

THE HILBERT MATRIX

In this homework assignment we will have some fun with the Hilbert matrix. The Hilbert matrix is interesting because it is an extremely ill-conditioned matrix. This makes the numerical calculation of its inverse very troublesome. At the same time an exact theoretical equation for the inverse of the Hilbert matrix can be derived!

The Hilbert matrix is defined as the matrix $H_n = [H_{\alpha\beta}] \in \mathbb{R}^{n \times n}$ such that its elements satisfy

$$H_{\alpha\beta} = \frac{1}{\alpha + \beta - 1}, \quad \forall \alpha, \beta \in [n]. \quad (1)$$

The inverse of the Hilbert matrix $H_n^{-1} = [G_{\alpha\beta}]$ is given by

$$G_{\alpha\beta} = (-1)^{\alpha+\beta} (\alpha + \beta - 1) \binom{n + \alpha - 1}{n - \beta} \binom{n + \beta - 1}{n - \alpha} \binom{\alpha + \beta - 2}{\alpha - 1}^2, \quad \forall \alpha, \beta \in [n]. \quad (2)$$

Here, we define

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (3)$$

as the well-known coefficients from the binomial theorem. These can be calculated recursively via the ‘‘Pascal triangle’’ method as follows:

$$\binom{0}{k} = 1 \quad (4)$$

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad (5)$$

To complete this assignment I recommend that you use either Matlab or Octave or C combined with the GSL library. Turn in your discussion paper, results and source code.

- (1) First, write a subroutine that generates the Hilbert matrix.
- (2) Second, write a subroutine that generates the inverse Hilbert matrix. It will be necessary to have another subroutine that can precalculate the binomial theorem coefficients needed for building inverse Hilbert matrices up to order n . These can then be passed on to the routine that will calculate the inverse Hilbert matrix. This way one can save a lot of calculation time. A question that you must discuss in your assignment is: how many binomial theorem coefficients are needed to generate all the Hilbert matrices up to order n ?
- (3) Verify the correctness of your implementation so far by multiplying your Hilbert matrix and the inverse Hilbert matrix together. Compare the result against the identity matrix.
- (4) Now, use your algorithm of choice (available in Matlab or GSL) to calculate the inverse Hilbert matrix numerically. Compare that against the exact inverse Hilbert matrix using the infinity matrix norm. See what happens as you increase n ?

- (5) Calculate the infinite condition number $\kappa_\infty(H_n) = \|H_n\|_\infty \|H_n^{-1}\|_\infty$ using the exact inverse Hilbert matrix. You might need a separate subroutine to calculate the norms. Make a plot of the condition number against the size of the Hilbert matrix n . It may be necessary to show $\log \kappa_\infty(H_n)$ against n as the condition number will probably increase quite rapidly. How is $\kappa_\infty(H_n^{-1})$ related with $\kappa_\infty(H_n)$?
- (6) **Optional:** Feel free to indulge yourself with the Hilbert matrix. If you can think of some interesting calculation that you can do, try it and discuss what happens. For example, you can compare the numerical behavior of the Hilbert matrix with the behavior of a random orthogonal matrix Q (which would be optimally well-conditioned). You can generate a random orthogonal matrix using the QR decomposition on some random matrix A .