coursework2

Problem3

In lecture, we examined the stability of multiplying vectors with vectors and triangular matrices. We now consider the problem of computing the general matrix product $Ax = b$, where $A \in \mathbb{C}^{n\times n}$ and $x, b \in \mathbb{C}^n$. We use the following simple algorithm to compute b for given A, x

Algorithm 3 Matrix-Vector Multiplication

$b = \mathbf{0}$
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $b_i = b_i + A_{ij}x_j$
  end for
end for

Perform a backward stability analysis of this algorithm. You may assume the given data are floating-point numbers (errors are only introduce by arithmetic operations)

we assume the floating point precise of the machine that this algorithm will run on is $\epsilon_{\text{machine}}$, and the floating point operations to be $\oplus, \otimes$ for add ans multiply, and fl() for convert the numerical values to the real value used in the program

according to the question

- input data can be seen as x
- output is Ax

in order to check an algorithm is backwards stable, we need to check

- $\tilde{f}(x) = f(\tilde{x})$
- $\frac{||\tilde{x}-x||}{||x||} = \mathcal{O}(\epsilon_{\text{machine}})$

for clarity

we first analyse the inner cycle, or line 3 to 5,

then the inner loop will iteratively increment b_i by $A_{ij}x_j$

therefore, the inner loop does $b_i = b_i \oplus (A_{ij} \otimes x_j)$ iteratively

Since

$$\begin{aligned} fl(A_{ij}) \otimes fl(x_j) &= A_{ij} \otimes x_j \\ &= (1+\delta_1)A_{ij}x_j \end{aligned}$$

where, as the question states, we assume everything is floating point, or fl is the identity function. δ_1 is the error caused during multiplying floating point numbers and $|\delta| \leq \epsilon_{\text{machine}}$

therefore we can show that, the inner cycle will turn b_i to

$b_i + \sum_{j=1}^{n}(1+\delta_1)A_{ij}x_j$

combined with the outer loop, which does the same operation for each i, independently

the program turns b (initially 0), to $(1+\delta_1)Ax$

therefore $\tilde{f}(x) = (1+\delta_1)Ax$

to satisfy $f(\tilde{x})$ which $f(x) = Ax$, we let $\tilde{x} = (1+\delta_1)x$

then

$$\begin{aligned} \frac{||\tilde{x} - x||}{||x||} &= \frac{||(1+\delta_1)x - x||}{||x||} \\ &= \frac{||(\delta_1)x||}{||x||} \\ &= \mathcal{O}(\epsilon_{\text{machine}}) \end{aligned}$$

therefore, we can safely conclude that this algorithm is backward stable

Problem 4

We defined the growth factor for Gaussian elimination of a matrix $A \in \mathcal{L}^{m\times m}$ to be

$$\rho = \frac{\max_{i,j}|u_{i,j}|}{\max_{i,j}|a_{i,j}|}$$

where $A = LU$, $a_{i,j}$ is the element $A[i,j]$ and $u_{i,j}$ is the element $U[i,j]$. Show that with partial pivoting this growth factor is bounded by $\rho \leq 2^{m-1}$

when we swap rows with partial pivoting

consider the following example, we switch the rows of

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \to \begin{bmatrix} c & d \\ a & b \end{bmatrix}$$

then in gaussian eliminate, we compute a multiplier of $\frac{a}{c}$ and turn the matrix into

$$\begin{bmatrix} c & d \\ 0 & b - \frac{a}{c}d \end{bmatrix}$$

by subtracting $\frac{a}{c}$ times the first row from the second row

or, if we use the syntax we used for ρ

$u_1 = a_1 - \frac{a}{c}a_2$

since we only swap when c > a, this guarantees that the scaling factor $|\frac{a}{c}| \leq 1$ (for negatives, it is similar same)

and again we generalise this process, let m be the scaling factor with $|m| \leq 1$

then $u_i = a_i - m \bullet a_{pivot}$

notice that since $|m| \leq 1$, the maximum possible growth, or $\max \rho_{\text{one step}} = \max \frac{\max_{i,j} |u_{i,j}|}{\max_{i,j} |a_{i,j}|} = 2$

this happens when $m = -1$

so at every step we can at most double the ρ, since A is $m * m$ matrix, we at most do $m - 1$ operations

optimally, all the steps double rho, and ρ can be seen as having a initial value of 1

therefore, $\rho \leq 2^{m-1}$

or ρ is bounded by 2^{m-1}