CSC338 Midterm.

Question 2

Part I. (2 marks)

Let x be a real positive number not equal to 1, within the range of the representation of some floating point system. Recall $fl(x) = x(1+\epsilon)$ where : $\epsilon \leq \epsilon_{mach}$.

Estimate the absolute value of the absolute error when computing natural logarithm function $f(x) = \ln x$.

Please do typeset your solution in latex in the answer box. Your result must be supported with the correct mathematical arguments. You must show all your work and also typeset it in latex for full marks.

Solution.

$$\Delta f(x) = \ln x(1+\epsilon) - \ln x \text{ (1 mark)}$$
$$= \ln(1+\epsilon) \leq |\epsilon| \leq \epsilon_{mach} \text{ (1 mark)}$$

Part II. (3 marks).

Now consider the function: $g(x) = \ln|x|$ for nonzero values of x. Assuming there is a small error in the given value for x, write a formula that gives the conditioning number for the comuptation of $g(x)$. For what values of x this problem is not well-conditioned? Please typeset your solution using latex in the answer box.

Solution.

$$C_N(g) = \left|\frac{xg'(x)}{g(x)}\right| = \left|\frac{1}{\ln x}\right| \text{ (1 mark)}$$

The problem is not well conditioned for $C_N(g) > 1$, so:

$$-e < x < -\frac{1}{e} \text{ (1 mark)}$$

$$\frac{1}{e} < x < e \text{ (1 mark)}$$

Question 3

Part I. (4 marks)

It is a know fact that the determinant of a diagonal matrix:

$$\begin{vmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & d_n \end{vmatrix} = d_1 d_2 \dots d_n.$$

equals the product of its diagonal elements. Complete the code of the following Python function that computes the determinant of a diagonal matrix in single precision avoiding overflow and undeflow. For full marks, your function needs to minimize the relative error. You are not allowed to use any library function that computes the determinant directly.

Paste the completed code in the answer box.

```
def detf32(dg):
    """
    :param dg: 2-dimensional numpy.float32 diagonal matrix
    :return: determinant of dg
    >>> big = np.finfo('float32').max
    >>> small = np.float32(1/big)
    >>> A = np.diag([big, -big, small, small])
    >>> detf32(A) == np.float32(-1.0)
    True
    >>> A = np.diag([big, small, -big, -small])
    >>> detf32(A) == np.float32(1)
    True
    >>> A = np.diag([1, big, 2, small, 3])
    >>> math.isclose(detf32(A),6, rel_tol=0.0001)
    True
    """
    diag = np.diagonal(dg)
    signs = [-1 if d < 0 else 1 for d in diag]
    s = sum(sorted([np.log(abs(d)) for d in diag]))
    sign = reduce(lambda a, b: a*b, signs, 1)
    return sign*np.exp(s)
```

Part II. (4 marks)

Estimate the relative error of your implementation of detf32 function. Please do typeset your solution in latex in the answer box. For full marks, your solution must be well argued, show clearly why the relative error of your implementation is minimal, and also typeset it in latex.

Solution

To minimize relative error, suffices to minimize absolute error. Please note the absolute error of exponential function is $e^{x+\delta x} - e^x \approx \delta x$. Hence we need to minimize δs where s is the summation variable in the Python function. But δs is the error computing the summation of logarithms of diagonla elements, and we know how to minimize this - we need to sort the array of diagonal elemnts in increasing order hence $\delta s = (n-1)\epsilon_{mach}$ by lecture material.

Question 4

Consider the matrix

$$A = \begin{bmatrix} 2 & -9 & 0 & 4 \\ 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

with conditioning number 60. Let $b = \begin{bmatrix} 0 & .234 & 0 & 0 \end{bmatrix}^T$ where the entries valued zero are known precisely (without error) and the entry .234 contains a small error. Without solving the system $Ax = b$, give a precise value for the ratio of relative errors of x and b. Give full details how did you compute the ratio. Typeset your solution in latex in the answer area.

Solution.

Observing that

$$A = \begin{bmatrix} 2 & -9 & 0 & 4 \\ 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ a \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ a \\ 0 \\ 0 \end{bmatrix}$$

we conclude the required relative error ratio is precisely 1.
1 mark for saying ratio is 1, another mark for the explanation.

Question 5

Solve $A_1x_1 = \begin{bmatrix} 1 & 3 \\ 1 & 3.00001 \end{bmatrix} x_1 = \begin{bmatrix} 4 \\ 4.00001 \end{bmatrix}$ and $A_2x_2 = \begin{bmatrix} 1 & 3 \\ 1 & 2.99999 \end{bmatrix} x_2 = \begin{bmatrix} 4 \\ 4.00001 \end{bmatrix}$

Compute $\|A_1 - A_2\|_\infty$ and $\|x_1 - x_2\|_\infty$. Explain why a small perturbation in the matrix of the system induces a large perturbation in the solution. Write up your solution in latex using the answer area.

Solution.

It's not hard to see that $x_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}, x_2 = \begin{bmatrix} 7 & -1 \end{bmatrix}$. Also $\|A_1 - A_2\|_\infty = 0.00002, \|x_1 - x_2\|_\infty = 6$. (1 mark).

The explanation: Conditioning number for A_1 is $\|A_1\|_\infty \|A_1^{-1}\|_\infty \approx 2400000$(1 mark) so the probelm is ill conditioned (1 mark).

Question 6

Compute underflow, overflow, and ϵ_{mach} for the normalized floating point system $L(10, 3, -9, 9)$.

Solution
Applying textbook/lecture notes formulas: $\epsilon_{mach} = 10^{-2}$ by chopping, $\epsilon_{mach} = \frac{1}{2}10^{-2}$ by rounding (1 mark),
$UFL = 10^{-9}, OFL = 999 \times 10^{7}$ (1 mark each).

Question 7

Recall the following code from tutorials:

```
def backward_substitution(A, b):
    """Return a vector x with np.matmul(A, x) == b, where
        * A is an nxn numpy matrix that is upper-triangular and non-singular
        * b is an nx1 numpy vector
    """
    n = A.shape[0]
    x = np.zeros_like(b, dtype=np.float)
    for i in range(n-1, -1, -1):
        s = 0
        for j in range(n-1, i, -1):
            s += A[i,j] * x[j]
        x[i] = (b[i] - s) / A[i,i]
    return x

def eliminate(A, b, k):
    """Eliminate the k-th row of A, in the system np.matmul(A, x) == b,
    so that A[i, k] = 0 for i < k. The elimination is done in place."""
    n = A.shape[0]
    for i in range(k + 1, n):
        m = A[i, k] / A[k, k]
        for j in range(k, n):
            A[i, j] = A[i, j] - m * A[k, j]
        b[i] = b[i] - m * b[k]

def gauss_elimination(A, b):
    """Return a vector x with np.matmul(A, x) == b using
    the Gauss Elimination algorithm, without partial pivoting."""
    for k in range(A.shape[0] - 1):
        eliminate(A, b, k)
    x = backward_substitution(A, b)
    return x
```

Part I. (4 marks).

Assume we need to solve the system: $A^m x = b$. (m may be any positive natural constant.) Write an efficient Python function for that purpose. Your function may call any of the provided functions and nothing else. That means, you may not use other library functions to solve the problem. Here is the function signature:

```
def gauss_pow(A, b, m):
    """Return a vector x with np.matmul(A^m, x) == b"""
    # Your code here
```

Please paste your solution code in the solution area.
Solution

```
def gauss_pow(A, b, m):
    """Return a vector x with np.matmul(A^m, x) == b
    the Gauss Elimination algorithm, without partial pivoting."""
    x = gauss_elimination(A, b)
    while m > 1:
        b1 = np.array(x[:])
```

```
        x = backward_substitution(A, b1)
        m = m - 1
    return x
```

Part II. (2 marks).

Assume run-time of given functions is:

`backward_substitution`: bn^2, b a positive constant
`eliminate`: cn^2, c a positive constant
`gauss_elimination`: dn^3 a positive constant
Compute the runtime of your implementation of `gauss_pow`. You may assume one Python line is an elemntary instruction (for the purpose of line-counting). For full marks, show your work. Typeset your solution in latex in the answer area.

Solution
Using our implementation, we conclude the time complexity is $dn^3 + m(bn^2 + 3)$.
(1 mark for the cubic term 1 mark for the m and the square term).

Question 8

Does the matrix $\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ have a Cholesky decomposition? For full marks show all your steps in your answer (either way, show computationally why you decide to say "yes" or "no"). Latex your answer.

Solution Yes, because it is positive definite:

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= x^2 + y^2 + (x - y)^2 > 0$$

for all nonzero $\begin{bmatrix} x \\ y \end{bmatrix}$.