FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Řešiče soustavy lineárních rovnic pro intervalovou aritmetiku
Student:	Michal Demko
Supervisor:	doc. Ing. Ivan Šimeček, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	until the end of summer semester 2022/2023

Instructions

1. Familiarize yourself with existing libraries that support interval arithmetics on the x86 architecture [1,2].
2. Study and analyze alternative representations of interval arithmetics.
3. Implement the Gaussian elimination method using interval arithmetics. Represent the right side by random variables with a given distribution (uniform, normal, ...). Implement a direct solver of a system of linear equations (e.g. Gaussian elimination method) and iterative solver (Gauss-Streidel method)
4. Analyze the performance and correctness of solvers on testing systems of linear equations and analyze the effects of row and column pivotization.

[1] Kulle, Martin: Použití intervalové aritmetiky v řešičích soustav lineárních rovnic. Bakalářská práce. Praha: ČVUT FIT, 2015.
[2] Kulle, Martin: Pokročilé algoritmy pro řešení soustav lineárních intervalových rovnic. Diplomová práce. ČVUT FIT, 2017.

Electronically approved by prof. Ing. Pavel Tvrdík, CSc. on 15 December 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Solvers of systems of linear equations for interval arithmetic

Michal Demko

Department of Information Security
Supervisor: doc. Ing. Ivan Šimeček, Ph.D.

May 10, 2022

Acknowledgements

I would like to thank my supervisor doc. Ing. Ivan Šimeček, Ph.D. for practical suggestions, helpful advice and for the opportunity to work on such interesting thesis.

Thanks should also go to my parents that always supported my decision to study computer science. Last but not least, I would also like to extend my gratitude to my high school classmates that also helped me to get through university, I would never make it without them.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work for nonprofit purposes only, in any way that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 10, 2022 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Demko, Michal. *Solvers of systems of linear equations for interval arithmetic.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstrakt

Cieľom tejto bakalárskej práce je preskúmať efektivitu a následky riešenia systémov lineárnych rovníc s nepresnou pravou stranou. Tá je reprezentovaná náhodnými veličinami a výsledky sú porovnávané s použitím intervalovej aritmetiky. Riešiče implementujeme použitím Gaussovej eliminačnej metódy a tiež Gauss-Seidelovej iteračnej metódy. Cieľ je porovnať výkon týchto metód a skúmať ich použiteľnosť s týmito reprezentáciami. Napriek tomu, že presnosť informácie o skutočnej hodnote sa zvyšuje pomocou oboch metód pri použití náhodnej premennej, taktiež sa zvyšuje náročnosť a čas výpočtov. Najlepšie výsledky v rámci rýchlosti a presnosti vykazuje použitie Gaussovej eliminačnej metódy s pivotáciou. Gauss-Seidelova metóda generuje podobne presné intervaly za niekoľkonásobný čas.

Klíčová slova sústava lineárnych rovníc, náhodná veličina, intervalová aritmetika, Gaussova eliminačná metóda, Gauss-Seidelova metóda

Abstract

The goal of this thesis is to explore efficiency and consequences of solving the systems of linear equations with imprecise right hand side. These are represented by random variables and the results are compared to usage of interval arithmetic. We implement the solvers using Gaussian elimination method and also Gauss-Seidel iteration method. The aim is to compare the performance of these methods and to explore their viability in these representations. While information about real value is increased in the end by both approaches using random variables, the time and difficulty of computation is also increased. The best results within computing speed and precision are reported by Gassian elimination method with pivoting. Gauss-Seidel method generates similarly accurate results, but it is generally slower.

Keywords system of linear equations, random variable, interval arithmetic, Gaussian elimination method, Gauss-Seidel method

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Experimentation often comes with inaccuracies of measured values and therefore can create even more uncertainty when performing numerous calculations over them. Interval is a mean to represent said numbers, but when calculating, we need to take in consideration the correctness of arithmetic operations and the width of the interval, which means its informative value. Solving systems of linear equations is a task used in a wide variety of science fields and applications where inaccuracy is created and therefore we will be representing the right hand side of systems by intervals. We will be comparing two representations of the intervals:

- as an interval with given bounds, where the real value resides
- as a random variable, with the given distribution

The goal of this thesis is going to be the comparison of these two representations of the right hand side of system of linear equations. We will test whether we can use the random variable as a suitable replacement of traditional intervals considering the accuracy of results and the performance of the solvers.

For this purpose we will use two solving methods for linear equation systems, namely the Gaussian elimination method and also the Gauss-Seidel method. We will compare these methods, consider their limitations and also their individual advantages and disadvantages when using random variable and the effect of pivoting. This work will expand on the theses and themes explored in [1] and also [2].

This thesis is divided into five main chapters. The first is an introduction into basic terms of linear algebra and interval arithmetic. For out purposes, the interval arithmetic serves mainly as a baseline comparison due to its exploration in [1].

The next three chapters are dealing with core concepts of this thesis, namely the random variables and the solving methods. These chapters are divided into following sections:

Theory Here we introduce mathematical theory for the chapter

Design This section describes design decisions for the given program aspect

Implementation In the last section we show some implementation details

The last chapter interprets the results of our testing and discusses differences in performance. We also comment on some new observations.

Chapter 1

Mathematical Basics

This chapter introduces mathematical terms and principles needed for the implementation of solvers of systems of linear equations and we will need to introduce terms from linear algebra, that will help us create compact mathematical notations. The chapter than explores and explains two solving methods for these systems. Since we will be representing the right side of equations as random variable distributions, it also introduces necessary definitions from interval arithmetic and random variables.

1.1 Linear Algebra

Linear algebra gives us means of creating systems of linear equations that are easy to read and understand. Matrices and other useful definitions will be introduced in this section to create simple notations for our linear equation systems. Using linear algebra, we will also explain two solving methods for the systems, namely the Gaussian elimination method and the Gauss-Seidel method [3] [4] [5].

Definition 1.1.1 (Matrix) *Let $m, n \in \mathbb{N}$ and $a_{ij} \in F$ for all $i \in \{1,...,m\}$ and all $j \in \{1,...,n\}$. The rectangle-shaped array*

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

*of m rows and n columns is called a **matrix** of the type $m \times n$ with entries from the field F.*

Definition 1.1.2 (Matrix multiplication) *Let $m, n, p \in \mathbb{N}, \mathbb{A} \in \mathbb{R}^{m,n}$ matrix with elements $\mathbb{A}_{ij}$ and $\mathbb{B} \in \mathbb{R}^{n,p}$ matrix with elements b_{ij}. Multiplication of matrices $\mathbb{A}$ and $\mathbb{B}$ is matrix $\mathbb{D} \in \mathbb{R}^{m,p}$ with elements d_{ij}*

$$d_{ij} = \sum_{k=1}^{n} \mathbb{A}_{ik}\mathbb{B}_{kj}$$

Denoted $\mathbb{D} = \mathbb{A}\mathbb{B}$

Definition 1.1.3 (Identity matrix) *We call matrix $\mathbb{A} \in \mathbb{R}^{n,n}$ the* **identity** *matrix, if*

$$\begin{aligned} i \neq j &\rightarrow a_{i,j} = 0 \\ i = j &\rightarrow a_{i,j} = 1 \end{aligned}$$

We represent this matrix by $\mathbf{I}$

Definition 1.1.4 (Regular matrix) *Let $\mathbb{A} \in \mathbb{R}^{n,n}$. If there is matrix $\mathbb{B} \in \mathbb{R}^{n,n}$, that*

$$\mathbb{A}\mathbb{B} = \mathbb{B}\mathbb{A} = \mathbf{I}$$

we can call $\mathbb{A}$ ***regular****. If matrix is not regular, then we call it* ***singular****.*

Definition 1.1.5 (Matrix forms) *The square matrix $\mathbb{A} \in \mathbb{R}^{n,n}$ is called*

upper triangular *if all the entries below the main diagonal are equal to zero:*

$$(\forall i, j \in \{1, \ldots, n\})(i > j \Rightarrow \mathbb{A}_{i,j} = 0)$$

lower triangular *if all the entries above the main diagonal are equal to zero:*

$$(\forall i, j \in \{1, \ldots, n\})(i < j \Rightarrow \mathbb{A}_{i,j} = 0)$$

diagonal *if it is both upper and lower triangular:*

$$(\forall i, j \in \{1, \ldots, n\})(i \neq j \Rightarrow \mathbb{A}_{i,j} = 0)$$

Definition 1.1.6 (Diagonally dominant matrix) *The matrix $\mathbb{A}$ is diagonally dominant if*

$$|a_{ii}| \leq \sum_{j \neq i} |a_{ij}|$$

for all i, where a_{ij} denotes the entry in row i and column j.

Definition 1.1.7 (Symmetric matrix) *The matrix $\mathbb{A}$ is symmetric if $\mathbb{A} = \mathbb{A}^T$.*

$$\mathbb{A} \text{ is symmetric} \iff \mathbb{A} = \mathbb{A}^T$$

Definition 1.1.8 (Positive-definite matrix) *Let $\mathbb{A} \in \mathbb{R}^{n,n}$ be symmetric. $\mathbb{A}$ is said to be positive-definite if*

$$x^T \mathbb{A} x > 0; \forall x \in \mathbb{R}^n \setminus \{0\}$$

Definition 1.1.9 (System of linear equations) *By the system of linear equations is meant*

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ a_{3,1}x_1 + a_{3,2}x_2 + \cdots + a_{3,n}x_n &= b_3 \\ \vdots & \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n &= b_m \end{aligned}$$

where $a_{i,j}$ are coefficients of the system, x_j are unknowns and b_i are called constants. System of linear equations can have zero, one or multiple solutions. We can rewrite these systems using matrix notation 1.1.1 to the matrix form as matrix $\boldsymbol{A}$ and vectors $\boldsymbol{x}$ and $\boldsymbol{b}$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

To represent this system we can use short notation

$$\boldsymbol{Ax = b}$$

We will explain solving methods for these systems in their own chapters. By solving systems of linear equations, we are looking for the respective vector $\mathbf{x}$ that complies notation $\mathbf{Ax = b}$. From now on in this thesis, we will only consider square regular matrices, because they only have one solution, which is enough to illustrate general principles and goals stated in the beginning.

1.2 Interval Arithmetic

Interval arithmetic is a mean to compute with intervals. These are defined as bounds on a real axis, which means that we use the lower bound and the upper bound. The real value of a number is somewhere in between these bounds with no information where exactly. Arithmetic operations on intervals are well defined and expect this uncertainty. Computing with intervals usually means increasing this uncertainty by widening the intervals [6] [7].

Definition 1.2.1 (Interval) *Closed interval $\boldsymbol{I}$, with lower bound $a \in \mathbb{R}$ and upper bound $b \in \mathbb{R}$, is a set*

$$\boldsymbol{I} = \{x \in \boldsymbol{R} | a \leq x \leq b\}$$

Definition 1.2.2 (Binary operations) *Let $\boldsymbol{I}_1, \boldsymbol{I}_2$ be intervals and $\circ$ is binary operator. Then*

$$\boldsymbol{I}_1 \circ \boldsymbol{I}_2 = \{x_1 \circ x_2 | x_1 \in \boldsymbol{R}, x_2 \in \boldsymbol{R}\}$$

In general, the result of operation does not have to be interval. But operations +,-,·,/ (except division by 0) always result in another interval. We can also define precise formulation for these relations.

$$\begin{array}{rcl} [a,b] + [c,d] & = & [a+c, b+d] \\ [a,b] - [c,d] & = & [a-d, b-c] \\ [a,b] \cdot [c,d] & = & [min(ac,ad,bc,bd), max(ac,ad,bc,bd)] \end{array}$$

Because we will be solving systems of linear equations with the left hand side consisting of real numbers, we will also define binary operations on intervals with intervals of wid(I)=0. These intervals can be treated as constants, which does not interfere with the regular binary operation definition 1.2.2.

Definition 1.2.3 *Let $\boldsymbol{I}_1 = [a,b], \boldsymbol{I}_2 = [c,c]$ be intervals. We can treat $\boldsymbol{I}_2$ as a constant and express operations +,-,· by relations:*

$$\begin{array}{rcl} [a,b] + c & = & [a+c, b+c] \\ [a,b] - c & = & [a-c, b-c] \\ [a,b] \cdot c & = & [min(ac,bc), max(ac,bc)] \end{array}$$

We will be using the interval arithmetic as a baseline performance indicator for our implementation of systems of linear equations solvers. Because it is well defined and explored, it is well suited to serve as our test sample.

Chapter 2

Analysis of Random Variable

The random variables and operations over them are the base of this thesis. This interpretation of the intervals provides means of giving values certain probability inside them. Using only general representation of intervals, we can only assume that the real value we look for is within given bounds. Nevertheless, this is usually not enough in science experiments, where observations for example, have far more probability to be closer to the real value, than to be far from it. Therefore when we represent this interval as the random variable, we may get much better idea of the real form of the interval and its correct distribution.

2.1 Theory

Here we will explore basic definitions needed to understand what the random variable is, how to represent it and how to compute with it. We will also introduce some special cases of random variable distributions [8] [9] [10] [11].

Definition 2.1.1 (Random variable) *Random variable X on a probability space ($\Omega, \mathcal{F}, P$) is a function $X : \Omega \to \mathbb{R}$, assigning a number X($\omega$) to each outcome $\omega \in \Omega$ with the property that:*

$$\{X \leq x\} \in \mathcal{F}, \forall x \in \mathbb{R}$$

Such function is said to be $\mathcal{F}$-measurable.

Definition 2.1.2 (Discrete random variable) *X is called discrete random variable, if it takes only values from some countable set $\{x_1, x_2, \dots\}$. Probabilities of the values of a discrete random variable X are given by*

$$P(X = x_k), k = 1, 2, \dots$$

When assigning probabilities to the values x_k, the normalization condition must hold:

$$\sum_{all x_k} P(X = x_k) = 1$$

Definition 2.1.3 (Continuous random variable) *Random variable X is called continuous, if there exists a non-negative function $f_X : \mathbb{R} \to [0, +\infty)$ such that for all $x \in \mathbb{R}$ the distribution function F_X can be expressed as*

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\mathrm{d}t$$

Function f_X is called the probability density of the random variable X.

Definition 2.1.4 (Uniform distribution) *Let [a,b] be a real interval. Random variable X has a uniform distribution, if its density function is:*

$$f_X(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & elsewhere \end{cases}$$

Definition 2.1.5 (Normal distribution) *Random variable X has the normal (Gaussian) distribution with parameters μ and $\sigma^2 > 0$, if the density has the form:*

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

These distributions will serve as our RHS representations and we will observe their effects on the results.

Definition 2.1.6 (Operations over random variables) *Let X, Y be random variables with known distribution and f be defined on Cartesian product of X and Y values. Then $Z=f(X,Y)$ is random variable and its distribution function is:*

$$F_Z(x) = P(Z < x) = P(f(X, Y) < x)$$

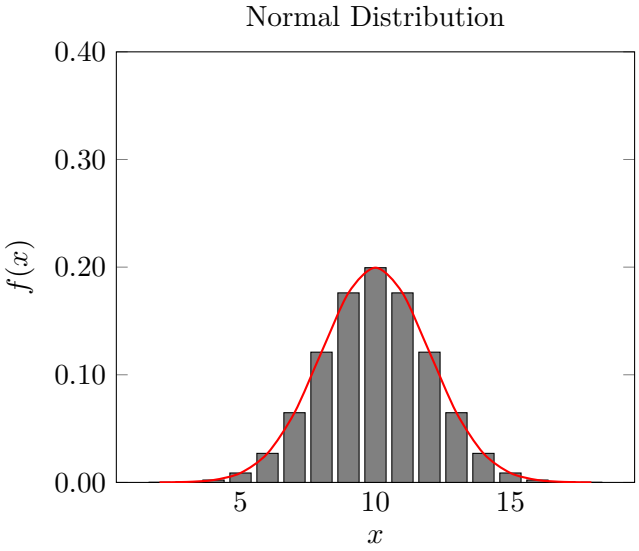
Definition 2.1.7 (Linear transformations) *Let X be a random variable taking values from an interval $I \in \mathbb{R}$ and that X has a continuous distribution on I with probability density function f. Let $Y = a + bX$, where $a \in \mathbb{R}$ and $b \in \mathbb{R} \setminus 0$. The Y than takes values:*

$$T = \{y = a + bx; x \in I\}$$

2.2 Design

The design of random variable in the program is based on an approximation of the continuous random variable by the discrete version. In case we have discrete values 2.1.6, we can say $P(Z = x) = P(f(X, Y) = x)$, which also allows us to make our random variables in a certain precision.

Say we have a continuous random variable X defined on the interval I, e.g. $P(X \in I) = 1$. We can then cut this interval into smaller intervals and represent those by their discrete values.2.2 Therefore we do not need to represent the probabilities as $P(X \in [x_i, x_i + 1])$, but instead we use simpler notation $P(X = x_i)$, creating a discrete random variable. These can have only countably many possible values and the more we have, the closer we converge to the original continuous random variable.



In our solution, we decided to treat the separation of intervals into smaller pieces by defining a unit and a number of steps. The unit being an interval $I = [a, b]; a, b \in \mathbb{Z}$ and $wid(I) = 1$. Steps is a number of parts that the unit is supposed to be separated into, so the bigger the number, the smaller the intervals it creates and therefore our approximation is more precise. This system allows us to create arbitrary options for precision, but for testing purposes, we defined six step numbers:

(1) $\rightarrow wid(I_i) = wid(I) = 1; i = 1$

(2) $\rightarrow wid(I_i) = 0.5; \forall i \in \{1, 2\}$

(4) $\rightarrow wid(I_i) = 0.25; \forall i \in \{1, 2, 3, 4\}$

(5) $\rightarrow wid(I_i) = 0.2; \forall i \in \{1, 2, 3, 4, 5\}$

$$(10) \rightarrow wid(I_i) = 0.1; \forall i \in \{1, \ldots, 10\}$$

$$(100) \rightarrow wid(I_i) = 0.01; \forall i \in \{1, \ldots, 100\}$$

We always have these lower intervals set on the same values between integers.

This approach also creates an issue of imprecise lower and upper bounds of the interval. For example, if we had an interval of $I = [1.5, 2]$ and steps number of 1, the resulting representation would be $I_i = [1, 2]$ and no matter what distribution, in our representation, it would create only one probability $P(X_i = 1) = 1$, which does not hold any informative value. This issue is only solvable by increasing the steps value and we will look at this more closely in the last chapter.

The real problem with usage of random variables as representations of intervals comes with their arithmetic operations [8]. Let us have random variables X, being represented by n values, and Y, being represented by m values. Because we cannot assume any dependencies between these variables, adding or multiplying them together has complexity of $O(nm)$. If represented by the very high step number, this might take a lot of space and performance. Considering we want to represent RHS of the SLE by these, which means many operations of addition over them, it is not ideal due to limitations in performance. This issue might be improvable by parallelization of these operations, however we will not examine this approach in this thesis.

2.3 Implementation

We implement our approximation of random variable in C++ and its Standard Template Library to represent it as a `std::map<double,double>` structure. The key being the representation of the given subinterval and the value of it, its probability. The biggest obstacle of this implementation was the correct placement of key resulting from multiplication of two random variables. The basic algorithm behind addition can be seen in pseudocode 2.1.

Given that addition or multiplication of keys does not result into predefined step definitions we wanted to create, but actually falls into those intervals $(x_{key} \circ y_{key}) \in I_i$, we had to create a way to align these keys, to correct place and value so that $f(x_{key} \circ y_{key})$ is equal to lower bound of I_i. In theory, it might work in this state, but the problem could be extreme spacial requirements. We also need to take in consideration our representation of the key. Because we represent it by `double`, it would almost never create the same two keys and therefore every new multiplication only creates more elements.

For this purpose we take the original result of operation and make additional rounding based on the step size we predefined. Let us say we have predefined step value $s = 0.25$ and we get a result of key multiplication of $z_i = 6.5132$. We can take the value after the floating point from z_i and divide

it by s. We multiply the result integer again by s and add it to the original integer from z_i, which creates our aligned key value of 6.5. This algorithm can be seen in pseudocode 2.2.

It is worth noting, that for representation of SLE we do not actually need to implement random variable multiplication, because as described in next two chapters, elements of RHS are not multiplied among each other.

Algorithm 2.1 Random variable addition

```
1: for x_key, x_value in X do
2:     for y_key, y_value in Y do
3:         NewKey ← x_key + y_key
4:         z_NewKey ← x_val * y_val
5:     end for
6: end for
```

Algorithm 2.2 Steps align

```
1: z_float ← floatremainder(z_i)
2: z_int ← integer(z_i)
3: Result ← z_int + integer(z_float/s) * s
```

Chapter 3

Analysis of Gaussian Elimination Method

The first solving algorithm to be used in this thesis is the Gaussian elimination method. It consists of series of elementary row operations used on the matrix to find the general solution of the matrix. On the assumption that numerical operations defined on the representation of the left hand side of the equations are also defined on the representation of the right hand side, we are able to apply this solving method.

3.1 Theory

This section will introduce definitions needed for the algorithm and the algorithm itself [3].

Definition 3.1.1 (Row echelon form) *Let* $\mathbf{A} \in \mathbf{F}^{m,n}$ *and* $\boldsymbol{\theta} \in \mathbf{F}^{1,n}$ *be a zero vector. The matrix* $\mathbf{A}$ *is called to be in row echelon form, if it satisfies the following conditions:*

1. *There exists row index* $k \in 0, 1, \ldots, m$ *such that*

$$(\forall i \leq k)(\mathbf{A}_{i:} \neq \boldsymbol{\theta}) \wedge (\forall i > k)(\mathbf{A}_{i:} = \boldsymbol{\theta})$$

   i.e., all non-zero rows are above any zero rows.

2. *For indexes*

$$j_i := min\{l \in \{1, \ldots, n\} | \mathbf{A}_{il} \neq 0\}, i \in \{1, \ldots, k\}$$

   it holds that $j_1 < j_2 < \cdots < j_k$*, i.e., the first non-zero entry from the left (called pivot) of a non-zero row is always strictly to the right from the pivot of the row above.*

This definition is broadly specified for every matrix, but considering only square regular matrices in this thesis, we only need its second part. First part of the GEM algorithm is to get the LHS of system to the row echelon form, using elementary operations (G1)–(G3). These operations can be described as:

(G1) Swap positions of two rows

(G2) Multiply i-th row by a non-zero scalar $\alpha \in F$, i.e., replace the row

$$\left(\begin{array}{cccc|c} a_{i1} & a_{i2} & \dots & a_{in} & b_i \end{array} \right)$$

with the row

$$\left(\begin{array}{cccc|c} \alpha a_{i1} & \alpha a_{i2} & \dots & \alpha a_{in} & \alpha b_i \end{array} \right)$$

(G3) Add the scalar multiple of j-th row to i-th row, i.e., replace the row

$$\left(\begin{array}{cccc|c} a_{i1} & a_{i2} & \dots & a_{in} & b_i \end{array} \right)$$

with the row

$$\left(\begin{array}{cccc|c} a_{i1} + \alpha a_{j1} & a_{i2} + \alpha a_{j2} & \dots & a_{in} + \alpha a_{jn} & b_i + \alpha b_j \end{array} \right)$$

Definition 3.1.2 (Gaussian elimination method) *Let $\mathbb{A} \in \mathbb{R}^{n,n}$ be regular matrix with vector b as its RHS. Then by using operations (G1)–(G3) do these steps:*

1. *Add to all rows $p \in \{i+1, \dots, n\}$, where i denotes first row, including RHS, row i multiplied by number $((-\mathbb{A}_{p,i})/\mathbb{A}_{i,i})$*
2. *Repeat part (1) for each consecutive row*

This algorithm creates the row echelon form of the matrix $\mathbb{A}$, assuming that we do not create zero on the diagonal, otherwise division operations would not be defined. This effect should be repressed by use of pivotation.

Definition 3.1.3 (Reverse GEM) *Let regular matrix $\mathbb{A} \in \mathbb{R}^{n,n}$ be in REF, with vector b as its RHS. Then by using operations (G1)–(G3) do these steps:*

1. *Add to all rows $p \in \{1, \dots, i-1\}$, where i denotes the last row, including RHS, row i multiplied by number $((-\mathbb{A}_{p,i})/\mathbb{A}_{i,i})$*
2. *Divide the row i by $\mathbb{A}_{i,i}$*
3. *Repeat part [1] and [2] for each consecutive row from last to first*

Combination of these two algorithms creates the solution of the SLE. In case of regular square matrices, it finds the only solution the given SLE has. We will be applying Reverse GEM only on the RHS as seen in pseudocode 3.2. Using this approach we can get better performance.

Definition 3.1.4 (Row pivoting) *Let $\mathbb{A} \in \mathbb{R}^{n,n}$ be square matrix. Let p be column index of the matrix element, such as*

$$|\mathbb{A}_{i,p}| = max(|\mathbb{A}_{i,1}|, |\mathbb{A}_{i,2}|, \ldots, |\mathbb{A}_{i,n}|)$$

The matrix element $\mathbb{A}_{i,p}$ is called pivot of row i. We say that matrix $\mathbb{B}$ is created by pivoting row i, if we exchange said columns.

Definition 3.1.5 (Column pivoting) *Let $\mathbb{A} \in \mathbb{R}^{n,n}$ be square matrix. Let p be the row index of the matrix element, such as*

$$|\mathbb{A}_{p,j}| = max(|\mathbb{A}_{1,j}|, |\mathbb{A}_{2,j}|, \ldots, |\mathbb{A}_{n,j}|)$$

The matrix element $\mathbb{A}_{p,j}$ is called the pivot of column j. We say that matrix $\mathbb{B}$ is created by pivoting column j, if we exchange said rows.

By combining row and column pivoting we create completely pivoted matrix. Important part of applying column pivoting on the matrix in SLE is that we also need to switch elements in the RHS vector on their respective positions.

3.2 Design

The design of our SLE solver with GEM is a direct consequence of the definitions in previous section. It is designed in four different parts:

1. GEM
2. Reverse GEM
3. Column pivoting
4. Row pivoting

This way we can create every combination of pivoting with ease and decide which one to use, based on arguments. There is no need for runtime condition decisions, which in theory improves performance of individual solvers.

There are also other possibilities of improving the performance of the solver. For example, GEM is also a great candidate for parallelization of the main algorithm. We will not be implementing this technique in this thesis, but it is important to mention and take this in consideration when we rate its performance.

3.3 Implementation

Implementation of these parts can be seen in pseudocode 3.1 3.2 3.3 3.4. As in entire project, we use C++ and its Standard Template Library to represent these structures. We implement the RHS of SLE as a vector. In this vector are stored either our own representations of random variables or intervals implemented by PROFIL/BIAS library [12].

Algorithm 3.1 GEM

```
1: for i = 0 to n do
2:     for j = i + 1 to n − 1 do
3:         Ratio ← A_{i,j}/A_{i,i}
4:         for k = i to n do
5:             A_{k,j} ← A_{k,j} − Ratio ∗A_{i,i}
6:         end for
7:         b_j ← b_j − ratio ∗ b_i
8:     end for
9: end for
```

Algorithm 3.2 Reverse GEM

```
1: for i = n − 1 to 0 do
2:     for j = n − 1 to i + 1 do
3:         b_i ← b_i − A_{j,i} ∗ b_j
4:     end for
5:     b_i ← b_i ∗ (1/A_{i,i})
6: end for
```

If we want to find the column/row pivot of the matrix during computation, we can insert respective code 3.4 3.3 between lines 1 and 2 in the GEM algorithm 3.1. This operation can improve the precision of the result and consequentially the performance of our solver. Because higher precision means narrower intervals, we will need less operations to use by our representation of random variable. This effect will be explored more in the last chapter.

Algorithm 3.3 Row pivoting

```
1: Max ←|𝔸_{i,i}|
2: index ← i
3: for j = i to n do
4:     if |𝔸_{j,i}| > Max then
5:         Max ← 𝔸_{j,i}
6:         index ← j
7:     end if
8: end for
9: SwitchMatrixColumns(i, index)
```

Algorithm 3.4 Column pivoting

```
1: Max ←|𝔸_{i,i}|
2: index ← i
3: for j = i to n do
4:     if |𝔸_{i,j}| > Max then
5:         Max ← 𝔸_{i,j}
6:         index ← j
7:     end if
8: end for
9: SwitchMatrixRows(i, index)
10: SwitchElements(b_i, b_index)
```

Chapter 4

Analysis of Gauss-Seidel Method

The other algorithm used to solve systems of linear equations in this thesis is the Gauss-Seidel method. It is an iteration method, which means that we use a sequence of vectors, that converge to solution by applying numerical operations on them. We also need to make sure, that numerical operations used are defined on the representations of the RHS. This method also requires some matrix attributes and we will explore this in next the section.

4.1 Theory

This section introduces the iteration principle of solving the SLE and two methods. The Jacobi method and also the Gauss-Seidel method which derives from it and is the main method observed by this chapter. These are described in [13] [14] [1].

4.1.1 Iteration Methods

The iteration methods use the consecutive vectors of partial solutions to converge to the solution of SLE. The result of these operations is therefore not the exact solution, but mere approximation of it. However, we can assume the maximum error of the resulting vector. For the SLE denoted $Ax = b$, using iteration method we can find vectors $x_1, \ldots, x_\infty$ as for this sequence of vectors applies:

$$x = \lim_{k \to \infty} x_k$$

and the x is the correct solution for SLE. While the infinite sequence of vectors cannot be realistically found, we usually settle for the result that assures maximal error ϵ. We would stop the calculation whenever $\|x_{k+1} - x_k\| < \epsilon$, where the $\|.\|$ is the vector norm.

4.1.2 Jacobi Method

This method is the base iteration method used to solve SLE. These transform the system

$$\mathbf{A}x = b$$

to the

$$x_{k+1} = \mathbf{B}x_k + c$$

where $x_0 \in IR^n$. From this we get infinite sequence $\{x_k\}_{k=0}^{\infty}$, that in special cases converges to

$$x = \lim_{k\to\infty} x_k$$

For this method, the sequence converges, if $\mathbf{A}$ is diagonally dominant1.1.6. We also need to convert the matrix $\mathbf{A}$ into $\mathbf{B}$ and find vector $\mathbf{c}$. This method transforms the matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$$

where $\mathbf{D}$ is diagonal matrix, L is lower triangular and $\mathbf{U}$ is upper triangular matrix 1.1.5. The SLE $Ax = b$ can be then denoted as:

$$(\mathbf{D} - \mathbf{L} - \mathbf{U})x = b$$

we can then gradually adjust this to:

$$\mathbf{D}x = (\mathbf{L} + \mathbf{U}) + b \to x = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})x + \mathbf{D}^{-1}b$$

When we then denote $\mathbf{B} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$ and $c = \mathbf{D}^{-1}b$, we then get the correct equation introduced in the beginning:

$$x = \mathbf{B}x + c$$

This equation can be also formulated for the i-th element of k-th step:

$$x_i^k = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1,j\neq i}^{n} a_{i,j}x_j^{k-1}\right)$$

This form can be further rewritten as subtraction of two sums, which gives us final form for the equation.

$$x_i^k = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j}x_j^{k-1}\right)$$

The vector x_0 can be chosen arbitrarily. We usually choose zeroed vector as a starting point, however we also can use our own estimate, which could result into faster converging to the result.

4.1.3 Gauss-Seidel Method

This method is based on the previous Jacobi method. To compute the x_i^{k+1} is used vector x^k in combination with the already computed elements from vector x^{k+1}. This means that to compute element x_i^{k+1}, we use the values $x_{1,\ldots,i-1}^{k+1}$ and $x_{i+1,\ldots,n}^{k}$. The following equation is the mathematical formulation of this element:

$$x_i^k = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^k - \sum_{j=i+1}^{n} a_{i,j} x_j^{k-1} \right)$$

The main assumption for this method to converge to the result is that matrix **B** is either diagonally dominant 1.1.6 or symmetric 1.1.7 and positive definite 1.1.8. The main advantage over Jacobi method is that we can store values of x^{k+1} and x^k only in single temporary vector. The disadvantage is that we need to compute these values sequentially, therefore we cannot use parallelization.

4.2 Design

The design of the solver is quite simple in comparison with the Gaussian elimination method. To stop the run of the algorithm, we do not use vector norm, but we have set number of cycles to calculate. The reason is that we will compare how close the result gets to the answer using given cycles count in comparison with the GEM. This gives us better benchmark performance to compare and more flexibility during the testing.

The algorithm itself follows the definition of equation in previous section 4.1.3. We have three nested cycles. The main uses defined number of Gauss-Seidel runs. The second runs through the width of the matrix and computes the value of x_i. The last is decomposed into two sub-cycles, which means we do not need to have the condition to check whether $j = k$. We can see this algorithm as the pseudocode 4.1.

4.3 Implementation

The implementation of this solver can be again seen in pseudocode 4.1. We use C++ language and the Standard Template Library to represent all major objects, namely RHS. That is again represented as a vector of elements, in our case mainly the random variables.

When computing, we first declare vector x_0 with zeroed equivalent of elements inside RHS of the SLE. Then we iterate through the algorithm as many times as specified during the testing.

Here we can see the inherent problem of this method, resulting into slower performance. The main issue being constant addition of RHS values for whole

row. So by having the matrix $A \in \mathbf{R}^{n,n}$ with n elements in each row, we need to do $n-1$ multiplications of the vector x element followed by $n-1$ additions of these elements. After that we subtract the sum for every row from the element of RHS and again multiply this by constant. This gets us to many operations on our implementation of random variable and can be potential performance issue. We will discuss this in the next chapter.

Algorithm 4.1 GS Method

```
1: for j = 0 to n − 1 do
2:     Sum ← 0
3:     for k = 0 to j − 1 do
4:         Sum ← Sum + a_{j,k} ∗ x_k
5:     end for
6:     for k = j + 1 to n − 1 do
7:         Sum ← Sum + a_{j,k} ∗ x_k
8:     end for
9:     x_j ← (b_j − Sum) ∗ 1/a_{j,j}
10: end for
```

Chapter 5

Analysis of Results

This chapter will analyze results that our implementations of the Gaussian elimination method, the Gauss-Seidel method and the random variable created. At first, we will introduce our sample matrices and also the intervals used. We will briefly comment on the benchmark results of the interval arithmetic, that will serve as a baseline and we will consider these as correct. Then we will comment on the different settings for the random variables and how it affects the solvers.

5.1 Test Data

We generated test data using Python script, creating random matrices, but with some rules to comply computability by our methods. We especially had to generate matrices that are calculable by the Gauss-Seidel method, so we made diagonally dominant1.1.6 matrices, or more precisely matrices that can be adjusted to this form by some simple valid matrix manipulation. This way we can also test the effect of pivoting the matrix when using Gaussian elimination method.

Using this script, we created thirty 4x4 matrices, twenty 5x5 matrices and also four 10x10 matrices. We also tried to implement and test 20x20 matrices, but we were unable to calculate these SLE due to performance limitations.

For the RHS of our systems of linear equations we also generated random intervals using Python script. For this purpose, we created various formats of their representations. First being regular intervals with upper and lower bound. These serve as our baseline to work with and are used by regular interval arithmetic to compute our benchmark results.

We then used these intervals and generated for them two types of random variables following normal 2.1.5 and uniform 2.1.4 distribution. For each of them were generated discrete distributions according to our steps 2.2 definition. We created these test distributions for steps:

- step = 1
- step = 2
- step = 4
- step = 10

To represent the general principles of our discoveries we chose these three matrices, but for the trends derived and described in this thesis, we generalized findings from all of the results:

$$A = \begin{pmatrix} 36 & 12 & 20 & 108 \\ 20 & 80 & 12 & 36 \\ 432 & 80 & 32 & 160 \\ 25 & 10 & 100 & 10 \end{pmatrix} B = \begin{pmatrix} 176 & 704 & 198 & 154 \\ 800 & 125 & 75 & 175 \\ 18 & 4 & 10 & 54 \\ 196 & 168 & 896 & 196 \end{pmatrix}$$

$$C = \begin{pmatrix} 42 & 126 & 840 & 210 & 168 \\ 180 & 72 & 180 & 720 & 180 \\ 4 & 36 & 8 & 5 & 5 \\ 780 & 104 & 156 & 52 & 130 \\ 36 & 72 & 54 & 54 & 324 \end{pmatrix}$$

And these are their associated RHS vectors which contain the intervals. We will consider their representations and distributions in other sections:

$$RHS(A) = \begin{pmatrix} [51, 62] \\ [9, 17] \\ [99, 108] \\ [42, 51] \end{pmatrix} RHS(B) = \begin{pmatrix} [95, 105] \\ [18, 26] \\ [33, 41] \\ [85, 96] \end{pmatrix} RHS(C) = \begin{pmatrix} [86, 97] \\ [31, 43] \\ [7, 19] \\ [69, 77] \\ [93, 103] \end{pmatrix}$$

5.2 Interval Arithmetic

Considering this topic to be thoroughly researched in previous thesis [1], we will not be analyzing the interval arithmetic implementations to the details. We use this as a baseline results to compare with our own representation of intervals, therefore we consider these as correct. For our implementation we used the PROFIL/BIAS [12] library and its class `INTERVAL`. We decided for this library for its performance in comparison with different libraries [15] and relatively simple use in linux environment.

Here we will demonstrate the results from every type of our computations. Considering we have four matrices with various number of rows and also six different types of computation, we will only show the vector element x_1 of each solution or some different element of interest. We will also display its computing time. The types of the computations we will compare:

GEM GEM without pivoting

GEMrow GEM with row pivoting

GEMcol GEM with column pivoting

GEMcom GEM with complete pivoting

GSM5 GSM with five cycles

GSM10 GSM with ten cycles

The values for the GSM method cycles were chosen to test how much the interval width and performance depend on them. We chose these values, because of random variable performance requirements. Therefore we used same values also with the interval arithmetic implementation, to compare their effect on both.

Table 5.1: Element x_1 results for matrix A

Method	Value	Width	Computing time (ms)
GEM	[-0.950,0.679]	1.629	0.0109
GEMrow	[0.380,0.557]	0.177	0.0104
GEMcol	[0.396,0.541]	0.145	0.0127
GEMcom	[-0.194,-0.077]	0.117	0.012
GSM5	[0.358,0.579]	0.221	0.0125
GSM10	[0.357,0.580]	0.223	0.0167

Table 5.2: Element x_1 results for matrix B

Method	Value	Width	Computing time (ms)
GEM	[-3.251,2.983]	6.234	0.0107
GEMrow	[-0.013,0.063]	0.076	0.0147
GEMcol	[-0.004,0.054]	0.058	0.0133
GEMcom	[-0.045,0.094]	0.139	0.0114
GSM5	[-0.026,0.075]	0.101	0.0124
GSM10	[-0.026,0.076]	0.103	0.0177

Table 5.3: Element x_1 results for matrix C

Method	Value	Width	Computing time (ms)
GEM	[-4.280,4.299]	8.580	0.0754
GEMrow	[-0.033,0.082]	0.115	0.0435
GEMcol	[-0.296,0.954]	1.250	0.0636
GEMcom	[-0.118,0.167]	0.285	0.0580
GSM5	[-0.060,0.110]	0.170	0.0510
GSM10	[-0.064,0.114]	0.178	0.0997

5.3 Random Variables

In the following subsection will be compared different step sizes 2.2 of random variables with interval arithmetic and between each other. We will begin with the biggest step size, that being one, and further continue to the smaller steps.

The interpretation of these results will be shown by methods similar to the interval arithmetic, but we will also show histograms representing some results in their true form.

5.3.1 Testing of Numerical Operations

Using the random variables $x = [4, 6]$ and $y = [3, 5]$ with uniform distribution, we will demonstrate addition over random variables and multiplication by constant.

Addition combines all elements of the random variables, to the resulting random variable. Adding great number of random variables generally results into shape resembling normal distribution.

Figure 5.1: Addition of x and y

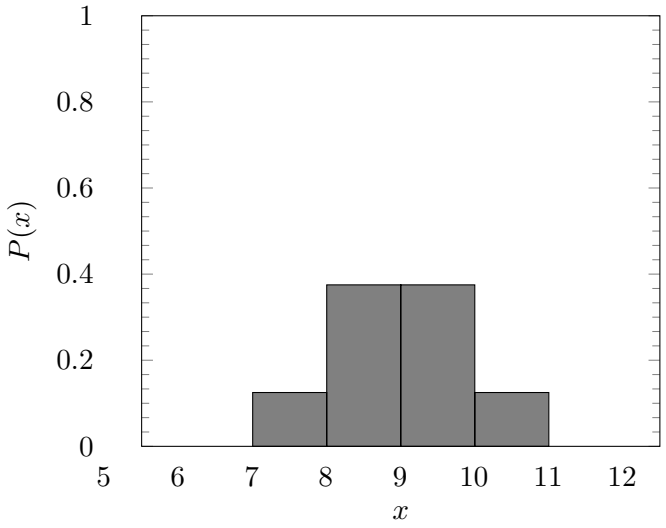
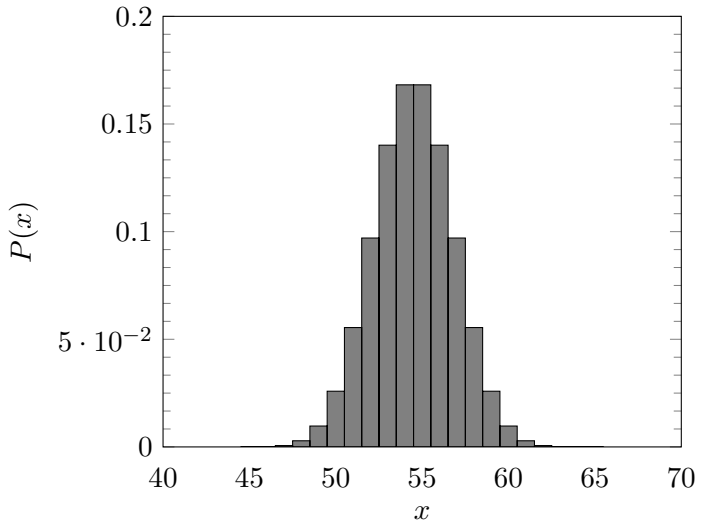
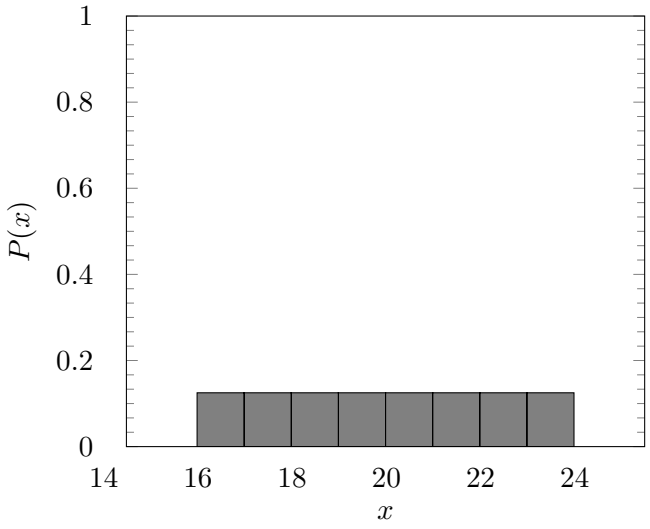


Figure 5.2: Ten additions of the x



Multiplying the random variable results into widening or narrowing the interval. The distribution preserves the shape as close as allowed by the step size.

Figure 5.3: Multiplying x by four



5.3.2 Step Size One

The results for this setting appeared to have the least informative value about the resulting intervals. Due to the nature of our matrices, that being ability to easily transform them to the diagonally dominant form, which is important for GSM, they appear to usually narrow our resulting intervals. That in most cases leads to the width of these intervals to be less than 1, where we loose all information about random variable. This effect can be seen table 5.4 with results for matrix B with uniform distribution configuration.

Table 5.4: Element x_1 results for matrix B (Random variable, step one)

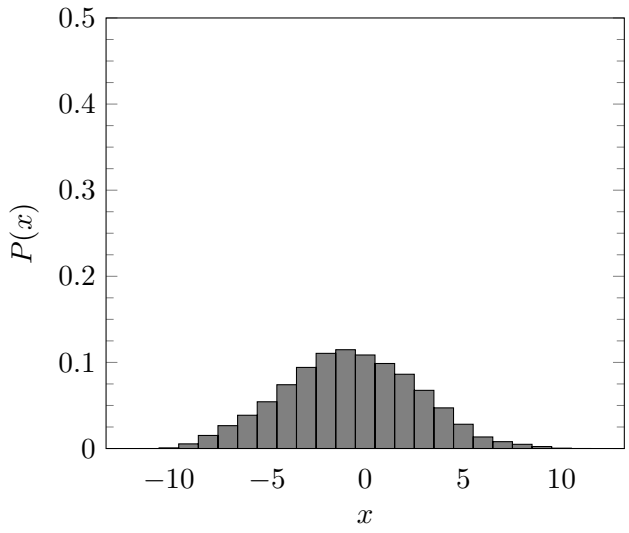
Method	Value	Width	Computing time (ms)
GEM	[-12,12]	24	14 405
GEMrow	[-1,1]	2	262
GEMcol	[-1,1]	2	247
GEMcom	[-1,1]	2	240
GSM5	[-1,1]	2	6521
GSM10	[-1,1]	2	14 277

The only viable results in this section were usually landed by the regular GEM method, due to no transformation of the matrices and therefore returning wider intervals than the other methods. However, we can see how was the resulting random variable distributed 5.4.

This histogram shows us, that even if this method returned wider intervals than interval arithmetic, the distribution of the random variable may be great indicator for the position of the real value. Due to operations over random variable, the resulting distribution is beginning to resemble normal distribution.

We also need to mention that with increasing number of operations with these random variables, there can also accumulate rounding error. This could potentially produce results too far from their expected real values. As seen in comparison of 5.4 and 5.2, this method produces several times grater intervals than regular interval arithmetic.

Figure 5.4: Probability function of x_1 in B (GEM)



5.3.3 Step Size Two

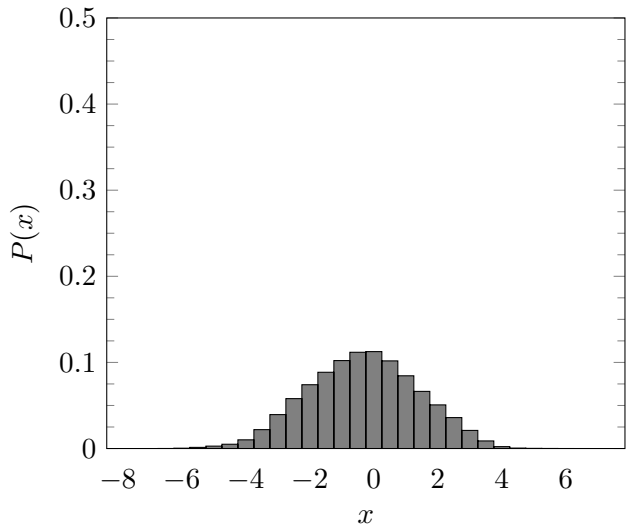
The results for the steps of size two, meaning that the width of sub-intervals was 0.5, were similar to the previous ones. In general, these random variables performed into better, more precise results, but this precision was still far from enough to provide relevant solutions of SLE in comparison with interval arithmetic.

Table 5.5: Element x_1 results for matrix B (Random variable, step two)

Method	Value	Width	Computing time (ms)
GEM	[-7.5,7.0]	14.5	20 676
GEMrow	[-0.5,0.5]	1	167
GEMcol	[-0.5,0.5]	1	248
GEMcom	[-0.5,0.5]	1	254
GSM5	[-0.5,0.5]	0.5	8013
GSM10	[-0.5,0.5]	0.5	18 446

The results for this section are in general very similar to the findings from previous section. The precision of these solutions is still not enough, with exception of regular GEM. We can compare results to the previous step size. These show improvement of the interval width, which seems to be a trend with increasing step size and therefore accuracy. The distribution of the random variable is again similar to the normal distribution 5.5.

Figure 5.5: Probability function of x_1 in B (GEM)



5.3.4 Step Size Four

We also applied the step size of four to the random variables. Again, this was not enough to provide very relevant information 5.6, with the exception of regular GEM. However, even if the results were proved to be too small, the computation time of these SLE was in general again more than doubled with every method. We will not analyze this step any further, due to the same general results as previous section.

Table 5.6: Element x_1 results for matrix B (Random variable, step four)

Method	Value	Width	Computing time (ms)
GEM	[-6,6]	12.0	68 549
GEMrow	[-0.25,0.25]	0.5	300
GEMcol	[-0.25,0.25]	0.5	480
GEMcom	[-0.25,0.25]	0.5	501
GSM5	[-0.25,0.25]	0.5	7367
GSM10	[-0.25,0.25]	0.5	14 676

5.3.5 Step Size Ten

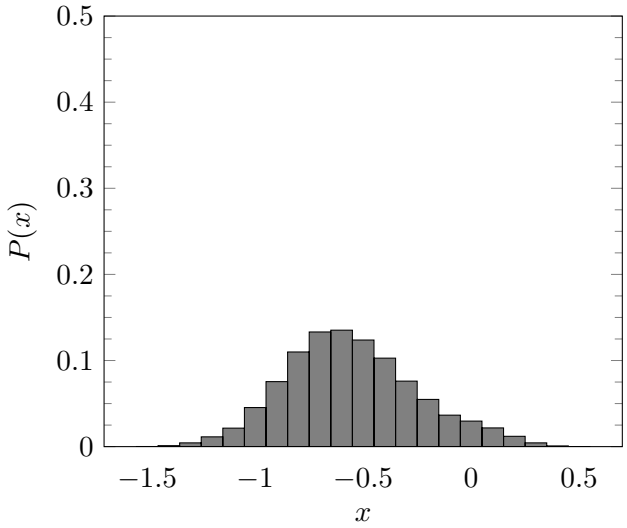
The results of this configuration made great improvement in information provided and can be considered usable for further exploration. We will use the matrix A and its element x_1 of RHS as an example, and compare it to the baseline solution 5.1. As in all previous sections we will use random variables with uniform distribution on the RHS.

Table 5.7: Element x_1 results for matrix A (Random variable, step ten)

Method	Value	Width	Computing time (ms)
GEM	[-1.6,0.6]	2.2	5874
GEMrow	[0.3,0.7]	0.4	115
GEMcol	[0.3,0.6]	0.3	464
GEMcom	[0.3,0.6]	0.3	487
GSM5	[0.2,0.7]	0.5	5772
GSM10	[0.2,0.7]	0.5	12 502

These values are comparable to the results of the interval arithmetic and we assume that it is the consequence of higher precision of computing. The rounding error of this precision allows for more reliable solutions for the linear equation systems than previous versions.

Figure 5.6: Probability function of x_1 in A (GEM)



As seen in the plots 5.6 5.7 5.8, the table 5.7 and the most test data, the regular GEM generates very wide intervals. This is strongly suppressed by using the pivoting of the matrix when using GEM. During our testing we also noticed some improvement of result precision when using complete pivoting but these observations were not seen in every case, therefore we cannot generalize this claim. Also if we compare the GSM with pivoting GEM, they generate very similar results, even with very small number of cycles. Generally we do not see any substantial improvement in precision by further increasing the cycles beyond five.

However, we need to acknowledge that our test data have been chosen with condition to be transformable to the diagonally dominant 1.1.6 form. This

Figure 5.7: Probability function of x_1 in A (GEMcom)

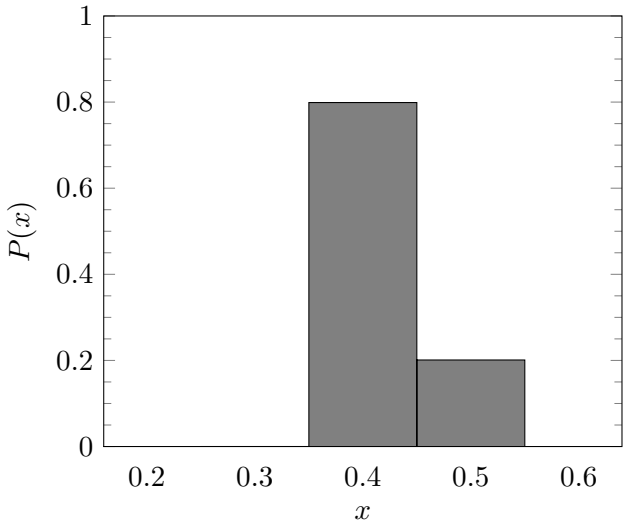
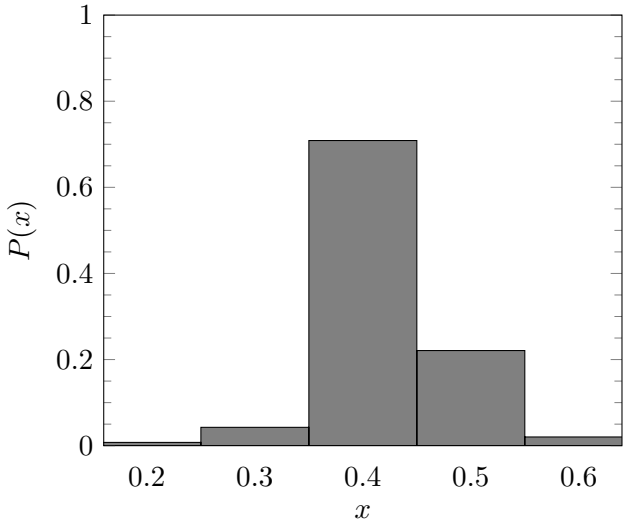


Figure 5.8: Probability function of x_1 in A (GS5)



allowed us to test these matrices against both methods, but could introduce some bias due to the nature of pivoting.

We also need to notice that this precision might still not be enough for the general use. We might be satisfied with the width of returned interval, where it can be just tenth of a unit, but we will not get any benefit from using the random variable. To highlight this effect, we can compare the results of the matrix C of random variables 5.8 with interval arithmetic 5.3, where the average width of result is around 0.1 and we can see a loss of accuracy.

Table 5.8: Element x_1 results for matrix C (Random variable, step ten)

Method	Value	Width	Computing time (ms)
GEM	[-7.2,6.8]	14.0	61 513
GEMrow	[-0.2,0.1]	0.3	2295
GEMcol	[-1.3,1.6]	2.9	6772
GEMcom	[-0.1,0.2]	0.3	1724
GSM5	[-0.2,0.3]	0.5	41 006
GSM10	[-0.2,0.3]	0.5	104 897

5.4 Performance

The performance varies based on the representation of random variable, the size of matrix and also the method used. The results were ambiguous, however we are able to deduce some general trends. We will comment on each of these aspects.

5.4.1 Methods

Using GEM, we can generally see major effect of any kind of pivoting applied to the matrix. Even if pivoting increases the number of operations over SLE, we can see improvement in speed by orders of magnitude. However the results show, that complete pivoting does not have generally same further effect on the speed of the computation, potentially even slowing it.

Comparing the solving methods between each other we can see that despite Gauss-Seidel method being better than regular Gaussian elimination method in computing time initially, the pivoting of matrices nullifies the advantage of this method and GEM with use of pivoting generally outperforms it significantly.

However, when comparing results of these methods, with the exception of regular GEM, they generally land very similar intervals and distributions. Therefore considering purely the numeric results, we can conclude that none of these methods is superior in this regard. This effect can bee also seen in 5.8 5.7.

5.4.2 Representation

Using the matrix A as an example, comparing the interval arithmetic 5.1 even with the random variable with step size of one 5.10, we can see great increase of computing time. This effect is further augmented by the raising precision of our random variable representation.

We can also note, that type of distribution, namely uniform and normal, appears to have the effect on the resulting performance, such as using normal

distribution as a default, we can see slight improvement of the computing speed. This effect can be seen in matrix A5.9 5.10.

Table 5.9: Performance (ms) for normal distribution, matrix A

Step	GEM	GEMrow	GEMcol	GEMcom	GSM5	GSM10
1	275	4	51	57	1246	2865
2	1079	7	110	107	1067	2419
4	2118	19	170	168	1514	3617
10	5033	98	388	381	4683	10 745

Table 5.10: Performance (ms) for uniform distribution, matrix A

Step	GEM	GEMrow	GEMcol	GEMcom	GSM5	GSM10
1	319	6	69	81	1560	3280
2	1306	11	155	166	1228	2910
4	2542	30	211	198	1733	4198
10	5874	115	464	487	5772	12 502

Conclusion

The goal of this thesis was the comparison of the two representations of inaccurate numbers used as the right hand side in systems of linear equations. For this purpose were used random variables that we compared with the interval arithmetic. As testing solving methods were used the Gaussian elimination method and the Gauss-Seidel iteration method.

We used several configurations for the random variables, ranging from the least accurate with relatively good computing speed, still being orders of magnitude higher than interval arithmetic, to the most accurate with gradually worsening performance. The general upside of the random variable was the improvement of the information about real position of value inside the interval, however this came at the cost of speed and efficiency, which decreased significantly with the increase in precision. In the end the downside of this representation is that it was very inefficient in comparison with interval arithmetic.

Both solving techniques were able to consistently solve the linear equations systems. The least accurate results were generated by the GEM with no pivoting, which was generally the slowest. This was very effectively suppressed by pivoting the rows and columns of the systems, often being orders of magnitude faster than original GEM and also much more accurate. The GSM method generated generally same results as the pivoting GEM, however the computing time is usually much slower.

Bibliography

[1] Kulle, M.: Použití intervalové aritmetiky v řešičích soustav lineárních rovnic. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[2] Kulle, M.: Pokročilé algoritmy pro řešení soustav lineárních intervalových rovnic. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[3] Hrabák, P.: Handbook to BIE-LIN. Praha: České vysoké učení technické v Praze, Fakulta informačních tehcnologií, Feb 2021.

[4] Horn, R. A.; Johnson, C. R.: *Matrix analysis*. Cambridge: Cambridge University Press, druhé vydání, 2013, ISBN 978-0-521-54823-6.

[5] Briggs, K.: "Diagonally Dominant Matrix." From MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein. Last visited on 5/4/2022. Dostupné z: `https://mathworld.wolfram.com/DiagonallyDominantMatrix.html`

[6] Moore, R. E.; Kearfott, R. B.; Cloud, M. J.: *Introduction to interval analysis*. SIAM, 2009.

[7] Hickey, T.; Ju, Q.; Van Emden, M. H.: Interval Arithmetic: From Principles to Implementation. *J. ACM*, ročník 48, č. 5, sep 2001: str. 1038–1068, ISSN 0004-5411, doi:10.1145/502102.502106. Dostupné z: `https://doi.org/10.1145/502102.502106`

[8] Dobronets, B.; Krantsevich, A.; Krantsevich, N.: Software Implementation of Numerical Operations on Random Variables. *Journal of Siberian Federal University - Mathematics and Physics*, ročník 2013, 01 2013: s. 168–173.

[9] Transformations of Random Variables. University of Alabama in Huntsville, Feb 2 2021, [Online; accessed 2022-04-20]. Dostupné z: https://stats.libretexts.org/Bookshelves/Probability_Theory/Probability_Mathematical_Statistics_and_Stochastic_Processes_(Siegrist)/03%3A_Distributions/3.07%3A_Transformations_of_Random_Variables

[10] Anděl, J.: *Základy matematické statistiky*. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, 2002.

[11] Blažek, R. B.; Kotecký, R.; Hrabáková, J.; aj.: Probability and Statistics Lecture 3 - Random variables I. [online presentation]. 2021. Dostupné z: https://courses.fit.cvut.cz/BIE-PST/media/lectures/BIE-PST-Lec03-Slides.pdf

[12] Knueppel, O.; Husung, D.; Keil, C.: PROFIL/BIAS (version 2.0. 8). 2009.

[13] Black, N.; Moore, S.: "Jacobi Method." From MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein. Last visited on 9/4/2022. Dostupné z: https://mathworld.wolfram.com/JacobiMethod.html

[14] Black, N.; Moore, S.: "Gauss-Seidel Method." From MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein. Last visited on 9/4/2022. Dostupné z: https://mathworld.wolfram.com/Gauss-SeidelMethod.html

[15] Dabrowski, R.; Kubica, B.: Comparison of Interval C/C++ Libraries in Global Optimization. 01 2009.

Appendix A

Acronyms

GEM Gaussian elimination method

GSM Gauss-Seidel method

LHS Left hand side

REF Row echelon form

RHS Right hand side

SLE System of linear equations

Appendix B

Contents of enclosed media

- `readme.txt` ..... the file with media contents description
- `Makefile` ..... makefile of the project
- `src` ..... the directory with the source codes
- `tests` ..... the directory containing all the testfiles
  - `intervals` ..... the directory containing all intervals
  - `matrices` ..... the directory containing all matrices
  - `outputs` ..... the directory containing all test outputs
- `thesis.pdf` ..... the thesis text in PDF format