Ryan Shìjié Dù
September 29th, 2023

Worksheet 3-1

1 Complexity of computing matrix condition numbers

(1.a) Recall the formulas for the 1- and ∞-norm. Assuming that taking the absolute value and determining the maximum does not contribute to the overall computational cost, calculate how many *flops* (floating point operations) are needed to calculate $\|A\|_1$ and $\|A\|_\infty$ for $A \in \mathbb{R}^{n\times n}$? By what factor will the calculation time increase when you double the matrix size?

(1.b) Now implement a simple code that calculates $\|A\|_1$ and $\|A\|_\infty$ for a matrix of any size $n \geq 1$. Try to do this without using loops[1]! Using system sizes of $n_1 = 100$, $n_{k+1} = 2n_k, k = 1, \ldots, 7$, determine how long your code takes to calculate $\|A\|_1$ and $\|A\|_\infty$ for a matrix $A \in \mathbb{R}^{n_i \times n_i}$ with random entries and report the results. Can you confirm the estimate from (1.a)?

(1.c) Python has the built-in functions `np.linalg.norm` to calculate matrix norms. Calculate for the system sizes in (1.b) $\|A\|_1$ and $\|A\|_\infty$ using both your implementation and the built-in `norm` function, determine for each n_i how long each code takes and plot the results in one graph. On average, by what factor is MATLAB or Python's implementation faster than yours?

2 Condition numbers based on different norms

(2.a) Let $A \in \mathbb{R}^{n\times n}$ be defined by

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Calculate $\kappa_1(A)$ and $\kappa_\infty(A)$. We see that a matrix can be well or ill-conditioned depending on the choice of norms.

(2.b) Indeed, we solve $A\boldsymbol{x} = \boldsymbol{b}$ and $A(\boldsymbol{x} + \Delta\boldsymbol{x}) = (\boldsymbol{b} + \Delta\boldsymbol{b})$ where

$$\boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{and} \quad \Delta\boldsymbol{b} = \begin{bmatrix} \epsilon \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Check that we have for both norms:

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \kappa(A)\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

[1] The commands needed in Python `np.abs` and `np.sum`. Most commands can not only applied to numbers, but also to vectors, where they apply to each component; this is typically much faster.

3 Conditional number for the Hilbert matrix

The Hilbert matrix $H \in \mathbb{R}^{n \times n}$ is a matrix with entries

$$h_{ij} = \frac{1}{i+j-1}.$$

(3.a) Using MATLAB or Python, compute the 2-norm-based condition numbers for $n = 3, 5, 10, 20, 25$.

(3.b) Let's consider a relative right hand side perturbation $\delta \boldsymbol{b}$ of a linear system with $\|\delta \boldsymbol{b}\|_2 / \|\boldsymbol{b}\|_2 \approx 10^{-15}$. Write down the corresponding bounds $\|\delta \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2$ from the theory we discussed in class.

(3.c) Now, let's compute the actual error. Use the right-hand side vector with entries $b_i = \sum_{j=1}^{n} (j/(i+j-1))$ chosen such that the solution vector has entries $x_i = i$. Now, Compute the numerical solutions[2] $\boldsymbol{x}$, then re-compute $\boldsymbol{b} = H\boldsymbol{x}$ and compare the relative right-hand side error and the relative error in the solutions. How much are these better than the estimates you got from the condition number?

[2] Note that all these computations contain tiny errors due to the final precision of computer computations.