THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MATHEMATICS

ACCURATE ARITHMETIC METHODS WITH MATRIX APPLICATIONS IN REAL FLOATING-POINT ARITHMETIC

NATHAN D. MURARIK
FALL 2024

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Mathematics
with honors in Mathematics

Reviewed and approved* by the following:

Dr. Thomas Robert Cameron
Professor of Mathematics
Thesis Supervisor

Dr. Daniel Galiffa
Professor of Mathematics
Honors Adviser

Dr. Derek Hanely
Professor of Mathematics
Faculty Reader

*Signatures are on file in the Schreyer Honors College.

Abstract

Compensated arithmetic is a summation technique designed to filter the error generated by a floating-point computation. The filtered error is then used to accurately estimate the computation's exact value. In this thesis, we use compensated arithmetic to construct accurate addition, multiplication, and division algorithms in real floating-point arithmetic. We demonstrate their accuracy by proving each algorithm's output is as accurate as if computed in k-fold precision and stored in k-parts. Moreover, we use the derived forward-error bounds to perform backward error-analysis on the dot product and on backward-substitution for upper-triangular systems. We further explore matrix applications by augmenting Gaussian Elimination with Partial Pivoting and iterative refinement with our k-parts arithmetic algorithms. We supplement our theoretical error bounds with numerical experiments on both ill-conditioned dot products and ill-conditioned linear systems.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to first acknowledge Dr. Thomas R. Cameron for his contribution to my thesis. Dr. Cameron actively supported my interest in numerical analysis throughout this project. He taught me the proof techniques, programming practices, and adaptability needed to produce lucrative numerical analysis research. Moreover, his compensated arithmetic GitHub repository is the building block for the programs used in the Chapter 5's numerical experiments. I want to extend my appreciation to the entire Penn State Behrend Mathematics department. This includes my Honors Advisor, Dr. Daniel Galiffa, who initially inspired me to pursue a career in mathematics research.

I also want to acknowledge the furry friends I met during my undergraduate studies. Our dogs at home are: Teddy, Penny, Dexter, Loki, Queen, Rambo, Scooter, Brownie, and Lilly. Bernie, our big orange cat, always took notes when I rambled to him about mathematics. Sally and our kitties, Pumpkin and Patch, remind me to find time for myself to relax. During my second year, I met both MooMoo and BunBun through my friend Gavin. I also made friends with Tori, Cicada, Ginger, and Milano during my time at Penn State Behrend. Finally, I am honored to be friends my with current partner's bunny Alice.

Finally, I want to thank my family for their encouragement over these past four years. They have wholeheartedly supported me during my time at Penn State DuBois and my studies at Penn State Behrend. My mother and father taught me the discipline needed to succeed in this project, but they also told me to have fun and relax. My sister Logan's passion for helping others inspires me to be the best person I can be for others as well.

This project was funded by the Pennsylvania State University 2024 Erickson Discovery Grant and the Penn State Behrend Undergraduate Research Grant.

Chapter 1

Introduction

1.1 Background

Consider the seemingly benign expression $1 + 10^{16} + 1 - 10^{16}$. On pen and paper we have $1+10^{16}+1-10^{16} = 2$, but a floating-point calculator states $1+10^{16}+1-10^{16} = 0$. The previous implies $0 = 2$, which is clearly a contradiction [1]. Numerical Analysis aims to study the effect of round-off error on algorithms and computational models derived from real-world phenomena. We develop our models with the drawback that most computational systems use floating-point numbers rather than the set of real numbers. Thus, floating-point arithmetic operations are subject to a degree of error that can be catastrophic for expressions with relatively small and/or large numbers such as $1 + 10^{16} + 1 - 10^{16}$.

Dekker, Gill, Golberg, Kahan, Knuth, and Møller [2, 3, 4, 5, 6] investigated floating-point error and discovered algorithms that account for the round-off error in floating-point arithmetic. Each of these algorithms were designed to make it unnecessary to upgrade hardware or use high-precision software, making error-free transformations both efficient and accessible. Ogita [7] expanded upon these algorithms to compute vector dot products with their VecSum, DotK, and SumK algorithms. Rump [8] also devised a summation routine, namely SumKK, that stored its output in multiple parts for increased accuracy. Moreover, Graillat [9] and Cameron [10, 11] applied Rump's error filtration technique to Horner's Method, the Ehrlich-Aberth method, and polynomial evaluation. The combined contribution of the aforementioned researchers has since been coined as compensated arithmetic.

1.2 Thesis Overview

In this thesis, we use Rump's error filtration strategy to develop addition, multiplication, and division algorithms that are as accurate as if computed in k-fold precision stored in k-parts. We first propagate an error vector with Kahan and Nievergelt's arithmetic algorithms [4, 12], which are described in Chapter 2. We, then, successively sum over the entries of the error-vector and track the result of each sum until we reach our desired precision.

We use the remainder of this chapter to provide a list of common notation used throughout this thesis. Chapter 2 will explain the particular algorithms developed by Kahan, Nievergelt, Ogita, and Rump [4, 7, 8, 12] with their associated theorems. We also describe how to designate a computation whose result is as accurate as if computed in l-fold precision and stored in k-parts. We take Chapter 3 to develop a k-parts addition, multiplication, fuse-multiply-add, and division algorithm. Chapter 4 delves into perturbation theory with k-parts computations. We use our perturbation results to derive the forward and backward errors of the dot product and the substitution methods used to solve triangular systems. We elaborate on our findings in Chapter 4 by conducting numerical experiments in Chapter 5. To conclude, we discuss our results and future work in Chapter 6.

1.3 Common Notation

We first establish the use of notation in this paper before discussing our results. This thesis works with data structures we refer to as floating-point collections. A floating-point collection is a list of floating point numbers. Table 1.1 outlines how we will designate both floating-point collections and floating-point numbers. The curly-brace notation was coined by Rump [8] in his paper on the inversion of ill-conditioned matrices, so we opt to use the same notation for the sake of consistency. Furthermore, we provide a working-notion of Big-O notation, which we frequently use in Chapters 4 and 5.

Notation	Description
$\{r\} = \{r_1, r_2, r_3, \ldots, r_k\}$	A floating-point collection of size k.
r_m or $\{r\}_m$	Denotes the mth entry in a floating-point collection. We opt for r_m if the floating-point collection is a scalar and use $\{r\}_m$ if the collection is an entry of a vector or matrix. For instance, $\{u_{ij}\}_m$.
$\hat{r}$	A floating-point number representing a computed result.
$\{r^{(i)}\}$ or $\hat{r}^{(i)}$	A floating-point number or collection computed on the ith iteration of an iterative process.
$\{\boldsymbol{r}\}$	A vector whose components are floating-point collections.
$\{R\}$	A matrix whose components are floating-point collections.
$\{r\} = \sum_{i=1}^{k} r_i$	Indicates how a floating-point collection is used as an operand.
$\alpha = O(\beta)$	Denotes $\alpha = \rho\beta$ for some small constant ρ. We will also say "α is on the order of β."
cond(c)	The condition number of operation c.

Table 1.1: Common Notation

Chapter 2

Preliminaries

2.1 Real Floating-Point Arithmetic

In this thesis, we assume floating-point arithmetic follows the IEEE 754 standard [13] with neither underflow nor overflow. Denote $\mathscr{F}$ as a set of floating-point numbers and μ as the working precision unit round-off. The unit round-off is $\mu = 2^{-53}$ in double precision and $\mu = 2^{-24}$ for single precision. Let $a, b \in \mathscr{F}$ and $\circ \in \{+, -, \cdot, /, \sqrt{}\}$. The IEEE 754 standard requires the result of $fl(a \circ b)$ to be rounded correctly; that is, to be as accurate as if computed exactly and then rounded into the working precision. Every operation must be performed with rounding to nearest, using round to even in case of a tie.

The aforementioned rounding-scheme implies that

$$fl(a \circ b) = (a \circ b)(1 + \epsilon)$$

where $|\epsilon| \leq \mu$. Furthermore,

$$|fl(a \circ b) - (a \circ b)| \leq \mu |fl(a \circ b)| \text{ and } |fl(a \circ b) - (a \circ b)| \leq \mu |a \circ b|.$$

Suppose c is an exact operation and $\overline{c}$ is the same operation with its operands replaced by their absolute value. The following definition describes a floating-point result as accurate as if computed in k-fold precision, a higher precision, and rounded into the working precision.

Definition 2.1.1 (k-fold precision)**.** We denote $r = \mathrm{fl}_{k,1}(c)$ to be a floating-point computation as accurate as if computed in k-fold precision and then rounded into the working precision; that is,

$$|r - c| \leq \rho_1 \mu |c| + \rho_2 \mu^k \overline{c},$$

where ρ_1 and ρ_2 are reasonably small positive values relative to μ^{-k}.

The next definition describes when a floating-point result, with operation c, is computed in l-fold precision and stored in k-parts for $0 < l \leq k$. Our goal, here, is to demonstrate the k-parts arithmetic operations satisfy the following definition for $l = k$, where some intermediate results may be stored in l-fold precision.

Definition 2.1.2 (l-fold precision, k-parts)**.** We denote $\{r\} = \{r_1, \ldots, r_k\} = \mathrm{fl}_{l,k}(c)$ to be a floating-point computation as accurate as if computed in l-fold precision and stored in k-parts; that is,

$$\left| \sum_{i=1}^{k} r_i - c \right| \leq \rho_3 \mu^l \overline{c}$$

and

$$\sum_{i=1}^{k} |r_i| \leq \hat{\rho}_3 \overline{c},$$

where ρ_3 and $\hat{\rho}_3$ are reasonably small positive values relative to μ^{-k}.

2.2 Error Free Transformations

Let $a, b \in \mathscr{F}$ and $\circ \in \{+, -, \cdot\}$. Then for any floating-point result $x = fl(a \circ b)$, there exists a $y \in \mathscr{F}$ so that $x + y = a + b$. The pair, (x, y), is called an Error Free Transformation (EFT). The first arithmetic operation that satisfies the definition of an EFT was made by Knuth in 1968 [5] and is shown as Algorithm (1).

Algorithm 1: Error Free Transformation for Addition where $(a, b) \in \mathscr{F}^2$
[5, Theorem B p.236]

```
1 procedure [x, y] = TwoSum(a, b)
2 |   x = fl(a + b)
3 |   z = fl(x − a)
4 |   y = fl(a − (x − z) + (b − z))
5 end
```

Algorithm (2), a multiplication EFT, was developed by Nievergelt in 2003. The Fuse-Multiply Add (FMA) operation, as used in Algorithm (2), determines the closest floating-point number to $a \cdot b + c \in \mathbb{R}$ for $a, b, c \in \mathscr{F}$. We assume FMA uses only one floating-point operation as was done in [9, 7, 11]. Thus, our assumption implies Algorithm (2) only requires two floating-point operations instead of the seventeen required by multiplication EFTs with the split routine.

Algorithm 2: Error Free Transformation for Multiplication where $(a, b) \in \mathscr{F}^2$
([12, Thrm 2])

```
1 procedure [x, y] = TwoProd(a, b)
2 |   x = fl(a · b)
3 |   y = FMA(a, b, −x)
4 end
```

Theorem 2.2.1 states that Algorithms (1) and (2) produce an Error Free Transformation. The theorem also outlines a few properties of the absolute error-bound in each operation.

Theorem 2.2.1 ([7, Thrm 3.4])**.** *Let $a, b \in \mathscr{F}$. Then, $[x, y] = TwoSum(a, b)$ requires 6 floating-point operations and satisfies*

$$a + b = x + y, \quad x = fl(a + b), \quad |y| \leq \mu|x|, \quad |y| \leq \mu|a + b|.$$

Additionally, $[x, y] = TwoProd(a, b)$ requires two floating-point operations and satisfies

$$a \cdot b = x + y, \quad x = fl(a \cdot b), \quad |y| \leq \mu|x|, \quad |y| \leq \mu|a \cdot b|.$$

2.3 Vector Summations

We now outline Ogita's [7] and Rump's [8] findings from their papers on an accurate dot product and inversion of ill-conditioned matrices. We begin our discussion by defining [14]

$$\gamma_n := \frac{n\mu}{1-n\mu}$$

where $n \in \mathbb{N}$ satisfies $n\mu < 1$. We also use the floating-point summation error-bound [14]

$$\left| fl\left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} x_i \right| \leq \gamma_{n-1} \sum_{i=1}^{n} |x_i| \tag{2.1}$$

so that

$$\frac{|fl\left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} x_i|}{|\sum_{i=1}^{n} x_i|} \leq \gamma_{n-1}\text{cond}\left(\sum_{i=1}^{n} x_i\right), \quad \text{cond}\left(\sum_{i=1}^{n} x_i\right) = \frac{\sum_{i=1}^{n} |x_i|}{|\sum_{i=1}^{n} x_i|}.$$

Moreover, we use the floating-point dot product error bound [14]

$$\left|fl(\boldsymbol{x}^T\boldsymbol{y}) - (\boldsymbol{x}^T\boldsymbol{y})\right| \leq \gamma_n \left|\boldsymbol{x}^T\right| |\boldsymbol{y}| \tag{2.2}$$

so that

$$\frac{\left|fl(\boldsymbol{x}^T\boldsymbol{y}) - \boldsymbol{x}^T\boldsymbol{y}\right|}{|\boldsymbol{x}^T\boldsymbol{y}|} \leq \gamma_n \text{cond}(\boldsymbol{x},\boldsymbol{y}), \quad \text{cond}(\boldsymbol{x},\boldsymbol{y}) = \frac{\left|\boldsymbol{x}^T\right| |\boldsymbol{y}|}{|\boldsymbol{x}^T\boldsymbol{y}|}.$$

Algorithm (3) describes a vector transformation that maintains the vector entry-wise sum of the original vector. We keep the interpretation of Algorithm (3) provided in [7, 11] for clarity. The important properties of Algorithm (3) are summarized as Theorem 2.3.1.

Algorithm 3: Vector Transform of $\boldsymbol{q} \in \mathscr{F}^n$ without changing its vector sum
[7, Alg 4.3]

```
procedure p = VecSum(q)
    p_1 = q_1
    for i = 2 to n do
        [p_i, p_{i-1}] = TwoSum(q_i, p_{i-1})
    end
end
```

Theorem 2.3.1 ([7, Lem 4.2])**.** *Let $\boldsymbol{q} \in \mathscr{F}^n$. Then, $\boldsymbol{p} = VecSum(\boldsymbol{q})$ requires $6(n-1)$ floating-point operations and satisfies*

$$\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i, \text{ and } p_n = fl\left(\sum_{i=1}^{n} q_i\right).$$

Additionally,

$$\sum_{i=1}^{n-1} |p_i| \leq \gamma_{n-1} \sum_{i=1}^{n} |q_i|$$

.

Algorithm (4) computes a vector sum as accurate as if computed in k-fold precision and then rounded into the working precision. Theorem 2.3.2 confirms $s = fl_{k,1}(\sum_{i=1}^{n} p_i)$ because $3\gamma_{n-1}^2$ is negligible compared to μ and γ_{2n-2}^k is a multiple of μ^k.

Algorithm 4: Vector Sum of $\boldsymbol{p} \in \mathscr{F}^n$ in k-fold precision and then rounded into the working precision ([7, Alg 4.8])

procedure *s = SumK(**p**, k)*
  for $i = 1$ *to* $k - 1$ **do**
    $\boldsymbol{p} = VecSum(\boldsymbol{p})$
  end
  $s = fl(p_n + \sum_{i=1}^{n-1} p_i)$
end

Theorem 2.3.2 ([7, Prop 4.5 and Prop 4.10])**.** *Let* $\boldsymbol{p} \in \mathscr{F}^n$, $4n\mu < 1$, *and* $k > 1$. *Then,* $s = SumK(\boldsymbol{p}, k)$ *requires (n - 1)(6k - 5) floating-point operations and satisfies*

$$\left| s - \sum_{i=1}^{n} p_i \right| \leq (\mu + 3\gamma_{n-1}^2) \left| \sum_{i=1}^{n} p_i \right| + \gamma_{2n-2}^k \sum_{i=1}^{n} |p_i|.$$

Additionally, we have the relative error-bound

$$\frac{|s - \sum_{i=1}^{n} p_i|}{|\sum_{i=1}^{n} p_i|} \leq \mu + \gamma_{2n-2}^k \operatorname{cond}\left(\sum_{i=1}^{n} p_i \right) + O(\mu^2).$$

Rump extended Algorithm (4) to have $\{s\} = fl_{k,k}(\sum_{i=1}^{n} p_i)$. We conclude this chapter by stating both Algorithm (5) and Theorem 2.3.3.

Algorithm 5: Vector Sum of $\boldsymbol{p} \in \mathscr{F}^n$ in k-fold precision and stored in k-parts [8, Alg 2.3]

procedure $\{s\}$ *= SumKK(**p**$^{(0)}$, k)*
  for $i = 0$ *to* $k - 2$ **do**
    $\boldsymbol{p}^{(i+1)} = VecSum(\boldsymbol{p}^{(i)}_{1,\ldots,n-i})$
    $s_{i+1} = p^{(i+1)}_{n-i}$
  end
  $s_k = fl(\sum_{i=1}^{n-k+1} p_i^{(k-1)})$
end

Theorem 2.3.3 ([8, Theorem 2.4]). *Let* $\boldsymbol{p} \in \mathscr{F}^n$, $(n-1)\mu < 1$, *and* $k > 1$. *Then,* $\{s\} = SumKK(\boldsymbol{p}, k)$ *satisfies*

$$\left| \{s\} - \sum_{i=1}^{n} p_i \right| \leq \gamma_{n-1}^{k} \sum_{i=1}^{n} |p_i|.$$

Additionally, we have the relative error-bound

$$\frac{|\{s\} - \sum_{i=1}^{n} p_i|}{|\sum_{i=1}^{n} p_i|} \leq \gamma_{n-1}^{k} \operatorname{cond} \left(\sum_{i=1}^{n} p_i \right).$$

Chapter 3

Arithmetic in K-Parts

We designate $\{q\}, \{r\}, \{s\} \in \mathscr{F}^k$ to be floating-point collections of size k. We will specify when $\{q\}$, $\{r\}$, and $\{s\}$ satisfy Definition 2.1.2, for $l = k$, as needed in this chapter.

3.1 K-Parts Addition

The first operation we consider in our k-parts arithmetic analysis is addition. We begin Algorithm (6) by storing the entries of $\{r\}$ and $\{s\}$ in vector $\boldsymbol{e} \in \mathscr{F}^{2k}$. We, then, apply Rump's SumKK algorithm, Algorithm (5), and store the result in $\{\mathrm{t}\}$.

Algorithm 6: Addition in K-Parts

procedure $\{t\} = SumParts(\{r\}, \{s\}, k)$
  for $i = 1$ *to* k **do**
    $e_i = r_i$ ▷ Store the entries of $\{r\}$ and $\{s\}$ into $\boldsymbol{e}$
    $e_{k+i} = s_i$
  end
  $\boldsymbol{e}^{(0)} = \boldsymbol{e}$ ▷ Apply SumKK to filter $\boldsymbol{e}$'s enteries
  for $i = 0$ *to* $k - 2$ **do**
    $\boldsymbol{e}^{(i+1)} = VecSum(\boldsymbol{e}^{(i)})$
    $t_{i+1} = e_{2k-i}^{(i+1)}$
    Delete $e_{2k-i}^{(i+1)}$ from $\boldsymbol{e}^{(i+1)}$
  end
  $t_k = \mathrm{fl}\left(\sum_{j=1}^{k+1} e_j^{(k-1)}\right)$
end

Our result is as accurate as if computed in k-fold precision and stored in k-parts. We confirm this by showing $\{t\} = \mathrm{fl}_{k,k}\left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i\right)$ with Lemma 3.1.1.

Lemma 3.1.1. *Suppose $\{r\} \in \mathscr{F}^k$ and $\{s\} \in \mathscr{F}^k$ are arbitrary floating-point collections. Then, provided $k > 1$ and $(2k-1)\mu < 1$, the floating-point collection $\{t\} = SumParts(\{r\}, \{s\}, k)$ satisfies*

$$\left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i\right)\right| \leq \gamma_{2k-1}^{k}\left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right).$$

Moreover,

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^2 + \cdots + \gamma_{2k-1}^{k})\left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right).$$

It follows that $\{t\} = \mathrm{fl}_{k,k}\left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i\right)$ since γ_{2k-1}^{k} is a multiple of μ^k

Proof. We demonstrate this lemma with a direct approach. The output $\{t\} = SumParts(\{r\}, \{s\}, k)$ is equivalent to Rump's output of $SumKK(\boldsymbol{e}, k)$, see Algorithm (5), where $\boldsymbol{e} \in \mathscr{F}^{2k}$ is con-

structed during SumParts. Therefore, we have [8, Theorem 2.4]

$$\left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i\right)\right| \leq \gamma_{2k-1}^{k} \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right). \tag{3.1}$$

We now determine an upper-bound for $\sum_{i=1}^{k} |t_i|$. For $i = 1, 2, \ldots, k$, the absolute error-bound for to each t_i is a floating-point summation error-bound, see (2.1), of the form

$$\left|t_i - \sum_{j=1}^{2k-i+1} e_j\right| \leq \gamma_{2k-i} \sum_{j=1}^{2k-i+1} |e_j|. \tag{3.2}$$

An application of the triangle inequality changes (3.2) to be

$$|t_i| \leq (1 + \gamma_{2k-i}) \sum_{j=1}^{2k-i+1} |e_j|. \tag{3.3}$$

In particular, the inequality in (3.2) for $i = 1$ is

$$\left|t_1 - \sum_{j=1}^{2k} e_j\right| \leq \gamma_{2k-1} \sum_{j=1}^{2k} |e_j| \implies \left|t_1 - \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i\right)\right| \leq \gamma_{2k-1} \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right)$$

as implied by lines 3 through 6 of SumParts. The triangle inequality changes the previous to

$$|t_1| \leq (1 + \gamma_{2k-1}) \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right). \tag{3.4}$$

The k-parts of $\{t\}$ are generated by lines 8 through 13 with VecSum, where the last entry of the VecSum result is stored. The previous let us use Theorem 2.3.1 with $e^{(i)}$ for $i = 2, \ldots, k$ to yield

$$\sum_{j=1}^{2k-i+1} \left|e_j^{(i)}\right| \leq \gamma_{2k-i+1} \sum_{j=1}^{2k-i+2} \left|e_j^{(i-1)}\right| \leq \cdots \leq \left(\prod_{v=2k-i+1}^{2k-2} \gamma_v\right) \sum_{j=1}^{2k-1} \left|e_j^{(1)}\right| \leq \gamma_{2k-2}^{i-2} \sum_{j=1}^{2k-1} \left|e_j^{(1)}\right|.$$

Theorem 2.3.1 further simplifies the previous inequality to

$$\sum_{j=1}^{2k-i+1} \left|e_j^{(i)}\right| \leq \gamma_{2k-2}^{i-2} \gamma_{2k-1} \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right) \leq \gamma_{2k-1}^{i-1} \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right). \tag{3.5}$$

We now use inequalities (3.3) and (3.5) to get

$$\begin{aligned} |t_i| \leq (1 + \gamma_{2k-i}) \sum_{j=1}^{2k-i+1} |e_j^{(i)}| &\leq (1 + \gamma_{2k-i}) \gamma_{2k-1}^{i-1} \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right) \\ &\leq (\gamma_{2k-1}^{i-1} + \gamma_{2k-1}^{i}) \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i|\right) \end{aligned}$$

and adding inequality (3.4) with the previous inequality for $i = 2, 3, \ldots, k$ yields

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^2 + \cdots + \gamma_{2k-1}^k) \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i| \right)$$

as desired. ■

We now demonstrate $\{t\} = \text{fl}_{k,k}(c_1 + c_2)$ when $\{r\} = \text{fl}_{k,k}(c_1)$ and $\{s\} = \text{fl}_{k,k}(c_2)$.

Theorem 3.1.2. *Suppose* $\{r\} = \text{fl}_{k,k}(c_1)$ *and* $\{s\} = \text{fl}_{k,k}(c_2)$ *with constants* $\rho_1, \hat{\rho}_1$ *for* $\{r\}$ *and* $\rho_2, \hat{\rho}_2$ *for* $\{s\}$. *Then, provided that* $k > 1$ *and* $(2k-1)\mu < 1$, *the floating-point collection* $\{t\} = SumParts(\{r\}, \{s\}, k)$ *requires* $9k^2 - 8k + 1$ *floating-point operations and satisfies*

$$\left| \sum_{i=1}^{k} t_i - (c_1 + c_2) \right| \leq \gamma_{2k-1}^k max\left(\{\rho_1 + \hat{\rho}_1, \rho_2 + \hat{\rho}_2\}\right)(\overline{c}_1 + \overline{c}_2).$$

Additionally,

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^2 + \cdots + \gamma_{2k-1}^k) max(\{\hat{\rho}_1, \hat{\rho}_2\})(\overline{c}_1 + \overline{c}_2).$$

It follows that $\{t\} = fl_{k,k}(c_1 + c_2)$ *since* γ_{2k-1}^k *is a multiple of* μ^k.

Proof. We demonstrate this result using a direct approach, but leave the floating-point operation count to the reader. The triangle inequality implies

$$\begin{aligned}
\left| \sum_{i=1}^{k} t_i - (c_1 + c_2) \right| &= \left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i \right) + \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i \right) - (c_1 + c_2) \right| \\
&\leq \left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i \right) \right| + \left| \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i \right) - (c_1 + c_2) \right| \\
&\leq \left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i + \sum_{i=1}^{k} s_i \right) \right| + \left| \sum_{i=1}^{k} r_i - c_1 \right| + \left| \sum_{i=1}^{k} s_i - c_2 \right|
\end{aligned}$$

Lemma 3.1.1 provides us with

$$\left| \sum_{i=1}^{k} t_i - (c_1 + c_2) \right| \leq \gamma_{2k-1}^k \left(\sum_{i=1}^{k} |r_i| + \sum_{i=1}^{k} |s_i| \right) + \left| \sum_{i=1}^{k} r_i - c_1 \right| + \left| \sum_{i=1}^{k} s_i - c_2 \right|.$$

We now can apply Definition 2.1.2 with $l = k$, where γ_{2k-1}^k is a multiple of μ^k, to find

$$\begin{aligned}
\left| \sum_{i=1}^{k} t_i - (c_1 + c_2) \right| &\leq \gamma_{2k-1}^k (\hat{\rho}_1 \overline{c}_1 + \hat{\rho}_2 \overline{c}_2) + \rho_1 \mu^k \overline{c}_1 + \rho_2 \mu^k \overline{c}_2 \\
&\leq \gamma_{2k-1}^k ((\rho_1 + \hat{\rho}_1)\overline{c}_1 + ((\rho_2 + \hat{\rho}_2)\overline{c}_2) \\
&\leq \gamma_{2k-1}^k max\left(\{\rho_1 + \hat{\rho}_1, \rho_2 + \hat{\rho}_2\}\right)(\overline{c}_1 + \overline{c}_2).
\end{aligned}$$

as desired. The additional inequality

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^2 + \cdots + \gamma_{2k-1}^k) max(\{\hat{\rho}_1, \hat{\rho}_2\})(\overline{c}_1 + \overline{c}_2)$$

is a consequence of Lemma 3.1.1 and Definition 2.1.2 for $\{r\}$ and $\{s\}$. ■

3.2 K-Parts Product

Algorithm (7) computes the multiplication of two-floating point collections by distributing each entry of $\{r\}$ into $\{s\}$. We add the result of each product together to yield $t_1 = \text{fl}\left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)$. The vector $\boldsymbol{e} \in \mathscr{F}^{2k^2}$ stores the error in each sum and product. We, then, apply Algorithm (5) to filter $\boldsymbol{e}$ and produce the output $\{t\}$.

Algorithm 7: Multiplication Stored in K-Parts

procedure $\boldsymbol{t} = ProdParts(\{r\}, \{s\}, k)$
  $p = 0$
  for $i = 1$ *to* k **do**
    for $j = 1$ *to* k **do**
      $[q, e_{(i-1)k+j}] = TwoProd(r_i, s_j)$ ▷ Distribute each r_i into $\{s\}$
      $[p, e_{k^2+(i-1)k+j}] = TwoSum(p, q)$ ▷ Store the error of each operation in $\boldsymbol{e}$
    end
  end
  $t_1 = p$ ▷ $t_1 = \text{fl}(\{r\} \cdot \{s\})$
  $\boldsymbol{e}^{(0)} = \boldsymbol{e}$ ▷ Apply SumKK to filter the error stored in $\boldsymbol{e}$
  for $i = 0$ *to* $k - 3$ **do**
    $\boldsymbol{e}^{(i+1)} = VecSum(\boldsymbol{e}^{(i)})$
    $t_{i+2} = e_{2k^2-i}^{(i+1)}$
    Delete $e_{2k^2-i}^{(i+1)}$ from $\boldsymbol{e}^{(i+1)}$
  end
  $t_k = fl\left(\sum_{j=1}^{2k^2-k+2} e_j^{(k-2)}\right)$
end

We first determine an upper-bound for the absolute vector sum of $\boldsymbol{e}$ from Algorithm (7). Our lemma below is reminiscent of Yamanaka et al's proof for DotK [15, See Theorem 2.10], but we write our own proof for completeness.

Lemma 3.2.1. *Suppose $\{r\} \in \mathscr{F}^k$ and $\{s\} \in \mathscr{F}^k$ are arbitrary floating-point collections. The vector $\boldsymbol{e}$ in ProdParts satisfies*

$$\sum_{r=1}^{2k^2} |e_r| \leq \gamma_{k^2} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i|\right).$$

Proof. We proceed in a direct approach. We first find a bound for the sum of the absolute errors ranging from $i = 1, 2, \ldots, k^2$. Line 5 uses TwoProd, so Theorem 2.2.1 implies for $i, j = 1, 2, \ldots, k$ that $|e_{(i-1)k+k}| \leq \mu |r_i \cdot s_j|$. The sum of each of these errors is

$$\sum_{i=1}^{k}\sum_{j=1}^{k} |e_{(i-1)k+i}| = \sum_{r=1}^{k^2} |e_r| \leq \mu \sum_{i=1}^{k}\sum_{j=1}^{k} |r_i \cdot s_j| . \tag{3.6}$$

We now determine a bound for the sum of the absolute errors ranging from $r = k^2 + 1, \ldots 2k^2$. Consider the vector $\boldsymbol{q} = [\text{fl}(r_1 \cdot s_1), \text{fl}(r_1 \cdot s_2), \ldots, \text{fl}(r_2 \cdot s_1), \text{fl}(r_2 \cdot s_2), \ldots, \text{fl}(r_k \cdot s_k)]^T$ where entry q_i is calculated on Line 5 of ProdParts. Line 6 of ProdParts, within the loop structure, produces a result as if we calculated $\boldsymbol{p} = VecSum(\boldsymbol{q})$ where the sum of $e_{k^2+1}, \ldots, e_{2k^2}$ is equal to the sum of the first n - 1 entries in $\boldsymbol{p}$. Thus, by Theorem 2.3.1 we have

$$\sum_{i=1}^{n-1} |p_i| = \sum_{r=k^2+1}^{2k^2} |e_r| \leq \gamma_{k^2-1} \sum_{i=1}^{k^2} |q_i| = \gamma_{k^2-1} \sum_{i=1}^{k}\sum_{j=1}^{k} |\text{fl}(r_i \cdot s_j)| . \tag{3.7}$$

The IEEE 754 specification $\text{fl}(a \circ b) = (a \circ b)(1 + \epsilon)$ for $|\epsilon| \leq \mu$ implies (3.7) becomes

$$\sum_{r=k^2+1}^{2k^2} |e_r| \leq \gamma_{k^2-1}(1 + \mu) \sum_{i=1}^{k}\sum_{j=1}^{k} |r_i \cdot s_j| . \tag{3.8}$$

Therefore, we can sum $\sum_{r=1}^{k^2} |e_r|$ and $\sum_{r=k^2+1}^{2k^2} |e_r|$ with (3.6) and (3.8) to yield

$$\begin{aligned}\sum_{r=1}^{k^2} |e_r| + \sum_{r=k^2+1}^{2k^2} |e_r| &\leq \mu \sum_{i=1}^{k}\sum_{j=1}^{k} |r_i \cdot s_j| + \gamma_{k^2-1}(1 + \mu) \sum_{i=1}^{k}\sum_{j=1}^{k} |r_i \cdot s_j| \\ &= (\gamma_{k^2-1}(1 + \mu) + \mu) \sum_{i=1}^{k}\sum_{j=1}^{k} |r_i \cdot s_j| .\end{aligned}$$

Since $\gamma_{k^2-1}(1 + \mu) + \mu \leq \gamma_{k^2}$, we conclude [14, Lemma 3.3]

$$\sum_{r=1}^{2k^2} |e_r| \leq \gamma_{k^2} \sum_{i=1}^{k}\sum_{j=1}^{k} |r_i \cdot s_j| = \gamma_{k^2} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right)$$

as desired. ■

We now continue with an analysis similar to the K-Parts Addition section. We do so by first showing that $\{t\} = \text{fl}_{k,k}\left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)$.

Lemma 3.2.2. *Suppose $\{r\} \in \mathscr{F}^k$ and $\{s\} \in \mathscr{F}^k$ are arbitrary floating-point collections. Provided that $k > 1$ and $(2k^2-1)\mu < 1$, the floating-point collection $\{t\} = ProdParts(\{r\}, \{s\}, k)$ satisfies*

$$\left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) \right| \leq \gamma_{2k^2-1}^{k} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right).$$

Moreover,

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k^2-1} + \cdots + 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k}) \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right).$$

It follows that $\{t\} = \mathrm{fl}_{k,k}\left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right)$ since $\gamma_{2k^2-1}^{k}$ is a multiple of μ^k.

Proof. We proceed in a direct approach, but leave the floating-point operation count to the reader. Repeated application of Theorem 2.2.1 implies lines 2 through 9 of ProdParts provides us with

$$\left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) = t_1 + \sum_{r=1}^{2k^2} e_r. \tag{3.9}$$

By Theorem 2.3.1, lines 10 through 16 imply that

$$\sum_{r=1}^{2k^2} e_r = \sum_{i=2}^{k-1} t_i + \sum_{r=1}^{2k^2-k+2} e_r^{(k-2)} \tag{3.10}$$

where VecSum for $k > 1$ yields

$$\begin{aligned} \sum_{r=1}^{2k^2-k+2} \left| e_r^{(k-2)} \right| \leq \gamma_{2k^2-k+2} \sum_{r=1}^{2k^2-k+3} \left| e_r^{(k-3)} \right| \leq \ldots \leq \left(\prod_{i=2k^2-k+2}^{2k^2-1} \gamma_i \right) \sum_{r=1}^{2k^2} |e_r| \\ \leq \gamma_{2k^2-1}^{k-2} \sum_{r=1}^{2k^2} |e_r|. \end{aligned} \tag{3.11}$$

Substituting (3.10) into (3.9) yields

$$\left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) = \sum_{i=1}^{k-1} t_i + \sum_{r=1}^{2k^2-k+2} e_r^{(k-2)} \tag{3.12}$$

The floating-point summation error-bound (2.1) implies our algorithm's remaining error satisfies

$$\left| t_k - \sum_{r=1}^{2k^2-k+2} e_r^{(k-2)} \right| \leq \gamma_{2k^2-k+1} \sum_{r=1}^{2k^2-k+2} |e_r^{(k-2)}|. \tag{3.13}$$

Thus, we start with the left-hand side of the inequality as stated in the lemma and use (3.12) to yield

$$\left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i \right) \left(\sum_{i=1}^{k} s_i \right) \right| = \left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k-1} t_i + \sum_{r=1}^{2k^2-k+2} e_r^{(k-2)} \right) \right| = \left| t_k - \sum_{r=1}^{2k^2-k+2} e_r^{(k-2)} \right|.$$

Equations (3.11) and (3.13) imply

$$\left| t_k - \sum_{r=1}^{2k^2-k+2} e_r^{(k-2)} \right| \leq \gamma_{2k^2-k+1} \sum_{r=1}^{2k^2-k+2} |e_r^{(k-2)}| \leq \gamma_{2k^2-k+1}\gamma_{2k^2-1}^{k-2} \sum_{r=1}^{2k^2} |e_r| .$$

We now invoke Lemma (3.2.1) and our assumption that $k > 1$ to find

$$\begin{aligned}\gamma_{2k^2-k+1}\gamma_{2k^2-1}^{k-2} \sum_{r=1}^{2k^2} |e_r| &\leq \gamma_{2k^2-k+1}\gamma_{2k^2-1}^{k-2}\gamma_{k^2} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right) \\ &\leq \gamma_{2k^2-1}^{k} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right) .\end{aligned}$$

Therefore, we have

$$\left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) \right| \leq \gamma_{2k^2-1}^{k} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right) .$$

To determine an absolute upper-bound for $\sum_{i=1}^{k} |t_i|$, consider the vectors

$$\begin{aligned}\boldsymbol{r} &= \left[r_1, r_2, \ldots, r_k, r_1, r_2, \ldots, r_k, r_1, r_2, \ldots, r_k, \ldots, r_k \right]^T \\ \boldsymbol{s} &= \left[s_1, s_1, \ldots, s_1, s_2, s_2, \ldots, s_2, s_3, s_3, \ldots, s_3, \ldots, s_k \right]^T\end{aligned}$$

in which $\boldsymbol{r}, \boldsymbol{s} \in \mathscr{F}^{k^2}$. We can use the floating-point dot product bound, see (2.2), to find

$$\begin{aligned}\left| \text{fl}\left(\boldsymbol{r}^T \boldsymbol{s} \right) - \boldsymbol{r}^T \boldsymbol{s} \right| &= \left| t_1 - \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) \right| \\ &\leq \gamma_{k^2} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right) \leq \gamma_{2k^2-1} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right) .\end{aligned}$$

Using the triangle equality implies the previous becomes

$$|t_1| \leq (1 + \gamma_{2k^2-1}) \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right) . \tag{3.14}$$

For $i = 2, 3, \ldots, k$, we can use the floating-point summation upper-bound, see (2.1), to get

$$\left| t_i - \sum_{j=1}^{2k^2-i+2} e_j \right| \leq \gamma_{2k^2-i+1} \sum_{j=1}^{2k^2-i+2} |e_j|$$

The triangle equality as well as Lemma 3.2.1 implies

$$\begin{aligned}
|t_i| &\leq (1+\gamma_{2k^2-i+1}) \sum_{j=1}^{2k^2-i+2} |e_j| \\
&\leq (1+\gamma_{2k^2-i+1})(\gamma_{2k^2-i+2})\dots(\gamma_{2k^2-1}) \sum_{j=1}^{2k^2} |e_j| \\
&\leq (\gamma_{2k^2-1}^{i-1} + \gamma_{2k^2-1}^{i}) \left(\sum_{j=1}^{k} |r_j| \cdot \sum_{j=1}^{k} |s_j| \right)
\end{aligned}$$

Therefore, the sum of each upper-bound for $i = 1, 2, \dots, k$ is

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k^2-1} + \dots 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k}) \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i| \right)$$

as desired. ■

We conclude this section by extending our previous result to make $\{t\} = \mathrm{fl}_{k,k}(c_1 c_2)$ when $\{r\} = \mathrm{fl}_{k,k}(c_1)$ and $\{s\} = \mathrm{fl}_{k,k}(c_2)$.

Theorem 3.2.3. *Suppose* $\{r\} = \mathrm{fl}_{k,k}(c_1)$ *and* $\{s\} = \mathrm{fl}_{k,k}(c_2)$ *with constants* $\rho_1, \hat{\rho}_1$ *for* $\{r\}$ *and* $\rho_2, \hat{\rho}_2$ *for* $\{s\}$. *Then, provided that* $k > 1$ *and* $(2k^2-1)\mu < 1$, *the floating-point collection* $\{t\} = ProdParts(\{r\}, \{s\}, k)$ *requires* $12k^3 - 11k^2 + 8k - 4$ *floating-point operations and satisfies*

$$\left| \sum_{i=1}^{k} t_i - c_1 c_2 \right| \leq \gamma_{2k^2-1}^{k} (2\hat{\rho}_1\hat{\rho}_2 + \rho_1(1+\hat{\rho}_2) + \rho_2(1+\hat{\rho}_1)) \overline{c}_1 \overline{c}_2.$$

Moreover,

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k^2-1} + \dots 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k}) \hat{\rho}_1\hat{\rho}_2\overline{c}_1\overline{c}_2.$$

It follows that $\{t\} = fl_{k,k}(c_1 c_2)$ *since* $\gamma_{2k^2-1}^{k}$ *is a multiple of* μ^k.

Proof. We demonstrate this result using a direct approach, but leave the floating-point operation count to the reader. The triangle inequality implies

$$\begin{aligned}
\left| \sum_{i=1}^{k} t_i - c_1 c_2 \right| &= \left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) + \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) - c_1 c_2 \right| \\
&\leq \left| \sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) \right| + \left| \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i \right) - c_1 c_2 \right|.
\end{aligned}$$

Lemma 3.2.2 gives us

$$\left|\sum_{i=1}^{k} t_i - c_1c_2\right| \leq \gamma_{2k^2-1}^{k}\left(\sum_{i=1}^{k}|r_i|\cdot\sum_{i=1}^{k}|s_i|\right) + \left|\left(\sum_{i=1}^{k} r_i\cdot\sum_{i=1}^{k} s_i\right) - c_1c_2\right|$$
$$\leq \gamma_{2k^2-1}^{k}\left(\sum_{i=1}^{k}|r_i|\cdot\sum_{i=1}^{k}|s_i|\right) + \frac{1}{2}\left|\sum_{i=1}^{k} r_i - c_1\right|\left|\sum_{i=1}^{k} s_i + c_2\right| + \frac{1}{2}\left|\sum_{i=1}^{k} r_i + c_1\right|\left|\sum_{i=1}^{k} s_i - c_2\right|.$$

We can now apply Definition 2.1.2, since $\gamma_{2k^2-1}^{k}$ is a multiple of μ^k, to find

$$\left|\sum_{i=1}^{k} t_i - c_1c_2\right| \leq \gamma_{2k^2-1}^{k}\hat{\rho}_1\hat{\rho}_2\overline{c}_1\overline{c}_2 + \frac{1}{2}\rho_1\mu^k\overline{c}_1\left|\sum_{i=1}^{k} s_i + c_2\right| + \frac{1}{2}\rho_2\mu^k\overline{c}_2\left|\sum_{i=1}^{k} r_i + c_1\right|.$$

The triangle inequality further implies

$$\left|\sum_{i=1}^{k} t_i - c_1c_2\right| \leq \gamma_{2k^2-1}^{k}\hat{\rho}_1\hat{\rho}_2\overline{c}_1\overline{c}_2 + \frac{1}{2}\rho_1\mu^k\overline{c}_1(\hat{\rho}_2\overline{c}_2 + \overline{c}_2) + \frac{1}{2}\rho_2\mu^k\overline{c}_2(\hat{\rho}_1\overline{c}_1 + \overline{c}_1)$$
$$\leq \gamma_{2k^2-1}^{k}(\hat{\rho}_1\hat{\rho}_2 + \frac{1}{2}\rho_1(1+\hat{\rho}_2) + \frac{1}{2}\rho_2(1+\hat{\rho}_1))\overline{c}_1\overline{c}_2$$
$$\leq \gamma_{2k^2-1}^{k}(2\hat{\rho}_1\hat{\rho}_2 + \rho_1(1+\hat{\rho}_2) + \rho_2(1+\hat{\rho}_1))\overline{c}_1\overline{c}_2.$$

The additional inequality

$$\sum_{i=1}^{k}|t_i| \leq (1 + 2\gamma_{2k^2-1} + \ldots 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k})\hat{\rho}_1\hat{\rho}_2\overline{c}_1\overline{c}_2.$$

is a consequence of Lemma 3.2.2 and Definition 2.1.2 for $\{r\}$ and $\{s\}$. ■

3.3 K-Parts Fuse-Multiply-Add

Algorithm (8) implements our previous addition and multiplication algorithms to compute a FMA when the input is two floating-point collections. The intermediate steps are computed to satisfy Definition 2.1.2, so we can apply Theorems 3.1.2 and 3.2.3 as well as Lemmas 3.1.1 and 3.2.2 in our analysis.

Algorithm 8: Fuse-Multiply Add stored in k-parts

```
procedure {t} = FMAKKP({q}, {r}, {s}, k)
    {u} = ProdKKP({r}, {s}, k)
    {t} = SumKKP({q}, {u})
end
```

As with the previous sections, we start by showing $\{t\} = \text{fl}_{k,k}\left(\sum_{i=1}^{k} q_i + \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)\right)$.

Corollary 3.3.1. *Suppose $\{q\} \in \mathscr{F}^k$, $\{r\} \in \mathscr{F}^k$, and $\{s\} \in \mathscr{F}^k$ are arbitrary floating-point collections. Provided that $k > 1$ and $(2k^2-1)\mu, (2k-1)\mu < 1$, the floating-point collection $\{t\} = FMAParts(\{q\},\{r\},\{s\},k)$ satisfies*

$$\left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} q_i + \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)\right)\right| \leq (\alpha\gamma_{2k-1}^{k} + \gamma_{2k^2-1}^{k}) \left(\sum_{i=1}^{k} |q_i| + \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i|\right)\right)$$

where $\alpha = 1 + 2\gamma_{2k^2-1} + 2\gamma_{2k^2-1}^2 + \cdots + \gamma_{2k^2-1}^{k}$. Moreover,

$$\sum_{i=1}^{k} |t_i| \leq \alpha(1 + 2\gamma_{2k-1} + \cdots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^{k}) \left(\sum_{i=1}^{k} |q_i| + \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i|\right)\right).$$

It follows $\{t\} = \mathrm{fl}_{k,k}\left(\sum_{i=1}^{k} q_i + \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)\right)$ since $\alpha\gamma_{2k-1}^{k} + \gamma_{2k^2-1}^{k}$ is a multiple of μ^k.

Proof. We demonstrate this proof with a direct approach. Lemma 3.1.1 from line 3 and Lemma 3.2.2 from line 2 give us the following inequalities.

$$\left|\sum_{i=1}^{k} u_i - \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)\right| \leq \gamma_{2k^2-1}^{k} \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i|\right) \tag{3.15}$$

$$\sum_{i=1}^{k} |u_i| \leq \alpha \left(\sum_{i=1}^{k} |r_i| \cdot \sum_{i=1}^{k} |s_i|\right) \tag{3.16}$$

$$\left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} q_i + \sum_{i=1}^{k} u_i\right)\right| \leq \gamma_{2k-1}^{k} \left(\sum_{i=1}^{k} |q_i| + \sum_{i=1}^{k} |u_i|\right) \tag{3.17}$$

$$\sum_{i=1}^{k} |t_i| \leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^2 + \cdots + \gamma_{2k-1}^{k}) \left(\sum_{i=1}^{k} |q_i| + \sum_{i=1}^{k} |u_i|\right) \tag{3.18}$$

where $\alpha = 1 + 2\gamma_{2k^2-1} + \cdots + 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k}$. We start with the left-hand side of the inequality as stated in the theorem and use the triangle inequality to yield

$$\begin{aligned}
&\left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} q_i + \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)\right)\right| \\
&\leq \left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} q_i + \sum_{i=1}^{k} u_i\right)\right| + \left|\sum_{i=1}^{k} u_i - \left(\sum_{i=1}^{k} r_i\right)\left(\sum_{i=1}^{k} s_i\right)\right|.
\end{aligned}$$

Inequalities (3.15), (3.16), and (3.17) simplify the previous to

$$\begin{aligned}
&\left|\sum_{i=1}^{k} t_i - \left(\sum_{i=1}^{k} q_i + \left(\sum_{i=1}^{k} r_i \cdot \sum_{i=1}^{k} s_i\right)\right)\right| \\
&\leq \gamma_{2k^2-1}^{k}\left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right) + \gamma_{2k-1}^{k}\left(\sum_{i=1}^{k}|q_i| + \sum_{i=1}^{k}|u_i|\right) \\
&\leq \gamma_{2k^2-1}^{k}\left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right) + \gamma_{2k-1}^{k}\left(\sum_{i=1}^{k}|q_i| + \alpha\left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right)\right) \\
&\leq \gamma_{2k-1}^{k}\sum_{i=1}^{k}|q_i| + (\alpha\gamma_{2k-1}^{k} + \gamma_{2k^2-1}^{k})\left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right) \\
&\leq (\alpha\gamma_{2k-1}^{k} + \gamma_{2k^2-1}^{k})\left(\sum_{i=1}^{k}|q_i| + \left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right)\right).
\end{aligned}$$

Moreover, we can use inequality (3.16) to rewrite (3.18) as

$$\begin{aligned}
\sum_{i=1}^{k}|t_i| &\leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^{2} + \cdots + \gamma_{2k-1}^{k})\left(\sum_{i=1}^{k}|q_i| + \alpha\left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right)\right) \\
&\leq \alpha(1 + 2\gamma_{2k-1} + \cdots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^{k})\left(\sum_{i=1}^{k}|q_i| + \left(\sum_{i=1}^{k}|r_i| \cdot \sum_{i=1}^{k}|s_i|\right)\right)
\end{aligned}$$

as desired. ■

Additionally, theorems 3.1.2 and 3.2.3 provide us the error-bounds to confirm $\{t\} = \text{fl}_{k,k}\left(c_1 + c_2 \cdot c_3\right)$, as outlined in the brief proof below.

Corollary 3.3.2. *Suppose* $\{q\} = fl_{k,k}(c_1)$, $\{r\} = fl_{k,k}(c_2)$, *and* $\{s\} = fl_{k,k}(c_3)$ *with constants* $\rho_1, \hat{\rho}_1$ *for* $\{q\}$, $\rho_2, \hat{\rho}_2$ *for* $\{r\}$, *and* $\rho_3, \hat{\rho}_3$ *for* $\{q\}$. *Then, provided that* $k > 1$ *and* $(2k-1)\mu < 1$, *the floating-point collection* $\{t\} = FMAParts(\{q\}, \{r\}, \{s\}, k)$ *requires* $12k^3 - 2k^2 - 3$ *floating-point operations and satisfies*

$$\left|\sum_{i=1}^{k} t_i - (c_1 + c_2c_3)\right| \leq \gamma_{2k-1}^{k} max\left(\{\rho_1 + \hat{\rho}_1, \alpha + \hat{\alpha}\}\right)(\overline{c}_1 + \overline{c}_2\overline{c}_3)$$

where $\alpha = \gamma_{2k^2-1}^{k}(2\hat{\rho}_2\hat{\rho}_3 + \rho_2(1+\hat{\rho}_3) + \rho_3(1+\hat{\rho}_2))$ *and* $\hat{\alpha} = (1 + 2\gamma_{2k^2-1} + \ldots 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k})\hat{\rho}_2\hat{\rho}_3$. *Moreover,*

$$\sum_{i=1}^{k}|t_i| \leq (1 + 2\gamma_{2k-1} + 2\gamma_{2k-1}^{2} + \cdots + \gamma_{2k-1}^{k})max(\{\hat{\rho}_1, \hat{\alpha}\})(\overline{c}_1 + \overline{c}_2\overline{c}_3).$$

It follows that $\{t\} = fl_{k,k}(c_1 + c_2c_3)$ *since* γ_{2k-1}^{k} *is a multiple of* μ^k.

Proof. We demonstrate this proof in a direct approach, but leave the floating-point operation count to the reader. Line 2 of FMAParts permits us to use Theorem 3.2.3 to infer $\{u\} = \mathrm{fl}_{k,k}\left(c_2 \cdot c_3\right)$ with constants α and $\hat{\alpha}$. Therefore, this result follows by the use of Theorem 3.1.2, on line 3 of FMAKKP, with inputs $\{q\}$ and $\{u\}$. ■

3.4 K-Parts Division

We adapt the Newton-Raphson Division algorithm for k-parts in Algorithm (9), as the Newton-Raphson Division algorithm computes the reciprocal of a number using only multiplication and addition operations. In our analysis, note that multiplication by two produces no round-off error.

Algorithm 9: Netwon's Division Method in K-Parts

```
1 procedure {x} = DivParts({x^(0)}, {b}, k)
2     for l = 0 to ⌈log_2(k)⌉ do
3         {y} = ProdParts({x^(l)}, {x^(l)}, k)
4         {x^(l+1)} = FMAParts(2{x^(l)}, −{b}, {y}, k).
5     end
6 end
```

The proof of the following lemma is similar to the proof of Corollary 3.3.1 with the inequalities from Corollary 3.3.1 and Lemma 3.2.1.

Lemma 3.4.1. *Suppose $\{x^{(l)}\} \in \mathscr{F}^k$ and $\{b\} \in \mathscr{F}^k$ are arbitrary floating-point collections. Provided that $k > 1$ and $(2k-1)\mu, (2k^2-1) < 1$, the floating-point collection $\{x^{(l+1)}\}$ in $DivParts$ satisfies*

$$\left|\sum_{i=1}^{k} x_i^{(l+1)} - \left(2\sum_{i=1}^{k} x_i^{(l)} - \left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2\right)\right|$$
$$\leq (\alpha\gamma_{2k-1}^{k} + 2\gamma_{2k^2-1}^{k})\left(2\sum_{i=1}^{k}\left|x_i^{(l)}\right| + \left(\sum_{i=1}^{k}|b_i|\right)\left(\sum_{i=1}^{k}\left|x_i^{(l)}\right|\right)^2\right)$$

where $\alpha = 1 + 2\gamma_{2k^2-1} + 2\gamma_{2k^2-1}^2 + \cdots + \gamma_{2k^2-1}^{k}$. Moreover,

$$\sum_{i=1}^{k}\left|x_i^{(l+1)}\right| \leq \alpha^2(1 + 2\gamma_{2k-1} + \cdots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^{k})\left(2\sum_{i=1}^{k}\left|x_i^{(l)}\right| + \left(\sum_{i=1}^{k}|b_i|\right)\left(\sum_{i=1}^{k}\left|x_i^{(l)}\right|\right)^2\right).$$

It follows $\{x^{(l+1)}\} = \mathrm{fl}_{k,k}\left(2\sum_{i=1}^{k} x_i^{(l)} - \left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2\right)$ since $\alpha\gamma_{2k-1}^{k} + \gamma_{2k^2-1}^{k}$ is a multiple of μ^k.

If we assume $\{b\} = \mathrm{fl}_{k,k}\left(b\right)$ and $\{x^{(l)}\} = \mathrm{fl}_{l,k}\left(\left(\sum_{i=1}^{k} b_i\right)^{-1}\right)$, then the next lemma follows by expanding upon Lemma 3.4.1 with the inequalities provided by Definition 2.1.2 for $\{b\}$

and $\{x^{(l)}\}$. It's important to note the constants used in Lemma 3.4.2 are not from absolute difference between $\{x^{(l)}\}$ and $\{b\}$; that is, we only use the constant from the absolute upper-bound of $\sum_{i=1}^{k}\left|x_i^{(l)}\right|$.

Lemma 3.4.2. *Suppose $\hat{\rho}_1$ and $\{x^{(l)}\} = \mathrm{fl}_{l,k}\left(\left(\sum_{i=1}^{k} b_i\right)^{-1}\right)$ with constants $\rho_1, \hat{\rho}_1$. Then, Provided that $k > 1$ and $(2k-1)\mu, (2k^2-1)\mu < 1$, the floating-point collection $\{x^{(l+1)}\}$ in $DivParts$ satisfies*

$$\left|\sum_{i=1}^{k} x_i^{(l+1)} - \left(2\sum_{i=1}^{k} x_i^{(l)} - \left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2\right)\right| \leq (\alpha\gamma_{2k-1}^{k} + 2\gamma_{2k^2-1}^{k})\hat{\rho}_1(2+\hat{\rho}_1)\left(\sum_{i=1}^{k}|b_i|\right)^{-1}$$

where $\alpha = 1 + 2\gamma_{2k^2-1} + 2\gamma_{2k^2-1}^2 + \cdots + \gamma_{2k^2-1}^{k}$. Moreover,

$$\sum_{i=1}^{k}\left|x_i^{(l+1)}\right| \leq \alpha^2(1 + 2\gamma_{2k-1} + \cdots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^{k})\hat{\rho}_1(2+\hat{\rho}_1)\left(\sum_{i=1}^{k}|b_i|\right)^{-1}.$$

It follows $\{x^{(l+1)}\} = \mathrm{fl}_{k,k}\left(2\sum_{i=1}^{k} x_i^{(l)} - \left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2\right)$ since $\alpha\gamma_{2k-1}^{k} + \gamma_{2k^2-1}^{k}$ is a multiple of μ^k.

We conclude the analysis of our arithmetic algorithms by demonstrating $\{x^{(l+1)}\} = \mathrm{fl}_{2l,k}(b^{-1})$ when $\{x^{(l)}\} = \mathrm{fl}_{l,k}\left(\left(\sum_{i=1}^{k} b_i\right)^{-1}\right)$.

Theorem 3.4.3. *Suppose $\{b\} = \mathrm{fl}_{k,k}(b)$ with constants ρ_1 and $\hat{\rho}_1$ and $\{x^{(l)}\} = \mathrm{fl}_{l,k}\left(\left(\sum_{i=1}^{k} b_i\right)^{-1}\right)$ with constants ρ_2 and $\hat{\rho}_2$. Then, the floating-point collection $\{x^{(l+1)}\}$ in DivParts requires $24k^3 - 13k^2 + 8k - 7$ floating-point operations and satisfies*

$$\left|\sum_{i=1}^{k} x^{(l+1)} - b^{-1}\right| \leq \beta(\bar{b})^{-1}.$$

where

$$\beta = \hat{\rho}_1\left(\frac{\rho_2\bar{b}}{\sum_{i=1}^{k}|b_i|}\right)^2\mu^{2l} + \frac{\rho_2(\bar{b})^2}{\left|b\sum_{i=1}^{k} b_i\right|}\mu^k + (\alpha\gamma_{2k-1}^{k} + 2\gamma_{2k^2-1}^{k})\hat{\rho}_1(2+\hat{\rho}_1)\left(\frac{\bar{b}}{\sum_{i=1}^{k}|b_i|}\right)$$

and $\alpha = 1 + 2\gamma_{2k^2-1} + \cdots + 2\gamma_{2k^2-1}^{k-1} + \gamma_{2k^2-1}^{k}$.

Proof. We demonstrate this result with a direct approach, but leave the floating-point operation count to the reader. The left-hand side of the inequality can be expanded with the triangle inequality to yield

$$\left|\sum_{i=1}^{k} x^{(l+1)} - b^{-1}\right| \leq \left|\left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2 - 2\sum_{i=1}^{k} x_i^{(l)} + b^{-1}\right| + \left|\sum_{i=1}^{k} x_i^{(l+1)} - \left(2\sum_{i=1}^{k} x_i^{(l)} - \left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2\right)\right|. \tag{3.19}$$

The first inequality on the right-hand side simplifies to

$$\begin{aligned}
&\left|\left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2 - 2\sum_{i=1}^{k} x_i^{(l)} + b^{-1}\right| \\
&\leq \left|\sum_{i=1}^{k} b_i\right|\left|\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2 - 2\left(\sum_{i=1}^{k} b_i\right)^{-1}\sum_{i=1}^{k} x_i^{(l)} + \left(b\sum_{i=1}^{k} b_i\right)^{-1}\right| \\
&\leq \left|\sum_{i=1}^{k} b_i\right|\left|\left(\sum_{i=1}^{k} x_i^{(l)} - \left(\sum_{i=1}^{k} b_i\right)^{-1}\right)^2 + \left(b\sum_{i=1}^{k} b_i\right)^{-1} - \left(\sum_{i=1}^{k} b_i\right)^{-2}\right|
\end{aligned} \tag{3.20}$$

where the last line comes from completing the square for a quadratic polynomial in variable $\sum_{i=1}^{k} x_i^{(l)}$. Our assumptions on $\{x^{(l)}\}$ and $\{b\}$ further imply (3.20) becomes

$$\begin{aligned}
\left|\left(\sum_{i=1}^{k} b_i\right)\left(\sum_{i=1}^{k} x_i^{(l)}\right)^2 - 2\sum_{i=1}^{k} x_i^{(l)} + b^{-1}\right| &\leq \hat{\rho}_1\bar{b}\left(\frac{\rho_2\mu^l}{\sum_{i=1}^{k}|b_i|}\right)^2 + \frac{\left|\sum_{i=1}^{k} b_i - b\right|}{\left|b\sum_{i=1}^{k} b_i\right|} \\
&\leq \hat{\rho}_1\bar{b}\left(\frac{\rho_2\mu^l}{\sum_{i=1}^{k}|b_i|}\right)^2 + \frac{\rho_1\mu^k\bar{b}}{\left|b\sum_{i=1}^{k} b_i\right|}
\end{aligned} \tag{3.21}$$

Therefore, applying Lemma 3.4.2 and (3.21) to (3.19) provides us with

$$\left|\sum_{i=1}^{k} x^{(l+1)} - b^{-1}\right| \leq \beta(\bar{b})^{-1}.$$

where

$$\beta = \hat{\rho}_1\left(\frac{\rho_2\bar{b}}{\sum_{i=1}^{k}|b_i|}\right)^2\mu^{2l} + \frac{\rho_2(\bar{b})^2}{\left|b\sum_{i=1}^{k} b_i\right|}\mu^k + (\alpha\gamma_{2k-1}^k + 2\gamma_{2k^2-1}^k)\hat{\rho}_1(2+\hat{\rho}_1)\left(\frac{\bar{b}}{\sum_{i=1}^{k}|b_i|}\right)$$

as desired. ■

In practice, we use the division routine supported by the system's architecture to establish $\{x^{(0)}\} = \text{fl}_{1,k}\left(\sum_{i=1}^{k} b_i\right)^{-1}$. Theorem 3.4.3 permits us to use our approximation, $\{x^{(0)}\}$, to infer that on the $\lceil log_2(k) \rceil$th iteration of DivParts we have $\{x^{(\lceil log_2(k) \rceil)}\} = \text{fl}_{k,k}\left(b^{-1}\right)$. Moreover, the second inequality of Lemma 3.4.2 gives us

$$\sum_{i=1}^{k} \left| x_i^{(l+1)} \right| \leq \alpha^2(1 + 2\gamma_{2k-1} + \cdots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^{k})\hat{\rho}_1(2 + \hat{\rho}_1) \left(\frac{\overline{b}}{\sum_{i=1}^{k} |b_i|} \right) (\overline{b})^{-1}$$

for the absolute upper-bound of $\sum_{i=1}^{k} \left| x_i^{(l+1)} \right|$.

Chapter 4

Perturbation Theory with Matrix Applications

4.1 K-Parts Perturbation and The Dot Product

The error-bounds in Chapter 3 describe the forward-error produced by an arithmetic operation. Our next step is to develop a way to use Definition 2.1.2 for backward-error analysis. Theorem 4.1.1 provides an analogous model to the IEEE 754 standard for a working-precision of order μ^k.

Theorem 4.1.1. *Let $\{r\} \in \mathscr{F}^k$ be a floating-point collection and c be some non-zero operation. Then, $\{r\} = fl_{k,k}(c)$ only if*

$$\{r\} = \sum_{i=1}^{k} r_i = (1+\delta)c$$

for $|\delta| \leq \rho\mu^k$, where ρ is some relatively small constant in respect to μ^{-k}.

Proof. We demonstrate this result with a direct approach. Suppose $\{r\} = fl_{k,k}(c)$. Then,

$$\left| \sum_{i=1}^{k} r_i - c \right| \leq \rho\mu^k \overline{c}.$$

Another way to write the previous is $\sum_{i=1}^{k} r_i = c + \epsilon$ for $|\epsilon| \leq \rho\mu^k\overline{c}$. Therefore, we have that

$$\sum_{i=1}^{k} r_i = c + \epsilon = \left(1 + \frac{\epsilon}{c}\right) c = (1+\delta)c$$

and

$$\delta := \frac{\epsilon}{c} \implies |\delta| = \left|\frac{\epsilon}{c}\right| \leq \left(\frac{\rho\overline{c}}{|c|}\right)\mu^k.$$

■

We can appeal to Definition 2.1.2 to infer $\{r\} = fl_{k,k}(c)$ only if $\sum_{i=1}^{k} r_i = \delta$ for $|\delta| \leq \rho\mu^k\overline{c}$ when $c = 0$. Our goal is to apply the previous theorem to analyze the backward-error of Algorithm (10).

Algorithm 10: Dot Product Stored in K-Parts

```
1 procedure {s_n} = DotParts({x}, {y}, k, c)
2 |   {s^(0)} = {c, 0, 0, ..., 0}
3 |   for i = 1 to n do
4 |   |   {s^(i)} = FMAParts({s^(i-1)}, {x_i}, {y_i})
5 |   end
6 end
```

Assume $\{x_i\}$ and $\{y_i\}$ are arbitrary floating-point collections of size k, which could satisfy Definition 2.1.2. We first define $\{s^{(i)}\} := c + \{x_1\}\{y_1\} + \{x_2\}\{y_2\} + \cdots + \{x_i\}\{y_i\}$ so that

$$\begin{aligned}
\{s^{(1)}\} &= \mathrm{fl}_{k,k}\left(\{s^{(0)}\} + \{x_1\}\{y_1\}\right) = (1+\delta_1)(c + \{x_1\}\{y_1\}) \\
\{s^{(2)}\} &= \mathrm{fl}_{k,k}\left(\{s^{(1)}\} + \{x_2\}\{y_2\}\right) = ((1+\delta_1)(c + \{x_1\}\{y_1\}) + \{x_2\}\{y_2\})(1+\delta_2) \\
&= (1+\delta_1)(1+\delta_2)c + (1+\delta_1)(1+\delta_2)\{x_1\}\{y_1\} + (1+\delta_2)\{x_2\}\{y_2\}
\end{aligned}$$

by our application of Theorem 3.3.1. The value of each δ_i is not important to consider in our analysis, so we replace $(1+\delta_i) = (1 \pm \delta)$ as was done in [14, Section 3.1]. Then,

$$\{s^{(3)}\} = (1\pm\delta)^3 c + (1\pm\delta)^3\{x_1\}\{y_1\} + (1\pm\delta)^2\{x_2\}\{y_2\} + (1\pm\delta)\{x_3\}\{y_3\}.$$

This pattern repeats itself yield

$$\begin{aligned}
\{s^{(n)}\} = (1\pm\delta)^n c + (1\pm\delta)^n\{x_1\}\{y_1\} + (1\pm\delta)^{n-1}\{x_2\}\{y_2\} \\
+ (1\pm\delta)^{n-2}\{x_3\}\{y_3\} + \cdots + (1\pm\delta)\{x_n\}\{y_n\}.
\end{aligned} \tag{4.1}$$

The following result provides a clean way to simplify the previous expression.

Lemma 4.1.2. *If $|\delta_i| \leq \rho_i\mu^k$ for $i = 1, 2, \ldots, n$ and $\left(\sum_{i=1}^n \rho_i\right)\mu^k < 1$, then*

$$\prod_{i=1}^{n}(1+\delta_i)^{\pm 1} = 1 + \theta_n$$

where

$$|\theta_n| \leq \kappa_n := \frac{\left(\sum_{i=1}^n \rho_i\right)\mu^k}{1 - \left(\sum_{i=1}^n \rho_i\right)\mu^k}.$$

Proof. We demonstrate this result by induction on $n \in \mathbb{N}$. The case for when $n = 1$ follows from the premise and our definition of κ_n. Suppose, for the inductive step, the aforementioned statement holds true for some $n \in \mathbb{N}$. Then $(1+\delta_{n+1})^1$ provides us with

$$\prod_{i=1}^{n+1}(1+\delta_i)^{\pm 1} = (1+\theta_n)(1+\delta_{n+1}) = 1 + \delta_{n+1} + \theta_n + \delta_{n+1}\theta_n$$

for $|\delta_{n+1}| \leq \rho_{n+1}\mu^k$ and $|\theta_n| \leq \kappa_n$. Let $\theta_{n+1} = \delta_{n+1}(1+\theta_n) + \delta_{n+1}$ so that

$$\begin{aligned}
|\theta_{n+1}| &\leq \rho_{n+1}\mu^k\left(1 + \frac{\left(\sum_{i=1}^n \rho_i\right)\mu^k}{1 - \left(\sum_{i=1}^n \rho_i\right)\mu^k}\right) + \frac{\left(\sum_{i=1}^n \rho_i\right)\mu^k}{1 - \left(\sum_{i=1}^n \rho_i\right)\mu^k} \\
&\leq \frac{\rho_{n+1}\mu^k}{1 - \left(\sum_{i=1}^n \rho_i\right)\mu^k} + \frac{\left(\sum_{i=1}^n \rho_i\right)\mu^k}{1 - \left(\sum_{i=1}^n \rho_i\right)\mu^k} \\
&\leq \frac{\left(\sum_{i=1}^{n+1} \rho_i\right)\mu^k}{1 - \left(\sum_{i=1}^n \rho_i\right)\mu^k} \\
&\leq \frac{\left(\sum_{i=1}^{n+1} \rho_i\right)\mu^k}{1 - \left(\sum_{i=1}^{n+1} \rho_i\right)\mu^k} = \kappa_{n+1}
\end{aligned}$$

If we instead have $(1+\delta_{n+1})^{-1}$, then note

$$\prod_{i=1}^{n+1}(1+\delta_i)^{\pm 1} = \frac{1+\theta_n}{1+\delta_{n+1}} = 1 + \frac{\theta_n - \delta_{n+1}}{1+\delta_{n+1}}$$

for $|\delta_{n+1}| \leq \rho_{n+1}\mu^k$ and $|\theta_n| \leq \kappa_n$. Let $\theta_{n+1} = \frac{\theta_n - \delta_{n+1}}{1+\delta_{n+1}}$. Since $|1+\delta_{n+1}| \geq 1 - |\delta_{n+1}|$ we have

$$|\theta_{n+1}| \leq \frac{|\theta_n| + \rho_{n+1}\mu^k}{1 - \rho_{n+1}\mu^k}$$

where our definition for κ_n and some simplification yields

$$|\theta_{n+1}| \leq \frac{\left(\sum_{i=1}^{n+1}\rho_i\right)\mu^k - \rho_{n+1}\left(\sum_{i=1}^{n}\rho_i\right)\mu^{2k}}{1 - \left(\sum_{i=1}^{n+1}\rho_i\right)\mu^k + \rho_{n+1}\left(\sum_{i=1}^{n}\rho_i\right)\mu^k} \leq \kappa_n$$

as desired. ■

Equation (4.1) with Lemma 4.1 becomes

$$\begin{aligned}\{s^{(n)}\} = (1\pm\theta_n)c + (1\pm\tilde{\theta}_n)\{x_1\}\{y_1\} + (1\pm\theta_{n-1})\{x_2\}\{y_2\} \\ + (1\pm\theta_{n-2})\{x_3\}\{y_3\} + \cdots + (1\pm\theta_1)\{x_n\}\{y_n\}.\end{aligned} \tag{4.2}$$

The $1+\theta_i$ factor for $i = 1, 2, \ldots, n$ describes how an exact k-parts dot product is perturbed to instead produce $\{s^{(n)}\}$. This perturbation, as Higham notes, can apply to either $\{\boldsymbol{y}\}$ or $\{\boldsymbol{x}\}$ but not both. We can also change the order of operations in (4.2) and produce a similar result by noting $|\theta_i| \leq \kappa_n$. As a consequence, we have

$$\{s^{(n)}\} = (c+\Delta c) + (\{\boldsymbol{x}\} + \Delta\boldsymbol{x})^T\{\boldsymbol{y}\} = (c+\Delta c) + \{\boldsymbol{x}^T\}(\{\boldsymbol{y}\} + \Delta\boldsymbol{y}) \tag{4.3}$$

so that

$$|\Delta c| \leq \kappa_n |c|, \quad |\Delta\boldsymbol{x}| \leq \kappa_n |\{\boldsymbol{x}\}|, \quad |\Delta\boldsymbol{y}| \leq \kappa_n |\{\boldsymbol{y}\}|$$

where $|\{\boldsymbol{x}\}|$ denotes the component-wise absolute value of each entry in $\{\boldsymbol{x}\}$ and the inequality is component-wise. We now show the first inequality of Definition 2.1.2 holds true for $\{s^{(n)}\}$ by using (4.3). Consider the absolute forward-error

$$\begin{aligned}\left|\{s^{(n)}\} - (c + \{\boldsymbol{x}^T\}\{\boldsymbol{y}\})\right| &= \left|(c+\Delta c) + \{\boldsymbol{x}^T\}(\{\boldsymbol{y}\} + \Delta\boldsymbol{y}) - (c + \{\boldsymbol{x}^T\}\{\boldsymbol{y}\})\right| \\ &\leq |\Delta c| + \left|\{\boldsymbol{x}^T\}\Delta\boldsymbol{y}\right| \\ &\leq \kappa_n |c| + \kappa_n \left|\{x^T\}\right| |\{\boldsymbol{y}\}|\end{aligned}$$

Therefore, we have Theorem 4.1.3 as shown below.

Theorem 4.1.3. *Let $c \in \mathscr{F}$ and $\{\boldsymbol{x}\}, \{\boldsymbol{y}\} \in \mathscr{F}^n$ be defined so that $\{x_i\}, \{y_i\} \in \mathscr{F}^k$ are arbitrary floating-point collections. Then, $\{s^{(n)}\} = DotParts(\{\boldsymbol{x}\}, \{\boldsymbol{y}\}, k, c)$ satisfies*

$$\left|\{s^{(n)}\} - (c + \{\boldsymbol{x}^T\}\{\boldsymbol{y}\})\right| \leq \kappa_n(|c| + \sum_{i=1}^{n} |\{x_i\}| \cdot |\{y_i\}|).$$

4.2 A Brief Analysis of Triangular Systems

We now focus on the error produced in solving an upper-triangular system, where the results in this section extend to lower-triangular systems as well. We assume $\{U\}$ is a matrix whose entries are a floating-point collection, which may satisfy Definition 2.1.2. Algorithm (11) computes the solution to the linear system $\{U\}\{\boldsymbol{x}\} = \boldsymbol{b}$ by backward substitution. Then, Theorem 4.2.1 replicates our analysis in Section 4.1 with an additional division operation.

Algorithm 11: Upper-Triangular System Solver in K-Parts

```
1  procedure {x} = SolveUpperParts({U}, b, k)
2      for i = n to 1 do
3          {y} = {b_i, 0, 0, ..., 0}
4          for j = i + 1 to n do
5              {y} = FMAParts({y}, {u_ij}, -{x_j}, k)          ▷ y = y - u_ij x_j
6          end
7          {z} = {({u_ii}_1)^{-1}, 0, 0, ..., 0}     ▷ Approximate {u_ii}^{-1} in μ-precision
8          {z} = DivParts({z}, {u_ii}, k)
9          {x_i} = ProdParts({y}, {z}, k)                  ▷ x_i = x_i / u_ii
10     end
11 end
```

Theorem 4.2.1. *Let $\{x_i\}$ denote the result of Lines 3 through 9 in Algorithm 11. Then, $\{x_i\}$ satisfies*

$$(1+\theta_{n-i+1})\{u_{ii}\}\{x_i\} = b_i - \sum_{j=i+1}^{n} (1+\theta_{|i+1-j|})\{u_{ij}\}\{x_j\}$$

where $|\theta_j| \leq \kappa_{n-i+1}$ and $\left|\theta_{|i+1-j|}\right| \leq \kappa_{|i+1-j|}$.

Proof. Our analysis of Algorithm (10) and Line 5 of SolveUpperParts provides us the equality

$$\{y\} = (1+\delta_{i+1})(1+\delta_{i+2})\cdots(1+\delta_n)b_i - \sum_{j=i+1}^{n} (1+\delta_j)(1+\delta_{j+1})\cdots(1+\delta_n)\{u_{ij}\}\{x_j\}.$$

Line 9 furthermore establishes by ProdParts and Theorem 4.1.1 that

$$\{x_i\} = \mathrm{fl}_{k,k}\left(\frac{\{y\}}{\{u_{ii}\}}\right) = (1+\delta^*)\frac{\{y\}}{\{u_{ii}\}}$$

for $|\delta^*| \leq \rho^*\mu^k$. Note the constant ρ^* contains the constant from $\{z\} = \mathrm{fl}_{k,k}\left(\{u_{ii}\}^{-1}\right)$, which is computed on Lines 7 and 8. We can simplify $\{x_i\}$ by substituting $\{y\}$ to yield

$$\frac{\{u_{ii}\}\{x_i\}}{(1+\delta^*)(1+\delta_{i+1})\cdots(1+\delta_n)} = b_i - \sum_{j=i+1}^{n} \frac{(1+\delta_j)\ldots(1+\delta_n)\{u_{ij}\}\{x_j\}}{(1+\delta_{i+1})\cdots(1+\delta_n)}.$$

Therefore, Lemma 4.1.2 implies

$$(1+\theta_{n-i+1})\{u_{ii}\}\{x_i\} = b_i - \sum_{j=i+1}^{n}(1+\theta_{|i+1-j|})\{u_{ij}\}\{x_j\}$$

where $|\theta_j| \leq \kappa_{n-i+1}$ and $\left|\theta_{|i+1-j|}\right| \leq \kappa_{|i+1-j|}$ as desired. ■

We now know how an entry in $\{U\}$ is perturbed to make an exact $\{x_i\}$. Thus, we find

$$(\{U\}+\Delta U)\{\boldsymbol{x}\} = \boldsymbol{b}, \quad |\Delta U| \leq \begin{cases} \kappa_{n-i+1}\left|\{u_{ii}\}\right|, & i=j \\ \kappa_{|i+1-j|}\left|\{u_{ij}\}\right|, & i\neq j \end{cases} \tag{4.4}$$

as we did with the dot product. We want to acknowledge the order of operations can significantly effect the error-bounds shown in (4.4). Although, we defer the discussion on an arbitrary order of operations to [14, Chapter 8], where Higham accounts for an arbitrary order of operations with the floating-point model. To provide a relative error for $\{x_i\}$, we want to state an important theorem that relates the backward-error of a linear system to the relative-error of its solution. We modify the theorem to suit our needs for k-parts and μ^k precision.

Theorem 4.2.2. *([14, Theorem 7.4]) Let $\{A\}\{\boldsymbol{x}\} = \boldsymbol{b}$ and $(\{A\}+\Delta A)\{\boldsymbol{y}\} = \boldsymbol{b}$ where $|\Delta A| \leq \rho\mu^k\left|\{A\}\right|$, and assume that $\rho\mu^k\left\||\{A^{-1}|\}\{A\}\right\| < 1$ where $\|\cdot\|$ is an absolute norm. Then,*

$$\frac{\|\{\boldsymbol{x}\}-\{\boldsymbol{y}\}\|}{\|\{\boldsymbol{x}\}\|} \leq \frac{\rho\mu^k}{1-\rho\mu^k\left\|\left|\{A^{-1}\}\right|\left|\{A\}\right|\right\|}\left(\frac{\left\|\left|\{A^{-1}\}\right|\left|\{A\}\right|\left|\{\boldsymbol{x}\}\right|\right\|}{\left\|\left|\{\boldsymbol{x}\}\right|\right\|}\right)$$

and for the $\infty-norm$ this bound is attainable to first order in $\rho\mu^k$.

Let $\{\boldsymbol{y}\} = SolveUpperParts(\{U\}, \boldsymbol{b}, k)$ and note both κ_{n-i+1} and $\kappa_{|i+1-j|}$ are less than κ_n. Theorem 4.2.2 and (4.4) provide us with the forward-error result

$$\frac{\|\{\boldsymbol{x}\}-\{\boldsymbol{y}\}\|}{\|\{\boldsymbol{x}\}\|} \leq \frac{\text{cond}(\{U\},\{x\})\kappa_n}{1-\text{cond}(\{U\})\kappa_n} \tag{4.5}$$

where

$$\text{cond}(\{U\},\{\boldsymbol{x}\}) = \frac{\left\|\left|\{A^{-1}\}\right|\left|\{A\}\right|\left|\{\boldsymbol{x}\}\right|\right\|}{\left\|\left|\{\boldsymbol{x}\}\right|\right\|} \quad \text{and} \quad \text{cond}(\{U\}) = \left\|\left|\{A^{-1}\}\right|\left|\{A\}\right|\right\|.$$

Therefore, the first inequality in Definition 2.1.2 holds true for the entries in $\{\boldsymbol{y}\}$ as long as $\{U\}$ and its corresponding matrix U have a relatively small condition number relative to μ^{-k}. We specify both $\{U\}$ and U must have a relatively small condition number since we speculate from Definition 2.1.2 that $\text{cond}(U) \approx \text{cond}(\{U\})$.

4.3 Gaussian Elimination and Iterative Refinement

Higher precision computations have an application in the iterative refinement of a solution to a linear system. Consider the matrix equation $A\boldsymbol{x} = \boldsymbol{b}$. If $\hat{\boldsymbol{x}}^{(0)} = A^{-1}\boldsymbol{b}$ is the computed solution in u precision, then the following process improves the accuracy of $\hat{\boldsymbol{x}^{(0)}}$ on the ith iteration.

1. Compute $\boldsymbol{r} = A\hat{\boldsymbol{x}}^{(i)} - \boldsymbol{b}$ in u^2 precision.
2. Solve $A\boldsymbol{z} = \boldsymbol{r}$ in u precision.
3. Update $\hat{\boldsymbol{x}}^{(i+1)} = \hat{\boldsymbol{x}}^{(i)} + \boldsymbol{z}$ in u precision.

Our goal is to implement iterative refinement to calculate $\hat{\boldsymbol{x}}_i$ in $u = \mu^k$ precision. To do so, we first compute the LU factorization of A with pivoting and store the result in k-parts, which we denote as $\{A\}$. We then use forward and backward substitution to produce an initial estimate of $\{A\}\{\boldsymbol{x}^{(0)}\} = \boldsymbol{b}$. Finally, the process outlined above is computed until the backward relative error is small or the allotted number of iterations occurs.

Higham has an in-depth analysis of the forward error produced by iterative refinement under the assumption that a linear solver satisfies

$$(A + \Delta A)\hat{\boldsymbol{x}} = \boldsymbol{b}, \qquad |\Delta A| \leq uW$$

for some nonnegative matrix W depending on A, n, and u. Additionally, let $\overline{\gamma}_n = \frac{nu^2}{1-nu^2}$ and denote the identity matrix as I_n. Higham derived the forward error produced by iterative refinement, in [14, Theorem 12.1], to be

$$|\hat{\boldsymbol{x}}_{i+1} - \boldsymbol{x}| \lessapprox G_i|\hat{\boldsymbol{x}}_{i+1} - \boldsymbol{x}| + g_i$$

where

$$\begin{aligned} G_i &= u|A^{-1}|W + (u\overline{\gamma}_{n+1})(I_n + u|A^{-1}W|)|A^{-1}||A| \\ g_i &= 2\overline{\gamma}_{n+1}(I_n + u|A^{-1}|W)|A^{-1}||A||\boldsymbol{x}| + u|\boldsymbol{x}| \end{aligned}$$

and the absolute value is component-wise. Higham also uses $uW := \gamma_{3n}|\hat{L}||\hat{U}|$ to describe an upper-bound for ΔA with respect to the computed LU factorization of A [14, Theorem 9.4]. If each arithmetic operation in the iterative refinement process is replaced with their k-parts counterpart, then we use Higham's analysis to speculate

$$\left|\{\boldsymbol{x}^{(i+1)}\} - \boldsymbol{x}\right| \lessapprox G_i \left|\{\boldsymbol{x}^{(i)}\} - \boldsymbol{x}\right| + g_i$$

where $G_i = O(\mu^k) + O(\mu^{2k})$ and $g_i = \mu^k |\boldsymbol{x}| + O(\mu^{2k})$. Algorithm (12) computes the LU Decomposition of a matrix and Algorithm (13) conducts the iterative refinement process. We specify Gaussian Elimination with Partial Pivoting (GEPP) in K-Parts within the algorithms as needed, but it is not necessary to construct GEPP, assuming our input is an LU Decomposition, as it is an application of Algorithm (11) from Section 4.2. Each result is as accurate as if computed in k-fold precision and stored in k-parts, which we opt to justify numerically in this thesis.

Algorithm 12: LU Decomposition in K-Parts

procedure $[\{A\}, \sigma] = LUParts(A, k)$
  Construct $\{A\}$ so that $\{a_{ij}\} = \{a_{ij}, 0, 0, \ldots, 0\}$
  $\sigma = (1, 2, \ldots, n)$
  for $j = 1$ *to* $n - 1$ **do**
    Denote m to be the index of $max(\{|\{a_{jj}\}_1|, \ldots, |\{a_{nj}\}_1|\}$
    if $\{a_{mj}\} \equiv 0$ **then**
      Signal A is a Singular Matrix and End Program
    end
    if $m \neq j$ **then**
      Swap σ_m and σ_j
      for $l = 1$ *to* n **do**
        Swap $\{a_{jl}\}$ and $\{a_{ml}\}$
      end
    end
    for $i = j + 1$ *to* n **do**
      $\{z\} = \{(\{a_{jj}\}_1)^{-1}, 0, 0, \ldots, 0\}$ ▷ Approximate $\{a_{jj}\}$ in μ-precision
      $\{z\} = DivParts(\{z\}, \{a_{jj}\}, k)$
      $\{a_{ij}\} = ProdParts(\{a_{ij}\}, \{z\}, k)$ ▷ Compute $\{L\}$ in $A = \{L\}\{U\}$
      for $l = j + 1$ *to* n **do**
        $\{a_{il}\} = FMAParts(\{a_{il}\}, \{a_{ij}\}, \{a_{jl}\})$ ▷ Compute $\{U\}$ in $A = \{L\}\{U\}$
      end
    end
  end
end

Algorithm 13: Iterative Refinement of $\{\boldsymbol{x}\}$ in K-Parts

procedure $\{\boldsymbol{x}^{(0)}\} = IterRefParts(A, \{A\}, \{\boldsymbol{x}\}, \boldsymbol{b}, \sigma, \epsilon, MAX, k)$
  Construct $\{A^*\}$ so that $\{a^*_{ij}\} = \{a_{ij}, 0, 0, \ldots, 0\}$
  for $m = 0$ *to* MAX **do**
    for $i = 1$ *to* n **do**
      $\{r_i\} = DotParts(\{\boldsymbol{a}^*_i\}, -\{\boldsymbol{x}^{(m)}\}, b_i, 2k)$ ▷ Compute $\{\boldsymbol{r}\} = \boldsymbol{b} - \{A\}\{\boldsymbol{x}^{(m)}\}$
    end
    Solve $\{A\}\{\boldsymbol{z}\} = \{\boldsymbol{r}\}$ with GEPP in K-Parts with σ
    if $\|\{\boldsymbol{z}\}\|_1 < \epsilon$ **then**
      Return $\{\boldsymbol{x}^{(m)}\}$
    end
    else
      for $i = 1$ *to* n **do**
        $\{x^{(m+1)}_i\} = SumParts(\{x^{(m)}_i\}, \{z_i\}, k)$ ▷ $\{\boldsymbol{x}^{(m+1)}\} = \{\boldsymbol{x}^{(m)}\} + \{\boldsymbol{z}\}$
      end
    end
  end
  Signal ϵ Tolerance is Not Satisfied
end

Chapter 5

Numerical Experiments

We elaborate on Chapter 4 with numerical experiments on the k-part dot product and k-part iterative refinement. We use the Multi-Precision Floating-Point Reliable (MPFR) Library and Linear Alegebra Package (LAPACKE) in C to supplement our experiments. The IEEE 754 standard [13, See Table 3.5] recommends we set MPFR's precision to be

$$prec = 64k - \log_2(4 \times 64k) + 13, \quad k \geq 2$$

for both time and accuracy testing.

5.1 On Constructing K-Parts Numbers

We require a method to construct lists of size k whose exact sum is equal to a given floating-point number before our numerical experiments. Suppose we are given some arbitrary floating-point number, say $c \in \mathscr{F}$. The IEEE 754 standard [13, 14] implies

$$c = (-1)^s 2^e \left(1 + \sum_{i=1}^{52} d_i 2^{-i}\right)$$

where $s = \pm 1$, $d_i = 0, 1$ for all $i = 1, 2, \ldots, 52$, and $-1022 \leq e \leq 1023$. We can decompose c into $k = 2$ parts by assigning $\{c\} = \{\frac{1}{2}c, \frac{1}{2}c\}$. Let's also consider decomposing c into $k = 3$ parts. To do so, observe

$$\begin{aligned} c &= (-1)^s 2^e \left(1 - 1 + 1 + \sum_{i=1}^{52} d_{52-i} 2^{-i}\right) \\ &= \underbrace{(-1)^2 2^e \left(1 + \sum_{i=1}^{26} d_i\right)}_{c_1} + \underbrace{(-1)^2 2^e \left(1 + \sum_{i=27}^{52} d_i\right)}_{c_2} + (-1)^{s+1} 2^e. \end{aligned}$$

The previous produces $\{c\} = \{c_1, c_2, (-1)^{s+1}2^e\}$ with $c_1, c_2 \in \mathscr{F}$. We generalize our result for any $2 < k < 52$ by first partitioning the set $P = \{1, 2, 3, \ldots, 52\}$ into $P_1, P_2, \ldots, P_{k-1}$ so that $\bigcup_{i=1}^{k-1} P_i = P$, $P_i \neq \varnothing$ for $i = 1, 2, \ldots, k-1$, and $P_i \cap P_j = \varnothing$ for $i \neq j$. Then,

$$c = \sum_{i=1}^{k-1} \underbrace{(-1)^s 2^e \left(1 + \sum_{p \in P_i} d_p 2^{-p}\right)}_{c_i} + (k-2)(-1)^{s+1} 2^e$$

and $\{c\} = \{c_1, c_2, \ldots, c_{k-1}, (k-2)(-1)^{s+1}2^e\}$. We will only split floating-point numbers into $k = 2, 3, \ldots, 8$ parts for our numerical experiments. However, future experiments with the k-parts library can use this technique to extrapolate more parts.

5.2 Dot Product Numerical Experiment

Suppose the entries of $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy $\{x_i\} = \mathrm{fl}_{k,k}(x_i)$ and $\{y_i\} = \mathrm{fl}_{k,k}(y_i)$. Then, observe

$$\begin{aligned}\left|\{s^{(n)}\} - \boldsymbol{x}^T\boldsymbol{y}\right| &\leq \left|\{s^{(n)}\} - \{\boldsymbol{x}^T\}\{\boldsymbol{y}\}\right| + \left|\boldsymbol{x}^T\boldsymbol{y} - \{\boldsymbol{x}^T\}\{\boldsymbol{y}\}\right| \\ &\leq \kappa_n \left|\{\boldsymbol{x}^T\}\right| |\{\boldsymbol{y}\}| + O(\mu^k)\left|\boldsymbol{x}^T\right| |\boldsymbol{y}|.\end{aligned}$$

The left-hand portion of the last inequality comes from Theorem 4.1.3 while the right-hand portion comes from an argument similar to Lemma 3.2.2 and Lemma 3.2.3. Using Definition 2.1.2 on the previous yields

$$\left|\{s^{(n)}\} - \boldsymbol{x}^T\boldsymbol{y}\right| \leq O(\mu^k)\left|\boldsymbol{x}^T\right| |\boldsymbol{y}|$$

and

$$\frac{\left|\{s^{(n)}\} - \boldsymbol{x}^T\boldsymbol{y}\right|}{|\boldsymbol{x}^T\boldsymbol{y}|} \leq O(\mu^k)\mathrm{cond}(\boldsymbol{x},\boldsymbol{y}), \quad \mathrm{cond}(\boldsymbol{x},\boldsymbol{y}) = \frac{\left|\boldsymbol{x}^T\right| |\boldsymbol{y}|}{|\boldsymbol{x}^T\boldsymbol{y}|}. \tag{5.1}$$

We want to find when $\mathrm{cond}(\boldsymbol{x},\boldsymbol{y})$ is large enough to cancel with $O(\mu^k)$ and make our forward-error larger than 1. One way to accomplish this is to create as many cancellations as possible in $\left|\boldsymbol{x}^T\boldsymbol{y}\right|$ so that $\left|\boldsymbol{x}^T\boldsymbol{y}\right| << 1$. Both Ogita and Yamanaka [7, 15] have devised algorithms to produce extremely ill-conditioned dot products in this manner, which we now use in our own numerical experiments.

Figure 5.1 shows the input condition number versus each dot product's relative-error. We test for k-parts ranging from $k = 2, 3, \ldots, 8$ with a condition number that ranges from 2^1 to 2^{400}. The test generates 1000 pairs of random vectors with a size of 100. We, then, construct $\{\boldsymbol{x}\}$ and $\{\boldsymbol{y}\}$, for each generated pair of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, with the technique outlined in the Section 5.1. Any relative error in computing $\{s^{(n)}\} = \mathrm{fl}_{k,k}\left(\{\boldsymbol{x}^T\}\{\boldsymbol{y}\}\right)$ that exceeds 1.0 is rounded down to 1.0 for ease of viewing. We conclude the relative-error threshold, the point in which our relative-error exceeds 1.0, aligns with the theoretical bound established in (5.1). For instance, $k = 2$ makes $O(\mu^k) \approx 10^{-32}$, so our result should only be accurate up to condition numbers of order 10^{-32}. The approximate value for the condition number that makes our relative-error exceed 1.0 is $\mathrm{cond}(\boldsymbol{x},\boldsymbol{y}) \approx 5 \cdot 10^{31}$, implying that ρ is likely on the order of $O(10)$.

Additionally, we provide a time comparison between the MPFR and k-parts dot product, as shown in Table 5.1. Our findings highlight a k-part algorithm's key weakness, their execution time. We can see that up until $k = 3$ the k-parts dot product competes with the MPFR dot product. Unfortunately, the k-parts dot product is slower than the MPFR dot product after $k = 3$.

k	Average K-Time	Average MPFR Time	Ratio of Averages
2	7.44E-06	1.62E-05	4.58E-01
3	1.63E-05	1.91E-05	8.57E-01
4	3.32E-05	2.53E-05	1.32E+00
5	6.10E-05	2.70E-05	2.26E+00
6	1.02E-04	2.76E-05	3.68E+00
7	1.55E-04	2.96E-05	5.24E+00
8	2.31E-04	3.08E-05	7.51E+00
Overall	8.65E-05	2.51E-05	3.45E+00

Table 5.1: Time Comparison of K-Parts and MPFR Dot Product

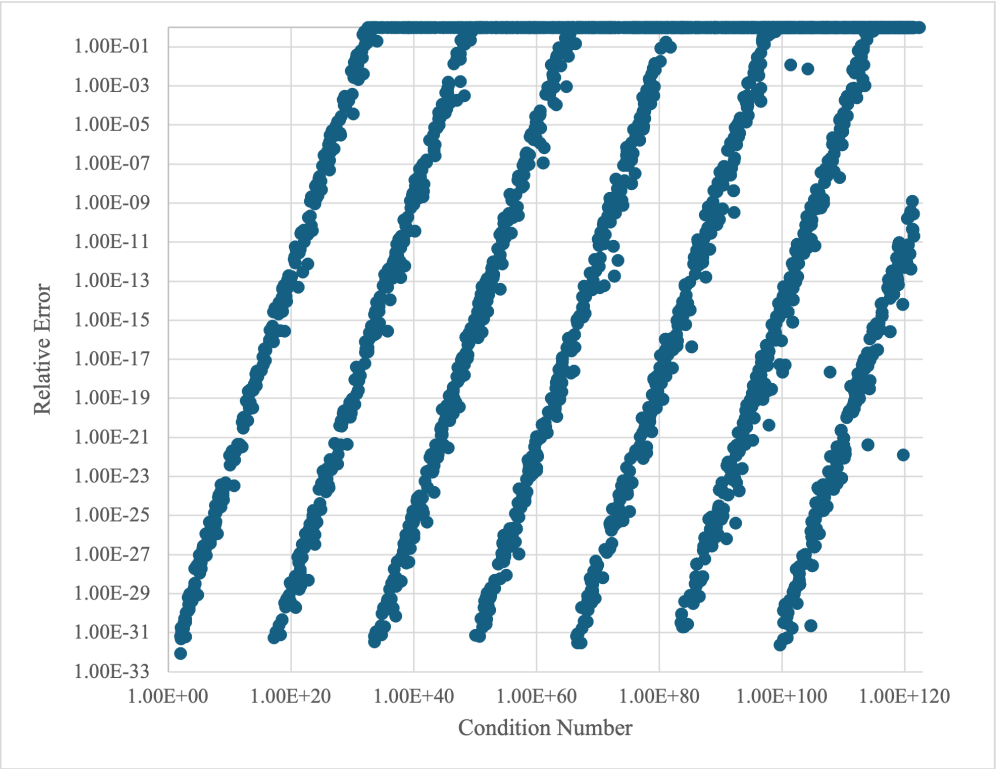

Figure 5.1: Dot Product in K-Parts with Random Ill-Conditioned Vectors

5.3 Iterative Refinement Numerical Experiments

We test our assertion in Chapter 4 against a square Hilbert Matrix. The condition number for an nxn Hilbert Matrix, which we denote H_n, grows on the order of

$$\frac{(1+\sqrt{2})^{4n}}{\sqrt{n}},$$

as can be seen in [14, Chapter 28.1]. Additionally, the Hilbert Matrix cannot be stored exactly in double-precision. We account for the previous fact by representing each entry in H_n as

$$\{h_{ij}\} = \mathrm{fl}_{k,k}\left(\frac{1}{i+j-1}\right)$$

and $\{\boldsymbol{x}\}$ as the k-parts solution to the linear system $\{H\}\{x\} = \boldsymbol{b}$. The exact solution for the linear system $H_n\boldsymbol{x} = \boldsymbol{e}_1$, where $\boldsymbol{e}_1$ denotes the vector whose entries are all zero except for the first entry, is the first column of the inverse Hilbert Matrix. The inverse Hilbert Matrix is an integer matrix [14, Chapter 28.1], so the first column of H_n^{-1} is exact in double-precision arithmetic. Similarly, we have $\boldsymbol{x}$ is the last column of H_n^{-1} in the linear system $H\boldsymbol{x} = \boldsymbol{e}_n$. We test with $\boldsymbol{e}_1$ and $\boldsymbol{e}_n$ in particular since they represent the smallest and largest column of H_n^{-1}. Table 5.2 shows the 1-norm relative error between the exact integer solution to $H_{50}\boldsymbol{x} = \boldsymbol{e}_1$ and the computed solution to $\{H_{50}\}\{\boldsymbol{x}\} = \boldsymbol{e}_1$.

The condition number for this experiment is on the order of $O(5.077 \cdot 10^{75})$. As such, we expect the relative-error to exceed 1.0 for $\mu^{-k} \leq 5.077 \cdot 10^{75}$. This would be the case when $k = 2$, $k = 3$, and $k = 4$ as demonstrated in Table 5.2. We see a similar result in Table 5.3 as well. The lower-bound on the relative error, in both cases, is likely a result of the round-off error generated by computing the forward-error between $\{\boldsymbol{x}\}$ and $\boldsymbol{x}$. Lastly, note how iterative refinement produces a less accurate answer than GEPP in this experiment.

k	GEPP Relative Error	Refinement Relative Error
2	1.00E+00	1.00E+00
3	1.00E+00	1.00E+00
4	1.00E+00	1.00E+00
5	1.08E-09	4.36E-09
6	2.90E-16	2.90E-16
7	2.90E-16	2.90E-16
8	2.90E-16	2.90E-16

Table 5.2: Relative Error of GEPP and Iterative Refinement with Hilbert Matrix and Vector $\boldsymbol{e}_1$

k	GEPP Relative Error	Refinement Relative Error
2	1.00E+00	1.00E+00
3	1.00E+00	1.00E+00
4	1.00E+00	1.00E+00
5	1.16E-09	2.72E-09
6	6.68E-16	6.68E-16
7	6.68E-16	6.68E-16
8	6.68E-16	6.68E-16

Table 5.3: Relative Error of GEPP and Iterative Refinement with Hilbert Matrix and Vector $\boldsymbol{e}_n$

We also tested our k-parts iterative refinement and GEPP with the floating-point representation of matrices generated by a Singular-Value Decomposition (SVD). We first state a desired condition number, and then construct the matrix [14, See Section 28.3]

$$A = U\Sigma V^T$$

so that U and V are from a Haar distribution and $\Sigma = \text{Diag}(\sigma_i)$. We set $\sigma_i = 1$ for $i = 1, \ldots, n-1$ and set σ_n equal to the reciprocal of the condition number. This construction of our SVD matrices is subject to round-off error, so we can only report a reliable condition number up to 10^{16} for this experiment. We also use MPFR to approximate the condition number of these systems and their exact solution. Figure 5.2 shows the relative error of GEPP and Figure 5.3 of iterative refinement with matrices whose dimension are $n = 10, 11, 12, \ldots, 100$. Both figures demonstrate how GEPP can provide an inaccurate solution, in respect to our theoretical error-bounds, but can be corrected with iterative refinement.

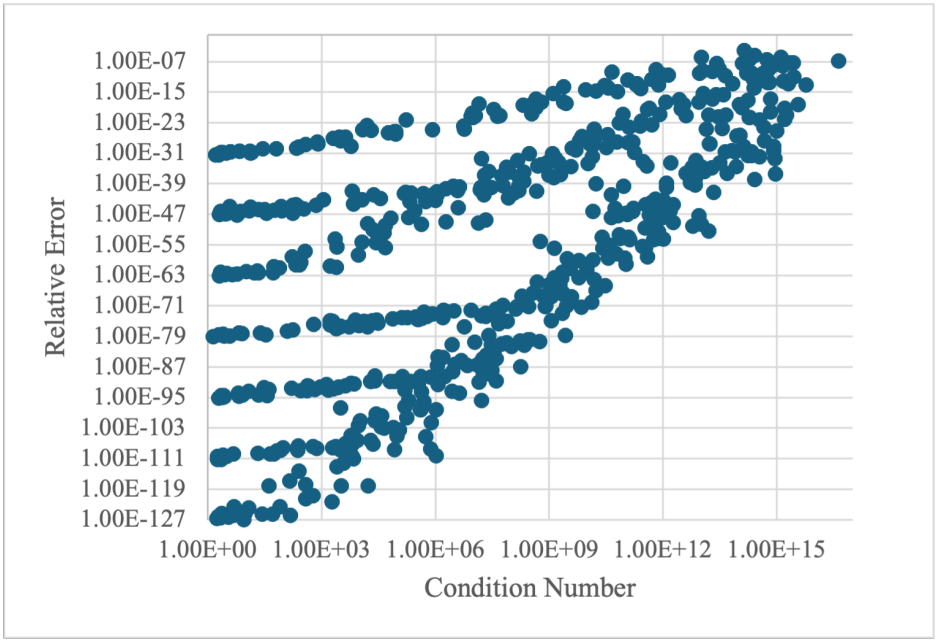

Figure 5.2: GEPP in K-Parts with Singular Value Decomposition Matrices

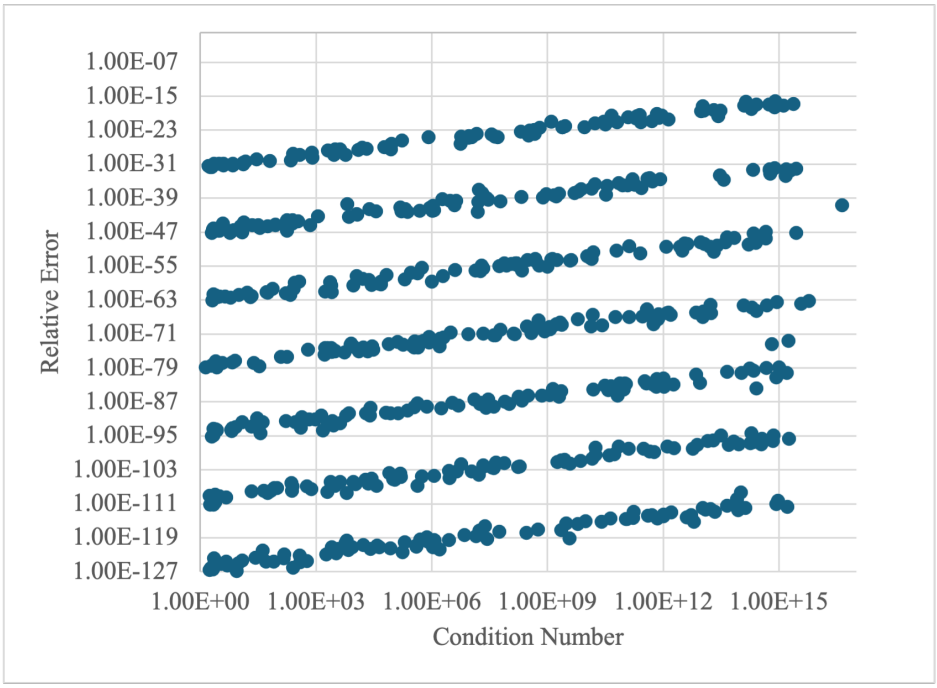

Figure 5.3: Iterative Refinement in K-Parts with Singular Value Decomposition Matrices

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis began with an exploration of the history behind compensated arithmetic. We, then, summarized the floating-point model, introduced Error Free Transforms (EFTs), and provided a literature review of compensated arithmetic research in Chapter 2. In Chapter 3, we leveraged Rump's [8] work to develop addition, multiplication, fuse-multiply-add (FMA), and Newton-Raphon division algorithms whose output satisfies Definition 2.1.2. Each algorithm, in essence, uses addition and multiplication EFTs to build a large error vector. Then, we filter the error by successively applying VecSum to the error vector, storing the last entry of its output vector in our k-parts number. Furthermore, we generalized our findings to show both arbitrary floating-point collections and floating-point collections with Definition 2.1.2 can be used in a k-parts arithmetic algorithm to produce an output satisfying Definition 2.1.2.

Chapter 4 explores backward-error analysis with k-parts numbers. We demonstrated that any k-parts floating-point collection can be represented as a perturbation of its operation, similar to how operations are represented in the floating-point model. We used the k-parts FMA to construct a dot product whose result upholds the forward-error inequality in Definition 2.1.2. We confirmed this by using our k-parts perturbation model to derive a backward-error formula. A consequence of our analysis was the creation of κ_n, an analogue to the γ_n defined for the floating-point model. We further applied our perturbation toolkit to the analysis of a backward substitution solver for upper-triangular systems. The relationship between the backward-error of the triangular system and the norm-wise forward-error of its solution was presented as well. Chapter 4 concludes with a brief discussion on an applying k-parts arithmetic to the accurate solution of a linear system.

We fleshed out our theoretical results with numerical experiments in Chapter 5. To do so, we constructed a method to decompose a floating-point number into k-parts by manipulating the number's IEEE 754 representation. We, then, conducted a numerical experiment with the dot product, where we use the aforementioned method to make each entry in the k-parts vector. The result of the dot product experiment highlighted the relationship between the value of k and the condition number in which the relative-error exceeded 1.0. We continued our numerical experiments by testing our k-parts GEPP and iterative refinement with the Hilbert Matrix. The Hilbert Matrix experiment was subject to round-off error complications that caused the iterative refinement algorithm to produce a worse relative-error than GEPP. Although, matrices generated by a Singular-Value Decomposition (SVD) produce the desired k-fold result after iterative refinement.

6.2 Future Work

The original goal of this research project was to develop a dominant eigenvalue method whose output is as accurate as if computed in k-fold precision and stored in k-parts. Even though we did not develop an eigenvalue method, the arithmetic operations devised in Chapter 3 give us the tools to develop such a solver in future work. In Chapter 4, we provided k-part analogues to the floating-point analysis made by Higham [14], showcasing how Higham's work can be extrapolated with k-parts perturbation analysis. Future work in this field ranges from floating-point interpolation to solving nonlinear systems of equations.

Moreover, the development in Chapter 4 can be expanded upon to explain the numerical results in Chapter 5. For instance, why does the Hilbert Matrix cause our k-fold iterative refinement to produce a worse forward-error than GEPP? What causes does iterative refinement to compute an accurate solution, in respect to our theoretical error-bounds, to linear systems generated by a SVD? Can we account for floating-point representation errors by computing their entries to satisfy Definition 2.1.2? Future work may also involve a formal approach to the errors discussed in Chapter 4.

Bibliography

[1] Thomas R. Cameron and Tim Chartier. Finite precision in an infinite world. *Math Horizons*, 27(1):12–14, August 2019.

[2] T. J. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, June 1971.

[3] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991.

[4] W. Kahan. Pracniques: further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, January 1965.

[5] Donald E Knuth. *The art of computer programming: Fundamental algorithms v. 1*. Addison-Wesley Educational, Boston, MA, February 1968.

[6] Ole Møller. Quasi double-precision in floating point addition. *BIT*, 5(1):37–50, March 1965.

[7] Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, January 2005.

[8] Siegfried M. Rump. Inversion of extremely ill-conditioned matrices in floating-point. *Japan Journal of Industrial and Applied Mathematics*, 26(2–3):249–277, October 2009.

[9] Stef Graillat and Valérie Ménissier-Morain. Accurate summation, dot product and polynomial evaluation in complex floating point arithmetic. *Information and Computation*, 216:57–71, July 2012.

[10] Thomas R. Cameron and Stef Graillat. On a compensated ehrlich-aberth method for the accurate computation of all polynomial roots. *ETNA - Electronic Transactions on Numerical Analysis*, 55:401–423, 2022.

[11] Thomas R. Cameron and Stef Graillat. Accurate horner methods in real and complex floating-point arithmetic. *BIT Numerical Mathematics*, 64(2), March 2024.

[12] Yves Nievergelt. Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit. *ACM Transactions on Mathematical Software*, 29(1):27–48, March 2003.

[13] IEEE/ANSI. IEEE standard for floating-point arithmetic. Technical report, IEEE/ANSI, Piscataway, NJ, USA, 2019.

[14] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2 edition, 2002.

[15] N. Yamanaka, T. Ogita, S.M. Rump, and S. Oishi. A parallel algorithm for accurate dot product. *Parallel Computing*, 34(6–8):392–410, July 2008.