Ill Conditioned Matrices

The *condition number* of a nonsingular matrix A relative to the norm $\|\cdot\|$ is

$$K(A) = \|A\| \cdot \left\|A^{-1}\right\|$$

where the norm that is usually used is the 1−norm for matrices:

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$$

If this value is large then a matrix is said to be *ill conditioned*. What this means in plain terms is that a small change in your matrix A can result in a large change in $\mathbf{b}$. This will affect your ability to accurately solve your system.

If $K(a) = \infty$ then the matrix is singular (i.e. the matrix does not have an inverse).

The Hilbert Matrix

A notable example of a poorly conditioned matrix is the Hilbert matrix. A Hilbert matrix is a square matrix with elements defined by

$$H(i,j) = \frac{1}{i+j-1}$$

For example, a 3×3 Hilbert matrix is

$$H_{3\times 3} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

Note that this matrix is symmetric and postive definite. Matlab has built in commands to produce a Hilbert matrix and its inverse.

Table 1

Command	Description
`hilb(n)`	Creates an $n \times n$ Hilbert matrix
`invhilb(n)`	Calculates the exact inverse of the Hilbert matrix for values of $n < 15$. For larger values of n, computes an approximate inverse.
`cond(A)`	Calculates the condition number of the matrix A
`inv(A)`	Calculates the inverse of the matrix A. Note that using `A^(-1)` is equivalent to this function.

Floating Point Arithmetic

Computers can only store values up to a certain level of accuracy. Past this level, the computer will round values. What this means is that arithmetic does not work exactly as we expect. Namely, arithmetic is no longer commutative, associative, or distributive. The lab exercises will demonstrate some of the issues that arise.

Lab Exercises

I. Ill Conditioned Systems

1. Open and run the script `lab08a.m`.
2. Change the value of `n` so that it greater than or equal to 14. What do you notice about the results?
3. Use `diary('lab08a_output')` to save the resulting output.

II. Finite Precision Point Arithmetic

Open the script file `lab08b.m`. Enter in the required calculations in the indicated spaces of the script file.

1. Calculate
$$s = \sum_{i=1}^{10} 0.1 = 0.1 + 0.1 + \cdots + 0.1$$
We would expect this to be equal to 1.
2. Calculate
$$b = 2 - 3\left(\frac{4}{3} - 1\right)$$
We would expect this to be equal to 1.
3. Calculate each side of the following equation in the variables `lhs` and `rhs`.
$$1 + a + a^2 + a^3 + a^4 + a^5 = \frac{1 - a^6}{1 - a}$$
for the value $a = 0.3$.
4. Calculate the following and compare their output.
```
x = 1e16 + 1 - 1e16;
y = 1e16 - 1e16 + 1;
z = 1e16 - (1e16 - 1);
```
5. Calculate the following and compare their output.
```
u = 1 + 0.1 - 1;
v = 1 - 1 + 0.1;
w = 1 - (1 - 0.1);
```
6. Use `diary('lab08b_output')` to record a single run of your script file. Submit your output and script files via the provided lab report template, uploaded to WyoCourses.