Midterm 553.386, March 4, 2019

Do all four of the following problems. Show all your work. Answers without supporting work may receive no credit.

Students may discuss the exam only with the instructor and the teaching assistant. No discussion of the exam contents, directly or indirectly, is permitted among students or with any third parties. Any book or internet resource may be used, as long as the book or the website are cited, along with the material taken from it.

You may use any numerical software available, unless you are specifically instructed in the problem statement to write your own code. All codes that are written by you should be turned in with the exam, either as paper printouts or preferably as a Matlab script sent by e-mail to the instructor. Numerical results without the code that produced them will receive no credit.

I attest that I have completed this exam without unauthorized assistance from any person, materials, or device:

Full name: ______________________________________

Signature: ______________________________________

(See the Johns Hopkins Handbook Academic Ethics for Undergraduates).
**Problem 1.** As we shall discuss later in the course, the *Euler method* for solving the initial-value problem

\[ \frac{dx}{dt} = f(t, x), \quad t_0 < t < t_1; \quad x(t_0) = x_0 \]

can be obtained by replacing the time-derivative in the above equation by a difference-quotient \[ x(t_1 + h) - x(t_1)/h \approx \dot{x}(t), \]
so that the approximate solution \( x_n = x(t_n) \) at a sequence of times \( t_n = t_0 + nh, \) \( n = 0, 1, 2, 3, \ldots \) is obtained by iterating

\[ \frac{x_{n+1} - x_n}{h} = f(t_n, x_n), \quad x_0 = x_0 \]

for some small value of \( h. \) At the following URL

[http://www.ams.jhu.edu/~eyink/SciCompODE/seuler.m](http://www.ams.jhu.edu/~eyink/SciCompODE/seuler.m)

you can download a Matlab function file `seuler.m` that implements the Euler approximation in *single-precision* arithmetic. It is written so that

\[ \gg [t,x]=seuler(f,[t0,t1],x0,N,0); \]

will output the vectors \( t = (t_n|, n = 0, 1, .., N), \) \( x = (x_n|, n = 0, 1, .., N), \) for given function \( f(t, x), \) time-interval \( [t_0, t_1], \) initial condition \( x_0, \) and number of steps \( N, \) or, equivalently, stepsize \( h = (t_1 - t_0)/N. \)

(a) Apply the Euler method in single-precision arithmetic to approximate the solution of the initial-value problem

\[ \frac{dx}{dt} = x, \quad 0 < t < 1; \quad x(0) = 1 \]

with exact solution \( X(t) = e^t. \) Use a number of steps \( N_m = 10^m \) for \( m = 1, 2, 3, ..., 6 \) and for each value of \( m \) calculate the maximum relative error

\[ \text{relerr}(m) = \max\{|x_n/X(t_n) - 1| : n = 0, 1, 2, \ldots, N_m\}. \]

Plot the relative errors using the command

\[ \gg \text{semilogy}(1:6,\text{relerr}) \]

Does the relative error decrease monotonically as \( N = 10^m \) increases? What value of \( m \) gives the minimum relative error? Are the Euler approximations for this problem ever accurate to single precision?
(b) To try to understand the observations in part (a), consider the local truncation error per step
\[ \tau(t_n) = \frac{X(t_n + h) - X(t_n)}{h} - f(t_n, X(t_n)) \]
obtained when the exact solution is substituted into the Euler method, with infinite-precision arithmetic. Use Taylor series to expand this error in powers of \( h \) for a general smooth \( f(t, x) \) and \( X(t) \). What is the leading-order term in \( h \)? Do you ever see the relative error in part (a) decreasing proportional to this power of \( h \)?

(c) Now consider the local truncation error per step for the exact solution \( \hat{X}(t) \) approximated in finite-precision arithmetic
\[ \hat{\tau}(t_n) = \frac{\hat{X}(t_n + h) - \hat{X}(t_n)}{h} - f(t_n, \hat{X}(t_n)) \].
Write this error as the sum of the truncation error in part (b) and a term that depends upon the round-off errors \( \delta X(t_n) = \hat{X}(t_n) - X(t_n) \). Obtain an estimate of the truncation error \( \hat{\tau}(t_n) \) in terms of \( h \), the unit round \( u \) of the machine arithmetic, and the absolute values \( |X(t_n)|, |\dot{X}(t_n)| \).

(d) What is the unit round \( u \) in IEEE standard single-precision arithmetic? Use this value and the result in part (c) for the specific choice \( f(t, x) = x, X(t) = e^t \) in order to estimate the optimal choice \( h_\ast \) of \( h \) which should minimize the local truncation error for Euler method in single-precision arithmetic. Use this estimate of \( h_\ast \) to explain your observations in part (a).
Problem 2. Here we consider the second-order initial-value problem

$$\frac{d^2x}{dt^2} = x, \quad x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0.$$ 

(a) Find an explicit formula for the solution $x(t; x_0, \dot{x}_0)$ as a function of the input parameters $x_0$, $\dot{x}_0$. (Hint: Recall that the general solution of such linear differential equations is a superposition of exponentials $e^{\lambda t}$, with suitable values of $\lambda$). Is this problem well-posed? Explain your answer.

(b) Calculate the relative condition numbers

$$K_{x_0}(x(t)) = \frac{x_0}{x(t)} \frac{\partial x(t)}{\partial x_0}, \quad K_{\dot{x}_0}(x(t)) = \frac{\dot{x}_0}{x(t)} \frac{\partial x(t)}{\partial \dot{x}_0}$$

for variations in $x_0$ and $\dot{x}_0$, respectively. What is the general definition of condition number and why do the above expressions represent condition numbers? What is the requirement on the inputs so that these condition numbers remain finite as $t \to \infty$?

(c) A particular solution of the initial-value problem is $x(t) = e^{-t}$ for the choice of inputs $x_0 = 1$, $\dot{x}_0 = -1$. Use the general results in part (b) to calculate $K_{x_0}(x(t))$, $K_{\dot{x}_0}(x(t))$ for this solution. If the solution is calculated in double-precision arithmetic, estimate the time $t_*$ at which all significant figures are lost in the approximation.
Problem 3. We study the use of the Newton algorithm to approximate the root $x_* = 0$ of the following two functions:

\[(i) \quad f(x) = 1 + x - \sqrt{1 + x^2}, \quad (ii) \quad f(x) = |x|^{1/2}\text{sign}(x),\]

(a) Plot the function $f(x)$ in (i) over the range $[-5, 5]$ and explain geometrically why the Newton method with $|x_0| \gg 1$ will result in order-unity values of $x_n$ for either $n = 1$ or for $n = 2$. In fact, it is known that the Newton method converges for this function to the root $x_* = 0$ globally for all choices of $x_0$.

(b) Apply the Newton method numerically for the function in (i) with $x_0 = 10^8$. Take tolerance $TOL = 10^{-15}$ and maximum number of iterations $MAXIT = 100$. What happens? Explain the source of the problem. Write Matlab function files \texttt{f.m} and \texttt{Df.m} with improved representations of the functions $f(x)$, $f'(x)$ that avoid the problem and verify that the Newton iteration then converges with $x_0 = 10^8$, $TOL = 10^{-15}$, $MAXIT = 100$.

(c) For the function $f(x)$ in (ii), calculate the Newton iterate $x_{n+1}$ explicitly in terms of the previous iterate $x_n$. Using this result, does the Newton method converge for this function with any choice of $x_0 \neq 0$?

(d) Why does the theorem on local convergence of the Newton method not apply to the function in (ii)? What assumption of the theorem is not valid?
Problem 4. We study the use of the Newton algorithm to approximate the root 
\(x^* = 0\) of the function 
\[ f(x) = e^{-1/x^2}, \ x \neq 0; \quad f(0) = 0 \]

(a) Apply the Newton method numerically for this function with \(x_0 = 1\), \(TOL = 10^{-15}\) and maximum number of iterations \(MAXIT = 500\). Is the method converging quadratically? Why not? It might be helpful to plot the function \(f(x)\) to understand the problem.

(b) Now apply the Newton method for this function numerically with \(x_0 = 0.01\) and all other inputs the same. What happens, and why?

(c) Find the smallest value \(x_m\) in IEEE double-precision arithmetic so that \(f(x_m) \neq 0\). Assume that denormalized numbers are represented. Now apply the Newton method for this function numerically with \(x_0 = x_m\) and all other inputs the same. What happens, and why?

(d) To avoid the NAN’s encountered in parts (b),(c), explicitly evaluate the ratio \(f(x)/f'(x)\) as a function of \(x\) and use this expression to implement the Newton iteration. Try this new implementation of Newton’s method numerically for \(x_0 = x_m\). Does the method converge within 500 iterations?

(e) Now apply the new implementation of Newton’s method in part (d) numerically with \(x_0 = 10^{-8}\). What happens? Does the algorithm find the root \(x^* = 0\)? Why is the convergence criterion satisfied?

(f) Find the smallest value \(x_m^*\) so that the convergence criterion is not immediately satisfied. Apply the implementation of Newton’s method in part (d) numerically again, now with \(x_0 = x_m^*\). Does the method converge within 500 iterations?