IIT Vadodara
WINTER 2019-20
MA202 Numerical Techniques
LAB#2 Numerical errors and System of linear equations[1]

1. **Condition number and singularity**

$\|A\|$ is the 2-norm of a matrix A. The MATLAB command norm(A,2) returns the 2-norm of A.The condition number of a matrix A,i.e., cond(A) = $\|A\|\|A\|^{-1}$with $\|A\|$ = largest eigenvalue of A^TA i.e., largest singular value of A. The MATLAB command cond(A) returns the 2-norm condition number (the ratio of the largest singular value of X to the smallest). Large condition numbers indicate a nearly singular matrix.

Q. 1: Write a MATLAB function docondition() to make use of matlab commands cond() and det() to compute the condition number and $det(A)det(A^{-1})$ where A is for example, Hilbert matrix defined by

$$A_{m\times n} = \{a_{mn}\} = \frac{1}{m+n-1}. \quad (1)$$

hilb(N) in MATLAB is the N-by-N matrix with elements 1/(i+j-1), which is a famous example of a badly conditioned matrix. Vary the dimension of A from N = 7 to 12 and see the degree of discrepancy between AA^{-1} and the corresponding identity matrix. (Note: The number RCOND following the warning message about near-singularity or ill-condition given by MATLAB is a reciprocal condition number, which can be computed by the rcond() command and is supposed to get close to 1/0 for a well-/badly conditioned matrix.)

2. **Numerical errors and analysis**

There are various numerical errors in the computing results made by digital systems, mostly coming from representing the numbers in finite bits, which is an intrinsic limitation of dig- ital world.

- Truncation Error: It is caused by adding up to a finite number of terms, while we should add infinitely many terms to get the exact answer in theory.
- Round-off Error: It is caused by representing/storing numeric data in finite bits.
- Overflow/Underflow: Caused by too large or too small numbers to be represented/stored properly in finite bits—more specifically, the numbers having absolute values larger/smaller than the maximum (fmax)/minimum(fmin) number that can be represented in MATLAB.
- Negligible Addition: It is caused by adding two numbers of magnitudes differing by over 52 bits.
- Loss of Significance: It is caused by a “bad subtraction,” which means a subtraction of a number from another one that is almost equal in value.
- Error Magnification: It is caused and magnified/propagated by multiplying/dividing a number containing a small error by a large/small number.

[1]submission deadline : 24th january 11 PM

- Errors depending on the numerical algorithms, step size, and so on.

Q. 2: consider the following two formulae:

$$f_1(x) = \sqrt{x}(\sqrt{x+1} - \sqrt{x} \tag{2}$$

$$f_2(x) = \frac{\sqrt{x}}{(\sqrt{x+1} + \sqrt{x}} \tag{3}$$

Write a MATLAB program to compute the values of the equation (4) and equation (5) at x = 1, 10, 100, 1000, 10000, 100000. Observe what happens as x increases and what kind of computing error has occured upon execution.

Q. 3: For 100 values of x over the interval $[10^{14}, 10^{16}]$, evaluate the following two expressions that are mathematically equivalent, plot them, and based on the graphs, tell which is better in terms of resisting the loss of significance.

$$\sqrt{2x^2+1} - 1 \tag{4}$$

$$\frac{2x^2}{\sqrt{2x^2+1}+1} \tag{5}$$

Q. 4: Write a matlab program to compute

$$y^n/e^{nx}, \tag{6}$$

with x = 36, y = 1e16, and for n = [-20 -19 19 20] with $x \succ 1$and $y \succ 1$, and also

$$(y/ex)^n. \tag{7}$$

Which is the right one to avoid overflow/underflow?

Q. 5: For $x = 9.8^{201}$ and $y = 10.2^{199}$, evaluate the following two expressions that are mathematically equivalent and tell which is better in terms of the power of resisting the overflow.

$$z = \sqrt{x^2+y^2} \tag{8}$$

$$z = \sqrt{(\frac{x}{y})^2 + 1} \tag{9}$$

Q. 6: Also for $x = 9.8^{-201}$ and $y = 10.2^{-199}$, evaluate the equation (8) and equation(9) and tell which is better in terms of the power of resisting the underflow.

3. **Systems of linear equations** There are several numerical schemes for solving a system

of equations $A_{m\times n}x_{n\times 1} = b_{m\times 1}$
Consider the three cases:
(i) $(m = n)$, i.e., the case where the number of equations (m) and the number of unknowns (n) are equal so that the coefficient matrix $A_{M\times N}$ is square.
(ii) $(m < n)$, i.e.,the case where the number of equations is smaller than the number of unknowns so that we might have to find the minimum-norm solution among the numerous solutions.
(iii) $(m > n)$, i.e.,the case where the number of equations is greater than the number of unknowns so that there might exist no exact solution and we must find a solution based on global error minimization, like the "least-squares error (LSE) solution."

Q. 7: Write a MATLAB routine lineq() to solve a given set of equations, covering all of the three cases

Q. 8: Solve the linear equations using lineq() function created in Q.7 with given A and b
(a) A = [1 2;3 4]; b = [-1;-1]
(b) A = [1 2;2 4]; b = [-1;-1]
(c)[1 2]; b = 3
(d) A = [1; 2]; b = [2.1; 3.9]