University of Colorado
Department of Computer Science

Numerical Computation **CSCI 3656**

Spring 2016

Exam 1 Solutions

1. Dana Codehacker has written the following chunk of code:

```
function answer = foo(a,b,tolerance)
function f = bar(x)
   f = sin(x) + 0.25;
end
answer = (a*f(b) - b*f(a))/(f(b) - f(a));
err = abs(f(answer));
prev = b;
while err > tolerance
if f(a)*f(answer)<0   ********
   answer = (answer*f(prev) - prev*f(answer))/(f(prev) - f(answer));
   err = abs(f(answer));
end
end
```

(a) *[5 pts]* What method was Dana *trying* to instantiate here? secant

(b) *[5 pts]* What *kinds* of problems is this method designed to solve? root finding ones

(c) *[6 pts]* There's a bug in that code. Find the offending line, circle it, and tell us how you'd have to modify it in order to get the code to properly instantiate the method that Dana had in mind.

There were two issues; we accepted either one. First: the line with the *****s is unnecessary. There is no check of this form in the secant method. Second, `prev` never gets updated in the loop.

2. (a) *[10 pts]* Perform Gaussian elimination on this matrix. Show your work, and be neat enough that we can see what you're doing. If you start feeling the need for a calculator, please check your work; we chose the numbers to make the arithmetic easy.

$$\begin{bmatrix} -8 & 4 & -4 \\ -4 & 2 & -5 \\ 6 & -7 & 2 \end{bmatrix}$$

After G.E. in the first step:

$$\begin{bmatrix} -8 & 4 & -4 \\ 0 & 0 & -3 \\ 6 & -7 & 2 \end{bmatrix}$$

Second step & row swap:

$$\begin{bmatrix} -8 & 4 & -4 \\ 0 & -4 & -1 \\ 0 & 0 & -3 \end{bmatrix}$$

(b) *[4 pts]* Is pivoting necessary to that procedure, or could you get away without it in this case? Explain. It is absolutely necessary because of that zero pivot in the (2,2) position of the matrix after the first step.

3. *[6 pts]* What is the $PA = LU$ factorization of the matrix in the previous problem?

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} -8 & 4 & -4 \\ 0 & -4 & -1 \\ 0 & 0 & -3 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -3/4 & 1 & 0 \\ 1/2 & 0 & 1 \end{bmatrix}$$

4. Consider a matrix B where $cond(B) = 100000$.

   (a) *[3 pts]* What can you say about B^{-1}? It can be calculated, but its entries will be inaccurate.

   (b) *[3 pts]* Is the value of $cond(B)$ likely to be accurate—i.e., is $cond(B)$ exactly equal (or very close) to 100000? No. Large condition numbers are, by definition, inaccurate.

   (c) *[3 pts]* If $\vec{x}$ and $\vec{x'}$ are the solutions to $B\vec{x} = \vec{b}$ and $B\vec{x'} = \vec{b'}$, respectively, and $||\vec{b} - \vec{b'}||$ is very small, what can you say about $||\vec{x} - \vec{x'}||$? It may not be very small.

5. *[8 pts]* If A is the matrix in the problem 2 above and $\vec{b_1} = (-17, 4, 1)^T$, would you have to go through the whole Gaussian elimination process again to solve $A\vec{x_1} = \vec{b_1}$,

or could you do that more efficiently using other information that you've computed in previous pages of this exam? Explain and justify your answer.

Since you have P, L, and U from previous problems, you can just multiply $\vec{b}_1$ by P to get $\vec{b'}_1$ and then solve $LU\vec{x} = \vec{b'}_1$ in two stages: first solve $L(\vec{c}) = \vec{b'}_1$ to get $\vec{c}$, then solve $U\vec{x} = \vec{c}$ to get $\vec{x}$. Those two solves are $O(n^2)$, which is much cheaper than the $O(n^3)$ cost of going Gaussian elimination all over again on $A\vec{x} = \vec{b}_1$.

6. *[8 pts]* Consider a variation of the IEEE standard in which the exponent is represented by three bits and the fractional part of the mantissa is represented by three bits. Determine whether or not the following decimal (base 10) numbers can be represented in this scheme without error: 0.2, 1.625, 11.5, and 17. Explain why or why not. Assume that all other rules of the IEEE standard apply, including standard normalization (e.g., no subnormals) and reserved exponent values.

   This is the same setup as in PS1, where the exponent can range from -2 to 3. The smallest positive number that can be represented in this system is 1.000×2^{-2}, which is 0.25, so 0.2 is out. The largest positive number is $1.111_2 \times 2^3$, which is 15, so 17 is out. 1.625 can be represented as $1.101_2 \times 2^0$, so that one is OK. 11.5 is out because it's between two machine numbers (10 and 12).

7. *[7 pts]* Why is it important to consider the effects of floating-point arithmetic in numerical computing problems? Explain.

   Because of the errors introduced by the "snapping" of the real numbers to the grid of machine numbers, which are discretely spaced. These errors can bite you especially hard if you are working with numbers that are smaller than the spacing of that grid.

8. *[8 pts]* Use the fixed-point method to find a root of $f(x) = -2x^2 - x + 6$ starting from the guess of $x_0 = 1$.

   Most people (including me) tried $h(x) = -2x^2 + 6$ first, but that iteration sequence diverges: $x_0 = 1; x_1 = -2(1) + 6 = 4; x_2 = -2(16) + 6; \ldots$

   $g(x) = \sqrt{\frac{-x+6}{2}}$ works better: $x_0 = 1; x_1 = \sqrt{\frac{-1+6}{2}} = 1.414; x_2 = \sqrt{\frac{-1.414+6}{2}} = 1.514; x_3 = \sqrt{\frac{-1.514+6}{2}} = 1.497, \ldots$

   This is converging to the root at $x = 1.5$.

9. *[4 pts]* What is a root, anyway? It's a value x such that $f(x) = 0$.

10. *[5 pts]* What is *reasoning behind* the next-term rule? (Not "what is it," but "why does it work?")

    The next-term rule is based on the assumption that terms in series approximations get smaller and smaller. The error in a numerical method that uses a *truncation* of a series is equal to the terms that you truncated; since the first of those terms is the largest, that's the lion's share of the error. The next-term rule follows that reasoning.

11. *[5 pts]* Name a rootfinding method that always converges at the same rate, regardless of the function involved (as long as the guesses satisfy the requirements of that method). Bisection

12. *[4 pts]* In general, does Newton's method converge faster than the secant method? Circle one: yes/no.

    Does that hold *for all* situations (functions, guesses)? Circle one: yes/ no.

13. *[5 pts]* Give an example (words and/or pictures) of a function $f(x)$ on which Newton's method will converge, but only *slowly*. A function with a multiple root, such as $(x-1)^4 = 0$