Faculty of Arts & Sciences
Department of Computer Science
CMPS 251—Numerical Computing
Fall 2013–14
Saturday, Oct 12th, 2013

| | |
|---|---|
| **Family Name** | |
| **First Name** | |
| **Id. Num** | |
| **Major** | |

**QUIZ I** **Duration: 60 Minutes**

| Problem | Grade | Your Grade |
|---|---|---|
| 1 | 20 | |
| 2 | 25 | |
| 3 | 15 | |
| 4 | 25 | |
| 5 | 25 | |
| **Total** | **110** | |

# Instructions

Write your name, Id number, and your major in the space provided above.

**THE EXAM IS CLOSED BOOK, CLOSED NOTES, & "CLOSED NEIGHBOR"!!**

Your answers must be presented on the question sheet itself. Your handwriting should be readable so it can be graded. Use the provided space for your answers. If the space is not enough, then most probably you are writing more than what is expected. In case you need more space, put an "explicit pointer" to where your answer is written or continued (e.g. draw an arrow, or say back of page or both !!!…)

There are 5 problems altogether and 7 pages (besides this front page). You may use the back of the sheets for scratch work.

**DO NOT CUT OFF ANY PAGE !!**

## Problem 1 (20 Points)

For the following expressions, state the numerical difficulties that may occur, and rewrite the expression in a way that is more suitable for numerical computation:

a. (10) $(x + 1/x)^{1/2} - (x - 1/x)^{1/2}$ , when $x >> 1$ ($x$ very large)

**SOLUTION:**

The Problem here is loss of significance cause by subtracting small numbers, so we rationalize.

$$\frac{x + \frac{1}{x} - x + \frac{1}{x}}{\sqrt{x + \frac{1}{x}} + \sqrt{x - \frac{1}{x}}} = \frac{\frac{2}{x}}{\sqrt{x + \frac{1}{x}} + \sqrt{x - \frac{1}{x}}}$$

b. (10) $(a^{-2} + b^{-2})^{1/2}$ where $a \approx 0$ and $b \approx 1$.

**SOLUTION:**

The problem is inverting the square of $a$ which causes an overflow

$$\sqrt{\frac{1}{a^2} + \frac{1}{b^2}} = \sqrt{\frac{a^2 + b^2}{a^2 b^2}} = \frac{1}{|a||b|}\sqrt{a^2 + b^2}$$

## Problem 2 (20 Points) Gaussian Elimination

Consider the system of linear equations (2 by 2):

$$10^{-4} x_1 + x_2 = b_1$$
$$x_1 + x_2 = b_2$$

where $b_1 \neq 0$ and $b_2 \neq 0$. The solution of this system is

$x_1 = ( -b_1 + b_2 ) / ( 1 - 10^{-4})$ and $x_2 = ( b_1 - 10^{-4} b_2 ) / ( 1 - 10^{-4})$.

a. (10) Let $b_1 = 1$, and $b_2 = 2$. Solve this system using naïve Gaussian elimination with 3-digit (rounded) arithmetic, and compare with the exact solution $x_1 = 1.00010....$ and $x_2 = 0.999899...$ . Show all your computations.

**<u>SOLUTION:</u>**

$$10^{-4}x_1 + x_2 = 1$$
$$-9999x_2 = -9998$$
$$-10000x_2 = -10000$$
$$x_2 = 1, \; x_1 = 0$$

b. (10) Repeat the preceding part, but using Gaussian elimination with scaled partial pivoting. Show the resulting scale array ***s*** used to store the scales, and the index array ***l*** that stores the permutations. Show your computations at all the steps.

**<u>SOLUTION:</u>**

$$s = (1, 1) \quad l = (1, 2)$$

Exchange rows 1 & 2 $\quad l = (1, 2)$

$$x_1 + x_2 = 2$$
$$10^{-4}x_1 + x_2 = 1$$

$$x_1 + x_2 = 2$$
$$(1 - 10^{-4})x_2 = 1 - 2 * 10^{-4}$$

$$x_1 = 1, \quad x_2 = 1$$

## Problem 3 (15 Points) Bisection Method

Consider $f(x) = x^2 - 3$ on the interval [1,2]. Obviously, $f$ has a simple zero $r = 3^{1/2}$ in this interval.

a. (10) Starting with [a,b] being [1,2], (with $f(1)<0$ and $f(2)>0$), and using the Bisection method, how many iterations are needed to approximate $r$, the zero of $f$, to within *tol*, where *tol* is a preset tolerance.

**SOLUTION:**

$$f(1) = 1 - 3 = -2$$
$$f(2) = 4 - 3 = 1$$

Therefore we can proceed with the Bisection Method.

$$|r - x_n| \le \frac{1}{2^n}(b-a) = 2^{-(n+1)}$$

and $x_0 = 1.5$

Choose the smallest $n$ such that $2^{-(n+1)} \le tol$

$$2^{n+1} \ge \frac{1}{tol}$$

$$n + 1 \ge \log\left(\frac{1}{tol}\right)$$

$$n = \lceil -\log(tol) \rceil - 1$$

b. (5) In particular, how many iterations are needed to reach machine precision, on a machine that uses a mantissa with $k$ bits.) ? Justify your answer.

**SOLUTION:**

Here

$$tol = 2^{-k}$$
$$-\log(tol) = \log(2^{-k}) = -k$$
$$n = k - 1$$

## Problem 4 (25 Points) Newton's Method

As in the previous problem, consider $f(x) = x^2 - 3$ on the interval [1,2]. Obviously, $f$ has a simple zero $r = 3^{1/2}$ in this interval.
Now we want to use Newton's method (as below) to approximate the zero of $f$ above.

$$x_{n+1} = x_n - f(x_n) / f'(x_n), \quad n = 0, 1, 2, \ldots$$
$x_0$ is given

a. (4)What is the recurrence formula that defines the Newton's iterates for the $f$ given above?

**SOLUTION**

$$x_{n+1} = x_n - \frac{x_n^2 - 3}{2x_n}$$

b. (5) Let $e_n = r - x_n$ denote the error in $x_n$ as an approximation to $r$. Suppose you know that

$$e_{n+1} = -(f''(\xi_n) / (f'(x_n)) (e_n)^2 / 2 \qquad n = 0, 1, 2, \ldots$$

where $\xi_n$ is a point between $r$ and $x_n$, $r$ being the root. Show that if $x_n$ is in [1,2], then

(*) $\qquad |e_{n+1}| \leq M \; |e_n|^2$ , where $M = ½$

**SOLUTION:**

$$|e_{n+1}| = \frac{1}{2}\frac{|f''(\xi_n)|}{|f'(x_n)|}|e_n|^2 \leq \frac{1}{2}\frac{\max|f''(\xi_n)|}{\min|f'(x_n)|}|e_n|^2$$

$$\min|f'(x_n)| = \min_{1<x<2}|2x_n| = \frac{1}{2}\frac{2}{\min|f'(x_n)|}|e_n|^2 = 2$$

$$|e_{n+1}| \leq |e_n|^2$$

c. (4)Using the inequality (*) above, prove (quadratic) convergence.

**SOLUTION:**

$$\begin{aligned}
|e_n| &\leq \frac{1}{2}|e_{n-1}|^2 \leq \frac{1}{2}\left(\frac{1}{4}|e_{n-2}|^4\right) \\
&\leq \frac{1}{2}\cdot\frac{1}{2^2}\cdot\frac{1}{2^4}|e_{n-3}|^8 \\
&\leq \frac{1}{2}\cdot\frac{1}{2^2}\cdot\frac{1}{2^4}\cdots\frac{1}{2^{2^{n-1}}}|e_0|^{2^n} \\
&= \left(\frac{1}{2}\right)^{[1+2+4+8+\cdots+2^{n-1}]}|e_0|^{2^n}
\end{aligned}$$

$$|e_n| \leq \left(\frac{1}{2}\right)^{2^n-1}|e_0|^{2^n} = 2^{-2^n+1} = 2\,.\,2^{-2^n}$$

d. (5) Assuming that $x_0$ is chosen to be 1.5, how many iterations are needed to approximate the zero of $f$ to within *tol*, where *tol* is a preset tolerance.

**SOLUTION:**

$$x_0 = 1.5 \Rightarrow |e_0| \leq \frac{1}{2}$$

$$|e_n| \leq \left(\frac{1}{2}\right)^{2^n-1} \left(\frac{1}{2}\right)^{2^n} = \left(\frac{1}{2}\right)^{2^{n+1}} \leq \frac{tol}{2}$$

$$2^{-(2^{n+1})} \leq \frac{tol}{2} \Rightarrow -2^{n+1} \leq 1 - \ln(tol)$$

$$n = \lceil \ln\,(1 - \ln\,(tol)\rceil - 1$$

e. (3) In particular, how many iterations are needed to reach machine precision, on a machine that uses a mantissa with $k$ bits.) ? Justify your answer.

**SOLUTION:**

$$tol = 2^{-k}$$

$$n = \lceil \ln\,(1 + k)\rceil - 1$$

f. (4) Compare your result for Newton's method in (e) above with the result for the Bisection method in (b) of the previous problem.

**SOLUTION:**

Bisection $n = k - 1$

Newton $n = \lceil \ln\,(1 + k)\rceil - 1$ which is much faster!

## Problem 5 (25 Points) Special Matrices

Consider an $n \times n$ matrix **A** that has a diagonal ($d_i$), a subdiagonal ($a_i$) and two superdiagonals ($c_i$) ($e_i$), so the matrix looks like:

$$\begin{bmatrix} d_1 & c_1 & e_1 & & & \\ a_1 & d_2 & c_2 & e_2 & & \\ & a_2 & d_3 & c_3 & e_3 & \\ & & \cdots & \cdots & & e_{n-2} \\ & & & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & & a_{n-1} & d_n \end{bmatrix}$$

You are supposed to adapt Gaussian Elimination *without pivoting,* to matrices of this kind. In particular, give precise answers to the following with a brief explanation:

a. (2) How many non-zero entries are there in the matrix?

**SOLUTION:**

$$n + n - 1 + n - 1 + n - 2 = 4n - 4$$

b. (2) How many of these entries need to be eliminated? Which entries are these?

**SOLUTION:**

$n - 1$: The $a_i{}'s$

c. (4) If the elimination process is to be thought of as the LU factorization of the matrix A, what will be the structures of L and U? Be precise. Where are the "multipliers" stored?

**SOLUTION:**

L would be have 1 one the diagonals and the n-1 multipliers would be stored right underneath the diagonal

U would be a tri-diagonal matrix!

d. (3)Suggest a representation for the matrix **A**, that is suitable for storing this kind of matrices.

**SOLUTION:**

4 one dimentional arrays:

$$(d_i)_{n\times 1}$$
$$(a_i)_{n-1\times 1}$$
$$(c_i)_{n-1\times 1}$$
$$(de_i)_{n-2\times 1}$$

e. (5) Give pseudo-code for the elimination process, using your suggestion in (d).

**SOLUTION:**

```
for i=2 to n-1
          m = a(i) / d(i);    a(i-1) = m;
          d(i) = d(i) – m * c(i-1);
          c(i) = c(i) – m * e(i-1);
          b(i) = b(i) – m * b(i-1);
end

 m = a(n-1)/d(n-1);
 d(n) = d(n) – m*c(n-1);
 b(n) = b(n) – m*b(n-1);
```

f. (5) Give corresponding pseudo-code for the back-solving process.

**SOLUTION:**

```
x(n) = b(n)/d(n);
x(n-1) = (b(n-1) – c(n-1)*x)/d(n-1);
for i= n-2  down to 1;
          x(i)  = (b(i) – c(i) * x(i+1) –e(i)*x(i+2))/d(i);
end
```

g. (4) What is the long operations count of the processes of (e) and (f) in terms of *n* ? Be precise.

**SOLUTION:**

3(n-2) for (e)
1+2+3(n-2) for part (f)