Math 4363/5383 – Numerical Analysis Fall 2023
Program 1 Due Friday, September 13, 2024

1. Write a routine to approximate the machine epsilon. Call this function subroutine macheps; the first line of macheps.m should be
   *function [mu] = macheps*
   This routine does not have any input variables.

2. Write a routine to find the roots of $ax^2 + bx + c$ with $a,\ b,\ c \in \mathbb{R}$ using (a variation of) the quadratic formula. Call this function subroutine quadroot; the first line of quadroot.m should be
   *function [x1,x2,flag] = quadroot(a,b,c)*

   This code should

   (a) Take as input the real numbers $a,\ b$ and $c$. Return as output the "roots" $x_1$ and $x_2$, and an information flag, *flag*.

   (b) Use real arithmetic (do not attempt to take the square root of a negative number). In the case of complex conjugate roots, your program should tell the user that $x_1$ is the real part and $x_2$ is the imaginary part. You should use the third output variable, flag, for such messages.

   (c) Avoid cancellation where possible.

   (d) Try to avoid overflow by (i) scaling, and then (ii) avoiding divides by anything "really-close-to-zero" (this is not the only way to get overflow, but it is definitely the most common). If $x_1$ and/or $x_2$ are not returned, they should be set to NaN's, and the error flag should indicate this.

   (e) Be documented. Most importantly, you need to explain – *leaving no ambiguity* – what each input and output variable is and how the output should be interpreted.

3. See https://arnold.hosted.uark.edu/NA/Pages/function_skel.pdf and https://arnold.hosted.uark.edu/NA/Pages/Matlab1.pdf for help with this assignment.

4. When you are satisfied your program works, run the testing routine NAProg1Test.m (https://arnold.hosted.uark.edu/NA/prog.html); it will generate an output file called prog1run.txt. Email macheps.m, quadroot.m and prog1run.txt to arnold@uark.edu. Also, hand in printouts of macheps.m, quadroot.m, and prog1run.txt (in class or slide under my office door).

Hint: The roots of a polynomial are not changed if the coefficients are multiplied by a non-zero constant (so we can scale this problem easily). Your problem should probably first be scaled so that the |largest| coefficient is between, say, 1/2 and 2 in magnitude.