Math 541: Final
Due: 05/13, 5:00 PM

1. (**15 pts**) For the function

$$f(x) = e^x \cos(x),$$

(a) (**5 pts**) Find a general formula for its Taylor series around $x_0 = 0$. Note, to make short work of this, use the fact that around $x_0 = 0$ you know

$$e^x = \sum_{j=0}^{\infty} \frac{x^j}{j!}, \;\; \cos(x) = \sum_{j=0}^{\infty} \frac{(-1)^j}{(2j)!} x^{2j}.$$

Then we know that if we have two infinite series say $\sum_j a_j x^j$ and $\sum_k b_k x^k$ then

$$\left(\sum_{j=0}^{\infty} a_j x^j \right) \left(\sum_{k=0}^{\infty} b_k x^k \right) = \sum_{l=0}^{\infty} c_l x^l, \;\; c_l = \sum_{j=0}^{l} a_j b_{l-j}.$$

(b) (**5 pts**) In Matlab, produce one plot showing $f(x)$, $T_1(x)$, $T_2(x)$, and $T_3(x)$ over the interval $x \in [-1, 1]$. Comment on the improvement of accuracy in using the higher order Taylor series approximations. Note, using different linestyles and colors will help here. Also, maybe use a legend in your figure...

(c) (**5 pts**) In Matlab, produce one plot showing $R_1(x)$, $R_2(x)$, and $R_3(x)$ over the interval $x \in [-1, 1]$. Comment on the improvement of accuracy in using the higher order Taylor series approximations. Likewise, using the Taylor series remainder theorem, find a value M such that

$$|R_3(x)| \leq \frac{M}{4!} |x|^4, \;\; x \in [-1, 1].$$

Plot $R_3(x)$, $M|x|^4/4!$, and $-M|x|^4/4!$ in one graph for $x \in [-1, 1]$. Comment on your results. Again, different line styles and colors will be super useful here.

2. (**20 pts**) The (p, q) hypergeometric function is defined as

$$ {}_pF_q(a_1, \cdots, a_p; b_1, \cdots, b_q; x) = \sum_{n=0}^{\infty} \frac{(a_1)_n \cdots (a_p)_n}{(b_1)_n \cdots (b_q)_n} \frac{x^n}{n!} $$

where $a_j \in \mathbb{R}$, $b_l \in \mathbb{R}$ and for any real value a_j we have that

$$ \begin{aligned} (a_j)_0 =& 1 \\ (a_j)_n =& a_j(a_j + 1)(a_j + 2) \cdots (a_j + n - 1), \ n \geq 1. \end{aligned} $$

So for example, we have that

$$ {}_0F_0(;;x) = \sum_{n=0}^{\infty} \frac{1}{1} \frac{x^n}{n!} = e^x, $$

and if we have the Bessel function $J_n(x)$ where

$$ J_n(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(m+n)!} \left(\frac{x}{2}\right)^{2m+n} = \frac{1}{n!} \left(\frac{x}{2}\right)^n \sum_{m=0}^{\infty} \frac{1}{(n+1)_m} \frac{1}{m!} \left(\frac{-x^2}{4}\right)^m, $$

since

$$ \begin{aligned} (m+n)! =& n!(n+1)(n+2) \cdots (n+m) \\ =& n!(n+1)(n+1+1) \cdots (n+1+m-1) \\ =& n!(n+1)_m, \end{aligned} $$

then we have

$$ J_n(x) = \frac{1}{n!} \left(\frac{x}{2}\right)^n \ {}_0F_1\left(;n+1;-\frac{x^2}{4}\right). $$

Write a program which computes the (p, q) hypergeometric function. It should take as input vectors **a** and **b** where

$$ \begin{aligned} \mathbf{a} =& (a_1, \ a_2, \ \cdots, \ a_p) \\ \mathbf{b} =& (b_1, \ b_2, \ \cdots, \ b_q) \end{aligned} $$

and an evaluation point x. Note, your program should determine p and q using the length of a and b respectively. Make sure you vectorize your function. Make sure you make efficient use of recursion. Clearly explain the stopping criteria you choose and why you choose it. (**10 pts**) Test your code by comparing your results to those you would get using the "bessel plotter" code. Provide plots to explain your validation. (**5 pts**)

Then, using Matlab's version of the functions as the "true" values, numerically prove the identities

$$\ln(1+x) = x \cdot_2 F_1\left(1, 1; 2; -x\right)$$

Generate semi-log plots of the difference of the two functions over meshes on $[-.5, .5]$ and argue why this shows the identities are true. Note, you may need to adapt your stopping condition to adequately answer this problem. (**5 pts**).

3. (a) (**5 pts**) Using Newton's method, for a positive number A, build an algorithm which finds

$$\frac{1}{2}\left(A + \sqrt{A^2+1}\right).$$

Hint, think about using quadratic equations of the form

$$f(x) = x^2 + bx + c.$$

(b) (**5 pts**) In class, we have shown we can think of Newton's method as an iterative scheme i.e.we define a function $g(x)$ such that

$$x_{j+1} = g(x_j) = x_j - \frac{f(x_j)}{f'(x_j)}.$$

Then, for a root x_* of $f(x)$ for which $f'(x_*) \neq 0$, we have shown

$$g(x_*) = x_*,\ g'(x_*) = 0,$$

and

$$|x_{j+1} - x_*| = \frac{1}{2}\left|g''(\xi)\right| |x_j - x_*|^2,\ \xi \in (x_*, x_j) \text{ or } (x_j, x_*).$$

Using your result in the first part, assuming that

$$x_j = \frac{1}{2}\left(A + \sqrt{A^2+1}\right) + \epsilon_j,\ \ |\epsilon_j| \text{ "is very small"},$$

show that

$$\left|g''(\xi)\right| \sim \frac{2}{\sqrt{A^2+1}}.$$

Using this result, what would you expect the effect on convergence would be for large choices of A?

(c) (**5 pts**) Choose several values of A and the impact of your choice on how quickly your algorithm for finding $\frac{1}{2}\left(A + \sqrt{A^2+1}\right)$ converges. While log-log plots help find the rate of convergence, here, you might think of a different approach. Does the theoretical result you derived in the previous part accurately determine which values of A produce slower or faster convergence? What is the role of your initial guess x_0 in determining how quickly your method converges?

4. (**15 pts**) In order to approximate the integral

$$\int_{x_j}^{x_{j+1}} f(x)dx,\ \Delta x = x_{j+1} - x_j,$$

we can use the Midpoint Method in which we use the approximation

$$f(x) \approx f(x_{j+1/2}),\ x_{j+1/2} = (x_j + x_{j+1})/2.$$

(a) (**5 pts**) Show that this choice yields the approxmation

$$\int_{x_j}^{x_{j+1}} f(x)dx \approx \Delta x f(x_{j+1/2}).$$

Further show that

$$\left| \int_{x_j}^{x_{j+1}} (f(x) - f(x_{j+1/2}))dx \right| \leq \frac{1}{2} \max_{\xi \in [x_j, x_{j+1}]} \left| f''(\xi) \right| \int_{x_j}^{x_{j+1}} (x - x_{j+1/2})^2 dx,$$

and thus that you have the local trunctation error

$$\left| \int_{x_j}^{x_{j+1}} (f(x) - f(x_{j+1/2}))dx \right| \leq \frac{1}{24} \max_{\xi \in [x_j, x_{j+1}]} \left| f''(\xi) \right| (\Delta x)^3.$$

(b) (**5 pts**) Provide an argument which shows that for the Midpoint Method, we have the global error result

$$\left| \int_a^b f(x)dx - \Delta x \sum_{j=0}^{N-1} f(x_{j+1/2}) \right| \leq \frac{(b-a)}{24} \max_{\xi \in [a,b]} \left| f''(\xi) \right| (\Delta x)^2,$$

where

$$x_0 = a,\ x_N = b,\ \Delta x = \frac{b-a}{N}.$$

(c) (**5 pts**) For the function $f(x) = \cos^2(x)$ on $x \in [0, 2]$, implement the Midpoint Method and using a semi-log plot confirm the error analysis you performed above. Do you get the predicted behavior for the error as a function of the number of mesh points N?

5. (**15 pts**) While we will talk about partial pivoting to solve $A\mathbf{x} = \mathbf{b}$, that is not the only game in town. We say a square, $n \times n$ matrix A is diagonally dominant if

$$|A_{jj}| > \sum_{k=1, k \neq j}^{n} |A_{jk}|.$$

If A is diagonally dominant and we write

$$A = D + (A - D) = D + R,$$

where D is the diagonal of A, i.e. $D_{jj} = A_{jj}$ so that R is just A with a zero diagonal, then we can solve

$$A\mathbf{x} = \mathbf{b}$$

using the iterative scheme

$$\mathbf{x}_k = -D^{-1}R\mathbf{x}_{k-1} + D^{-1}\mathbf{b},\ k > 1.$$

This is called Jacobi iteration. The idea is that if A is diagonally dominant, then as $k \to \infty$

$$\lim_{k\to\infty} ||\mathbf{x}_k - \mathbf{x}|| = 0,$$

so that we iterate towards a solution in the same way we iterate to roots in Newton's method. Also, just like with Newton's method, you need to provide an initial guess $\mathbf{x}_0$ to get the scheme started. Likewise, we need a stopping criterion such as iterate until

$$||\mathbf{x}_k - \mathbf{x}_{k-1}||_2 \leq \text{tol}.$$

(a) (**5 pts**) Write a program that takes in a matrix and determines if it is diagonally dominant. Note, Matlab commands like `diag` will be very useful.

(b) (**5 pts**) Write a program that takes in a matrix A and right hand side vector $\mathbf{b}$ and performs Jacobi iteration on a diagonally dominant matrix. If the input is not diagonally dominant, ake sure you let the user know they are being silly.

Also keep in mind that if D is a $n \times n$ diagonal matrix with non-zero diagonal entries d_{jj}, then D^{-1} is also diagonal with non-zero diagonal entries $1/d_{jj}$. Thus for example, instead of computing $D^{-1}\mathbf{b}$ at every iteration, you can just use the fact that

$$D^{-1}\mathbf{b} = \left(\frac{b_1}{d_{11}}, \frac{b_2}{d_{22}}, \cdots, \frac{b_n}{d_{nn}}\right)^T,$$

so that you compute this once and then use it over and over again. You should do something similar for $D^{-1}R$.

(c) (**5 pts**) Define the rate of convergence α for Jacobi's method to be

$$\lim_{k\to\infty} \frac{||\mathbf{x}_k - \mathbf{x}||_2}{||\mathbf{x}_{k-1} - \mathbf{x}||_2^\alpha} = \lambda.$$

Using the matrix A given by

$$A = \begin{pmatrix} 6 & .1 & 1.9 & 2.1 & .5 & .3 & 1 \\ 3 & -8 & 1 & -.4 & -.2 & -.7 & 2 \\ 1.9 & .1 & 6 & 2.1 & .5 & .3 & 1 \\ 3 & .4 & 1 & -8 & -.2 & -.7 & 2 \\ .5 & .1 & 1.9 & 2.1 & 6 & .3 & 1 \\ 3 & -.7 & 1 & -.4 & -.2 & 8 & 2 \\ 1 & .1 & 1.9 & 2.1 & .5 & .3 & -6 \end{pmatrix}$$

and $\mathbf{b}$ given by

$$\mathbf{b} = \begin{pmatrix} 7 \\ 3 \\ 2 \\ 1.3 \\ -2.3 \\ 4.1 \\ 6.36 \end{pmatrix},$$

for a tolerance of 10^{-12}, determine the rate of convergence of Jacobi's method. Produce a log-log plot which justifies your conclusions. Note, you can use $\mathbf{x} = A\backslash\mathbf{b}$ in Matlab for the exact solution.