UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ERROR PROPAGATION DUE TO APPROXIMATIONS IN SOFTWARE PIPELINES

Diploma Thesis

Dimitrios Samakovlis

Supervisor: Christos Antonopoulos

Volos 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ERROR PROPAGATION DUE TO APPROXIMATIONS IN SOFTWARE PIPELINES

Diploma Thesis

Dimitrios Samakovlis

Supervisor: Christos Antonopoulos

Volos 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΑΔΟΣΗ ΣΦΑΛΜΑΤΩΝ ΛΟΓΩ ΠΡΟΣΕΓΓΙΣΤΙΚΟΥ ΥΠΟΛΟΓΙΣΜΟΥ ΣΕ PIPELINES ΛΟΓΙΣΜΙΚΟΥ

Διπλωματική Εργασία

Δημήτριος Σαμακοβλής

Επιβλέπων: Χρήστος Αντωνόπουλος

Βόλος 2021

Approved by the Examination Committee:

Supervisor	**Christos Antonopoulos** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Spyros Lalis** Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Panagiota Tsompanopoulou** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: 8-7-2021

Acknowledgements

I would like to thank my supervisor, Associate Professor Christos Antonopoulos, for the support and guidance he provided me throughout the whole process of the Thesis development, as well as for his contribution to the selection of the subject. Your expertise was invaluable in formulating the research methodology, while your systematic approach and immediate response to any arising issue saved me unproductive time and simultaneously inspired me to become more professional.

Abstract

My diploma thesis explores the area of Approximate Computing (AC) and, in particular, attempts to model the propagation of errors that arise from approximation techniques, mainly precision scaling. We use two methods to model the approximation effect, namely Error Injection (EI) and Lossy Compression (LC). Both techniques target the distortion of the input data of the operation under test. EI consists of introducing error to data, using the normal or the uniform distribution, and LC includes compression and decompression of data using two popular compressors in High-Performance Computing (HPC) datasets, SZ (Sneeze) and ZFP. We conducted experiments on multiple applications like stereo vision matching (SPS-Stereo), Matrix Multiplication (MM), Singular Value Decomposition (SVD), and Conjugate Gradient (CG). The analysis of the error patterns highlights the complexity of simulating the precision scaling approximations, albeit the success of the models in achieving the desired metrics of quality. Nevertheless, we gain insight into the quality distortion and applicability of SZ and ZFP through the experiments.

Table of contents

Abbreviations

HPC	High Performance Computing
AC	Approximate Computing
EI	Error Injection
LC	Lossy Compression
CG	Conjugate Gradient
MM	Matrix Multiplication
SVD	Singular Value Decomposition
PSNR	Peak Signal-to-Noise Ratio
CR	Compression Ratio
SGM	Semi-Global Matching
SPS	Slanted Plane Smoothing

Chapter 1

Introduction

As large-scale software has been developed for various domains, such as scientific computing and social media, it is evident that computational and storage requirements of computer systems are continuously growing, resulting in excessive power consumption [1]. Moreover, it is expected that the information managed by those applications will increase at a much higher rate than the computational resources will [2].

Those facts suggest that over-provisioning of computing resources is about to pose a great challenge for the computer industry, rendering the call for energy-efficient computing stronger than ever [3]. Taking the above into consideration, and as traditional energy gains derived from the Moore's law are slowly diminishing [4] , computer engineers are obliged to seek for alternative and creative solutions.

Approximate Computing(AC), based on the observation that not every part of the computations requires high precision to achieve the desired target, offers a viable solution [5]. Many computationally expensive algorithms, usually derived from the area of data analytics, scientific computing, machine learning or computer vision, have been proven to produce satisfactory results with approximate implementations that can achieve energy gains of up to 50 times by only sacrificing 5% quality loss [6], [7].

In the last two decades, numerous approaches have been proposed in the context of AC. In general, approximation can be introduced at multiple levels, like algorithmic-, coding-, architectural- or microarchitectural- level, depending on the nature of the application. To that extent, a lot of research has been conducted to explore the potential of building approximate frameworks in error resilient applications, in order to trade off accuracy for energy or performance , while still maintaining acceptable quality of results [5].

Despite the great amount of work in the AC area, rigorous methods have not been established to quantify the effect of specific approximations , but rather the approaches lie on trial and error and on expert knowledge on the specific domain. Indeed, approximation is inherently application-specific and it is tricky to generalize the effect of a specific technique to a broader range of application domains. However, analyzing the error induced by an approximation and the way it is propagated in the computational pipeline is of high importance when evaluating the applicability of an approximation.

That said, we believe that more research should be conducted in the direction of understanding the propagation of error due to approximations. On top of that, we believe that a methodology capable of modeling the effect of approximation techniques, would be highly appreciated by the AC community, as it would guide future software developers in the decision-making process of selecting a suitable approximation method, by means of providing fast insights on the effects of the quality of the final result. In this sense, we investigate methods capable of modeling software approximations.

1.1 Scope of Thesis

The purpose of this work is to explore methods that can reliably model the effect of approximations at the software level. Our method of work is summarized conceptually in Fig. 1.1. On the top row, it depicts the fully accurate pipeline, which is essentially the original implementation of the algorithm using accurate computations. In the middle, there is the approximate version, which uses some approximation technique at coding- or algorithmic-level. Finally, at the bottom row lies our modeling framework, which is based on the accurate execution of the distorted data that have been provided as input to the approximate pipeline.

Both the approximate version and our modeling approach induce error relative to the fully accurate calculated results. Our work essentially analyzes approximate and modeling error patterns, in a bid to pinpoint which parts of the final result have been affected and to what extent. Moreover, metrics of the quality of the final result for the two approaches are directly compared. Evaluating both the error patterns and metrics of quality, we can infer the efficiency of our modeling approaches, in particular how accurately the effect of the approximation has been replicated or what insight about the distortion of quality has been gained.

Our modeling approach focuses on two main mechanisms, namely Error Injection (EI)

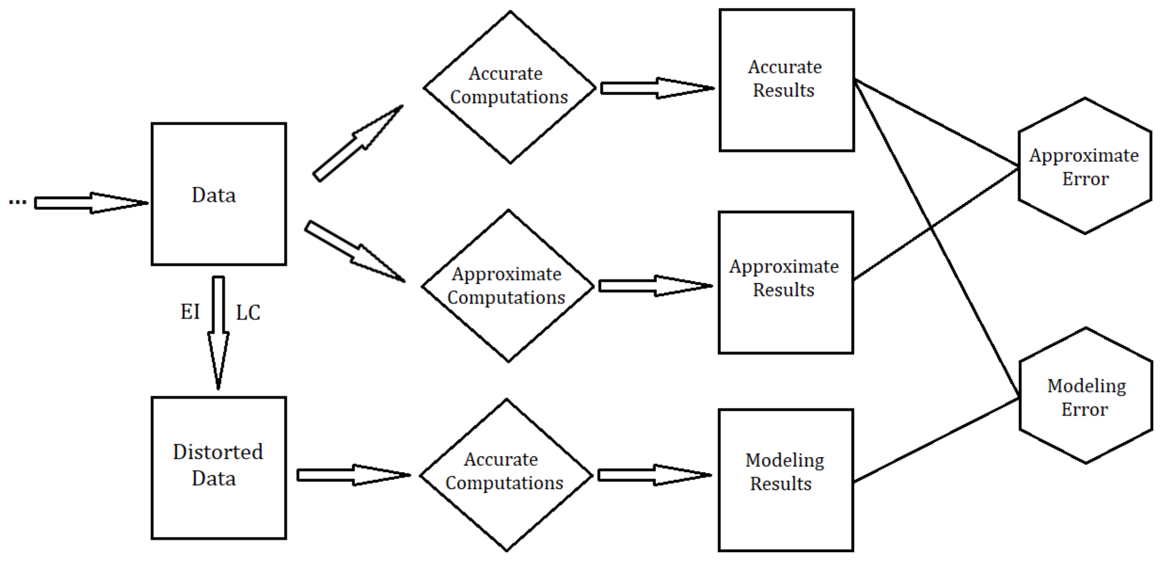


Figure 1.1: Conceptual schema of our work

and Lossy Compression (LC), applied to a variety of applications and domains. Both methods are based on the idea of distorting the data that are affected by the approximation technique. We expect the propagation of error from the distorted data to simulate the error derived from the approximation.

The EI method consists of adding error specified by two well-known statistical distributions, normal and uniform distribution, both centered at 0 but configurable regarding their standard deviation and range respectively. It should be noted that the results derived from EI are not reproducible and exhibit slight variability due to the randomness of the process. However, the effect of a specific configuration tends to follow a certain pattern which renders the EI a firm modeling method.

On the other hand, the LC mechanism lies on the idea that after compressing and decompressing a data structure using a lossy compressor, part of the information cannot be retrieved, resulting in data distortion. Consequently, LC can be used in a similar way to the EI, but rather than relying on statistical distributions, it is based on spatial continuity to remove redundancy and induce error-bounded distortion. The lossy compressors used are SZ and ZFP (see Chapter 2). Both of them, require an error metric to be specified to bound the error amongst the original value and the decompressed value of an element.

The experiments consist of applying the aforementioned methods, with different configuration parameters, on the data structure that is given as input to the approximate pipeline.

Practically, our work attempts to find out if there is a configuration of the parameters of EI and LC which can simulate the approximation and how to specify the correct parameters for a variety of input data.

We experiment with our models in various domains. The applications examined are:

- SPS-Stereo, a Stereo Vision Matching algorithm
- Matrix Multiplication of sparse matrices
- Singular Value Decomposition of sparse matrices and images
- Conjugate Gradient for solving linear systems with sparse matrices

The main approximation technique explored is precision scaling, which consists of using fewer bytes for the representation of real numbers and limits computational and storage demands. It has been extensively used in the literature [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. It is one of the most straightforward approximations, that can be employed both at the software and hardware level. We focus on applications that use double floating-point precision and experiment with single precision, half-precision, and in some cases, we even use integers in place of real numbers. All applications listed above, apart from the SPS-Stereo, are eligible to experiment with precision scaling.

In all cases, results are visualized using heat maps representing the absolute or relative difference in values amongst the fully accurate result and the result of each approximate method or modeling mechanism. Moreover, domain-specific metrics, such as the Frobenius norm and condition number for matrices or PSNR for images, are employed to assess the quality degradation of the final result and directly compare the modeling approaches with the approximate version.

Since performance is not of interest, we conduct all experiments on a desktop with CPU Intel i5 10500 (6 cores - 12 threads), GPU Nvidia GeForce GTX 1660 Super, and 16 GB of dual RAM. We develop software using C++, while we use python to produce the heat maps and the graphs. For the error injection framework, we utilize a fast parallel library which employs OpenMP [18], while for the lossy compressors we use the official Github CPU implementations. Extra libraries used are stated explicitly in each chapter.

1.1.1 Contribution

The contribution of this Thesis is two-fold. First, it offers an approach to model the effect of existing approximation techniques, which is a new approach in the context of Approximate Computing. Second, the analysis of the experiments provides valuable insight on the lossy compressors SZ, ZFP which are widely used in the High Performance Computing area to substantially reduce the footprint of the data and consequently improve the performance of applications.

1.2 Structure of paper

The rest of the Thesis is organized as follows. Chapter 2 provides an overview of the background needed to understand the methods employed. In particular, how to apply the frameworks, and the relevant metrics we use to evaluate the quality of results. Each subsequent section discusses a different application case study. Chapter 3 analyzes the SPS-Stereo matching algorithm and Chapter 4 refers to Matrix Multiplication. Chapters 5 and 6 examine the Singular Value Decomposition and Conjugate Gradient respectively. Finally, chapter 7 summarizes the Thesis and highlights the main observations.

Chapter 2

Background

2.1 Error Injection

Error Injection (EI) is the process of deliberately introducing error to computations or data. One potential application of EI is to study the sensitivity of a function or algorithm to the distortion of the input. In our case, we want to test whether the added error, which is derived from a statistical distribution, can simulate the effect that an existing approximation technique has in the final results. In practice, we inject error to the input data of the operation we test as illustrated in Fig. 2.1.

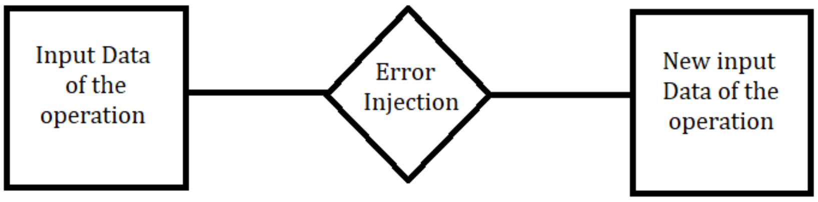


Figure 2.1: Error Injection framework

In our experiments we use normal and uniform distributions for the error. Normal distribution is used widely in practice to simulate noise, namely Gaussian noise. Normal distributions used here are defined as:

$$N(0, \sigma^2)$$

which means they have a mean of 0 and a standard deviation of σ. Similarly, the uniform distribution is used as:

$$U(-a, a)$$

meaning that they are centered around 0 and are limited to a range of $b = 2a$. Each value inside that space has an equal probability. For abbreviation, the normal distribution for the rest of the work will be defined by its standard deviation σ and the uniform by its range b. Both distributions are illustrated in Figure 2.1 for $\sigma = 1$ and $b = 1$.

It should be noted that for our experiments, we utilized a fast C++ library to produce pseudo-random numbers following the aforementioned distributions. The omprng library [18] employs a parallel OpenMP implementation which is crucial for performance efficiency when running multiple experiments.

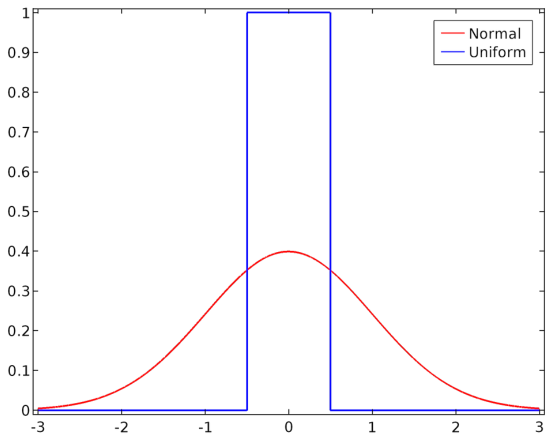


Figure 2.2: Power density function of $N(0, 1)$ and $U(-0.5, 0.5)$

2.2 Lossy Compressors

Lossy compressors have been widely used in the last decade to reduce the footprint of HPC datasets and thus speed up the intensive transfer of big data [19]. In contrast to a lossless compressor, a lossy compressor induces data distortion in the reconstructed or so-called uncompressed data, as shown in Fig. 2.3. The distortion of the original data is usually bounded in terms of error magnitude. On the other hand, lossy compressors can achieve much higher compression ratios than the state-of-the-art lossless compressors, a fact that makes them ideal in frameworks where the data transfer is the bottleneck of performance.

In this work, we examine whether the data distortion a lossy compressor induces after the decompression, can act similarly to the EI concept and form a firm model of the approximation error. We focus on two popular and successful lossy compressors, SZ and ZFP. Both have been proved to provide high compression ratios in spatially correlated data across multiple domains while preserving a user-specified metric of quality [19].

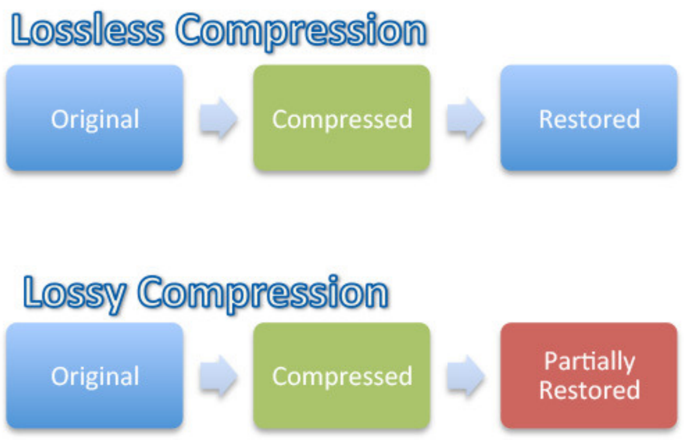


Figure 2.3: Difference between lossy and lossless compression

Fig. 2.4 illustrates the LC framework we employ in our work. First, we write the data of the program in a binary file. Then SZ or ZFP is employed via its terminal mode to compress and directly decompress the binary file into a new binary file. Finally, we read back the distorted data from the new binary file.

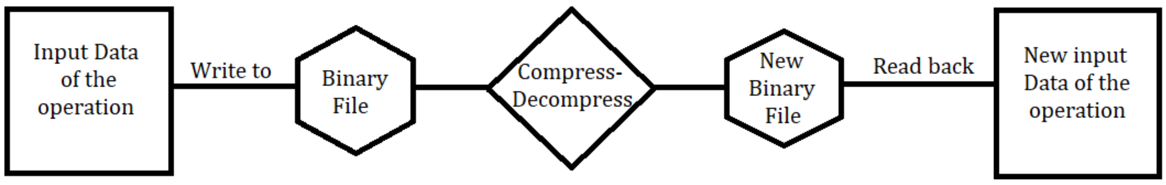


Figure 2.4: Lossy Compression framework

A variety of quality metrics are provided to the user by both compressors. Commonly used are the point-wise relative error, relative error, absolute error, and PSNR. In this work, we mainly focus on the simplest, which is the absolute error. However, in few cases where a different metric is employed, it will be explicitly clarified.

In the analysis below we briefly summarize the main concept and features of the two lossy compressors. It should be noted that the scope of this work does not focus on the performance aspect but rather on the distortion of the data when applying compression and decompression. Practically, we do not explore the applicability of the compressors in the context of AC, but rather target the interpretation of the approximation error.

2.2.1 SZ

The fundamental concept of Squeeze (SZ) compression is that it exploits locality by employing prediction curves on subsequent data. The initial release is thoroughly analyzed in [20]. Initially, SZ transforms multidimensional data into a 1D sequence. Then, predictable data points are replaced by two bits, defining one of three curve-fitting models, which pass through the preceding data points. On the contrary, unpredictable data points are normalized to be closer to zero, their mantissa bits are truncated and leading-zero based floating-point compression is applied on them. In both cases, the user-specified error bound is strictly respected.

Subsequent releases of SZ came with many improvements. The prediction model has been extended to work on multidimensional data using a multilayer approach, significantly increasing the percentage of predictable data points, while new strategies like quantization and Huffman coding have been implemented in [21]. Partitioning of data into blocks, new prediction models for larger error bounds, and optimization techniques in model selection have been introduced in [22]. The current version, which we use in our experiments, is 2.1 and is available on Github [23].

2.2.2 ZFP

ZFP follows a different principle than the prediction-based SZ, namely the transform-based approach. The procedure is summarized in [24]. First, the data are partitioned into 4^d - sized blocks, where d is the dimension of the data, and are converted to fixed precision using a common exponent, that of the largest value. As a result, all values are in the range of (-1,1). Then a reversible orthogonal transform is applied, changing the basis of representation and producing coefficients with small magnitude. Finally, embedded coding is applied to the coefficients, producing a stream of bits with decreasing significance, thus allowing truncation with respect to the desired error bound. The code we used is available on Github [25].

2.3 Metrics

Apart from the heat maps of the error provided in almost all applications, there are some metrics that are widely used to assess the data distortion. Obviously the nature of each application suggests a different metric to evaluate data distortion. However, we focus on the most frequently used metrics for each case, which are the Frobenius norm for matrices and the PSNR for images. Moreover, regarding the LC application the compression ratio establishes an expressive metric for the redundancy of the data.

2.3.1 Frobenius Norm

The Frobenius norm is an element-wise norm widely used to assess the size of the elements of a matrix. Let A be an mxn matrix, then the Frobenius norm is defined as:

$$||A||_{frob} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}$$

In our case, we use it to quantify the size of error between the original matrix A and its' approximate version A', by calculating $||A - A'||_{frob}$. In the same manner, we use it to calculate the error between the accurate result matrix and our models' result matrices.

2.3.2 PSNR

Peak signal-to-noise ratio (PSNR) is a popular metric for evaluating the quality of signals, like images and videos. It expresses the ratio between the maximum possible power of a signal and the power of noise. This ratio can be used as a quality measurement between the original and a distorted image. The higher the PSNR, the better the quality of the compressed, or reconstructed image. In our case, we use it on grayscale images where the bit depth is 8 bits.

PSNR is usually calculated as a logarithmic quantity using the dB scale. Let I be the original $m \times n$ image and K be the distorted one. Then:

$$PSNR = 20 \times log_{10}(255) - 10 \times log_{10}(MSE),$$

where

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

MSE is the mean squared error and 255 is the maximum possible pixel value. Typically, PSNR values between 30 and 50 are acceptable for compression of such images [26].

2.3.3 Compression Ratio

The compression ratio (CR) of a lossy compressor is equal to the ratio of the size of uncompressed data to the size of compressed data. Taking bytes as the measure unit it is defined as:

$$CR = \frac{bytes\, of\ uncompressed\, data}{bytes\, of\ compressed\, data}$$

Evidently, a high CR indicates strong spatial coherency in data and is a verification that the compressor is efficient. However, one must take into account that the looser the error bounds specified to a lossy compressor, the higher the CR can be. Finally, it is a metric that can be directly used to compare the efficiency of different compressors.

Chapter 3

Stereo Vision Matching - SPS

3.1 Introduction

Stereo vision matching is a very popular and important application in the computer vision domain. It is used for extracting distance information from a pair of stereo cameras and is utilized in autonomous systems and image reconstruction. SPS - Stereo (SPS) [27] is a popular approach to calculate the disparity image, flow estimation and perform occlusion labeling and joint segmentation. It combines the Semi-Global Matching (SGM) [28] and slanted plane algorithm which assumes that the 3D scene is piece-wise planar and the motion is rigid or piece-wise rigid.

In our work, we focus only on the disparity image, which essentially is the image representing the distances of every point in the left image with relevance to the right image. Since objects with large distance between the two captures are closer to the camera, the depth of an object is essentially represented in the disparity image. The challenge of calculating disparity is to identify the same object in two different captures of the scene.

SGM has been proved to efficiently provide an initial estimation of the disparity image and is the part of the algorithm which is interesting to us. Fig. 3.1 shows the main part of the SGM pipeline. In short, SGM is based on the idea of pixel-wise matching and cost aggregation across multiple optimization paths in the image. Pixel-wise matching is performed using the sobel filter to highlight the edges and using a bitmap of a square window around a pixel to form a similarity metric. This provides an initial disparity estimation.

Given the intial disparity estimation, cost aggregation is performed by means of summing minimum path costs across multiple 1D scanlines while taking into account a penalty factor to deal with noise issues. Mathematically this optimization task is formulated as finding the disparity image with the minimum global energy metric. In practice, this means finding the globally optimum path for each pixel. On top of that, it is interesting that this process is executed twice , each time using a different image as base and as matching, and the final disparity image is derived from a consistency check amongst the two runs.

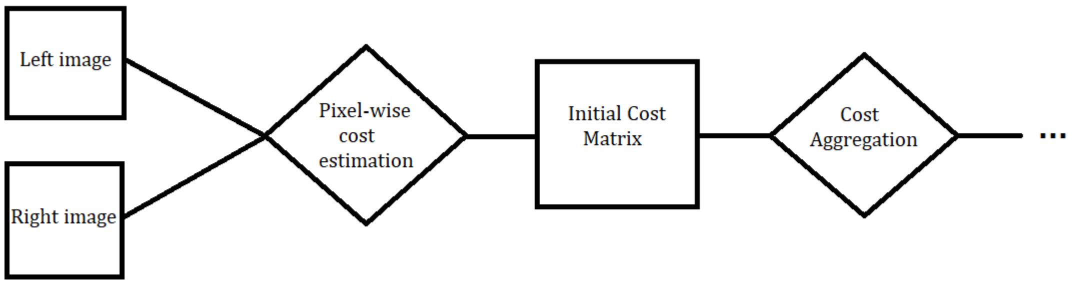


Figure 3.1: SGM pipeline

In the context of AC, the work of AcHEe [29] showcases how an approximate parallel implementation in the cost aggregation of the SGM algorithm can contribute to performance gains while maintaining acceptable disparity image quality. The approximate version suggests substituting the global minimum along multiple scanlines with a local minimum in the cost aggregation process, which eliminates synchronization and thus allows a massively parallel implementation, in this case a GPU implementation.

In this work, this approximation is replicated in CUDA C programming on top of a C++ SPS implementation publicly accessible on Github [30]. The purpose of our work is to model the effect of this approximation on the final disparity image with EI and LC.

3.2 Methodology

To model the effect of the approximation, errors are induced in two intermediate matrices derived from the cost aggregation process. Each matrix, which corresponds to the initial cost matrix in Fig. 3.1, is derived from each run. Those matrices contain values in the range of 0 to 2^{16}, as they express a 16-bit pixel representation. They are composed of integer values, hence of the two lossy compressors we can only utilize the ZFP compressor.

Regarding the LC methodology, we employ the LC framework once for each matrix, before continuing the execution with the distorted data. The fixed precision mode used in

ZFP, refers to the number of bits of the decompressed values [31]. On the other hand, EI is applied directly on the matrix via a function call with suitable parameters.

3.2.1 Parameter Configuration

After running multiple experiments and analyzing the error patterns and scale, we concluded to the configuration of the models with the most accurate results. The parameters used for our models are illustrated in Table 3.1.

Table 3.1: Parameter Configuration

Method	Parameters
ZFP	2D mode - fixed precision of 26 bits
Normal	$\sigma = 512$
Uniform	$b = 1024$

3.3 Evaluation

The results of the experiments highlight the ability of EI and LC to estimate the approximation of local minimum cost aggregation of AcHEe, by inducing artifacts on the boundaries of objects. Fig. 3.2 depicts the accurate disparity image, Fig. 3.3 the approximate disparity image, while Fig. 3.4 and Fig. 3.5 show the results of normal distribution EI modeling and ZFP modeling respectively. Albeit not replicating accurately the artifacts, both EI and LC can be useful in pinpointing the areas of the final disparity image that will be affected, which have been reported to primarily be the object boundaries in [29].

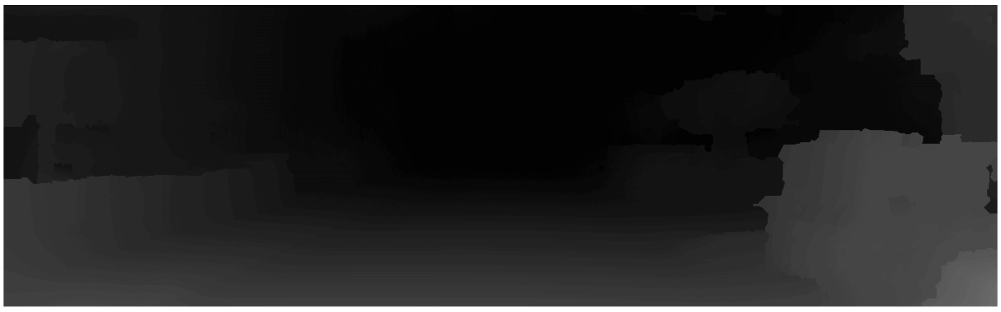


Figure 3.2: Fully accurate

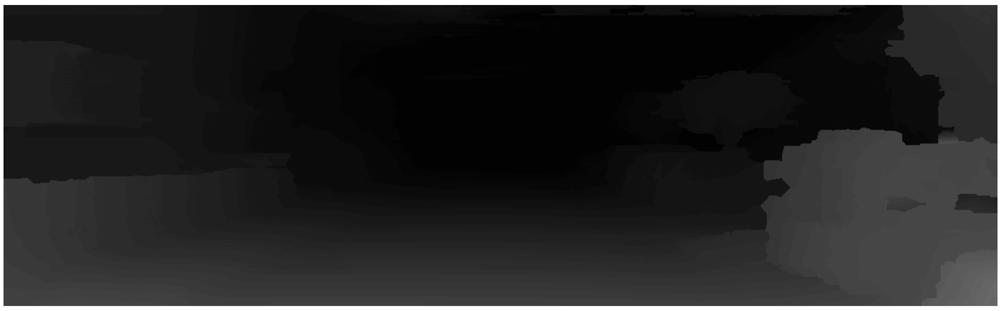


Figure 3.3: AcHEe approximate

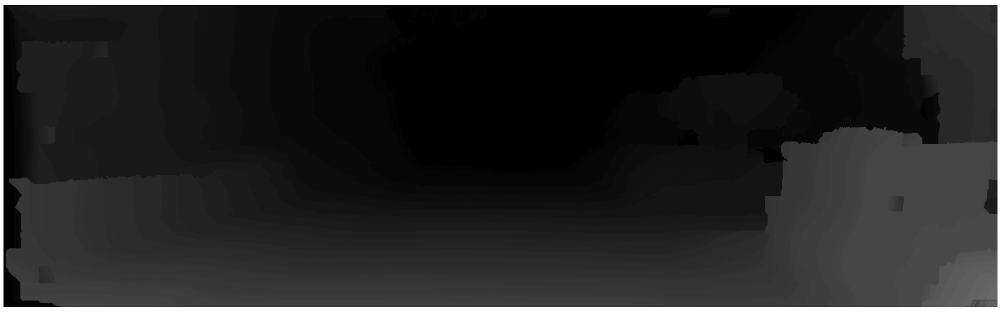


Figure 3.4: Normal distribution approximate

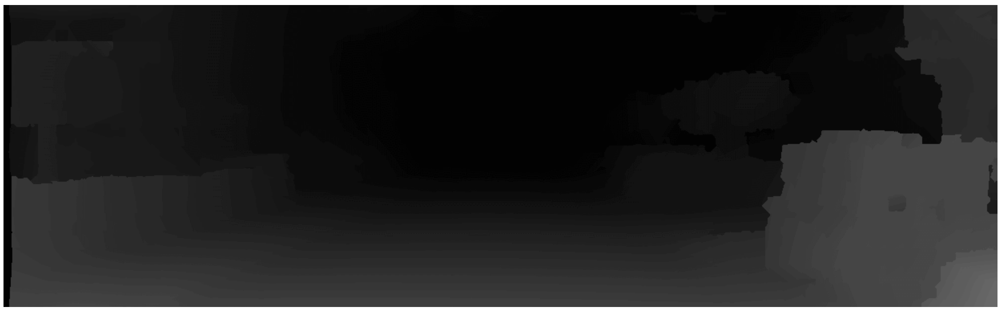
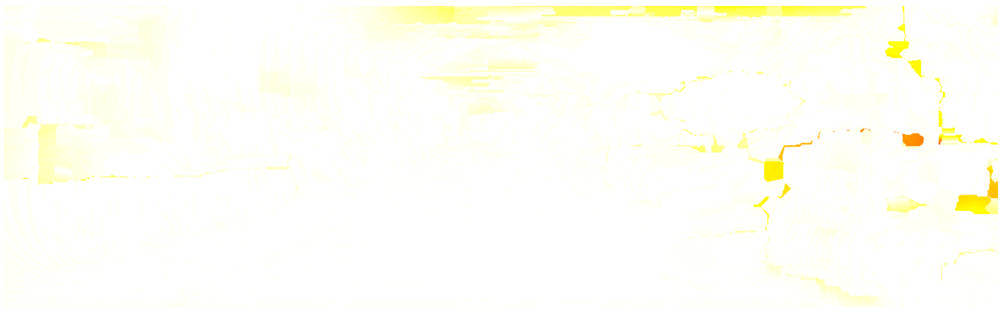
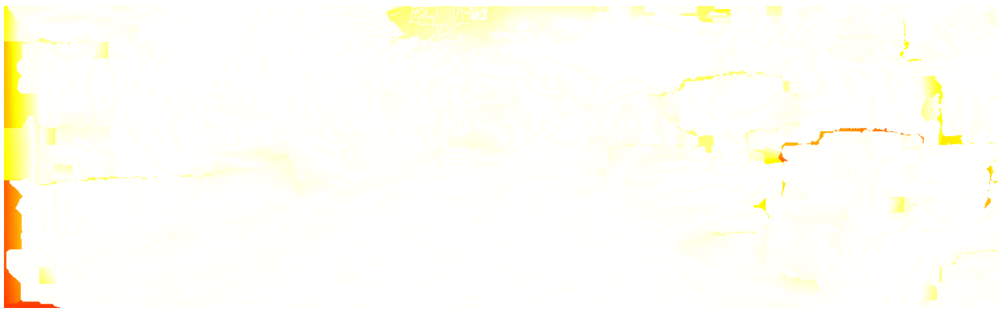


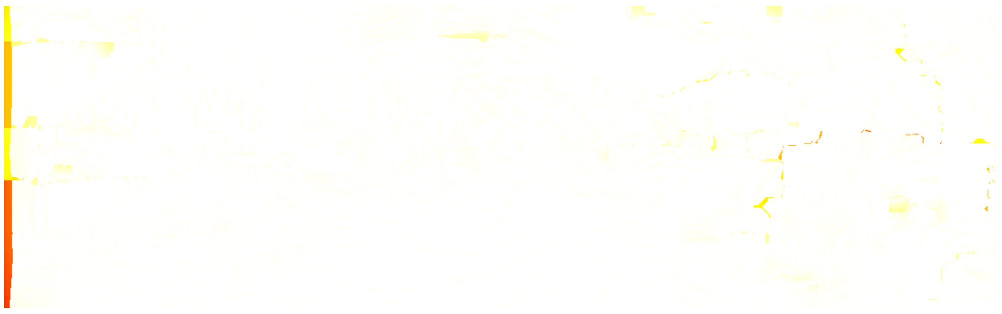
Figure 3.5: ZFP approximate

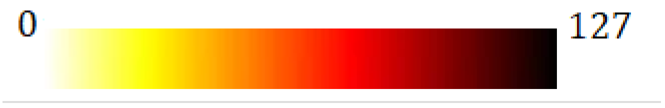
For better understanding, heat maps relative to the fully accurate disparity image have been provided in Fig. 3.6. Using an 8-bit per pixel representation, the heat maps are constructed by the absolute pixel differences, and the scale is shown in Fig. 3.6iv. Fig. 3.6i shows the heat map of the approximation, Fig. 3.6ii the heat map of the normal distribution EI method and Fig. 3.6iii the heat map of the ZFP method. Uniform distribution has been omitted for brevity, as it has fairly similar effect to the normal distribution on the final result.

To interpret the results, one must first identify the parts of the disparity image which have been distorted by the initial approximation (Fig. 3.6i). The parts of the image that have been affected by the initial approximation are the borders of the objects at the right side of the image and to a lesser extent the top of the image, which essentially is the sky. Finally, there


(i) AcHEe heat map


(ii) Heat map of normal distribution


(iii) Heat map of ZFP



(iv) Scale of heat maps

Figure 3.6: Absolute error to accurate disparity image

is a small amount of distortion at the left side of the image.

Observing the normal distribution results (Fig. 3.6ii), all of the areas mentioned above have also been affected , but in a different manner. Admittedly, the artifacts are not the same, however the EI does a decent job in finding all the areas of the image that are prone to errors. The problem with this approach is that it induces extreme distortion at the left side, which is the part where the cost aggregation process starts.

Regarding the ZFP usage, the distortion is more conservative compared to the EI methods. For example, the ZFP final disparity image (Fig. 3.6iii) shows a very large band of distortion on object borders at the right side of the image, while it exhibits negligible distortion at the top of the image. That said, it is interesting to note the compression ratio achieved by the compressor, which is 10.2 . Tackling 2D integer data with spatial continuity suggests high compression potential and that can be further improved with efficient preprocessing and a different mode, as it will be shown in chapter 5.

3.4 Challenges

In spite of the positive observations, we cannot neglect the negative side. Both EI and LC exhibited sensitivity to extreme distortion of the final image on the left boundary, as evident in Fig. 3.6ii, 3.6iii). This phenomenon is more extreme in some pairs of images and limited to others and may as well not be limited only to the left side of the image. To deal with this, we provide a simple approach that proves efficient.

We introduce a simple yet efficient solution for this problem. However, including this improvisation in the generic framework requires further work as it is image-dependent. The extreme boundary distortion can be eliminated by applying EI only in a window of the image, excluding the border pixels at the side where the problem is evident. The size and shape of the window are image-dependent and would require further exploration.

Concerning the example image analyzed, simply omitting the left 50 columns of the image, accounting for 5% of the total columns, solves the problem as shown in Fig. 3.7, 3.8. Fig. 3.7 depicts the disparity image and Fig. 3.8 its' heat map.

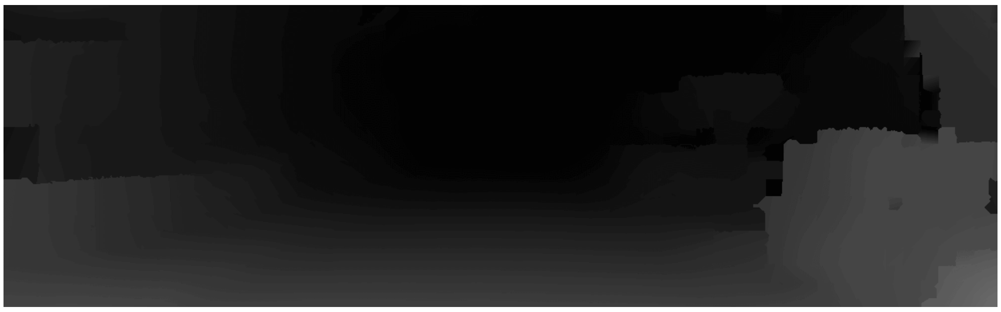


Figure 3.7: Windowed normal distribution

Finally, it must be outlined that the EI approach does not provide reproducible results because of the pseudo-random procedure that starts each time from a different seed. In practice, most of the runs with the same configuration exhibit the same trend, however there are cases

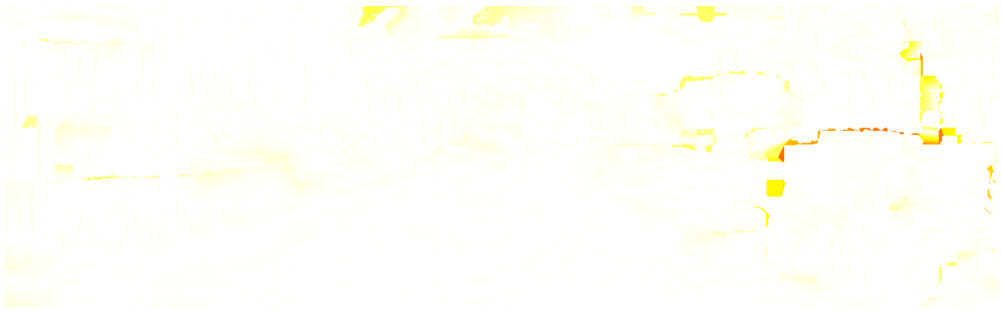


Figure 3.8: Heat map of windowed normal distribution

where areas of the image are excessively distorted, most frequently the parts of the image that experience the larger distortion by default, which are the red regions at the heat maps.

An instance of extreme distortion can be seen in the heat map of Fig. 3.9, where the extreme distortion at the left side of the image has expanded and also a strong artifact at the right side of the image has been induced, at a place which did not seem susceptible to errors in the previous analysis.

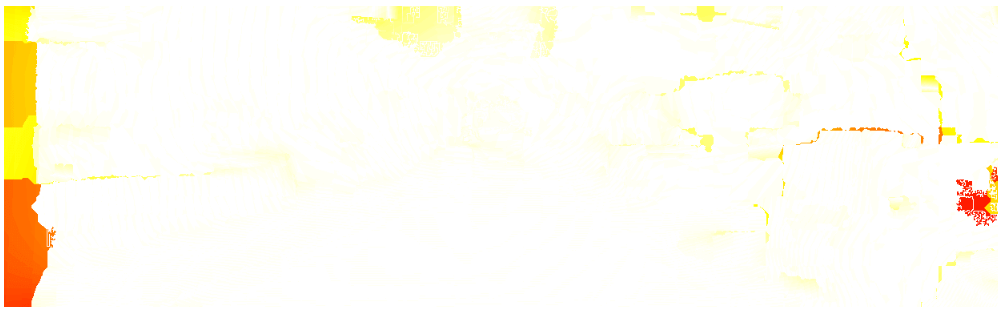


Figure 3.9: Heat map of extreme distortion case in normal distribution

Finally, the PSNR metric quantifies the quality of the produced disparity images and is also used in the AcHEe paper. PSNR is calculated with respect to the fully accurate disparity image. Table 3.2 summarizes the PSNR values of the AcHEe approximate implementation and our approaches. It seems that the EI and ZFP approaches lack the quality achieved by the AcHEe approximate, but this is linked to the phenomenon of extreme distortion analyzed above. This assertion is proven by the fact that the windowed EI method produces slightly better PSNR than the AcHEe approximation.

Table 3.2: PSNR relative to fully accurate

Method	PSNR
AcHEe approx	37.19
Normal	33.22
ZFP	35.62
Normal windowed	39

3.5 Conclusion

To conclude, our models fail to simulate the effect of the approximation accurately. The artifacts that have been induced by the approximation cannot be replicated by our models in detail. Moreover, the left side of the disparity image is sensitive to extreme distortion in our approach and although a solution is proposed, it cannot be embedded in the generic implementation safely. On top of that, the variability of EI experiments, which can lead to extreme distortions in particular parts of the image, also renders the EI models inconsistent. However, it must be noted that our approach is consistent in detecting the parts of the disparity image that will be distorted by the approximation.

Chapter 4

Matrix Multiplication

4.1 Introduction

Matrix Multiplication is a basic linear algebra operation widely used in many scientific areas. Although the operation itself is not error resilient, approximation in matrix multiplication can be used to reduce the computational cost through precision scaling. Understanding the propagation of error in case of such approximation is important to decide whether this technique is effective in a given application. In this regard, a series of experiments is conducted aiming to model the error induced by the use of floats, or in extreme cases integers, in place of doubles.

4.2 Methodology

We explore the multiplication of three sparse square matrices from matrix market [32], by themselves. The fully approximate multiplication uses only doubles while the approximation techniques consist of direct typecasting to floats and rounding to nearest integers respectively.

Regarding the modeling attempts, EI is exclusively applied to the non-zero elements of the sparse matrix, whereas LC is applied first to the full 2D matrix and then to the 1D sequence of non-zero elements, as saved by the matrix market format in column-major style. The application of LC to the full matrix will be referred to as global (i.e. ZFP-global).

The three matrices used are derived from different domains and have a different range of values. The first matrix, named bcsstm27 [33], is a symmetric indefinite matrix used for buckling analysis on an engine inlet of a Boeing jetliner and its' values range is $(-717, 1.44e3)$.

We work only on the lower diagonal part of that matrix in order to save time in the experiments. Also, for brevity, we will analyze only the experiments on the bcsstm27 matrix and will generalize our conclusions.

For reference, the other two real and unsymmetric matrices are the bfw398a [34] and impcol-a [35]. The bfw398a matrix is derived from the domain of electrical engineering. It appears in a generalized eigenvalue problem regarding the millimeter wave technology. Its' values range in $(-2.43, 6.3)$. The impcol-a matrix regards the chemical engineering domain. It is an initial Jacobian approximation for a sparse nonlinear equation modeling a chemical process system. Its' values range is $(-376, 680)$.

For visualization purposes, we use heat maps of absolute differences between the fully accurate implementation and each approximate one. From those heat maps, we have been able to gain insight about the pattern of error and some interesting properties of the LC methods.

Apart from the visualization, the Frobenius norm of the difference of the fully accurate and approximate matrix is calculated, as it expresses the sum of squared error for the elements of the matrix. In particular, the Frobenius norm is calculated twice, first to evaluate the distortion of the initial matrix due to approximation, EI or LC, and then to quantify the error on the final result matrix.

4.2.1 Parameter Configuration

Running multiple experiments with various configurations, we targeted for the configurations which achieve an error scale similar to the approximations. Table 4.1 shows the parameters for the float approximation, while Table 4.2 illustrates the parameters for the integer approximation.

4.3 Evaluation

4.3.1 Float approximation

The results of the experiments highlight the complexity of modeling the error induced by the single-precision (float) approximation. Although, able to achieve similar error range in final result, the modeling attempts cannot follow the error pattern of floats relative to doubles. The error pattern of floats shows large outliers at scattered points throughout the non-zero

Table 4.1: Parameter Configuration - Float approximation

Method	Parameters
SZ	2D mode - absolute error of 7e-5
ZFP	2D mode - absolute error of 5e-4
SZ-global	2D mode - absolute error of 7e-5
ZFP-global	1D mode - absolute error of 5e-4
Normal	$\sigma = 3.3e-5$
Uniform	$b = 1.44e-4$

Table 4.2: Parameter Configuration - Integer approximation

Method	Parameters
SZ	2D mode - absolute error of 0.5
ZFP	2D mode - absolute error of 1
SZ-global	2D mode - absolute error of 0.5
ZFP-global	1D mode - absolute error of 1
Normal	$\sigma = 0.2$
Uniform	$b = 1.04$

elements of the matrix and negligible error for the majority of non-zero elements (Fig. 4.1). The maximum absolute error is 0.281, while the maximum relative error reaches 0.41% and the average relative error is $1.85 \times 10^{-5}\%$ when excluding zero elements.

In contrast, our modeling attempts form a more uniform error pattern, without showing very large errors at distinct points (red points in heatmaps). Indeed, ZFP (Fig. 4.2), normal (Fig. 4.3) and uniform distribution (Fig. 4.4) achieve similar maximum error to floats, but rather have a higher overall error added to the final result. This can also be verified by the Frobenius norm values of the second column of Table 4.3, which highlight that the float approximation has a much lower error norm value in the result matrix.

Interestingly, observing the Frobenius norm values, it is evident that the SZ fails to compress the stream of non-zero elements and add any distortion to the initial matrix. Given absolute error bound of 7e-5 , which works fine when applied to the whole matrix, it achieves a

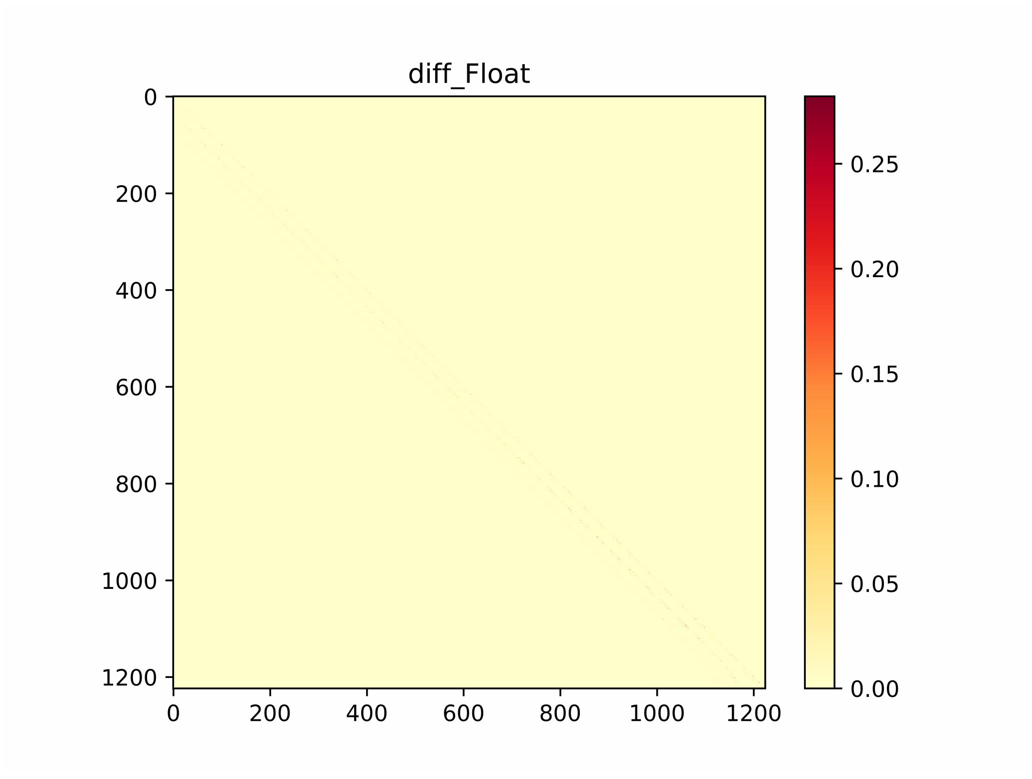


Figure 4.1: Float approximation result - Absolute error to accurate result matrix

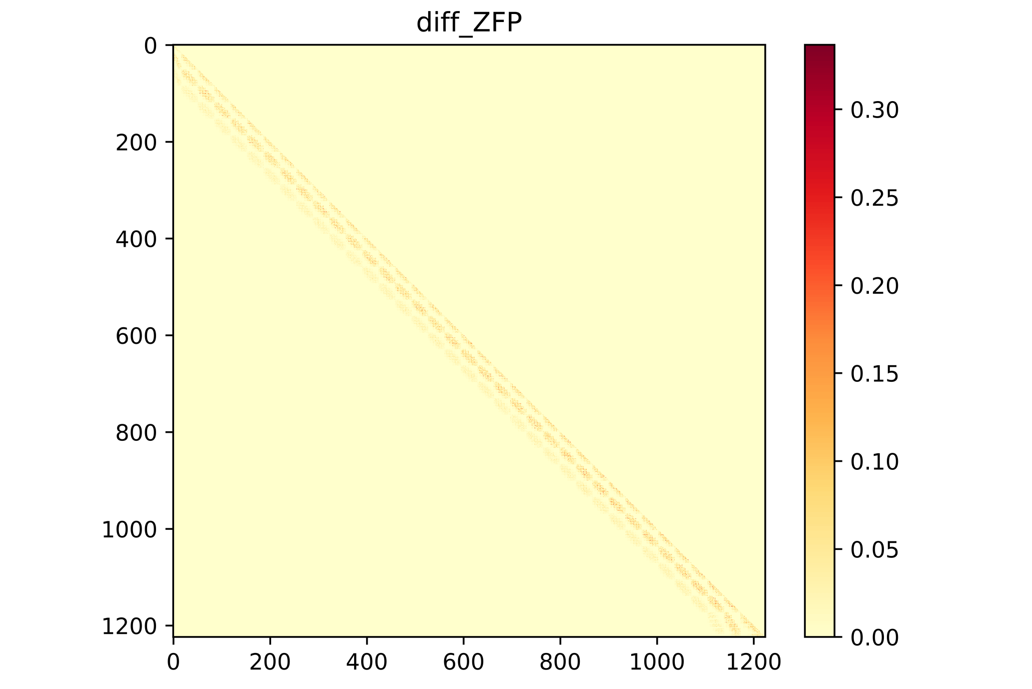


Figure 4.2: ZFP float approximation - Absolute differences from fully accurate result matrix

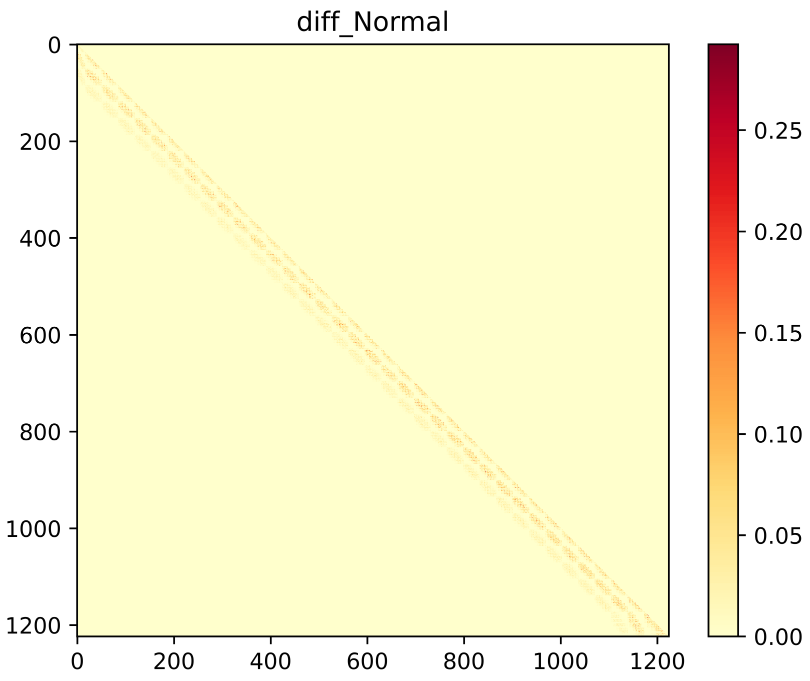


Figure 4.3: Normal float approximation - Absolute differences from fully accurate result matrix

Table 4.3: Frobenius norm of error - Float modeling

Method	$\|\|A - A_{approx}\|\|_{frob}$	$\|\|A^2 - A^2_{approx}\|\|_{frob}$
Floats	0.0007	2.5465
SZ	0	0
ZFP	0.007	9.6404
Normal	0.0056	7.6664
Uniform	0.007	9.5394

CR of 1.03. This behavior is a direct outcome of lack of continuity in the stream of non-zero elements.

Overall, we believe that the inability of our approaches to resemble the float approximation is directly related to the non-linear nature of float representation, which is denser in lower values and sparser in higher values. In contrast, both EI and LC handle elements of data in a uniform manner, regardless of their absolute values.On top of that, the modeling difficulty increases due to the nature of the application, which has inherent error amplification proper-

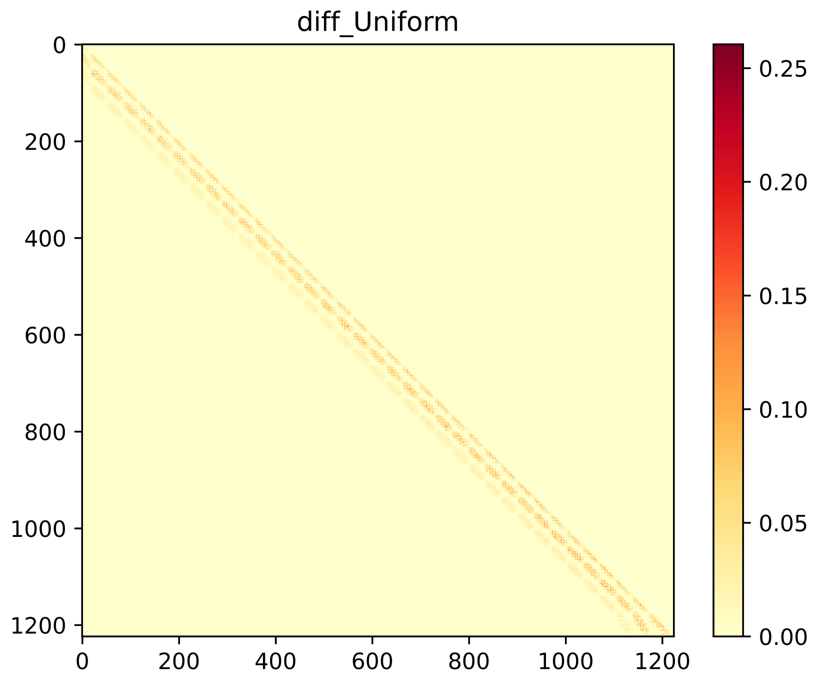


Figure 4.4: Uniform float approximation - Absolute differences from fully accurate result matrix

ties, as shown by the Frobenius norm values in Table 4.3, where the size of error is always higher for the result matrix than the original matrix.

Finally, another important observation is that the configurations of the modeling techniques significantly vary amongst different matrices. In practice, higher absolute values in matrix elements suggest higher error range in the EI process and higher error bounds in the LC framework to achieve the same scale of error in final result. This fact is directly linked to the sparser representation of floats in higher values.

4.3.2 Lossy Compressors

First, we would like to highlight the difference in the configuration parameters of SZ and ZFP. Although using the absolute error bound for both, we observe that ZFP requires a looser error bound than SZ to achieve similar size of distortion in final results. This is directly linked to the fact that ZFP is very conservative on the error bound specified, in order to bound the worst case scenario, as illustrated in [36]. This observation holds true for the rest of the applications analyzed and will not be stated explicitly again.

Concerning the LC application, it is obvious that applying LC only on the non-zero elements is much more accurate than applying it to the full matrix. Interestingly, applying LC to a full sparse 2D matrix highlights the weakness of SZ and ZFP in discontinuous data, by means of distorting zero elements that are near non-zero elements.

Fig. 4.5 depicts the absolute values of the original matrix, while Fig. 4.6 and Fig. 4.7 show the absolute differences of the distorted matrix, by ZFP-global and SZ-global respectively, with respect to the original matrix. As evident, zero elements are affected by their neighboring non-zero elements in a square shape. Moreover, Fig. 4.8 shows the absolute values of the accurate result matrix, while Fig. 4.9 and Fig. 4.10 illustrate the absolute differences of the results of ZFP-global and SZ-global, with respect to the accurate result. Obviously, after the matrix multiplication the band of error is widened even more. This property of LC must be realised to avoid large errors in applications involving sparse matrices.

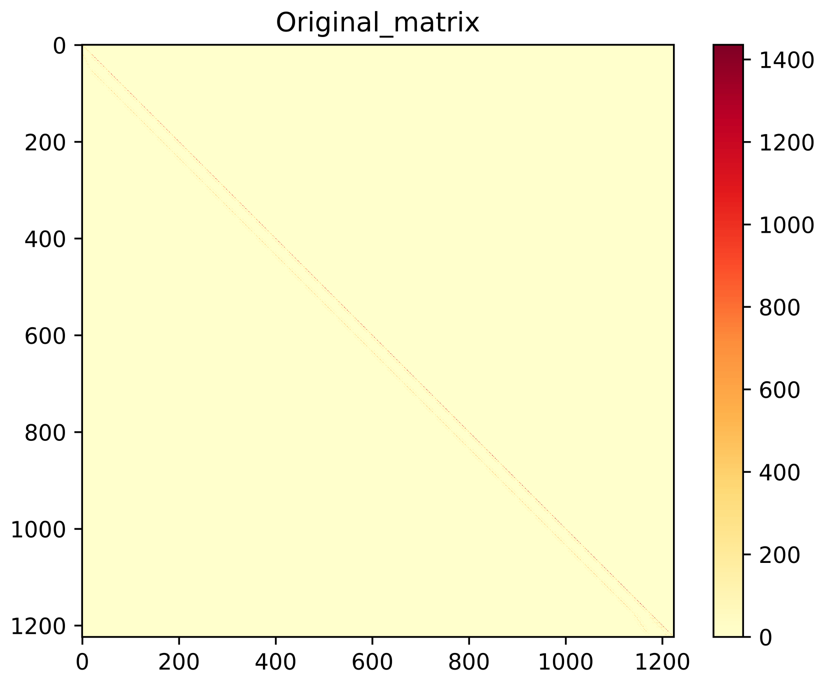


Figure 4.5: Original matrix - Absolute values

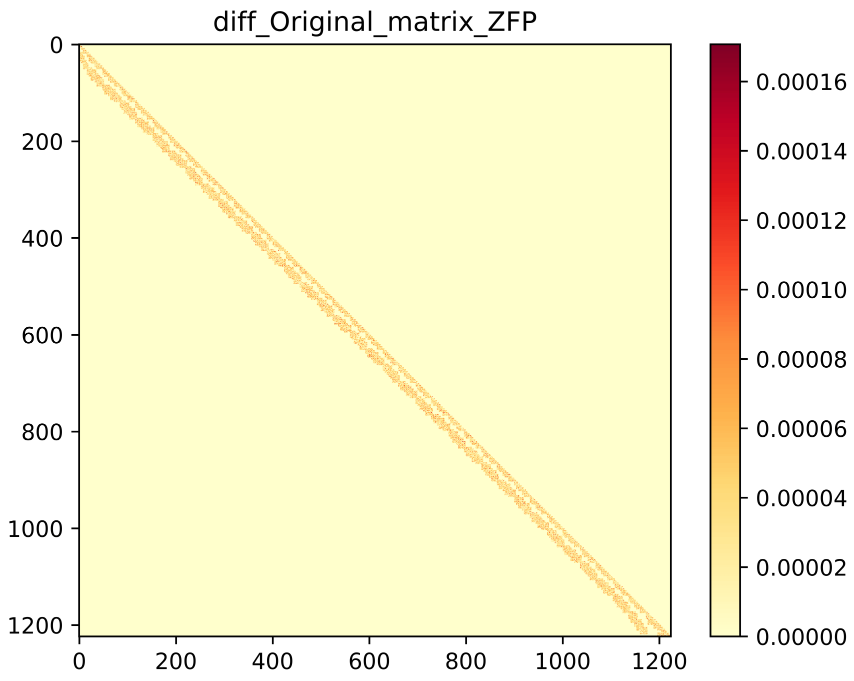


Figure 4.6: ZFP-global on original matrix - Absolute error to original matrix

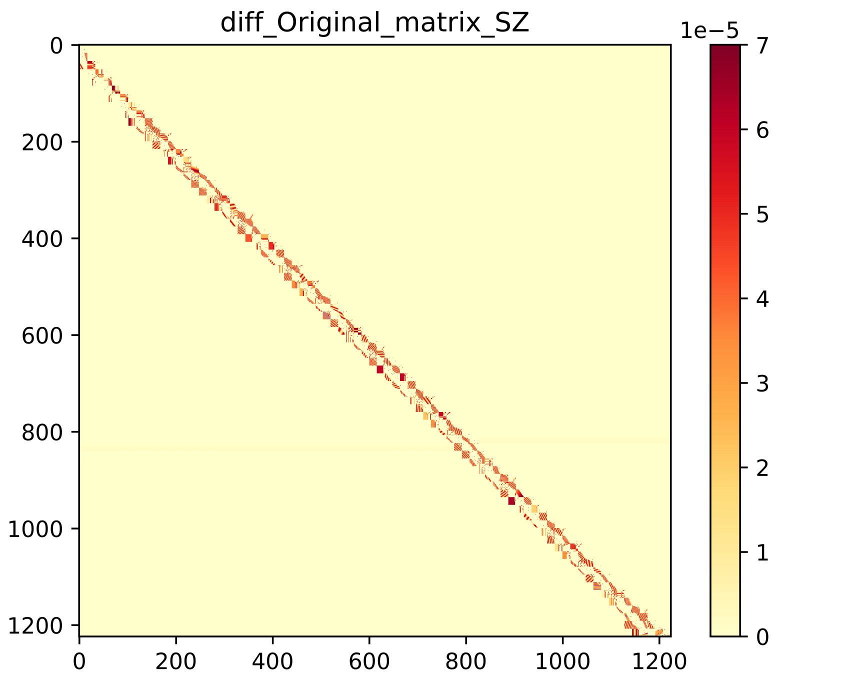


Figure 4.7: SZ-global on original matrix - Absolute error to original matrix

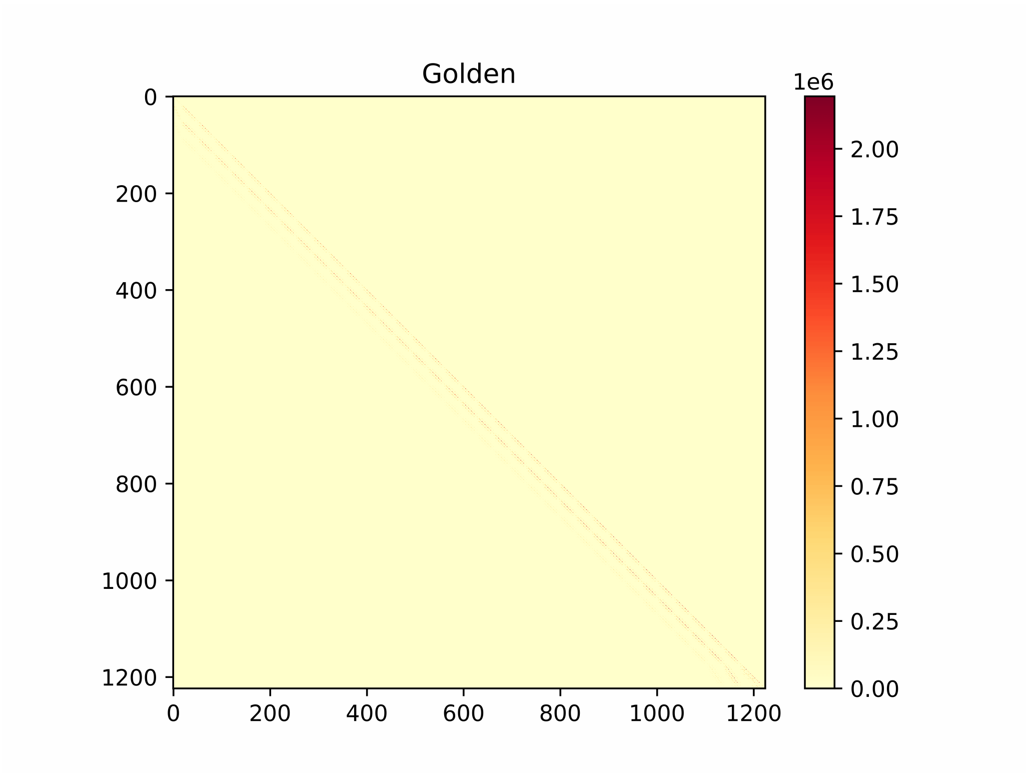


Figure 4.8: Result matrix - Absolute values

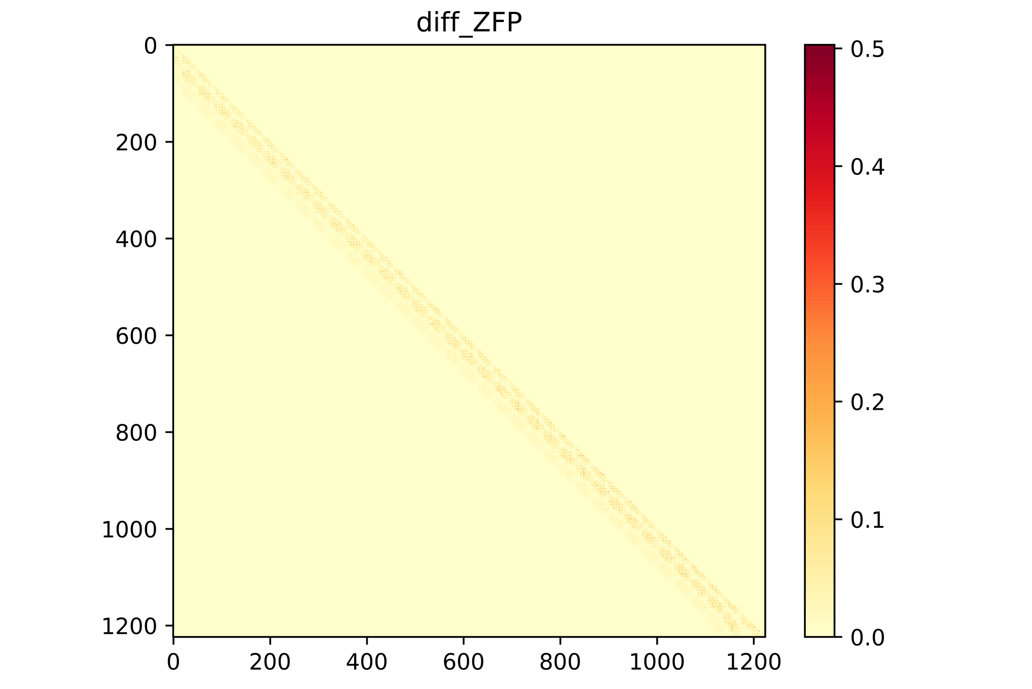


Figure 4.9: ZFP-global on result matrix - Absolute error to accurate result matrix

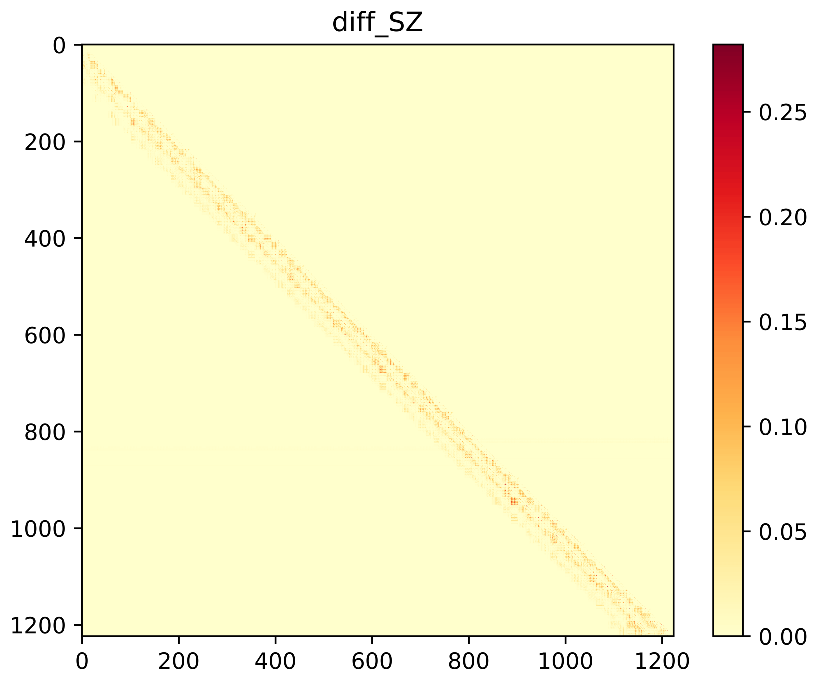


Figure 4.10: SZ-global on result matrix - Absolute error to accurate result matrix

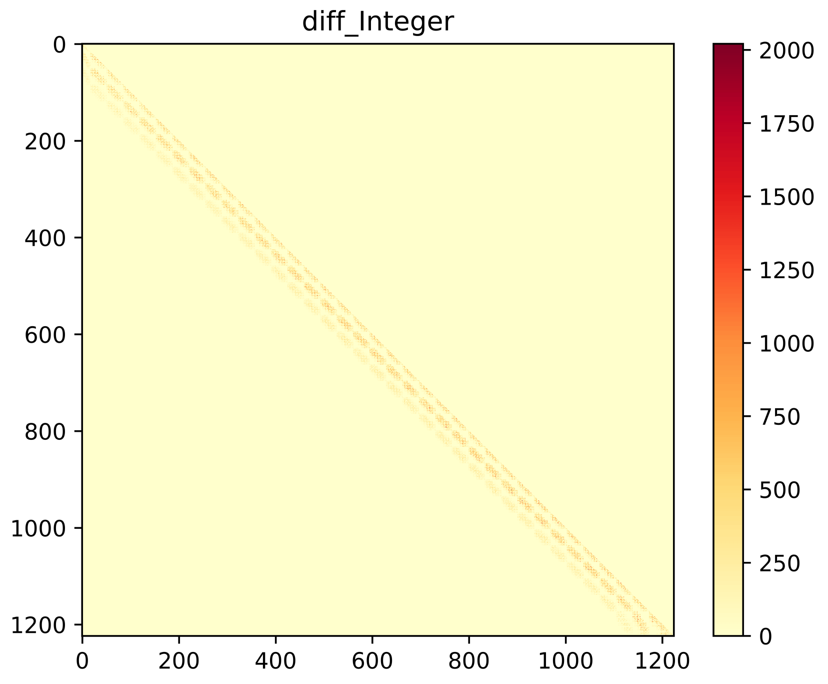


Figure 4.11: Integer approximation - Absolute differences from fully accurate result matrix

4.3.3 Integer approximation

The integer approximation results in very large errors, as illustrated in Fig. 4.11. It reaches a maximum absolute error of 2020.7 and a maximum relative error of $38 \times 10^4\%$. The average relative error is 78% disregarding the zero elements of the matrix.

Nevertheless, it is interesting how EI is able to replicate efficiently the pattern of error in Fig. 4.13, 4.14 and even more so, how the SZ ,applied on the 1D sequence of non-zero data and operating in 2D mode and 0.5 absolute error bound, is replicating exactly the effect of rounding to nearest integer approximation (Fig. 4.12. This proves that this configuration of SZ has identical behavior to the round to nearest integer approximation.

On top of the visualization, the Frobenius norm of error introduced in Table 4.4 , confirms our conclusions. SZ is a perfect model for this approximation, while the uniform and normal EI also form a strong model with similar norms to the integer approximation. On the other hand, ZFP fails to achieve the same scale of error with a much lower error norm value, a fact also visible in Fig. 4.15.

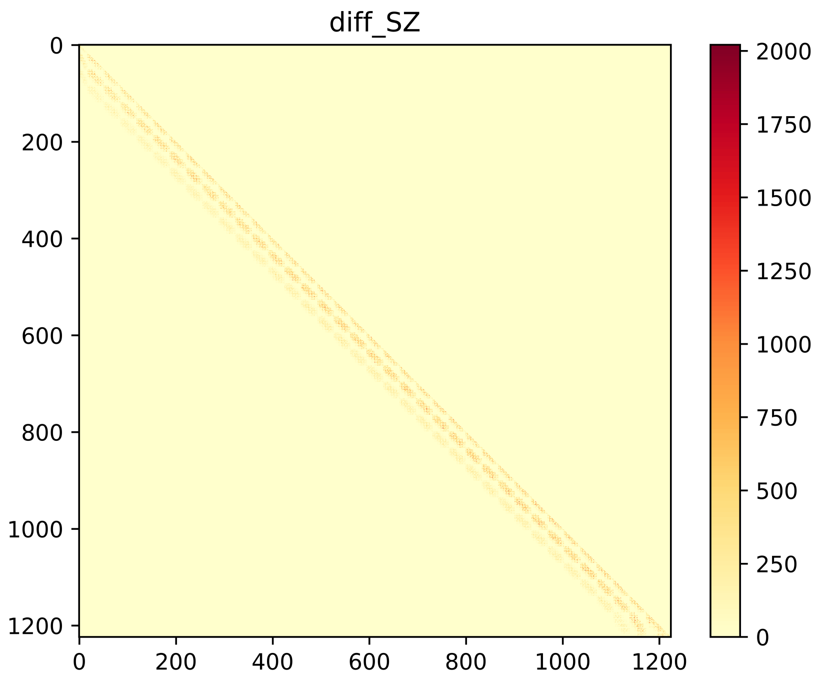


Figure 4.12: SZ integer approximation - Absolute differences from fully accurate result matrix

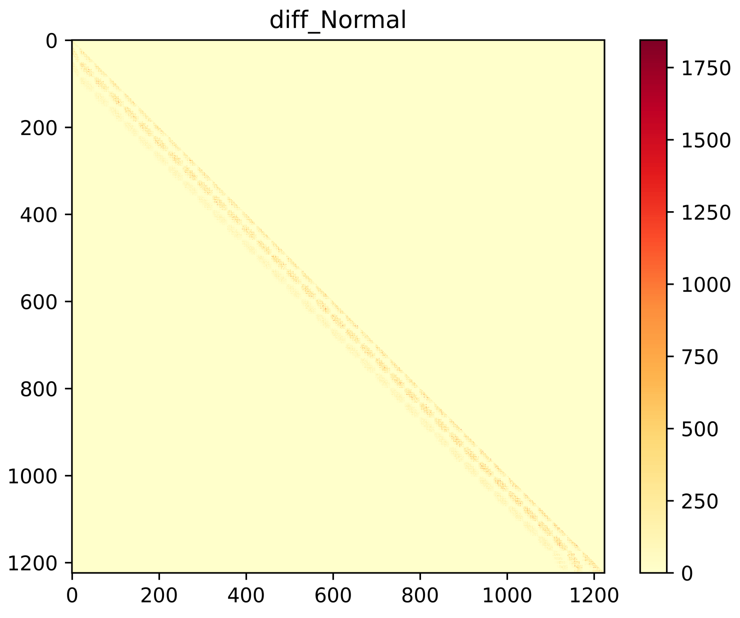


Figure 4.13: Normal integer approximation - Absolute differences from fully accurate result matrix

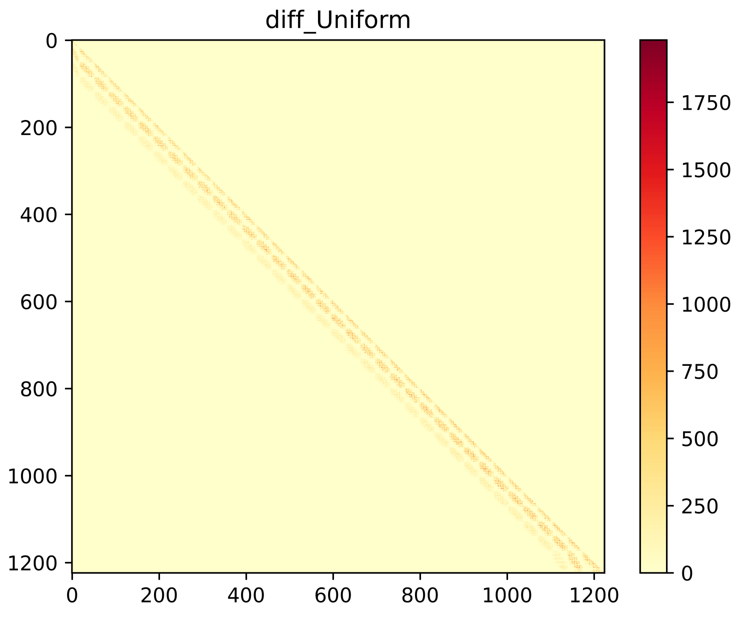


Figure 4.14: Uniform integer approximation - Absolute differences from fully accurate result matrix

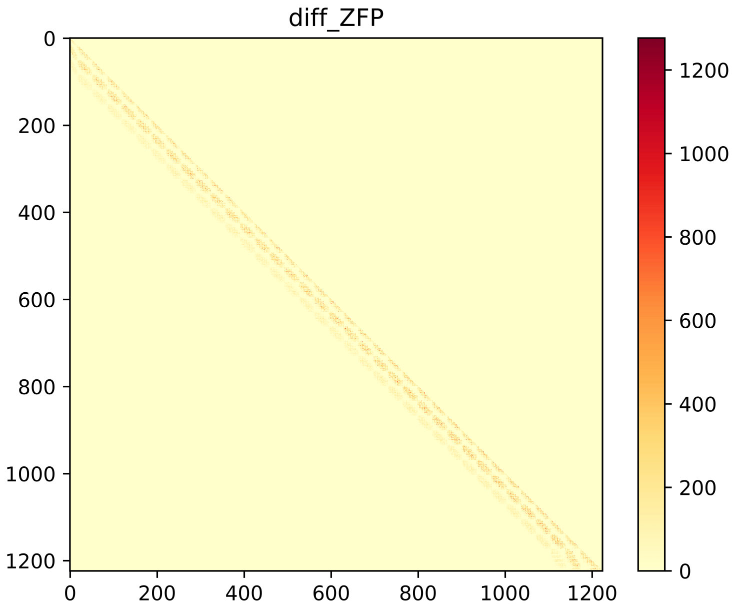


Figure 4.15: ZFP integer approximation - Absolute differences from fully accurate result matrix

Table 4.4: Frobenius norm of error - Integer modeling

Method	$\|\|A - A_{approx}\|\|_{frob}$	$\|\|A^2 - A^2_{approx}\|\|_{frob}$
Integers	42.7254	60672.1
SZ	42.7254	60672.1
ZFP	28.5549	38737.6
Normal	33.8246	45731.3
Uniform	50.9448	69214

4.4 Conclusion

To sum up, our methods fail to simulate the float approximation effect in the matrix multiplication case. Although able to replicate the same scale of error, they cannot achieve similar values in the Frobenius norm of the error neither produce the same patterns of error in the result matrix. On the other hand, regarding the round to nearest integer approximation, we observe that SZ, employed with 2D mode and absolute error of 0.5, is a perfect model. Finally, EI also forms a strong predictor achieving similar size and patterns of error.

Chapter 5

Singular Value Decomposition

5.1 Introduction

Singular Value Decomposition (SVD) is a well-known factorization technique to reduce the storage space of a large matrix while accepting some quality loss. It essentially decomposes a matrix into the product of 3 matrices. Let A be an $m \times n$ real matrix, then SVD factorizes the matrix in the form $U\Sigma V^T$, where U, V are orthogonal matrices of dimensions $m \times n$ and $n \times n$ respectively, whereas Σ is a diagonal matrix whose elements are called singular values.

Interestingly, the number of non-zero singular values is equal to the rank of matrix A, while singular values are stored in descending order. This way, one can use only a portion of them, and the corresponding columns of U and rows of V^T, to perform the reconstruction via multiplication. The more singular values used, the more accurate the reconstruction is, but the less singular values used the less storage is required to store the matrix.

That being said, there are different algorithms to calculate the matrices U, Σ, V. In this case, we focus on the iterative method of Golub and Reinsch introduced in [37]. This method comprises three steps. First, an initial factorization is calculated where the middle matrix is bidiagonal [38]. Second, the bidiagonal matrix is reduced to a diagonal matrix via an iterative method, while the other two matrices are processed simultaneously to satisfy the equation. Finally, the diagonal matrix is sorted to hold elements of descending order.

The actual version of the SVD algorithm used is described in [39]. We use the c++ library of [40] , which includes fully accurate implementation using doubles. To implement an approximate version of this algorithm, we had to extend the library.

Similar to matrix multiplication, precision scaling is an interesting method to induce approximation in the SVD process described above. Our experiments involve both the use of single precision and half precision arithmetic, which uses just 16 bits to represent floating-point values. The latter is not natively supported by the machine nor the programming language, but can be simulated using the library described in [41]. The software is available on Github [42].

5.2 Methodology

The approximation method is based on the use of reduced precision, wherever there is a need for floating point arithmetic in the SVD library referenced above. As always, our modeling approaches lie on the fact of distorting the initial matrix using EI or LC, before applying the fully accurate SVD version and comparing the error patterns. The error patterns of the three matrices U, Σ, V with respect to the fully accurate decomposition of the original matrix are visualized as heat maps.

Using this approach, we ran the first series of experiments on the matrices of Chapter 4 using only floats for the SVD algorithm. The error patterns of U, V were similar for the float approximation and the rest of the techniques, whilst the error patterns for the Σ matrix showed different patterns, yet an error of order 10^{-5}, which is negligible for singular values of order up to 10^3. The visualizations of the above observations are omitted for brevity, since a similar approach will be analyzed below.

After promising experiments on sparse matrices, we decided to tackle an SVD application. Image compression with SVD decomposition is a popular application, where reduced number of singular values is used to reconstruct the initial image. We chose this application, because an image is a fitting example of dense matrix where we can also assess the behavior of LC in continuous data. Moreover, using different percentages of singular values for reconstruction we can observe where most of the error lies in the decomposed matrices.

That said, we examine the case of a 16-bit grayscale image (Fig. 5.1) and decide to normalize the pixel values to the [0,1] range. This way we can make good use of the LC which is more efficient on floating-point data and also use more aggressive precision scaling for the SVD algorithm. After experiments with floats showed negligible distortion in the reconstructed image, as PSNR values were equal to the fully accurate implementation, we decided to experiment with half precision.

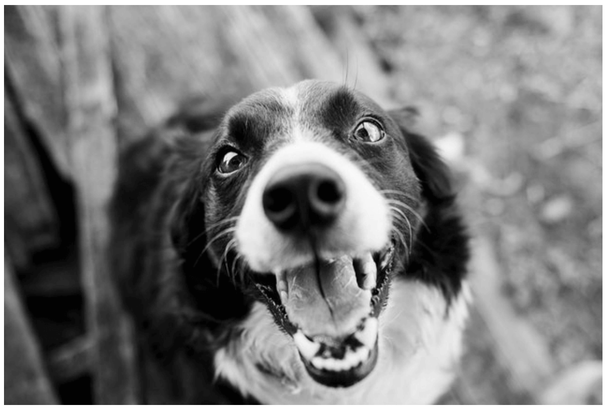


Figure 5.1: Original picture

Implementing the SVD only with half precision resulted in lack of convergence in the iterative process of the algorithm, even with increased iterations. In addition, using floats in the SVD and half precision in the reconstruction operations resulted in unacceptably large errors. Finally, the use of half precision for the first and third step of the SVD, in combination with the use of floats for the iterative second step, is the ideal approximation for this application, as it induces acceptable error. We have to credit the work of [17] because it inspired our mixed precision approach.

We refer to this pipeline as the mixed precision approximation from now on. The following sections give the configuration of the modeling techniques used to simulate this approximation and analyze the results of the experiments.

Table 5.1: Parameter Configuration

Method	Parameters
SZ	2D mode - absolute error of 3e-2
ZFP	2D mode - absolute error of 3e-1
Normal	$\sigma = 1.5e - 2$
Uniform	$b = 6e - 2$

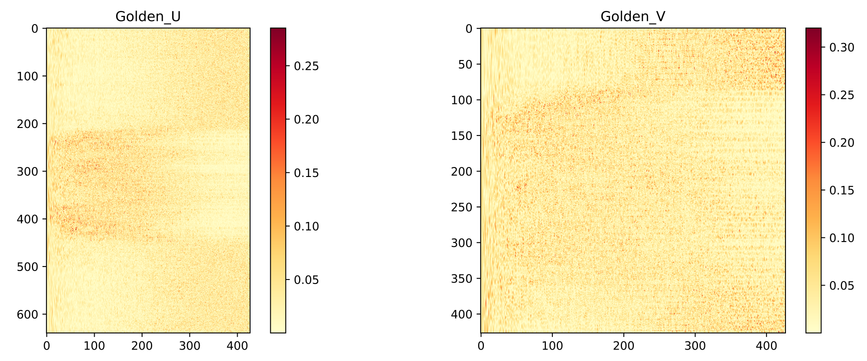
5.2.1 Parameter Configuration

The parameters used for the modeling methods are deliberately chosen to produce similar PSNR values to the mixed precision pipeline. The actual parameters used can be seen in Table 5.1.

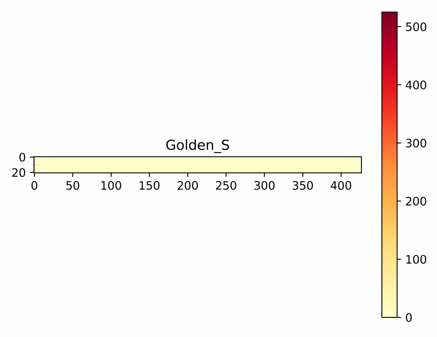
5.3 Results

5.3.1 Error patterns in decomposed matrices

First of all, we examine the range of values in the U, Σ and V matrices of the accurate execution of SVD. As shown in Fig. 5.2, the absolute values of elements range from 0 to 0.28 in U and from 0 to 0.31 in V, while the singular values range from 0 to 515.



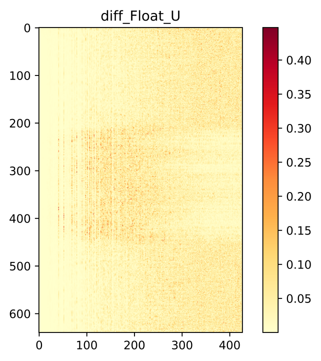
(i) U (ii) V



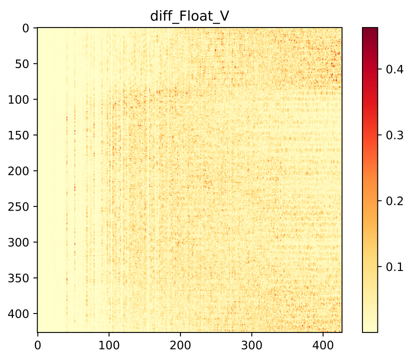
(iii) Σ

Figure 5.2: Fully accurate SVD - Absolute values

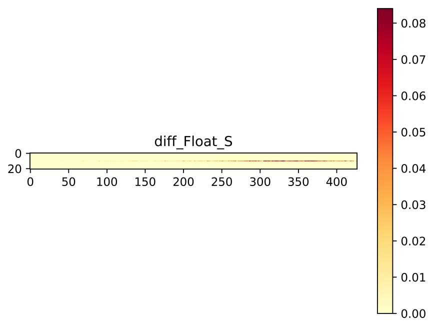
The absolute error of the mixed precision approximation with relevance to the accurate, can be seen in Fig. 5.3. It ranges from 0 to 0.45 for both U and V matrices, which is significant bearing in mind the range of values of those matrices. However, for the singular values relative error has been calculated, because we believe it is more meaningful due to the variability of singular values. It can be noticed the first half of the singular values shows a relative error between 0 to 5%, while the second half has a larger error that ranges between 5% to 8%.



(i) U



(ii) V



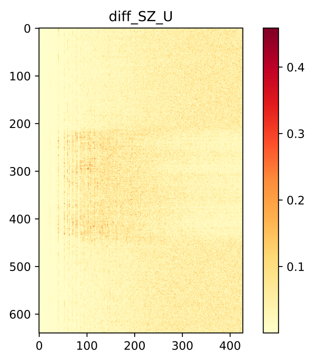
(iii) Σ

Figure 5.3: Mixed Precision SVD - Absolute error to accurate

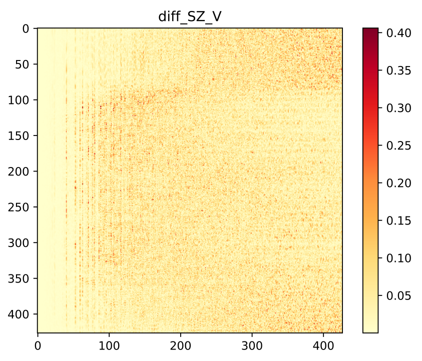
Regarding the modeling techniques, we observe that the error range and pattern in U and V is almost identical to the mixed precision case for the SZ, the ZFP and the Normal, as illustrated in Fig. 5.4. The ZFP method introduces slightly larger error values at distinct points while the Uniform case is similar to the Normal and is omitted for brevity.

Despite the success in simulating error patterns in U and V, our modeling attempts fail to replicate the same error scale in the singular values. As shown in Fig. 5.5, all modeling methods exhibit relative error above 300% in the smallest singular values, apart from the ZFP which shows a maximum relative error of 100%.

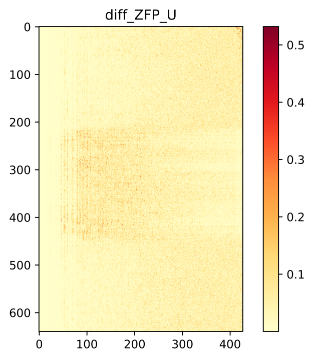
Comparing the above to the singular values of the approximation (Fig. 5.3iii), it is apparent that there is significant divergence in the error introduced in the singular values. The relative error of our methods is two orders of magnitude larger. However, the effect of this variation in the final result needs to be tested in the final reconstructed image, since matrix multiplication is involved.



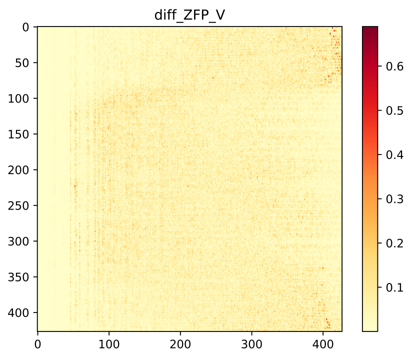
(i) SZ U



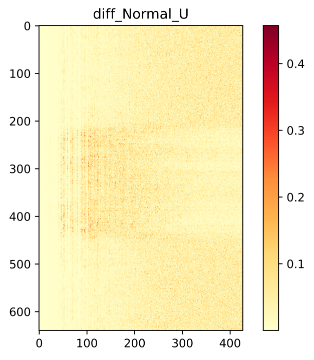
(ii) SZ V



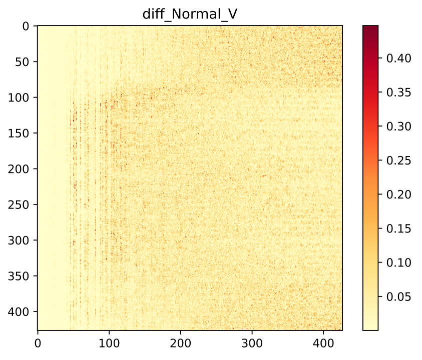
(iii) ZFP U



(iv) ZFP V

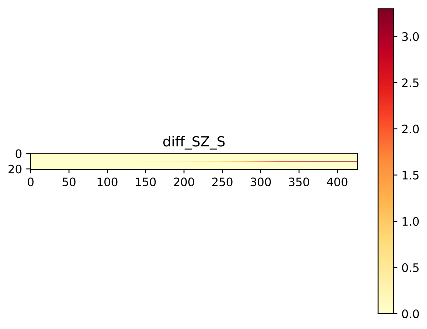


(v) Normal U

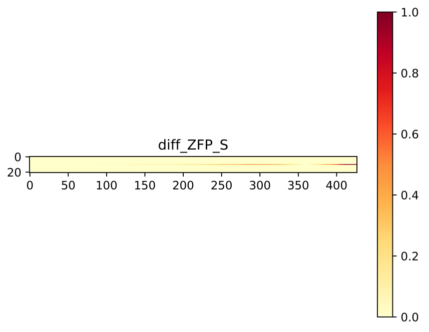


(vi) Normal V

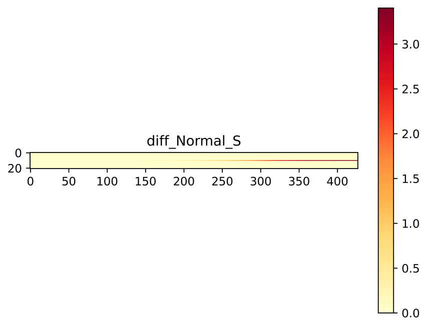
Figure 5.4: Absolute error to accurate U, V



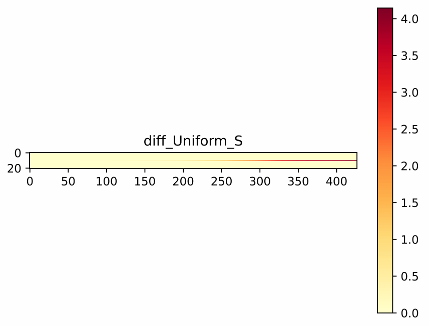
(i) SZ



(ii) ZFP



(iii) Normal



(iv) Uniform

Figure 5.5: Absolute differences to accurate Σ

5.3.2 Reconstructed Images

The reconstructed images using all singular values for each approach are listed in Fig. 5.6. All the images are visually the same, since we attempted to retain a PSNR higher than 40. Nevertheless, slight differences are visible in the heat maps of absolute pixel differences in Fig. 5.7. It must be noted, that the heat map of the fully accurate case, has been calculated relevant to the original image, while for the rest relevant to the fully accurate case.

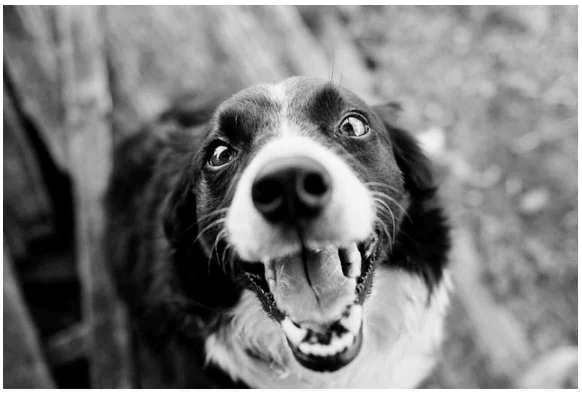
Using an 8-bit per pixel representation for the reconstructed images, the absolute pixel differences have similar scale for all instances. However, the mixed-precision approximation (Fig. 5.7ii) has a unique error pattern that cannot be replicated by any of the modeling techniques.

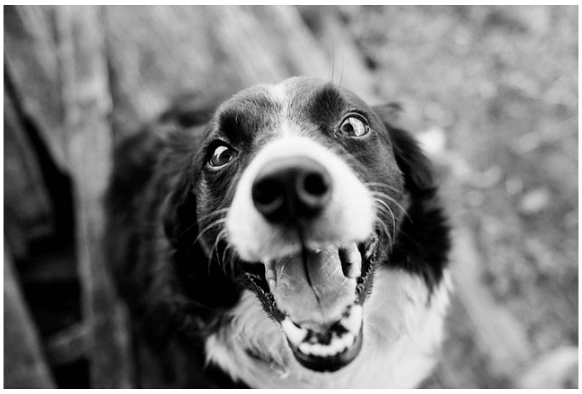
Firstly, the approximation induces a vertical line of distortion at the left side, which includes pixel differences over 25 and does not exist in any other method. Moreover, the general error pattern shows horizontal locality, by means of similar errors in pixels of the same row.

(i) Fully accurate

(ii) Mixed Precision

(iii) SZ


(iv) ZFP


(v) Normal

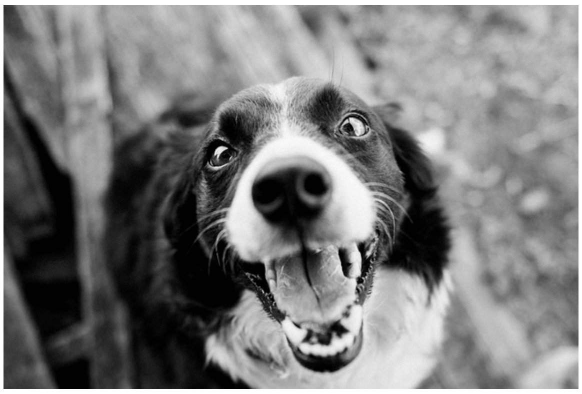

(vi) Uniform

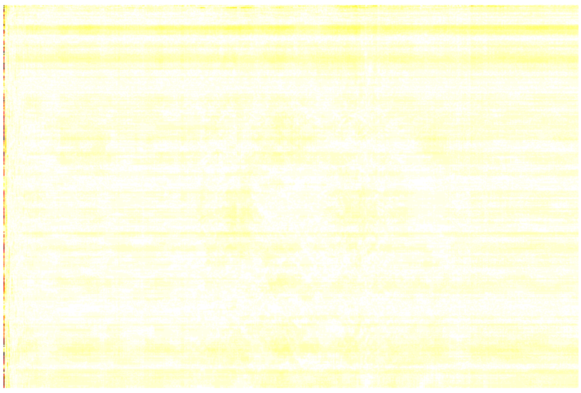
Figure 5.6: Full SVD reconstruction

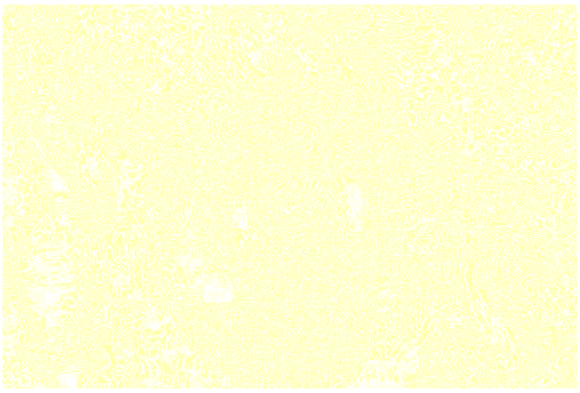
Again this horizontal continuity is not present in any other method.

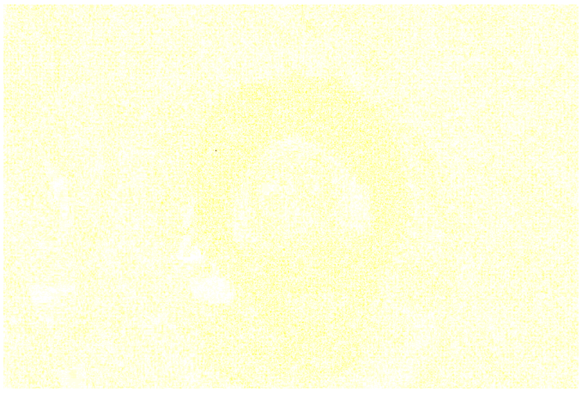
Regarding the EI methods (Fig. 5.7v, 5.7vi), the error pattern is uniform around all parts of the image and is of no interest to us. On the other hand, the LC techniques display interesting error patterns that require elaborate analysis.

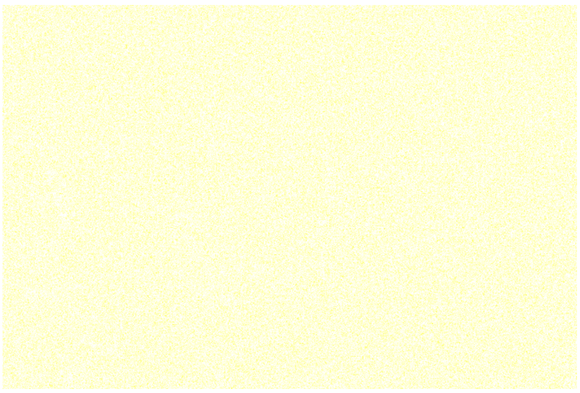
The SZ (Fig. 5.7iii) demonstrates a distribution of error that resembles the uniform. It includes several regions of zero distortion, regions with moderate error, and some areas with

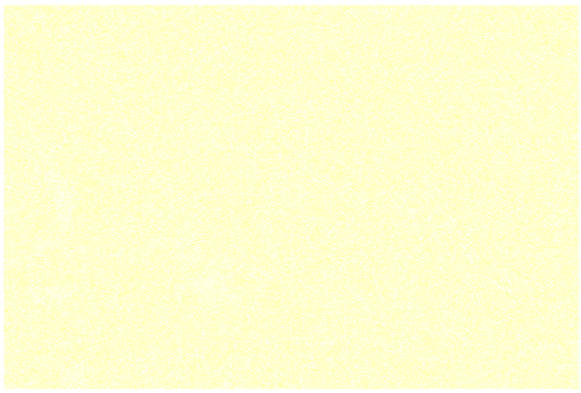
(i) Fully accurate

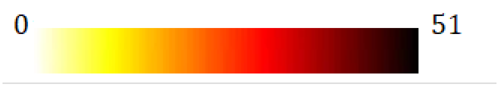

(ii) Mixed Precision


(iii) SZ


(iv) ZFP


(v) Normal


(vi) Uniform

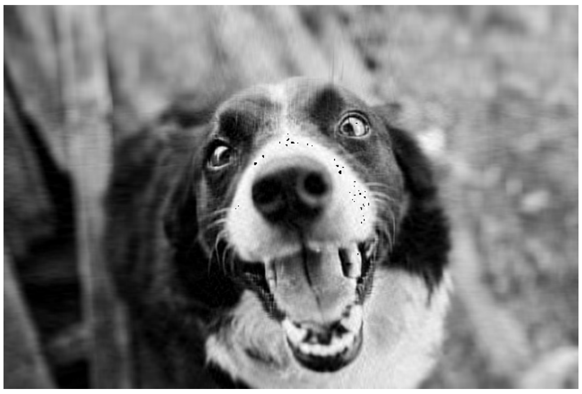


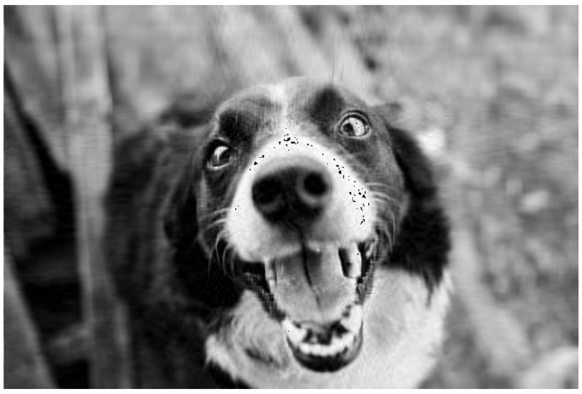
(vii) Scale

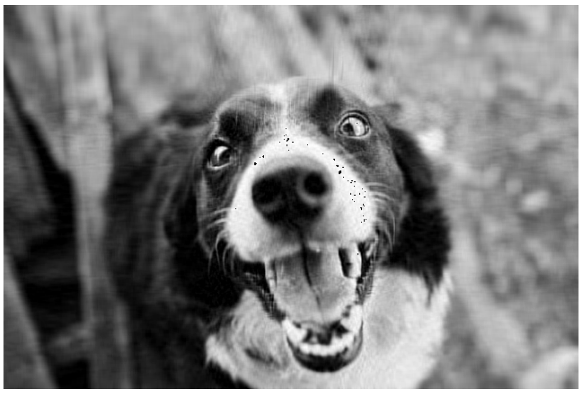
Figure 5.7: Absolute pixel differences with full SVD reconstruction

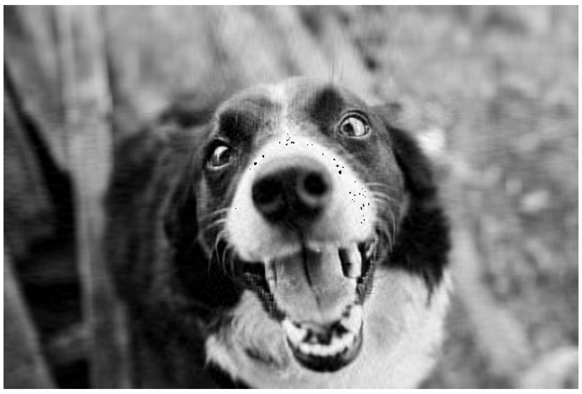
larger distortion that are usually around areas of zero distortion. On the contrary, the ZFP (Fig. 5.7iv) follows a distribution that mostly resembles the normal, since most pixels have an error near zero. Moderate error values are scattered around the background while larger error values exist at the face of the dog where detailed texture lies.

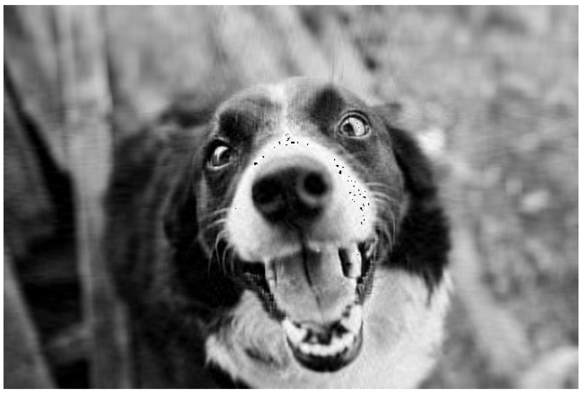
Regarding the areas of zero distortion, we confirm that those areas of the picture, show little variability in pixel values. As a result, the compressors, which partition the image into blocks, can provide compression without introducing errors. Since all values are similar, SZ can predict them accurately based on the preceding values and ZFP can effectively truncate their floating-point representation.


(i) Fully accurate


(ii) Mixed Precision


(iii) SZ


(iv) ZFP


(v) Normal

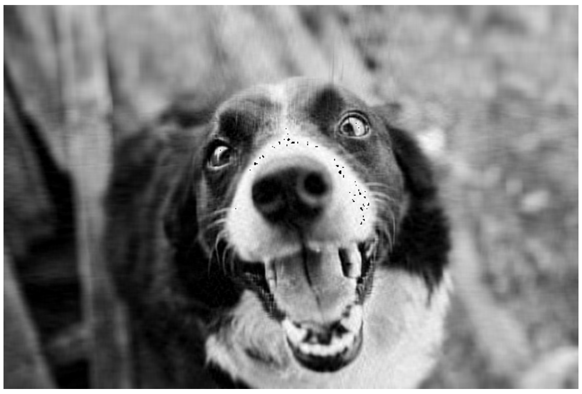
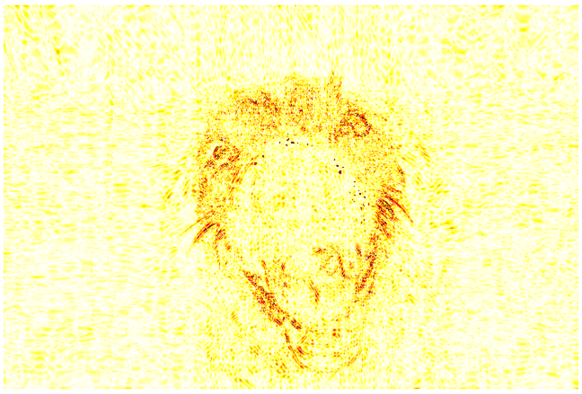
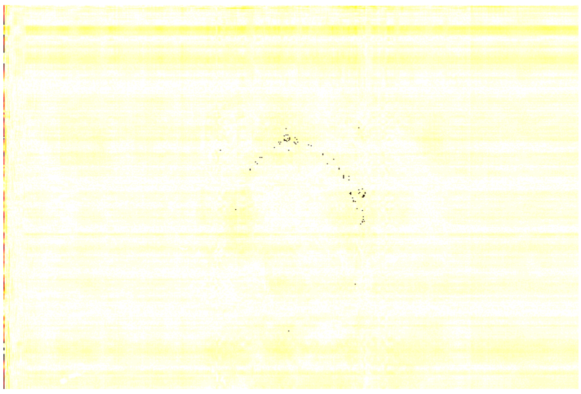

(vi) Uniform

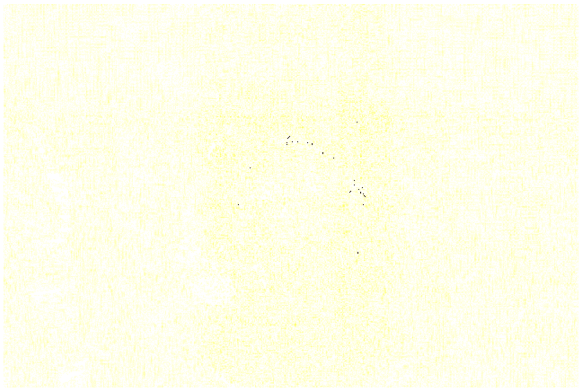
Figure 5.8: SVD reconstruction with 50(12%) singular values

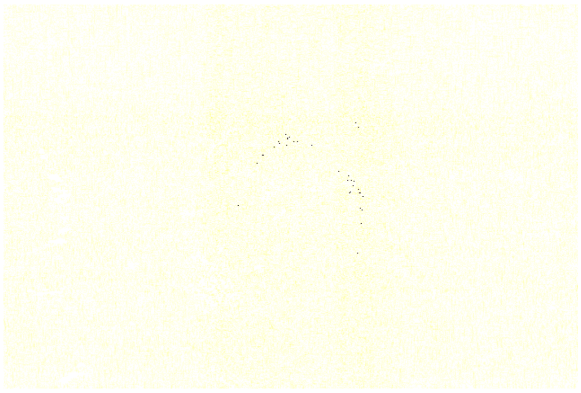
Similarly, the error pattern of the approximation cannot be simulated for fewer singular values. We include the reconstructed images using only 12% of the singular values in Fig. 5.8. It is evident that even the accurate SVD results in blurry image and also includes some corrupted pixels around the nose of the dog. The blur is present in all methods, yet the corrupted pixels' location varies.


(i) Fully accurate


(ii) Mixed Precision

(iii) SZ


(iv) ZFP


(v) Normal

(vi) Uniform

(vii) Scale

Figure 5.9: Absolute pixel differences of SVD reconstruction with 50(12%) singular values

The relevant heat maps are provided in Fig. 5.9 and provide valuable insight. In Fig. 5.9i, the distortion of the accurate reconstruction is substantially larger than in Fig. 5.7i and reaches differences of more than 30 around the nose of the dog, which is the most sensitive part of the image. Moreover, the horizontal lines in the mixed precision heat map (Fig. 5.9ii) still exist, whereas all methods show increased number of corrupted pixels around the nose with relevance to the accurate image.

Regarding the LC techniques (Fig. 5.9iii, 5.9iv), the error is less uniform across the image and also the areas of zero distortion have been partially or completely eliminated. This is the result of the combination of errors derived from the distortion of the initial image due to LC and the error induced by the use of fewer singular values. In a similar manner, the EI approaches (Fig. 5.9v, 5.9vi) have partially lost their homogeneity of error across the image.

To conclude, we have summarized the PSNR values of the images in Fig. 5.10 for different singular values. It is evident that our models are consistently producing similar PSNR values to the approximate method.

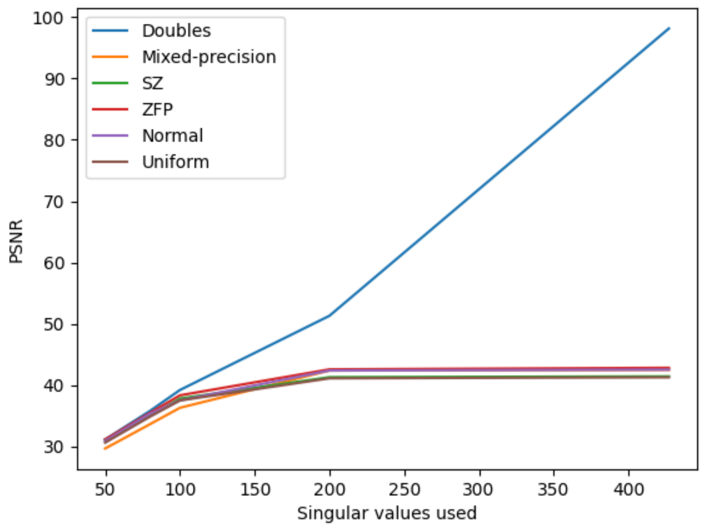


Figure 5.10: PSNR relative to original for various singular values

5.3.3 Effect of Lossy Compressors on Images

First of all, we clarify that the distortion induced above, by both LC and EI in the final reconstructed image using all singular values, is primarily based on the distortion of the initial image and not on the SVD process itself. We have confirmed that observation by omitting the SVD process and calculating the heat-maps of the distorted images by LC or EI. As a result, evaluating the results of the LC experiments on the reconstructed image with full reconstruction is equivalent to evaluating the effect of direct application of LC on the images.

Regarding the error patterns analyzed above, we confirm that our observations on the patterns of error match the observations of [36]. In particular, the SZ exhibits a uniform distribution of error in image pixels whereas the ZFP tends to follow a normal distribution, with higher pixel differences around the center of the image where most detailed texture exists.

At last, we report the CRs for SZ and ZFP to be 37.4 and 26.6 respectively. This means that we have introduced a framework, where a grayscale 16-bit png image has been compressed by a factor of 37, while achieving a PSNR of 42 with no obvious visual differences. The key to our approach lies on the downscale of the pixel values to the $[0, 1]$ interval, essentially converting integer values to floating point values.

5.4 Conclusion

Our modeling methods are capable of providing similar error patterns to the approximate method in the decomposed U and V matrices. However, there are significant differences in the singular values calculated. As a result, despite achieving similar PSNR values in the reconstructed image for various singular values used, our modeling attempts fail to replicate the effect of the approximation or provide some valuable insight about it, for any number of singular values used.

Chapter 6

Conjugate Gradient

6.1 Introduction

Conjugate Gradient (CG) is an iterative method for solving systems of linear equations, whose matrix is positive-definite. As most iterative methods, it is not highly sensitive to errors because the solution is improved at each iteration. Consequently, modeling the effect of an approximate implementation with floats is practically useful for future users exploring performance gains on the CG method.

The algorithm we developed is based on the first algorithm described in [43], which is the simplest and most commonly used version of CG and is illustrated in Fig. 6.1. We define maximum iterations to equal the number of rows of the matrix, while we use a tolerance of $1e - 10$ as our termination criterion. Moreover, we extend our terminating conditions to include both the norm of the residual and the norm of subsequent solution vectors.

Essentially the problem can be formulated as:

$$Find\, x : Ax = b$$

where A is a real positive definite $n \times n$ matrix and b is a known $n \times 1$ vector. The approach we follow and the experiments we conducted are analyzed below.

6.2 Methodology

First, we employ the fully accurate CG with double precision to have the reference solution vector. Second, we employ the approximate implementation with float precision and

```
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
    $\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
    $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
    $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
    if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
    $\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
    $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
    $k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result
```

Figure 6.1: The Conjugate Gradient algorithm used

then we run the accurate pipeline on the distorted matrix by EI or LC. Finally, we construct heat maps of the approximate solution and our modeling methods' solutions with relevance to the accurate solution.

This way we can analyze the error pattern and pinpoint any similarity, which would suggest that our methods can simulate the approximation. However, results are highly dependent both on the matrix A and the vector b used. We decided to experiment on three different matrices, each with a different range of values. For each matrix, we first manually specify the solution vector x in order to produce the vector b and test the algorithm, but we also experiment with random number generation for the vector b. Finally, the initial estimation of x is $\vec{0}$, which in all cases results in convergence.

Regarding the matrices used, once again we use matrices from matrix market. The matrices are namely, bcsstk14 [44], nos3 [45] and nos5 [46]. Their elements' absolute values are of order $1e9$, $1e3$ and $1e5$ respectively. Taking into account the high number of experiments due to different matrices and setups, we will fully analyze the results of a particular matrix and setup. However, we will discuss about the results of all experiments and highlight the most noteworthy observations.

Regarding the data distortion, we apply EI to the elements as saved by the matrix market format, in column major order. This means that only the non-zero elements of the lower triangular part of the matrix are saved, since the matrix is symmetric. Similarly, LC is applied to the 1D stream of elements, as provided by the matrix market format.

Taking into account initial experiments with manually built solutions and using $x = \vec{0}$ as initial estimation, we acquired valuable insight about our methods. First, we realized that the higher the absolute values of the matrix the harder the modeling attempts get. In particular, for the bcsstk14 matrix, it was not possible to even achieve the same error scale with the approximation. With the nos5 case the scale was similar, but the error pattern substantially different and for the nos3 matrix the error scale was also similar, as was the error pattern for some solutions.

To that extent, we regarded our modeling attempts as unsuccessful in high-value ranges and decided to carry on more experiments on the nos3 matrix, which has the highest modeling potential. Additional experiments include random number generation for the solution vector. It has to be noted that nos3 has the same condition number as nos5, yet two orders of magnitude smaller element values. We believe that the sparser representation density of floating-point precision in higher absolute values renders the float approximation more complex to model, as was the case in the matrix multiplication case in Chapter 3.

Below we analyze the error pattern on the CG of the nos3 matrix with solution vector $x = [0.5\ 0.5 \ldots 0.5]^T$ and initial estimation $x = \vec{0}$. This example is a case of strong modeling potential of the approximation by our methods. However, a more challenging example follows. Finally, we conclude in the last subsection the challenges faced by our approach.

6.2.1 Parameter Configuration

Table 6.1 includes the parameters of our modeling methods for the setup analyzed above are shown:

6.3 Results

Table 6.2 illustrates the Frobenius norm of the difference amongst the solution reached by the approximation or modeling technique and the fully accurate solution achieved by doubles. This metric of error suggests that all modeling methods achieve a similar sum of squared errors to the approximate version in the solution vector. Yet, the pattern of error is highly

Table 6.1: Parameter Configuration

Method	Parameters
SZ	2D mode - absolute error of 3e-7
ZFP	2D mode - absolute error of 2e-6
Normal	$\sigma = 5e - 6$
Uniform	$b = 1.4e - 5$

important to assess the success of our methods.

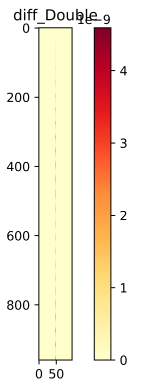
Table 6.2: Frobenius norm of error

Method	$\|\|x - x_{approx}\|\|_{frob}$
Floats	3.98×10^{-4}
SZ	4.12×10^{-4}
ZFP	4.13×10^{-4}
Normal	3.65×10^{-4}
Uniform	4.61×10^{-4}

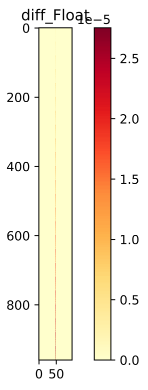
Figure 6.2 displays heat maps representing the absolute differences with respect to the solution reached by the accurate CG with doubles. In case of doubles, the heat map represents absolute error with relevance to the actual solution. The error is of order $1e - 9$, suggesting that the solution is accurate and the CG has converged. Similarly, the scale of error for the approximation and the modeling methods is of order $1e - 5$, also confirming that those implementations converge.

The accurate implementation, as well as our models, converge in 510 iterations, where the solution vector does not change. Interestingly, the CG with floats converges in 280 iterations, with the same termination condition. This phenomenon applies to all experiments, proving how efficient this approximation is. On average, the float implementation requires 40% fewer iterations.

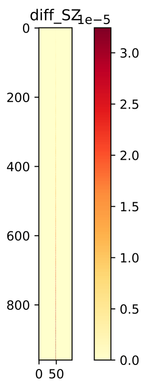
Observing the error patterns of Fig. 6.2iii, 6.2iv, 6.2v, 6.2vi, one can claim that all modeling approaches resemble the pattern of error of the approximation , as shown in Fig. 6.2ii. Indeed, the pattern of error in all cases progressively grows from the top to the bottom of the column vector. Although element-wise one can identify different error values, the general pattern of error is arguably simulated accurately.



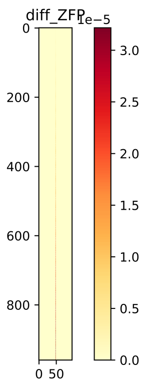
(i) Doubles relative to actual solution



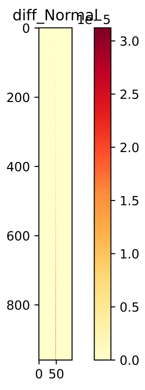
(ii) Float approximation



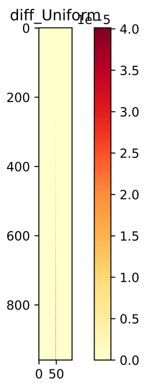
(iii) SZ



(iv) ZFP



(v) Normal



(vi) Uniform

Figure 6.2: Absolute error with relevance to accurate solution vector

Regarding the lossy compressors, we would like to report a significant difference in CRs for the two compressors. SZ achieves a CR of 48 while ZFP only reaches 2. These numbers are very contradicting to what we have seen so far in our experiments, where both compressors achieve similar CRs and never have an order of magnitude difference. This fact can be linked to high serial correlation of the particular stream of elements , which is better exploited by the prediction-based SZ.

To sum up, we regard our approaches as highly accurate in simulating the approximation effect in this particular setup. However, not every setup of solutions and matrices confirms this efficient simulation.

6.3.1 Counter-example

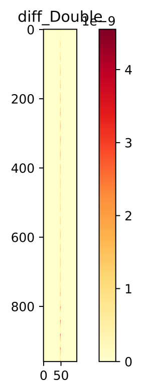
Despite the success of our approaches in the example analyzed above, results have not always been so accurate. For example, for the same matrix NOS3, given the solution vector $x = [2\ \ -1\ \ 2\ \ -1 \ldots 2\ \ -1]^T$, our models fail to simulate the approximation. Table 6.3 introduces the Frobenius norm of errors in solution vectors. It is obvious that the LC methods, on their best configuration, fail to produce error values close to the float approximation. On the contrary, the EI approach replicates similar error values.

Table 6.3: Frobenius norm of error

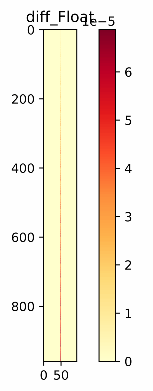
Method	$\|\|x - x_{approx}\|\|_{frob}$
Floats	1.1×10^{-3}
SZ	4.21×10^{-4}
ZFP	5.22×10^{-4}
Normal	1.22×10^{-3}
Uniform	1.06×10^{-3}

Moreover, the heat maps of the absolute error have also been included in Fig. 6.3. Fig. 6.3i proves that the CG with doubles has converged to the actual solution, since error is of order 10^{-9}. Regarding the float approximation, Fig. 6.3ii depicts the error, while our modeling methods errors are shown in Fig. 6.3iii, 6.3iv, 6.3v, 6.3vi.

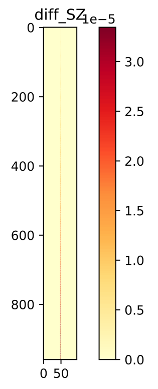
Although ,in general, the error patterns are similar, one must observe that in the middle of the vector, float approximation exhibits a special pattern, where per 40 consecutive elements, large error rapidly diminishes to small error. This pattern is not apparent in any of our modeling attempts. In addition, no model has exactly the same scale of error with float approximation. In particular, the LC methods and the normal distribution have a maximum error of 3.3×10^{-5}, the uniform distribution has a maximum of 4×10^{-5}, while the float has a maximum of only 2.2×10^{-5}.



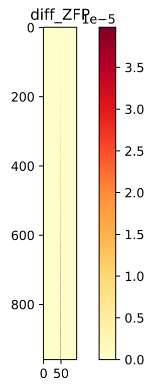
(i) Doubles relative to actual solution



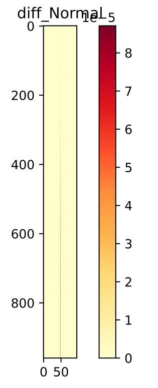
(ii) Float approximation



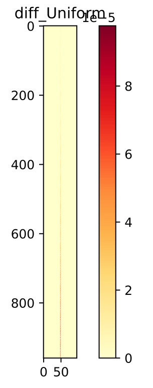
(iii) SZ



(iv) ZFP



(v) Normal



(vi) Uniform

Figure 6.3: Absolute error with relevance to accurate solution vector

6.4 Challenges

Having analyzed two setups of our experiments, it is not clear whether EI and LC can form strong models for the simulation of the approximation. To clarify this, we have concentrated the challenges faced by our approach, which prove the inconsistency of it.

The first problem is the variability of our configuration parameters for different solution vectors. This means that for the same matrix, there is no configuration for the LC and EI methods that produces the same size of error, but rather a trial and error approach is required. There is no obvious pattern on how the parameters should be modified for different solution vectors.

In addition, variability is existent in subsequent EI runs with the same configuration. We have noticed that the scale of error in the solution vector, as well as the Frobenius norm of error, are significantly varying amongst consecutive runs of CG, with the same setup and configuration. This phenomenon diminishes the modeling capabilities of the EI method, as it cannot provide reliable results in this application.

Finally, a significant issue with our method is its' dependency on the input matrix. After experimenting with three different matrices of different range of values, we conclude that our modeling attempts can only be helpful when dealing with low absolute values. In our case, we found that for a matrix with order of values of 10^3 there is strong potential.

6.5 Conclusion

To sum up, taking into account both the successful and unsuccessful cases of modeling, as well as the challenges that arose, we conclude that there is no generic parameter configuration of our models capable of consistently simulating the approximation effect. The issues stated above make the simulation of the approximation effect challenging, even for matrices where the models seem to succeed under certain circumstances.

Chapter 7

Conclusion

7.1 Summary

We have explored an approach to model the effect of approximation techniques on a variety of applications, employing error injection and lossy compression. Our approach provides valuable insight into the effect of the approximation in the case of the stereo vision matching application, yet fails to deliver accuracy and consistency when it comes to scientific computing applications that employ precision scaling as an approximation. We believe the complexity of modeling precision scaling approximations lies in the fact that the error induced has a nonlinear nature, as floating-point representation becomes sparser in higher absolute values. However, through the experiments, valuable insight has been gained on the functionality of the two lossy compressors, SZ and ZFP, reviewing and even extending the findings of existing literature. To summarize our work, we outline our main observations below.

7.2 Keypoints

The most noteworthy observations in our work are listed below:

- Our models in SPS - Stereo detect the areas of the image that will be affected by the approximation, yet do not replicate the details of the distortion accurately (Chapter 3).
- Our models do not have adequate fidelity in simulating precision scaling approximations in matrix multiplication, singular value decomposition and conjugate gradient (Chapters 4, 5, 6).

- We have highlighted the squared manner in which SZ and ZFP work on 2D data (Chapter 4).
- We have illustrated how SZ and ZFP can lead to large errors in sparse matrices (see Chapter 4).
- We identified a configuration of SZ that rounds floating-point values to the nearest integer (see Chapter 4).
- We have verified that the error values that arise by SZ and ZFP application, resemble the uniform and the normal distribution respectively, which is also reported in [36] (see Chapter 5).
- We have illustrated that ZFP introduces larger errors in detailed texture when compressing grayscale images, while both SZ and ZFP do not distort low-textured areas (see Chapter 5).
- We have showcased a framework to compress gray-scale png images, by a factor of 36 for SZ and 27 for ZFP, with negligible distortion (see Chapter 5). The idea is to normalize pixel values to $[0, 1]$ so that the compressors work on floating-point data.
- We have confirmed that SZ introduces errors closer to the maximum error bound specified while ZFP is conservative and only reaches the maximum error at distinct data points.
- We have shown that SZ has consistently higher compression ratio than ZFP for spatially correlated data, in one case even an order of magnitude higher, but also fails to provide any compression for discontinuous data.

7.3 Discussion

Modeling the effect of approximation techniques is challenging. We provided an initial approach based on the idea of distorting the input data of the operation. Although partially successful in an algorithmic approximation, precision scaling approximations proved highly challenging. We believe that a more efficient approach to the precision scaling would be to distort data regarding the absolute values of the elements, albeit our experiments using lossy compression with relative error bound also being unsuccessful. Still, the problem with this approach is that the error introduced by precision scaling is not linearly correlated with the absolute values, as a large value could happen to be represented accurately by a low precision representation. Finally, we believe that the idea of data distortion could provide efficient modeling attempts in other types of approximation, more algorithmic ones, and requires further research.

Bibliography

[1] Sparsh Mittal. Power management techniques for data centers: A survey. April 2014.

[2] John Gantz and David Reinsel. Extracting value from chaos. *IDC IView*, pages 1–12, January 2011.

[3] Hrishav Barua and Dr-Kartick Mondal. Approximate computing: A survey of recent trends—bringing greenness to computing and communication. *The Journal of the Institution of Engineers (India): Electronics and Telecommunication Division*, 100:619–626, June 2019.

[4] Mark Papermaster. Improving the energy efficiency of computing as moore's law slows—are the energy-related benefits from moore's law slowing down?, 2015.[Online].

[5] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Comput. Surv.*, 48(4), March 2016.

[6] Vinay Kumar Chippa, Debabrata Mohapatra, Kaushik Roy, Srimat T. Chakradhar, and Anand Raghunathan. Scalable effort hardware design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(9):2004–2016, 2014.

[7] Beayna Grigorian, Nazanin Farahpour, and Glenn Reinman. Brainiac: Bringing reliable accuracy into neurally-implemented approximate computing. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 615–626, 2015.

[8] Thomas Y. Yeh, Petros Faloutsos, Milos D. Ercegovac, Sanjay J. Patel, and Glenn Reinman. The art of deception: Adaptive precision reduction for area efficient physics acceleration. In *40th Annual IEEE/ACM International Symposium on Microarchitecture*

(MICRO-40 2007), 1-5 December 2007, Chicago, Illinois, USA, pages 394–406. IEEE Computer Society, 2007.

[9] Antonio Roldao-Lopes, Amir Shahzad, George Constantinides, and Eric Kerrigan. More flops or more precision? accuracy parameterizable linear equation solvers for model predictive control. pages 209 – 216, 05 2009.

[10] Mohammad Anam, Paul Whatmough, and Yiannis Andreopoulos. Precision-energy-throughput scaling of generic matrix multiplication and convolution kernels via linear projections. volume 24, pages 21–30, 10 2013.

[11] Peter Düben, Parishkrati, Sreelatha Yenugula, John Augustine, K. Palem, Jeremy Schlachter, Christian Enz, and T. N. Palmer. Opportunities for energy efficient computing: A study of inexact general purpose processors for high-performance and big-data applications. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 764–769, 2015.

[12] Chih-Chieh Hsiao, Slo-Li Chu, and Chen-Yu Chen. Energy-aware hybrid precision selection framework for mobile gpus. *Computers & Graphics*, 37(5):431–444, 2013.

[13] Arnab Raha, Swagath Venkataramani, Vijay Raghunathan, and Anand Raghunathan. Quality configurable reduce-and-rank for energy efficient approximate computing. 2015:665–670, 04 2015.

[14] Abbas Rahimi, Andrea Marongiu, Rajesh K. Gupta, and Luca Benini. A variability-aware openmp environment for efficient execution of accuracy-configurable computation on shared-fpu processor clusters. In *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, 2013.

[15] Byonghyo Shim, S.R. Sridhara, and N.R. Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(5):497–510, 2004.

[16] Ye Tian, Qian Zhang, Ting Wang, Feng Yuan, and Qiang Xu. Approxma: Approximate memory access for dynamic precision scaling. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, page 337–342, New York, NY, USA, 2015. Association for Computing Machinery.

[17] Manuel Le Gallo, Abu Sebastian, Roland Mathis, Matteo Manica, Heiner Giefers, Tomas Tuma, Costas Bekas, Alessandro Curioni, and Evangelos Eleftheriou. Mixed-precision in-memory computing. *Nature Electronics*, 1(4):246–253, Apr 2018.

[18] Matthew Bognar. OMPRNG - Fast Parallel Random Number Generation. `http://www.stat.uiowa.edu/~mbognar/omprng`. Accessed: 2021 - 03 - 12.

[19] Franck Cappello, Sheng Di, Sihuan Li, Xin Liang, Ali Murat Gok, Dingwen Tao, Chun Hong Yoon, Xin-Chuan Wu, Yuri Alexeev, and Frederic T Chong. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications*, 33(6):1201–1220, 2019.

[20] Sheng Di and Franck Cappello. Fast error-bounded lossy hpc data compression with sz. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 730–739, 2016.

[21] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1129–1139, 2017.

[22] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, 2018.

[23] Mathematics and Argonne National Laboratory Computer Science. SZ. `https://github.com/szcompressor/SZ`, 2016.

[24] Peter Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[25] Lawrence Livermore National Laboratory. ZFP. `https://github.com/LLNL/zfp`, 2014.

[26] S. Ilic, Mile Petrovic, Branimir Jaksic, P. Spalevic, L. Lazic, and M. Miloševi'c. Experimental analysis of picture quality after compression by different methods.

[27] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014.

[28] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[29] Panos Koutsovasilis, Christos Kalogirou, Christos Konstantas, Manolis Maroudas, Michalis Spyrou, and Christos D. Antonopoulos. Achee: Evaluating approximate computing and heterogeneity for energy efficiency. *Parallel Computing*, 73:52–67, 2018. Parallel Programming for Resilience and Energy Efficiency.

[30] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. SPS - Stereo. `https://github.com/siposcsaba89/sps-stereo`, 2014.

[31] Peter Lindstrom. ZFP documentation. `https://zfp.readthedocs.io/en/release0.5.3/index.html`, 2018.

[32] National Institute of Standards and Technology. Matrix Market. `https://math.nist.gov/MatrixMarket/`, 2007.

[33] Boeing Computer Services. BCSSTM27: BCS Structural Engineering Matrices Buckling analysis, symmetric half of engine inlet from Boeing jetliner. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc4/bcsstm27.html`.

[34] University of Kentucky. BFW398A: Bounded Finline Dielectric Waveguide. `https://math.nist.gov/MatrixMarket/data/NEP/bfwave/bfw398a.html`.

[35] Imperial College. IMPCOL A: Chemical engineering plant models Heat exchanger network. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/chemimp/impcol_a.html`.

[36] Peter Lindstrom. Error distributions of lossy floating-point compressors. October 2017.

[37] G.H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, March 1970.

[38] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. In *Milestones in Matrix Computation*, 2007.

[39] J. H. Wilkinson and C. Reinsch. Handbook for automatic computation. vol ii, linear algebra. *Mathematics of Computation*, 27:134–151, 1973.

[40] Mathematics Source Library C & ASM. Singular Value Decomposition. `http://www.mymathlib.com/matrices/linearsystems/singular_value.html`, 2004.

[41] Goran Flegar, Florian Scheidegger, Vedran Novaković, Giovani Mariani, Andrés E Tom′ s, A Cristiano I Malossi, and Enrique S Quintana-Ortí. Floatx: A c++ library for customized floating-point arithmetic. *ACM Transactions on Mathematical Software (TOMS)*, 45(4), December 2019.

[42] Goran Flegar, Florian Scheidegger, Vedran Novaković, Giovani Mariani, Andrés E Tom′ s, A Cristiano I Malossi, and Enrique S Quintana-Ortí. FloatX. `https://github.com/oprecomp/FloatX`, 2019.

[43] Wikipedia contributors. Conjugate gradient method. [Online; accessed 30-April-2021].

[44] Georgia Institute of Technology. BCSSTK14: BCS Structural Engineering Matrices Roof of the Omni Coliseum in Atlanta. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc2/bcsstk14.html`, 1982.

[45] Boeing Computer Services. NOS3: Lanczos with partial reorthogonalization - Finite element approximation to biharmonic operator on a rectangular plate with one side fixed and the others free. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/lanpro/nos3.html`, 1982.

[46] Boeing Computer Services. NOS5: Lanczos with partial reorthogonalization 3 story building with attached tower. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/lanpro/nos5.html`, 1982.