

# floating-point precision and linear systems

Magnus Chr. Hvidtfeldt

Technical University of Denmark, Lyngby, DK,  
s255792@dtu.dk

**Summary.** This document analyzes the numerical stability and precision of linear systems within floating-point environments, providing a structured overview of error metrics and analysis techniques.

## 1 Information

Many students have graphing calculators that can solve difficult questions with minimal ease. The goal of a numerical mathematical course is for the students to learn how the software found the answer.

Numerical algorithms have limitations, as in, it is approximating, so it is important to understand the limitations and for example, note the error estimations also.

## 2 Significant digits of precision

Significant digits are digits beginning with the leftmost nonzero digit, and ending with the rightmost correct digit.

It is important to keep significant digits in a calculation and not round down, as the results can vary greatly.

### 2.1 Absolute and relative error

Absolute error = |exact value - approximate value|

Relative error:

$$\frac{|\text{exact} - \text{approx}|}{|\text{exact}|}$$

Where exact value is the correct value.

The relative error is a good indication of the amount of significant digits.

## 3 Accuracy and rounding

Rule of thumb: Keep as many significant digits as there are in the least accurate number involved in the calculations.

### 3.1 Rounding

When rounding a 5, we look at the preceding number. If it is even, we round down. If it is odd, we round up. (Consider  $9.265 = 9.26$  or  $1.75 = 1.8$ ).

## 4 Floating point representation

Scientific notation:  $x = r \times 10^n$  where  $0.1 \leq r < 1$ .  $r$  is called the Mantissa, and  $n$  is the exponent.

Most real numbers cannot be represented exactly in a computer, because there are a finite number of space for each word length.

### 4.1 Single precision (32-bit representation)

Write the 32-bit representation of  $x$  as eight hexadecimal digits.

The largest number representable is:  $3.4 \times 10^{38}$  The smallest number representable is:  $1.2 \times 10^{-38}$

Any number outside these numbers are called overflow numbers.

Approximately 7 significant decimal digits is possible in single precision.

### 4.2 Double precision (64-bit representation)

The largest number representable is:  $1.8 \times 10^{308}$  The smallest number representable is:  $2.2 \times 10^{-308}$

Approximately 15 significant decimal digits is possible in double precision.

For integer calculations, we have 9 digits for single and 18 for double precision. For high accuracy, we want to use double precision floating point arithmetic.

## 5 Loss of significance

In the case where  $x = 0.3721498 \times 10^{-5}$ , we have that 3 is the most significant digit, and 8 is the least significant digit, because they have different powers of 10.

If  $x$  is an exact number, we can write as many digits as we like.

If  $x$  is a measured quantity, it depends on the measuring device. Only digits that are believed to be correct or the least significant digit should have an error by at most 5 units, such that it is rounded correctly.

## 6 Theorem on Loss of precision

The computer lose significant precision when subtracting two numbers that are near equal to each other. This can be a serious problem.

**Theorem.** *Let  $x$  and  $y$  be normalized floating-point machine numbers, where  $x > y > 0$ . If  $2^{-p} \leq 1 - (y/x) \leq 2^{-q}$  for some integers  $p$  and  $q$ , then at most  $p$  and at least  $q$  significant binary bits are lost in the subtraction  $x - y$ .*

**Example.** Given  $y = .6311$  and  $x = .6353$ , how many significance are lost? We take  $1 - y/x \rightarrow 1 - .6311/.6353 = .00661$ , which lies between  $2^{-8} < .00661 < 2^{-7}$ . Thus 7-8 significant binary bits are lost.

### 6.1 Avoiding loss of precision

Removing the radical in the numerator, if there is a root.

Rewriting a term, for example,  $\sin(x)$  to the Taylor series (series expansion).

Range reduction:  $\sin(x) = \sin(x + 2n\pi)$ . Thus we only need to know the values of  $\sin(x)$  in it's  $2\pi$  interval to compute it for any  $n$ .

**Example.** We have  $\sin(12532.14 - 2k\pi)$  and want  $12532 = 2k\pi$ , get  $k = 12532/2\pi \approx 1994$ . Thus we get  $\sin(12532.14 - 2 * 1994 * \pi) = \sin(3.468)$

## 7 Linear systems

We want to create an algorithm for solving a system of  $n$  linear equations of  $n$  unknowns. In compact form we write a system as

$$\sum_{j=1}^n a_{ij}x_j = b_i, \quad (1 \leq i \leq n)$$

Where  $x_j$  are our unknowns.

### 7.1 Naive gaussian elimination

Forward elimination (get 0s in the first col except row i)

Back substitution (take last pivot row and find  $x$ , and so on up to first row)

Note that Naive gaussian elimination fails if  $a_{11} = 0$ , and likewise fails for some small  $\epsilon x_1$  in row 1. Not very reliable.

### 7.2 Gaussian elimination with partial pivoting (PP)

This method looks at the first column, and picks the row with the largest absolute value, and moves it to the top of the system. Then we perform the same system as naive method.

Pivoting is a way of changing either row or col to get a pivot in a certain place. This is needed if we come across a 0 element in the naive method. Partial pivoting we can only change the rows, but complete pivoting we can change both rows and cols. If the cols are changed also, it will cause the unknown variables to change order.

Note that this fails with very large numbers (near overflow).

### 7.3 Gaussian elimination with Scaled partial pivoting (SPP)

SPP involves first scaling the rows with a certain factor, and then picking the largest value to be in the pivot position, and then from there it's forward elimination and back substitution.

Find the scaling factor by  $s_i = \max_{1 \leq j} |a_{ij}|$ , which is then recorded in a scaling vector  $s = [s_1, \dots, s_n]$ .

Find the scaling ratio by dividing the first col each row  $a_{i1}/s_i$

**Example.** Given a system

$$\left[ \begin{array}{cc|c} 2 & 20000 & 20000 \\ 1 & 1 & 2 \end{array} \right]$$

We have scaling constants  $s = [20000, 1]$  and thus scaling ratios are  $(2/20000, 1/1)$ . Scaled partial pivoting continues to select the second equation as the pivot equation. Then perform forward elimination and back substitution to get your answer with all the significant digits intact.

SPP is not the best solution in all cases, a lot of times id argue PP does its job.

Solving large linear equation systems using SPP involves  $n^3/3$  operations whereas back substitution only requires  $n^2$  operations.

Gaussian elimination and variations are some of the most frequently used algorithms in computational mathematics. Mathematical software often use a variant of partial pivoting and back substitution to solve a linear system.

## 8 Review

- Example 9, sec 1.4.
- Naive gauss algorithm pp. 99-105
- p. 116 SSP pseudocode
- Learning to read pseudocode