + dt;
end

Absolute_error = abs(t-t)                                          %|I_n - I|
```

```
Absolute_error = 0
```

```
Relative_error =  abs((t-t)/t)                                      %|(I_n - I)/I|
```

```
Relative_error = 0
```

(b) Change the time increment to 0.125 and again run the simulation to 1 day. What are the relative and absolute error in the time?

```
dt = 0.125;          % time step
t = 0;             % initialize time
Nsteps = 86400;  % 86,400 is the number of steps to take for 1 day

% loop in time
%
for j=1:Nsteps

    %
    % SOME SIMULATION
```

```
    %

    % update time
    %
      t = t + dt;
end

Absolute_error = abs(t-t)                                    %|I_n - I|
```

```
Absolute_error = 0
```

```
Relative_error =  abs((t-t)/t)                                %|(I_n - I)/I|
```

```
Relative_error = 0
```

(c) Explain the difference in the results from parts (a) and (b).

Problem 9

(22 points) Suppose that you can only use addition, subtraction, multiplication, division, rounding and integer powers of numbers. You decide to use a Taylor Series to evaluate $y = e^x$ with only these operations, since you learned in calculus that it converges for all x. Below gives an example of such code.

(a) Assess the accuracy of the algorithm below 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.

```
% 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;
```

Problem 10

(25 points) Suppose we want to approximate the function $f(x) = \ln x$ on the interval $[1, 3]$ using a second degree polynomial.

(a) First, let q_2 denote the second degree Taylor polynomial of f about $x = 1$. Derive an upper bound for the magnitude of error in using q_2 to approximate f on $[1, 3]$, i.e. bound the maximum of $|q_2(x) - \ln x|$.

(b) Next, find the second degree polynomial, p_2, that interpolates f at the points $x = 1$, $x = 2$, and $x = 3$.

(c) Derive an upper bound for the magnitude of error in using p_2 to approximate f on $[1, 3]$.

(d) Plot f, q_2, and p_2 for $x \in [1, 3]$ on the same figure.

(e) Plot $|q_2(x) - \ln x|$ and $|p_2(x) - \ln x|$ 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 $[1, 3]$.

Problem 11

(24 points) Given a set of distinct points x_k for $k = 0...n$, 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 $k = 0...n$; 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:

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

(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?

Problem 12 (Bonus)

(BONUS 20 points) Suppose that you can only use addition, subtraction, multiplication, division, rounding and integer powers of numbers. You decide to use a truncated Taylor Series to evaluate $y = e^x$ with only these operations.

(a) Create a function that approximates e^x by truncating the series to n terms. Use your function for $n = 12$ to approximate e^x for some values of x between -100 and 100. Repeat for $n = 50$. Comment on your results.

(b) By exploiting properties of the exponential, design an algorithm for accurately computing the value of e^x using your truncated series for x values between -100 and 100. Explain your algorithm and implement it. Report your relative error for e^x for $x = \pm 0.5$ and $x = \pm 100$, and compare to the previous part. You should be able to acheive relative errors below 10^{-13}. Hint: Based on the Taylor Series remainder, for what values of x do you expect the series to be the most accurate? Exploit properties of the exponential to make an algorithm that only uses the series on this range of x values.

Problem 13 (Bonus)

(BONUS 16 points) Write a program to use the Barycentric form to evaluate a Lagrange interpolation polynomial. The polynomial degree, and the interpolation points should be inputs.

(a) Use this program to use a Lagrange interpolation polynomial to approximate the function $f(x) = e^{\frac{x}{3}}$ with a polynomial of degree 10 at the point $x = \pi/6$. First use the points given in Problem (11a). Then repeat for the Chebyshev points given in (11b).