

Numerical Methods, project A, Number 31

Krzysztof Rudnicki
Student number: 307585
Advisor: dr Adam Krzemieniowski

October 28, 2021

Contents

1	Problem 1 - Finding machine epsilon	2
1.1	Problem	2
1.2	Theoretical Introduction	2
1.2.1	Definition of machine epsilon	2
1.2.2	Practical applications of machine epsilon	2
1.3	Solution	3
1.3.1	Matlab code	3
1.4	Discussion of the result	3
2	Problem 2 - Solving a system of n linear equations - indicated method	5
2.1	Problem	5
2.2	Theoretical Introduction	5
2.2.1	Transform system of equation into an upper-triangular matrix	5
2.2.2	Backward substitution	7
2.2.3	Partial Pivoting	7
2.3	Solution	7
2.4	Discussion of the result	7
3	Problem 3 - Solving a system of n linear equations - iterative algorithm	8
3.1	Problem	8
3.2	Theoretical introduction	8
3.3	Solution	8
3.4	Discussion of the result	8
4	Problem 4 - QR method of finding eigenvalues	9
4.1	Problem	9
4.2	Theoretical introduction	9
4.3	Solution	9
4.4	Discussion of the result	9

Chapter 1

Problem 1 - Finding machine epsilon

1.1 Problem

Write a program finding macheps in the MATLAB environment

1.2 Theoretical Introduction

1.2.1 Definition of machine epsilon

Machine epsilon is the maximal possible relative error of the floating-point representation. (Tatjewski, p.14) Machine epsilon is equal to 2^{-t} where t is number of bits in the mantissa. In our case when we use IEEE Standard 754, mantissa is 53 bits long with first bit omitted as it is always equal to '1', so we technically work with 52 bits mantissa which makes the machine epsilon equal to: $2^{-52} = 2.220446\text{e-}16$

1.2.2 Practical applications of machine epsilon

Since macheps is connected to IEEE754 standard it is always equal to the same number, which means that we can safely compare results from different machines without worrying about their individual errors.

Macheps is also essential when we calculate cumulation of errors of given mathematical operation.

1.3 Solution

1.3.1 Matlab code

```
1 macheps = 1;
2 while 1.0 + macheps / 2 > 1.0
3     macheps = macheps/2;
4 end
```

Code above shifts macheps one bit to the right each iteration (by dividing by 2), it ends when we run out of mantissa bits which renders us unable to save smaller number. Due to underflow the value of macheps becomes 0 and therefore $1.0 > (\text{macheps} / 2) > 1.0$ will become false.

1.4 Discussion of the result

```
1 format long
2 disp(Display calculated macheps:)
3 disp(macheps);
4 disp(Display actual eps:)
5 disp(eps);
6 disp(Display 2^-52)
7 disp(2^-52)
8 disp(Display difference between calculated macheps and actual eps:)
9 disp(macheps - eps)
10 disp(Display difference between 2^-52 and actual eps:)
11 disp(2^-52 - eps) \
12 disp(Display difference between calculated macheps and 2^-52:)
13 disp(macheps - 2^-52)
```

Display calculated macheps:

2.220446049250313e-16

Display actual eps:

2.220446049250313e-16

Display 2^{-52} :

2.220446049250313e-16

Display difference between calculated macheps and actual eps:

0

Display difference between 2^{-52} and actual eps:

0

Display difference between calculated macheps and 2^{-52} :

0

As expected they are all equal to eachother. It means that our method of calculating macheps was correct.

Chapter 2

Problem 2 - Solving a system of n linear equations - indicated method

2.1 Problem

Write a program solving a system of n linear equations $Ax = b$ using the indicated method (Gaussian elimination with partial pivoting).

2.2 Theoretical Introduction

Gaussian elimination with partial pivoting consists of 3 main steps:

2.2.1 Transform system of equation into an upper-triangular matrix

Starting conditions

We start with the system of linear equations looking like this:

$$\begin{array}{cccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & = & b_n. \end{array}$$

In order for this method to work all the elements of **diagonal** line - $a_{11}, a_{22}, \dots, a_{nn}$ must be different from zero since we will be dividing by them.

We will denote rows as ' w_i ' where 'i' is number of the row.

Zeroing first column

We start transforming the system by **zeroing** elements in first column excluding first row element. We do it by multiplying first row by l_{i1} , where:

$$l_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$$

And then subtracting what we got ($l_{i1}w_1$), from i row.

Doing so we obtain a system of linear equations:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1, \\ 0 & + & (a_{22} - a_{12}l_{21})x_2 & + & \dots & + & (a_{2n} - a_{1n}l_{21})x_n & = & b_2 - b_1l_{21}, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ 0 & + & (a_{n2} - a_{12}l_{n1})x_2 & + & \dots & + & (a_{nn} - a_{1n}l_{n1})x_n & = & b_n - b_1l_{n1}. \end{array}$$

Zeroing second column

We continue onto the second column, this time we will zero all elements except first and second rows. Row multiplier becomes:

$$l_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}$$

Where:

$$a_{22}^{(2)} = (a_{22} - a_{12}l_{21})$$

And:

$$a_{i2}^{(2)} = (a_{i2} - a_{12}l_{i1})$$

They are modified values obtained from previous step. We continue as in the first step and we end up with:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1, \\ 0 & + & a_{22}^{(2)}x_2 & + & \dots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)}, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ 0 & + & 0 & + & \dots & + & a_{nn}^{(3)}x_n & = & b_2^{(3)}, \end{array}$$

Zeroing next columns

We repeat this process $n - 1$ times and we end up with upper triangular matrix:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1, \\ 0 & + & a_{22}^{(2)}x_2 & + & \dots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)}, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ 0 & + & 0 & + & \dots & + & a_{nn}^{(n)}x_n & = & b_2^{(n)}, \end{array}$$

2.2.2 Backward substitution

After transforming the system we solve the system from last to first.
First we calculate value of last element:

$$x_n = \frac{b_n}{a_{nn}}$$

Then one above:

$$x_{n-1} = \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}$$

And so on, for x_k :

$$x_k = \frac{b_k - \sum_{j=k+1}^n a_{kj}x_j}{a_{kk}}$$

2.2.3 Partial Pivoting

Gaussian elimination method has one flaw, where it can come into halt if:

$$a_{kk}^{(k)} = 0$$

To avoid it we use method of pivoting, in our case we will use partial pivoting method. Before each Gaussian elimination step, we do it before each Gaussian elimination step since this will lead to smaller error.

We first find a row i such that:

$$|a_{ik}^k| = \max_j \{|a_{kk}^k|, |a_{k+1,k}^k|, \dots, |a_{nk}^k|\}$$

Then we swap this row with k -th row. Since the matrix we use is assumed to be nonsingular then $|a_{ik}^k| \neq 0$ will be always true. After that we continue with the Gaussian elimination method.

2.3 Solution

2.4 Discussion of the result

Chapter 3

Problem 3 - Solving a system of n linear equations - iterative algorithm

3.1 Problem

3.2 Theoretical introduction

3.3 Solution

3.4 Discussion of the result

Chapter 4

Problem 4 - QR method of finding eigenvalues

4.1 Problem

4.2 Theoretical introduction

4.3 Solution

4.4 Discussion of the result

Bibliography

- [1] Piotr Tatjewski (2014) *Numerical Methods*, Oficyna Wydawnicza Politechniki Warszawskiej