Name: Rajat Jaiswal **Entry Number:** 2017CS50415

1. Let $f : X \to Y$ and $g : Y \to Z$ be continuous functions between arbitrary vector spaces, and let $h = g \circ f$. Suppose $y = f(x)$ and $z = g(y) = h(x)$.

   (a) Find the tightest possible upper bound on $\kappa_h(x)$ in terms of $\kappa_f(x)$ and $\kappa_g(y)$, where κ_f , κ_g, κ_h are the condition numbers of f , g, h respectively.

Solution: $\kappa_f(x) = \dfrac{\|x\| \cdot \|J_f(x)\|}{\|f(x)\|}$, and $\kappa_g(y) = \dfrac{\|y\| \cdot \|J_g(y)\|}{\|g(y)\|}$. Similarly, $\kappa_h(x) = \dfrac{\|x\| \cdot \|J_h(x)\|}{\|h(x)\|}$. Given, $y = f(x)$. Multiply the numerator and denominator in the RHS of $\kappa_h(x)$ by y and $f(x)$ respectively. Also given, $g(y) = h(x)$, this becomes,

$$\kappa_h(x) = \frac{\|x\| \cdot \|y\| \cdot \|J_h(x)\|}{\|f(x)\| \cdot \|g(y)\|}$$

Let A_i be the $i-th$ row of matrix A and A^j be the $j-th$ column. Let A_{ij} be the $(i,j)th$ entry in the matrix. Using the chain rule of derivatives we know that $J_h(x)_{ij} = J_g(f(x))_i \cdot J_f(x)^j$. Hence, $J_h(x) = J_g(f(x)) \cdot J_h(x) = J_g(y) \cdot J_f(x)$.

Using the inequality property of norms, $\|AB\| \leq \|A\| \cdot \|B\|$, we know,

$$\|J_h(x)\| = \|J_g(y) \cdot J_f(x)\| \leq \|J_g(y)\| \cdot \|J_f(x)\|.$$

This gives us the relation,

$$\kappa_h(x) = \frac{\|x\| \cdot \|y\| \cdot \|J_h(x)\|}{\|f(x)\| \cdot \|g(y)\|} \leq \frac{\|y\| \cdot \|J_g(y)\|}{\|g(y)\|} \cdot \frac{\|x\| \cdot \|J_f(x)\|}{\|f(x)\|} = \kappa_g(y) \cdot \kappa_f(x).$$

This can also be shown using the supremum definition of condition numbers whereby,
$\kappa_f(x) = \dfrac{\|x\|}{\|f(x)\|} \sup_{\delta x} \dfrac{\|\delta f(x)\|}{\|\delta x\|}$, $\kappa_g(y) = \dfrac{\|y\|}{\|g(y)\|} \sup_{\delta y} \dfrac{\|\delta g(y)\|}{\|\delta x\|}$, and $\kappa_h(x) = \dfrac{\|x\|}{\|h(x)\|} \sup_{\delta x} \dfrac{\|\delta h(x)\|}{\|\delta x\|}$.

$$\kappa_h(x) = \frac{\|x\|}{\|h(x)\|} \sup_{\delta x} \frac{\|\delta h(x)\|}{\|\delta x\|} = \frac{\|x\|}{\|f(x)\|} \cdot \frac{\|y\|}{\|g(y)\|} \cdot \sup_{\delta x} \frac{\|\delta g(y)\| \cdot \|\delta f(x)\|}{\|\delta x\| \cdot \|\delta y\|}, \because y = f(x) \ \& \ h(x) = g(y)$$

$$\implies \kappa_h(x) \leq \frac{\|x\|}{\|f(x)\|} \cdot \frac{\|y\|}{\|g(y)\|} \cdot \sup_{\delta x} \left(\frac{\|\delta f(x)\|}{\|\delta x\|} \cdot \sup_{\delta y} \left(\frac{\|\delta g(y)\|}{\|\delta y\|} \right) \right)$$

$$\implies \kappa_h(x) \leq \left(\frac{\|y\|}{\|g(y)\|} \cdot \sup_{\delta y} \frac{\|\delta g(y)\|}{\|\delta y\|} \right) \cdot \left(\frac{\|x\|}{\|f(x)\|} \cdot \sup_{\delta x} \frac{\|\delta f(x)\|}{\|\delta x\|} \right)$$

$$\implies \kappa_h(x) \leq \kappa_f(x) \cdot \kappa_g(y)$$

   (b) The actual condition number can be far smaller than the bound derived above. Give a concrete example where $\kappa_f(x)$, $\kappa_g(y) \geq 10$ but $\kappa_h(x) \leq 1$.

Solution: Take $f(x) = \begin{bmatrix} 10 \cdot x_1 - 10 \cdot x_2 \\ 1 \end{bmatrix}$ for the vector $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{C}^2$. Take $g(y) = \begin{bmatrix} y_1^{10} \\ y_2^{10} \end{bmatrix}$ for $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \in \mathbb{C}^2$. Let us use $\|.\|_\infty$ norms. Take $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \implies f(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. $\|x\|_\infty = max(|x_1|, |x_2|) = 1$.
$J_f(x) = \begin{bmatrix} 10 & -10 \\ 0 & 0 \end{bmatrix} \implies \|J_f(x)\|_\infty = 20$. $\kappa_f(x) = \dfrac{\|x\|_\infty \cdot \|J_f(x)\|_\infty}{\|f(x)\|_\infty} = \dfrac{1 \cdot 20}{1} = 20 \geq 10$.
Continued on next page.

For $y = \begin{bmatrix}0\\1\end{bmatrix} \implies g(y) = \begin{bmatrix}0\\1\end{bmatrix}$. $\|y\|_\infty = max(|y_1|, |y_2|) = 1$. $J_g(y) = \begin{bmatrix}10 \cdot y_1^9 & 0\\ 0 & 10 \cdot y_2^9\end{bmatrix} = \begin{bmatrix}0 & 0\\0 & 10\end{bmatrix}$.
$\kappa_g(y) = \dfrac{\|y\|_\infty \cdot \|J_g(y)\|_\infty}{\|g(y)\|_\infty} = \dfrac{1 \cdot 10}{1} = 10 \geq 10.$
$h(x) = g(f(x)) = \begin{bmatrix}(10 \cdot (x_1 - x_2))^{10}\\ 1\end{bmatrix}$. $J_h(x) = \begin{bmatrix}100 \cdot (10 \cdot (x_1 - x_2))^9 & -100 \cdot (10 \cdot (x_1 - x_2))^9\\ 0 & 0\end{bmatrix}$.
$\implies J_h(x) = \begin{bmatrix}0 & 0\\0 & 0\end{bmatrix} \because x_1 = x_2. h(x) = \begin{bmatrix}0\\1\end{bmatrix}$. $\kappa_h(x) = \dfrac{\|x\|_\infty \cdot \|J_h(x)\|_\infty}{\|h(x)\|_\infty} = \dfrac{1 \cdot 0}{1} = 0 \leq 1.$

2. Suppose you want to compute the function $f(x) = x(1 - x)$ for some number $x \in \mathbb{R}$, so you proceed by rounding to a floating-point number $\hat{x} = fl(x) \in \mathbb{F}$ and evaluating $\hat{x} \otimes (1 \ominus \hat{x})$.
   (a) Derive an upper bound on the forward error of the result.

Solution: Let ϵ_m be machine epsilon. Given, $f(x) = x \cdot (1 - x)$ and $\hat{f}(\hat{x}) = \hat{x} \otimes (1 \ominus \hat{x})$.

$$\hat{x} = fl(x) = x(1 + \epsilon_1); \epsilon_1 \leq \epsilon_m$$
$$(1 \ominus \hat{x}) = (1 - \hat{x})(1 + \epsilon_2); \epsilon_2 \leq \epsilon_m.$$
$$\hat{y} = \hat{x} \otimes (1 \ominus \hat{x}) = \hat{x} \cdot (1 \ominus \hat{x})(1 + \epsilon_3); \epsilon_3 \leq \epsilon_m.$$
$$\implies \hat{y} = \hat{x} \cdot (1 - \hat{x})(1 + \epsilon_2)(1 + \epsilon_3)$$
$$\implies \hat{y} = x \cdot (1 - x(1 + \epsilon_1))(1 + \epsilon_1)(1 + \epsilon_2)(1 + \epsilon_3)$$

Let $\alpha = (1 + \epsilon_1), \beta = (1 + \epsilon_1)(1 + \epsilon_3)(1 + \epsilon_2)$. Given, $y = x(1 - x)$. Ignoring all the terms of $O(g(x)\epsilon_m^2)$, where $g(x)$ is any polynomial in x.

$$\text{Abs. Fwd. Error} = |\hat{y} - y| = |x \cdot ((1 - x \cdot \alpha)\beta - (1 - x))|$$
$$\beta = 1 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_1 \cdot \epsilon_2 + \epsilon_2 \cdot \epsilon_3 + \epsilon_1 \cdot \epsilon_3 + \epsilon_1 \cdot \epsilon_2 \cdot \epsilon_3$$
$$(1 - x \cdot \alpha) = 1 - x - x \cdot \epsilon_1$$
$$(1 - x \cdot \alpha)\beta \approx 1 - x - x \cdot \epsilon_1 + \epsilon_1 - x \cdot \epsilon_1 + \epsilon_2 - x \cdot \epsilon_2 + \epsilon_3 - x \cdot \epsilon_3$$
$$\implies |\hat{y} - y| \approx |x^2(2 \cdot \epsilon_1 + \epsilon_2 + \epsilon_3) - x(\epsilon_1 + \epsilon_2 + \epsilon_3)| \leq |4 \cdot x^2\epsilon_m| + |3x \cdot \epsilon_m| = O(x^2\epsilon_m) + O(x\epsilon_m)$$
$$\implies |\hat{y} - y| = O(x^2\epsilon_m).$$

   (b) For what value(s) of x does the result fail to be accurate? For what value(s) does it fail to be backward stable?

Solution: For accuracy, relative forward error $= O(\epsilon_m)$.
Relative forward error $= \left|\dfrac{4x\epsilon_m}{(1 - x)}\right| + \left|\dfrac{3\epsilon_m}{(1 - x)}\right|$. The relative forward error becomes very high in the vicinity of $x = 1$. Hence, the algorithm fails to be accurate in the vicinity of $x = 1$. When x becomes very high then the relative error comes out to be $O(\epsilon_m)$.

For backward stability, find $\tilde{x}$ such that, $f(\tilde{x}) = \tilde{x}(1 - \tilde{x}) = \hat{y} \approx x(1 - x - 2x \cdot \epsilon_1 + \epsilon_1 + \epsilon_2 - x \cdot \epsilon_2 + \epsilon_3 - x \cdot \epsilon_3)$.

$$\implies \tilde{x} \approx \tilde{x}^2 + x - x^2(1 + 2\epsilon_1 + \epsilon_2 + \epsilon_3) + x(\epsilon_1 + \epsilon_2 + \epsilon_3)$$
$$\implies (\tilde{x} - x)(1 - (\tilde{x} + x)) \approx -x^2(2\epsilon_1 + \epsilon_2 + \epsilon_3) + x(\epsilon_1 + \epsilon_2 + \epsilon_3)$$
$$\implies \left|\frac{\tilde{x} - x}{x}\right| \approx \left|\frac{(\epsilon_1 + \epsilon_2 + \epsilon_3) - x(2\epsilon_1 + \epsilon_2 + \epsilon_3)}{(1 - (\tilde{x} + x))}\right|$$

The algorithm is not backward stable whenever $x + \tilde{x} = 1$. This is expected behaviour since $f(x) = f(1 - x)$. But as per the definition of backward stability, if for each x we can find a $\tilde{x}$ in the close vicinity of x, such that $f(\tilde{x}) = \hat{y}$, the algorithm is backward stable. *Continued on next page.*

Here also we can find two such $\tilde{x}$, and one of them is close to x, hence it is backward stable. The algorithm is giving an exact answer to a nearby question i.e. $\tilde{x}$. For $x = 0.5$, a $\tilde{x}$ in the vicinity of 0.5 gives $x + \tilde{x} = 1$, satisfying the above said condition but here also we can say that the algorithm is backward stable stable even though RHS is very large. This is because $\tilde{x}$ is still in the vicinity of x and that is the main concern of backward stability. Hence the algorithm is backward stable for all values.

3. Consider the equation $\mathbf{UX} + \mathbf{XL} = \mathbf{Y}$, where $\mathbf{U}, \mathbf{L}, \mathbf{Y} \in \mathbb{C}^{m \times m}$ are known matrices, $\mathbf{U}$ is upper triangular, and $\mathbf{L}$ is lower triangular. Give an algorithm to find the unknown matrix $\mathbf{X} \in \mathbb{C}^{m \times m}$ in $O(m^3)$ time using backsubstitution.

Solution: The main idea is to evaluate the m^{th} column of $\mathbf{X}$ first, then $m-1^{th}$ column and so on.

The m^{th} column of $\mathbf{UX} + \mathbf{XL}$ is $\begin{bmatrix} \sum_{j=i}^{m}[U_{1j}X_{jm}] + X_{1m}L_{mm} \\ \sum_{j=2}^{m}[U_{2j}X_{jm}] + X_{2m}L_{mm} \\ \cdots \\ \sum_{j=m}^{m}[U_{mj}X_{jm}] + X_{mm}L_{mm} \end{bmatrix} = \begin{bmatrix} Y_{1m} \\ Y_{2m} \\ \cdots \\ Y_{mm} \end{bmatrix}$.

To solve the m^{th} column use backsubstitution, we can get $X_{mm} = \dfrac{Y_{mm}}{U_{mm} + L_{mm}}$, and use X_{mm} to solve $X_{(m-1)m}$ and so on til X_{1m}. As we know, solving one particular column will take $O(m^2)$ time. And therefore, solving for all columns will take $O(m \cdot m^2) = O(m^3)$ time.

Let `backsubstitution(T, y)` be a subroutine that solves $Tx = y$, where $T \in \mathbb{C}^{m \times m}$ is an upper triangular matrix, $y \in \mathbb{C}^{m \times 1}$ and return $x \in \mathbb{C}^{m \times 1}$. This subroutine is assumed to be given and in any case it can be computed as given here. The invariant in the algorithm below is that when solving for i^{th} column of $\mathbf{X}$, the values of $(i+1)^{th}$ column to m^{th} column of $\mathbf{X}$ is known.

Algorithm 1 Solving for $\mathbf{UX} + \mathbf{XL} = \mathbf{Y}$

```
1:  function SolveForX(U, L, Y)
2:      m ← U.size[0]
3:      Declare 2D empty array X of size (m,m)
4:
5:      for i from m to 1 do                                  ▷ Column number
6:          y ← Y[:, i]
7:          T ← U.copy()
8:          for k from 1 to m do                              ▷ For each row k
9:              T_kk ← T_kk + L_ii                            ▷ Diagonal element L_ii contributes to X_ki
10:             Sum ← 0                                       ▷ Storing previous computed values of column i+1 to m
11:             for j from i+1 to m do
12:                 Sum+ = X_kj L_ji
13:             end for
14:             y[k] ← y[k] − Sum
15:         end for
16:         X[:, i] = backsubstitution(T, y)
17:     end for
18:     return X
19: end function
```

- **Running time analysis:** For loop on Line 5 runs m time. The for-loop from line 8-15 takes $O(m^2)$ time in each iteration of i. The subroutine `backsubstitution(T, y)` as given in the question statement takes $O(m^2)$ time. Hence the overall time taken is $O(m^3)$ time.

4. When does a rectangular matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ have the property that $\|\mathbf{Ax}\|_2 = \|\mathbf{x}\|_2$ for all $\mathbf{x} \in \mathbb{C}^n$? Give necessary and sufficient conditions if it is possible for each of the two cases, $m > n$ and $m < n$.

Solution: Given, $\|x\|_2 = \sqrt{x^*x}$. $\|Ax\|_2 = \sqrt{(Ax)^*(Ax)} = \sqrt{x^*A^*Ax}$.

$$\|x\|_2 = \|Ax\|_2 \implies \sqrt{x^*x} = \sqrt{x^*A^*Ax} \implies x^*x = x^*A^*Ax.$$

Since $x \in \mathbb{C}^n$ and this is true for all x. Hence, this gives us that $A^*A = \mathbf{I}$. This means that A is a unitary matrix with orthonormal columns.

- **Case 1:** $m \geq n$ **-** n orthonormal vectors of length m each, where $m \geq n$ can exist. This is a standard result and the proof can also be found here(5.2 pg-91). Hence the necessary and sufficient condition for this case is A is a unitary matrix.
- **Case 2:** $m < n$ **-** n vectors of length m each, are linearly independent, where $m < n$. This cannot be true, these vectors span a vector space of dimension m and $\because m < n$, all the n vectors cannot be linearly independent and $n - m$ vectors will have to be a linear combination of other m vectors. Hence, it is not possible in this case.

5. Say I am interested in matrices of the form $\mathbf{M}(\lambda) = \mathbf{I} + \lambda\mathbf{A}$ for $\lambda \in \mathbb{C}$ and $\mathbf{A} \in \mathbb{C}^{m\times m}$. I know the value of $\|\mathbf{A}\|$ in some arbitrary induced norm. Find a positive real number δ such that $\mathbf{M}(\lambda)$ is nonsingular whenever $|\lambda| < \delta$.

Solution: Given that $\mathbf{M}(\lambda) - \lambda\mathbf{A} = \mathbf{I}$. Multiply both sides by $x \in \mathbb{C}^{m\times 1}$. We get.

$$\begin{aligned} & x = \mathbf{M}(\lambda)x - \lambda\mathbf{A}x && \because \mathbf{I}x = x \\ \implies & \|x\| = \|\mathbf{M}(\lambda)x - \lambda \cdot \mathbf{A}x\| \leq \|\mathbf{M}(\lambda)x\| + |\lambda| \cdot \|\mathbf{A}x\| && \text{(Cauchy-Schwarz inequality)} \\ \implies & \|x\| \leq \|\mathbf{M}(\lambda)x\| + |\lambda| \cdot \|\mathbf{A}\| \cdot \|x\| && \text{(Submultiplicativity of induced norm)} \\ & \implies \|x\|(1 - |\lambda| \cdot \|\mathbf{A}\|) \leq \|\mathbf{M}(\lambda)x\| \end{aligned}$$

Given that whenever $|\lambda| < \delta$, $\mathbf{M}(\lambda)$ is non-singular. The null-space of non-singular or invertible matrix contains only the $\vec{0}$. And hence for all $x \neq \vec{0}$, $\mathbf{M}(\lambda)x$ is non-zero and hence the norm of $\|\mathbf{M}(\lambda)x\| > 0$ and also $\|x\| > 0$. So for $|\lambda| < \dfrac{1}{\|\mathbf{A}\|}$, $\|\mathbf{M}(\lambda)\| > 0$ whenever $x \neq \vec{0}$. Therefore, $\delta = \dfrac{1}{\|\mathbf{A}\|}$.

6. Consider the sequence defined by the difference equation $x_{k+1} = 2.25x_k - 0.5x_{k-1}$, with starting values $x_0 = 1/3$ and $x_1 = 1/12$.

   (a) Write a function *computeSequence*(n) which computes the first n terms of this sequence, and returns them in a list $[x_0, x_1, ..., x_{n-1}]$.

Solution: Submitted `q6_2017CS50415.py` file.

   (b) Make a semi log plot of the computed terms as a function of k, up to $k = 80$. The exact solution to the equation is $x_k = 4^{-k}/3$; show this on the same plot.

Solution:

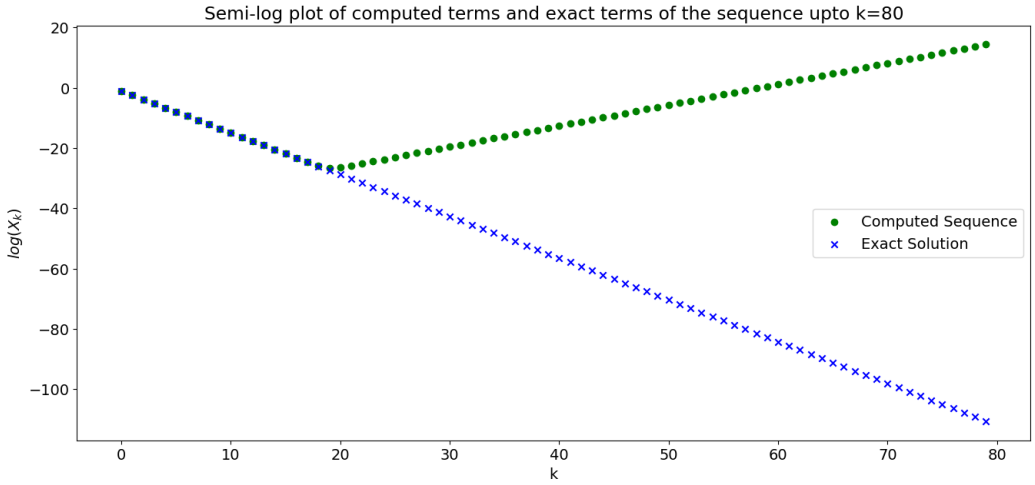

Figure 1: Semi log plot of computed terms and exact solution of the sequence up to $k = 80$.

   (c) The exact solution decreases monotonically as k increases. Does your computed solution have the same behaviour? Can you explain your results?

Solution: No, the computed solution do not have the same behaviour. A machine has a finite precision and hence it truncates floating point numbers. This error keeps on accumulating every time a calculation is performed and the value is stored. New values are being calculated using these truncated values and hence the error keeps on propagating and new error accumulates and after a point they start to magnify and the computed sequence diverges from the exact solution. These errors accumulate in the form that the constant of ϵ_m becomes very large. Since the exact solution is monotonically decreasing there comes a point when the exact value and the accumulated constant times ϵ_m becomes almost equal and the error term changes the actual value significantly such that the relative error is no longer in the bounds of machine epsilon. In the exact solution this truncation is happening only once when the value is being stored, and hence it is controlled and is within the bounds machine epsilon. Once they diverge the consecutive values also start to increase monotonically from that point because the difference equation is in that way. The coefficient of x_k is greater than the coefficient of x_{k-1} which is being subtracted and even if x_{k-1} becomes equal to x_k the overall value of x_{k+1} will be $\geq 1.75 \cdot x_k$(for $k > k'$) beyond that point(k') and hence it starts to increase monotonically as k increases.