COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

HIGH-PERFORMANCE MAGNETIC PARTICLE IMAGING IMAGE RECONSTRUCTION IN GPU USING CUDA

Klaus Natorf Quelhas

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Ricardo Cordeiro de Farias
Solomon Isaac Woods

Rio de Janeiro
Setembro de 2024

HIGH-PERFORMANCE MAGNETIC PARTICLE IMAGING IMAGE RECONSTRUCTION IN GPU USING CUDA

Klaus Natorf Quelhas

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Ricardo Cordeiro de Farias
Solomon Isaac Woods

Aprovada por: Dr. Ricardo Cordeiro de Farias
Dr. Solomon Isaac Woods
Dr. Cláudio Esperança
Dr. Weston Leo Tew
Dr. Esteban Walter Gonzalez Clua
Dr. Wagner Coelho de Albuquerque Pereira

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2024

Natorf Quelhas, Klaus

High-Performance Magnetic Particle Imaging Image Reconstruction in GPU using CUDA/Klaus Natorf Quelhas. – Rio de Janeiro: UFRJ/COPPE, 2024.

XII, 107 p.: il.; 29, 7cm.

Orientadores: Ricardo Cordeiro de Farias

Solomon Isaac Woods

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2024.

Referências Bibliográficas: p. 96 – 107.

1. Magnetic Particle Imaging. 2. Image Reconstruction. 3. Inverse problems. 4. Parallel computing. 5. GPU programming. 6. CUDA. I. Cordeiro de Farias, Ricardo *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

To my father José Carlos
in memoriam

Acknowledgements

The work presented in this thesis would not be possible without the support of a number of people, to whom I would like to express my deepest gratitude:

To my wonderful mother Míriam, who always believed that I would make it.

To my beloved wife Ninive, for her endless support and patience, and for being my source of inspiration and strength.

To my advisor Ricardo Farias, who has been with me since my master's degree, for taking me as his student right after college, and for always being supportive and patient.

To Weston Tew, to whom I am forever grateful for taking me as a guest researcher at NIST and for inviting me to join the *Thermal MagIC* project, which completely changed the course of my life. Thank you for sharing your knowledge, and for being a great supervisor, mentor and friend.

To Solomon Woods, also my advisor during this research, for welcoming me in the *Thermal MagIC* team, for his great leadership during the development of the project, and for guiding me through the development of this project.

To my friend Mark-Alexander Henn, who also became a project teammate, for the numerous discussions about MPI image reconstruction, for the help on the validation of MPI reconstruction models, and for being an amazing proofreader.

To my fellow *Thermal MagIC* teammates Cindi Dennis, Frank Abel, Eduardo Correa, Brianna Correa, Thomas Moffat, Eric Rus, Michael Donahue, Angela Walker, Adam Biacchi, and Thinh Bui. It was a privilege to be part of such a talented team.

To NIST staff Julia Scherschligt, Maria Nadal, Carrie Hart, Howard Yoon, Lisa Poole, Diane Neil and Claire Saundry, and Inmetro staff Valnei Cunha, Renato Machado, Mario Anselmo, Josiane Bemvindo and Helen Jesus, for all the administrative support they provided.

Finally, the research described in this thesis was funded by the Innovations in Measurement Science (IMS) grant awarded by the National Institute of Standards and Technology (NIST), as part of the *Thermal MagIC* project. This work was also supported by the Intituto Nacional de Metrologia, Qualidade e Tecnologia (Inmetro), Brazil's national metrology Institute.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

HIGH-PERFORMANCE MAGNETIC PARTICLE IMAGING IMAGE RECONSTRUCTION IN GPU USING CUDA

Klaus Natorf Quelhas

Setembro/2024

Orientadores: Ricardo Cordeiro de Farias
Solomon Isaac Woods

Programa: Engenharia de Sistemas e Computação

Magnetic Particle Imaging (MPI) é uma nova técnica desenvolvida para detectar nanopartículas magnéticas (magnetic nanoparticles, ou MNPs) no espaço, com diversas aplicações em potencial, desde imagens biomédicas até mapeamento térmico. A reconstrução de imagens é uma parte essencial do MPI, pois é responsável por traduzir os sinais medidos de magnetização gerados pelas MNPs em mapas discretos de concentrações em uma, duas ou três dimensões. O processo de reconstrução requer a solução numérica de grandes sistemas de equações, e a velocidade de computação é um fator limitante para a obtenção de imagens em tempo real com altas exatidão e resolução.

Esta tese apresenta o desenvolvimento de algoritmos paralelos para a reconstrução de imagens de MPI em GPU utilizando a plataforma CUDA. É mostrado como a adoção de algoritmos paralelos pode acelerar as reconstruções em até três ordens de grandeza, em comparação com algoritmos seriais tradicionais executados na CPU. Este documento também descreve o desenvolvimento de um novo algoritmo paralelo para o cálculo da matriz de calibração de MPI, com acelerações da ordem de 37× em comparação com a implementação serial, e como a combinação do cálculo paralelo da matriz de calibração e a reconstrução paralela de imagens podem fornecer medições de MPI flexíveis e em tempo real.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

HIGH-PERFORMANCE MAGNETIC PARTICLE IMAGING IMAGE RECONSTRUCTION IN GPU USING CUDA

Klaus Natorf Quelhas

September/2024

Advisors: Ricardo Cordeiro de Farias
Solomon Isaac Woods

Department: Systems Engineering and Computer Science

Magnetic Particle Imaging (MPI) is a novel technique developed for detecting magnetic nanoparticles (MNPs) in space, with several potential applications, ranging from biomedical imaging to temperature mapping. Image reconstruction is an essential part of MPI, as it is responsible for translating the measured magnetization signals generated by the MNPs into discretized concentration maps in one, two or three dimensions. The reconstruction process requires numerically solving massive systems of equations, and the computation speed is a limitation factor for achieving real-time images with high accuracy and resolution.

This thesis presents the development of parallel algorithms for executing MPI image reconstructions in GPU using the CUDA framework. It is shown how the adoption of the parallel algorithms can speed up the reconstructions by up to three orders of magnitude, in comparison to traditional serial algorithms executed in the CPU. This document also describes the development of a novel parallel algorithm for calculating the MPI calibration matrix, with speedups of the order of 37× in comparison to the serial implementation, and how the combination of the parallel calculation of the calibration matrix and the parallel image reconstruction can provide flexible and real-time MPI measurements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

2D micro and nanoscale imaging techniques comprise some well established technologies, such as optical microscopy and scanning electron microscopy. The first technique is limited by the wavelength of the light, which allows observing samples with dimensions bigger than about 100 nm; the second one obtains higher resolutions, but the cost of it are big investments to purchase and maintain such equipment, and the need of qualified personnel, special infrastructure and previous preparation of samples. Scanning probe microscopy allows obtaining 3D images of surfaces in nanoscale, but it doesn't provide any volumetric information about the sample, only its topography from 2D data.

3D volumetric imaging techniques are useful in applications where information about the internal structure of bodies or objects are required. In this context, computer tomography (CT), positron emission tomography (PET) and magnetic resonance imaging (MRI) constitute three of the most well established technologies that play a very important role in many applications, specially in medicine. The latter has the advantage over the previous two of not using ionizing radiation. Once again, the cost and expertise required to operate these equipment are the main disadvantages of these technologies.

Magnetic particle imaging (MPI) then arises as a promising alternative 3D imaging technique, with potential applications in biomedical imaging and materials research. It relies on the nonlinear response of *magnetic nanoparticles* (MNPs) when exposed to oscillating magnetic fields [1]. MPI allows determining the spatial distribution of these particles, used as tracers, with the important advantage of not employing ionizing radiation like CT and PET, allowing then the detection of particles in vivo for long periods of time without any harm due to radiation exposure. Also, in comparison to the other biomedical imaging techniques, MPI shows high sensitivity, better resolution, excellent response times, and the advantage of allowing particle quantification, since the measured signal is proportional to the concentration of the particles at a certain region inside the sample [2]. These characteristics

turn MPI into a technique suitable for real-time biomedical imaging, with numerous applications.

1.1 MPI Historical overview

MPI was first introduced in an article published in 2005 by Gleich and Weizenecker, from Philips Research group in Hamburg, Germany [1]. In this seminal paper, the authors describe how to employ contrast agents constituted of *superparamagnetic iron oxides* (SPIOs) nanoparticles, such as the ones used in MRI, to obtain stronger signals – and thus better image resolutions – by exploiting their nonlinear magnetization response when exposed to oscillating magnetic fields. They also introduced the concept of the *field-free point* (FFP), which allows measuring the response of the MNPs located in small limited regions of space. Since the signal strength is proportional to the concentration of MNPs within the FFP region, moving the FFP over the 2D or 3D space – either by mechanically moving the sample or moving the FFP itself by employing drive fields – allows obtaining images that describe the distribution of MNPs throughout the sample. The work described two scanner designs that allowed obtaining 1D, 2D or 3D images of samples containing MNPs. Image reconstruction, i.e., obtaining the MNP concentration map from the measured signal, is obtained by numerically solving an inverse problem where the variables are the MNP concentration map (the unknown), the measurement values, as well as the system response to a reference sample (later known as the *Calibration Matrix*, *System Matrix* or *System Function*) obtained through the calibration of the measurement system, leading to images with resolutions of the order of 1 mm.

After demonstrating the feasibility of the new technique, the Philips Research group performed a simulation study in 2007 in which they investigated the impact of changing parameters like the field strength, particle diameter and concentration, as well as the sampling time, over the output image quality (in this case, a 2D image), demonstrating the feasibility of MPI for monitoring biological processes through fast acquisition and encoding, aiming for future human applications [3]. In the sequence, they presented in 2008 experimental results of 2D MPI images obtained in 4 ms – even though showing low signal to noise ratios (SNRs) – with a resolution better than 1 mm [4] and, in 2009, the first results of 3D real-time *in vivo* imaging of the flow of a contrast agent injected into a mouse's bloodstream, showing good level of details – such as its beating heart – demonstrating that the technique is suitable for biomedical applications [5].

The first successful results of this new technique encouraged other groups to join the efforts towards the development of MPI. In 2008, the University of Lübeck's research team published a study where they investigated the use of the singular value

decomposition (SVD) method for MPI image reconstructions employing Tikhonov regularization [6]. In 2009, they simulated different FFP sweep trajectories and their influence over the sweep density, speed and over the quality of the reconstructed images [7], and also proposed a new design for MPI scanners [8]. Later in 2010, they proposed a MPI image reconstruction method based on modeling the measurement system (particle, excitation and receive coils, dimensions, scanning trajectory etc) in the so-called *model-based* approach [9, 10].

In the meantime, both groups studied a different measurement approach, based on experience gained from MRI and CT measurement techniques. In 2008, Philips research group published an article describing a simulation of MPI measurements employing a *field-free line* (FFL) rather than a FFP [11], and in 2010 another approach for generating a FFL, as well as experimental results obtained with a FFL MPI scanner [12] have been presented by the University of Lübeck. Both approaches resulted in better *signal-to-noise ratios* (SNR), when compared to the FFP approach.

In 2009, the Department of Bioengineering of the University of California, Berkeley, published the simulations and experimental results of a so-called *Narrowband MPI scanner*, which employed the concept of intermodulation to reduce the bandwidth requirements and improve the measurement SNR [13]. In the following years, they established the *X-Space* method, which employs the same instrumentation and measurement techniques employed in the Philips method described in [1] and [5], but with different scanning and reconstruction strategies; while the latter employs a method that later would be known as the *system function* method, which is based on a previous calibration of the measurement system, the former relies on the analytical modeling of the system, as well as on a procedure for recovering the fundamental frequency response, lost due to signal filtering [14, 15]. The Berkeley group also designed and implemented a FFL (or *Projection*) MPI scanner [16], demonstrating 3D reconstructions using the new *Projection X-Space* formulation [17].

The Department of Experimental Physics of University of Würzburg, Germany, published in 2014 the concepts, scanner design and experimental results of an new MPI method called *Traveling Wave MPI* (TWMPI), based on an alternative setup of drive field coils, called *dynamic linear gradient array* – or dLGA – that generates not only one but two FFPs simultaneously, with opposite gradient directions [18]. In the following year they showed the design and preliminary results of a micro-MPI (μMPI) scanner, with resolutions lower than 0.1 mm [19], by employing higher field gradients, and in 2019, the results obtained by joining both TWMPI and μMPI concepts while employing an optimized tracer in order to reach resolutions lower than 0.5 mm [20].

Philips Research group investigated other MPI-related techniques, and in 2015

released the first results of a method that allowed detecting different particles within the same sample, allowing them to be displayed in the final image with different colors – a method they called *multi-color* MPI [21]. The method requires prior calibration of the system for each individual particle to be detected.

Magnetorelaxometry (MRX) has also been investigated as an option for magnetic particle imaging. Germany's national metrology institute, PTB (Physikalisch-Technische Bundesanstalt) research team conducted experiments where the 3D distribution of MNPs was estimated by measuring their magnetic relaxation after removing external fields generated by a set of coils, with a measurement system consisted of a large number of SQUID (Superconducting Quantum Interference Device) magnetometers. The results, however, demonstrated that MRX imaging cannot compete with MPI in terms of spatial and temporal resolution [22].

Some studies also employed both MRI and MPI techniques, combined for more detailed biomedical images, since MRI is less sensitive but able to perform body tissue imaging, whereas MPI is more sensitive but it is only able to perform tracer imaging. In [5] and [16], MRI and MPI images were obtained using different scanners, and they were combined by using small MNP sample points that could be detected by both scanners. In 2014, a bimodal MRI-MPI scanner has been presented in [23]. In addition to body imaging, the technique also allows imaging of medical devices, such as guide wires and catheters during interventions [24].

Another recent development is the possibility of moving magnetic material by employing MPI techniques. It has been demonstrated that it is possible to use the gradient field to apply forces over MNPs and remotely manipulate them [25]. This approach could be used, for instance, to deliver drugs to a known position, or to move instruments during a procedure, without the need of direct intervention.

The measured signal-to-noise ratio (SNR) and the spatial resolution of MPI images is highly dependent on the magnetization dynamics of MNPs. Researchers from the University of California, Berkeley, presented a MNP system in which the particles exhibit strong magnetic response, as shown by regular ferromagnetic materials, with the reversibility characteristic of superparamagnetic materials. This new form of magnetism was called *superferromagnetism*, and is resulting from the alignment of MNPs within a fluid to form chains when exposed to certain magnetic fields. It has been demonstrated that superferromagnetic (SFM) particles can yield at least one order of magnitude of gain in the SNR and spatial resolution when used under certain conditions [26–28].

An emerging technique is MPI thermometry, or *T-MPI*. By exploiting the fact that the nonlinear response of MNPs is temperature-dependent, they could be used, in principle to generate temperature maps. This is particularly useful for searching for tumors, inflammations or any lesion that causes a local change in the body

temperature. Moreover, the same effect can be exploited also to generate heat by applying high-frequency magnetic fields over the MNPs inside the body, allowing a targeted magnetic hyperthermia therapy, heating up only the tissue that contains MNPs while preserving healthy tissues [24]. This approach could also be employed to promote targeted drug delivery [29] by, for example, covering the drug with a thermal sensitive coating which would release the drug only when heated to a certain temperature.

The first demonstration of the use of MNPs for temperature measurement dates back to 2009, in an article from the Dartmouth College, USA, where the authors proposed a method based on the ratio of different harmonics that compose the full MNP magnetic response spectrum, through magnetic particle spectroscopy (MPS) [30]. The work showed that the MNPs can be calibrated, and then used to determine temperature with an accuracy of about 0.3 K, being at least an order of magnitude better than the best previous attempts of remote temperature measurements. A similar approach was presented shortly after [31], in which AC and DC fields are combined for obtaining an accuracy of 0.79 K.

Their initiative was followed by Huazhong University of Science and Technology, China, in 2012, when they proposed a method of both concentration and temperature estimation based on the measurement of the MNPs susceptibilities when exposed to several different field intensities, solving then a system of equations for both variables with an accuracy of about 0,5 K [32]. Later in 2014, they explored different calculation methods, reporting accuracies lower than 0,1 K [33].

These first temperatures measurement attempts were performed employing MPS, which is a non-imaging technique. Philips' research group then investigated temperature measurements with MNPs, and proposed in 2016 an approach based on their recently proposed *multi-color* MPI method [34]. This was accomplished by calibrating the system with the MNPs at different temperatures, assuming that the MNPs at different temperatures show different properties so the signals can be separated and interpolated in order to calculate the temperature of the sample. Through this method, they were able to reconstruct a temperature map of a sample with an accuracy of about 0.5 K using a commercial MPI scanner.

PTB also investigated the influence of the temperature over the magnetization of different types of MNP suspensions using both MPS and MPI (but without imaging) systems, reporting strong agreement between measurements performed with both methods, and confirming the impact of temperature over the MNPs response spectra, specially the ratio between the fifth and third harmonic amplitudes, being the impact dependent on the particle and on the media in which it is diluted (air, liquid or solid) [35].

Another study performed at the Technical University of Braunschweig, Germany,

in 2018 employed a MPS system together with a mechanical scanner in order to perform MNP concentration and temperature imaging with an accuracy of about 1 K, stating that temperature accuracy may improve with the excitation frequency, and that even though temperature measurement is not dependent on the MNP concentration, since the harmonic ratio remains unchanged, it affects the measurement's SNR and thus temperature accuracy [29].

The National Institute of Standards and Technology (NIST, USA) is currently developing a project that aims to establish the instrumentation and techniques for remotely mapping and controlling temperature in 3D employing MPI [36] techniques. The *Thermal MagIC* project includes the development of novel MPI scanners, high-precision magnetic detectors, as well as MNP tracers tailored to maximize the thermal sensitivity of the measured signals [37–41]. The project also includes the development of MPI simulation and reconstruction methods [42–44] for allowing 3D temperature imaging, including the first uncertainty estimation for MPI reconstructions [45]. Other examples of temperature measurement techniques can also be found in the references [46–50].

As MPI studies advanced, researchers have also started to focus on two different but very important aspects: standardization and accessibility. In 2016, the first version of what can be considered a first attempt to create a standard for MPI data file format was published by a group of researches from several German institutes, being periodically revised and upgraded since then [51]. This initiative has been followed by the publication of an open-source software package written in Julia in 2019 [52] and the release of publicly available MPI measurement data to support the development of new reconstruction strategies, as well as CAD schematics of the phantoms used as samples for the measurements also in 2019 [53]. More recently, another initiative led by Harvard University and the Massachusetts Institute of Technology (MIT) introduced an open-source project, called OS-MPI, that aimed supporting new research teams to join the MPI field, by providing a "starter kit": schematics of a low-cost FFL MPI scanner, a spectroscopy platform and an electromagnet winding jig, including several designs, cost estimates and discussions regarding the chosen designs [54].

The main focus of MPI research is on its great potential as a biomedical imaging tool. Currently, commercial MPI setups are limited to pre-clinical scanners employed for imaging small rodents and phantoms. Recently a few head-sized MPI scanners were developed [55, 56], but scaling up MPI to human size is still challenging. Limiting factors include the increase of power needed for generating gradient fields strong enough for obtaining high spatial resolutions, the cooling that such increase of power demands, and concerns about the increase of the specific absorption rate (SAR) and magnetostimulation resulting from this increase of power [57].

MPI is still a relatively young technology, and is evolving at a remarkable speed. Additional research efforts focus on new MNP tracers, new applications and new image reconstruction algorithms [58, 59].

1.2 Image reconstruction: speed versus accuracy

Regardless of the method chosen for the signal measurement and data acquisition, image reconstruction plays a critical role in MPI. The measured signals have to be translated into discretized representations of the distribution of MNPs in space (or their temperatures, in the case of T-MPI). The reconstruction step involves the solution of systems of equations for optimal accuracy, which have the form $Ax = b$, where A is a calibration matrix or a convolution kernel representing a *point spread function* (PSF) scanning the sample, b is the measured signal vector and x is the unknown concentration vector. Depending on the scanning frequency, the sampling rate and the resolution of the resulting image, the matrix A can grow very large and occupy several gigabytes in the computer's memory. In addition, the measured signals contain noise, so the system is Ill-posed and its solution requires the use of regularization schemes such as the Tikhonov functional, and the reconstruction can take from minutes to hours to be executed.

Over the years that followed the development of MPI, several image reconstruction algorithms have been studied. Amongst the most popular ones, we can cite the singular value decomposition (SVD), the Kaczmarz algorithm, and the conjugate gradient normal residual (CGNR) method [6, 12, 15], implemented to deliver the most accurate reconstructions more efficiently. Usually the reconstruction is a trade-off between accuracy and speed, since more accurate reconstructions require bigger systems of equations and/or more iterations, compromising the execution times. Reconstruction speed and real-time applications have been the focus of several studies [5, 60–63], and after almost two decades of research, the Kaczmarz algorithm remains the most adopted one for MPI reconstructions, due to its fast convergence.

Even though real-time image reconstructions is clearly one of the main objectives in MPI research [5], no high-performance parallel approach had been proposed for MPI image reconstruction until the development of this work. High-performance parallel image reconstruction approaches are already a reality for several other biomedical imaging techniques [64], like MRI [65, 66] and CT [67], but that was not true for MPI yet. A number of authors acknowledged the potential of parallel approaches, suggesting C/C++ parallel implementations of the reconstruction algorithms using either OpenMP for multi-core CPUs or CUDA (*Compute Unified Device Architecture*, developed by Nvidia) for GPUs [63, 68, 69], and this potential

is explored in this work.

In early 2023, the first parallel implementations of MPI image reconstruction algorithms were reported in [70], as part of the research developed for this thesis. In the work, it was shown that the parallel CGNR algorithm can be executed 10× faster than the serial Kaczmarz algorithm, when reconstructing 2D MPI images from simulated data. It was also shown in that work that the parallel implementation of the Kaczmarz algorithm does not result in any acceleration, due to the highly sequential nature of the algorithm. Later in the same year, the poor performance of the parallel Kaczmarz algorithm was confirmed in a parallel implementation for accelerating multi-frame MPI reconstructions [71]. Finally, the parallel implementation of the CGNR algorithm for the reconstruction of 3D temperature images using the multi-color method was presented in [72].

1.3 Objectives

The main objective of this work is the development of strategies for accelerating MPI image reconstructions by implementing a number of algorithms to be executed in parallel in GPUs using the CUDA architecture. The parallel implementation is not limited to the reconstruction algorithms, but also includes the calculation of the model-based calibration matrix, which requires the proper modelling of MPI measurement systems. It will be shown how the parallelization of the calculation of the calibration matrix and a combination of it with parallel reconstruction algorithms can yield fast, accurate and memory efficient MPI image reconstructions.

1.4 Structure of the text

This document is organized as follows: chapter 2 presents the fundamentals of MPI, chapter 3 describes the reconstruction algorithms studied in this work, chapter 4 introduces parallel programming using the CUDA architecture, while chapter 5 describes the development stages of this research, chapter 6 presents and discusses the results and chapter 7 concludes this thesis.

Chapter 2

MPI principles

2.1 Magnetic nanoparticles and superparamagnetism

The first principle that supports MPI is the particular magnetic response of certain MNPs to an applied magnetic field. These particles are said to be *superparamagnetic*, meaning that they behave like ordinary paramagnetic materials, but with more pronounced magnetization responses. Under no external field, the magnetization directions of the particles (or their *magnetic moment* directions) are randomly oriented so the resulting magnetization - the total magnetic moment per unit volume - is zero [2]. Once an external field is applied, the moments progressively align with it as the field magnitude increases, until they reach saturation when all the moments are fully aligned. Figure 2.1 illustrates how the particles align with an increasing magnetic field. As the vertical applied field H increases, the particles align more with the field direction and the magnetization M increases. In the figure, M_S is the particle's saturation magnetization, and H_S is the field necessary to align the particles up to 99% of saturation. Once the external field is removed, the particles' moments scatter again and the magnetization returns to zero.

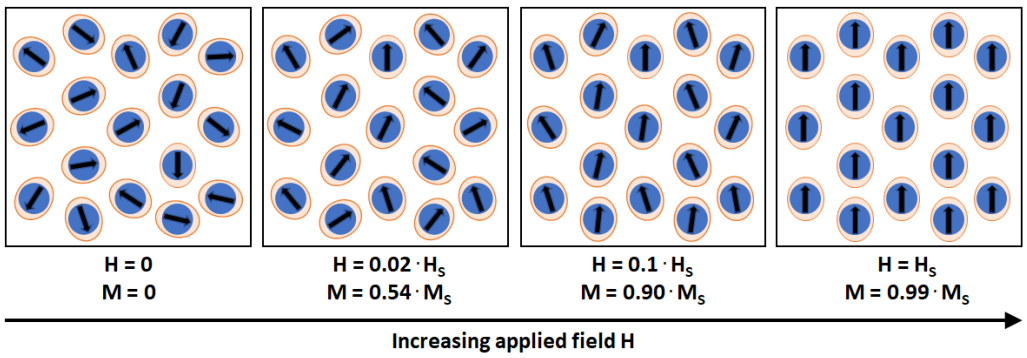

Figure 2.1: Illustration of the superparamagnetic behaviour of magnetic nanoparticles. The arrows show the orientation of the MNPs.

The particles are constituted by a magnetic core – usually iron-oxides like magnetite, which is why those particles are also know as *superparamagnetic iron oxides* (SPIOs). These cores are surrounded by an organic and magnetically neutral coating that prevents the particles from aggregating [2] due to mutual interactions. When the particle cores are sufficiently small (usually of the order of 20 nm or smaller), the core is composed of a single domain (i.e., there are no domain walls) and the thermal agitation is sufficiently strong to promote spontaneous moment reversals (and hence the superparamagnetic behaviour). The magnetization of such MNPs is well described by [73]:

$$\vec{M}(\vec{H}, T) = \rho m \mathcal{L}\left(\frac{\mu_0 m|\vec{H}|}{kT}\right)\frac{\vec{H}}{|\vec{H}|} \tag{2.1}$$

where $\mathcal{L}$ denotes the *Langevin function*:

$$\mathcal{L}(x) = \coth(x) - \frac{1}{x} \tag{2.2}$$

and ρ is the particle concentration in (m^{-3}), μ_0 is the vacuum permeability in $(\mathrm{N/A}^2)$, m is the particle's magnetic moment in $(\mathrm{A}\cdot\mathrm{m}^2)$, H is the applied field in (A/m), k is the Boltzmann constant in (J/K) and T is the absolute temperature in (K). The magnitude of the magnetic moment is given by:

$$m = M_{sat}\left(\frac{\pi}{6}\right)d^3 \tag{2.3}$$

where M_{sat} is the material's bulk saturation magnetization in (A/m) and d is the particle core diameter in (m). The magnetization according to equation 2.1 can then be seen as a balance between the magnetic Zeeman energy $\mu_0 mH$ that tries to align the particles with the field and the thermal energy kT that tries to scatter them.

Figure 2.2 shows a plot of the magnetization as function of the applied field according to the Langevin function. In the dynamic region near zero applied field, the magnetization changes almost linearly. As the applied field increases in magnitude, the plot moves to the saturation region, and the changes in the magnetization are significantly smaller.

Due to the non-linear magnetization response of the MNPs, if an oscillating excitation field is applied, the particles will exhibit a particular magnetization over time, $M(t)$, which will contain not only the excitation frequency, but also a series of harmonics, which can easily be analyzed in the Fourier domain, and separated by filtering. Usually the excitation field intensity is several orders of magnitude stronger than the magnetization response, so the excitation frequency (or base frequency) signal is often filtered out and the remaining signal is analyzed as the MNP response. Unfortunately, the magnetization response also contains information at the base

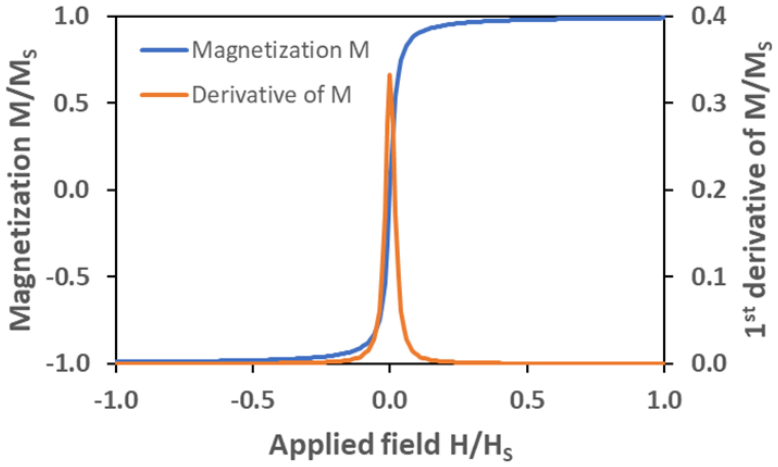

Figure 2.2: Magnetization as function of the applied field according to the Langevin function (blue), and the derivative of the magnetization (orange) representing the point spread function (PSF).

frequency, which means that part of the information is lost during the filtering, but there are some approaches to deal with this challenge.

On the other hand, if the MNPs are already exposed to a static magnetic field, with enough intensity to saturate them, then they will exhibit no harmonic response when exposed to an oscillating field, provided that the sum of the superimposed fields remains above the saturation limit. Therefore, controlling the offset field and consequently the saturation level means being able to turn on and off the harmonic response, and is the first key to Magnetic Particle Imaging [1].

2.2 Selection field, field-free point and field-free line

The second key element for MPI is the concept of *selection field*. As seen before, if the whole sample is subject to an offset field, it is possible to control the saturation level of the sample and thus its harmonic response. By employing two identical and opposite constant magnetic fields, generated by two electromagnets driven by DC currents, or two cylindrical permanent magnets placed in opposite directions, it is possible to generate a static gradient field that vanishes at the center. Such a field is called *selection field* and the region where the field is null is called *field free point* (FFP). The particles inside the FFP will exhibit their characteristic nonlinear magnetic response when an oscillating excitation field is applied, while the particles outside of it will remain saturated. Figure 2.3 illustrates this approach. This allows selectively measuring the signals generated by particles within a limited region of space, instead of the whole sample. Spatial encoding, i.e., the FFP movement relative to the sample in one, two or three dimensions, covering the desired field of view (FOV), is achieved by either mechanically moving the sample or magnetically

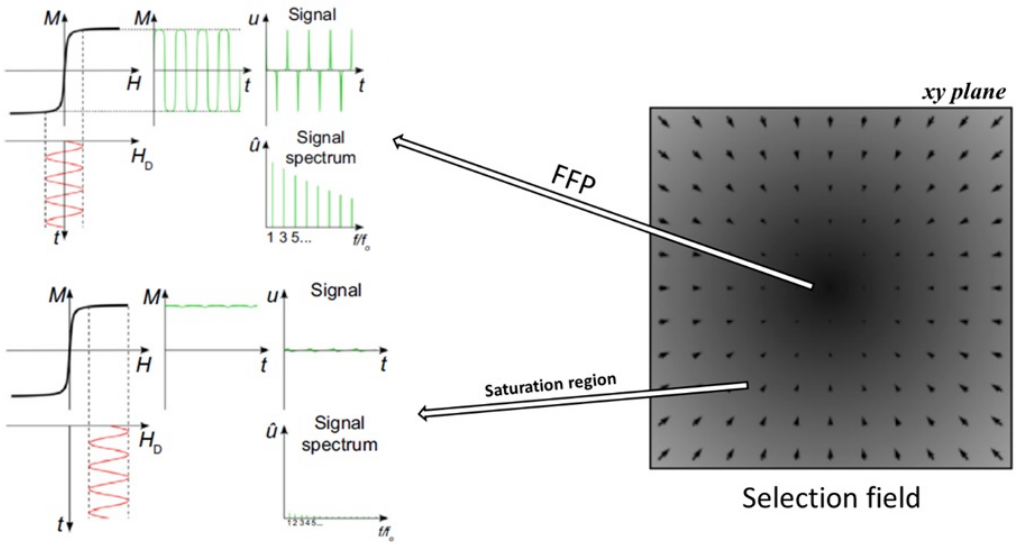

Figure 2.3: MPI FFP operating principle. Inside the FFP the MNPs will exhibit a magnetic response to an oscillating excitation field, whereas the particles outside it will not since they are all saturated by the gradient selection field (adapted from [12] and [74]).

moving the FFP using orthogonal sets of *drive fields* – homogeneous fields generated by two identical coils aligned with the same direction that superimpose with the selection field, changing then the position where the sum of intensities reaches zero, allowing then one, two or three-dimensional coverage, and consequently a map of particle concentration over the space [73]. A number of FFP trajectory patterns can be implemented for MPI scanning through the use of drive field coils [7], but the most widely employed are Cartesian (mostly used with x-space reconstructions, see 3.1.3) and Lissajous (mostly used with system function reconstructions, see 3.1.1) trajectories.

The use of drive fields to move the FFP yields a drastic increase in the scanning speed, allowing for real-time MPI signal acquisition [5]. Also, the fast movement of the FFP itself causes a rapid change in the MNPs magnetization in the covered regions, resulting in a measurable magnetic response, seen as harmonics of the drive field scanning frequency, turning the use of modulation fields unnecessary. Figure 2.4 illustrates a FFP measurement setup.

In addition to the FFP, another configuration that is widely used is the so-called *field-free line*, or FFL. Initially proposed in a simulation study in 2008 [11], its feasibility was demonstrated in 2010 [12]. In this approach, instead of generating a single region with zero intensity in the middle of a gradient field, a linear region is created, inside of which the field intensity is also zero. This region can be generated in a similar way the FFP is generated, but instead of using a pair of coils or permanent magnets in opposite directions, two orthogonally aligned pairs are used to generate

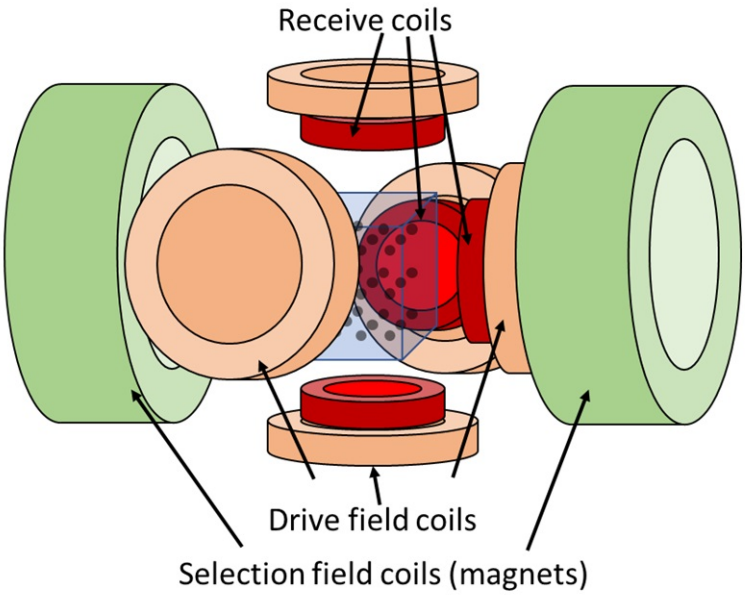

Figure 2.4: 3D MPI FFP setup example.

a gradient field that vanishes in a central linear region, resulting in a FFL that is orthogonal to both pairs main axis. Alternatively, a FFL can be also generated by a pair of rectangular permanent magnets, as shown in [16]

The greatest advantage of using a FFL over a FFP is that the signal-to-noise ratio (SNR) increases by a factor of $\sqrt{N}$, where N is the number of voxels covered by the FFL [11]. This leads to better image resolutions, as well as better results with lower MNP concentrations. However, the FFL should be rotated with relation to the sample in order to provide the extra degree of freedom necessary for image reconstructions, which requires accurately moving big and heavy sets of magnets.

One particular configuration is the so-called *travelling wave magnetic particle imaging* (TWMPI) method. It is based on the concept of what the authors named *dynamic Linear Gradient Array* - dLGA, constituted by an array of coils driven by sinusoidal currents. Each coil's current has a specific phase shift with relation to the other in such way that a full current wavelength is described along the array's length. As the currents of the coils oscillate, not only one, but two FFPs are generated, with opposite gradient directions and separated by half of the array's length [18]. Figure 2.5 illustrates the TWMPI design.

One of the advantages of employing two FFPs instead of one is the possibility of generating twice the amount of data due to the opposite gradient directions. When using drive fields to displace the two FFPs in the perpendicular directions, they move in opposite directions, scanning then two lines at a time. It happens because the FFPs are created by gradient fields with opposite directions. Figure 2.5 illustrates the two FFPs within the dLGA.

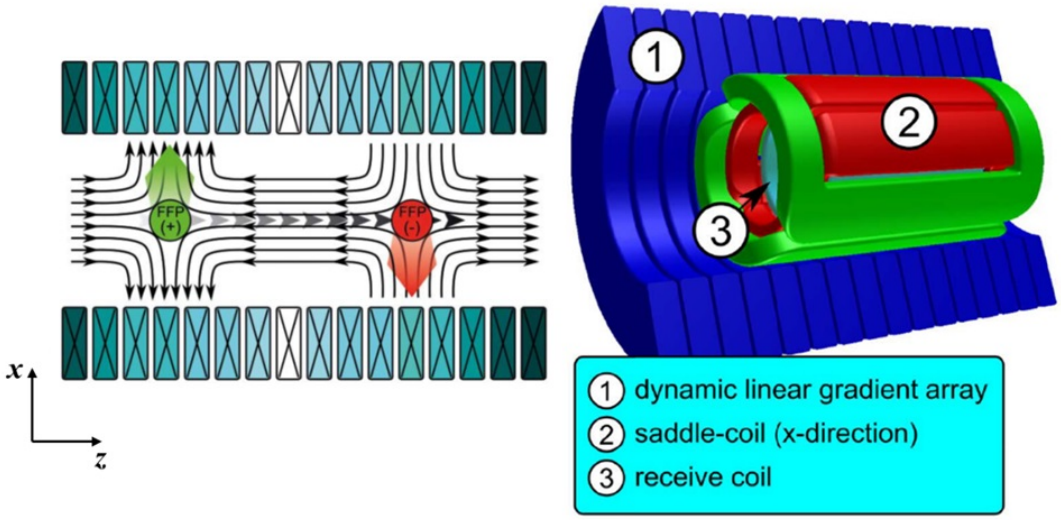

Figure 2.5: TWMPI design (adapted from [18] and [20]).

2.3 Signal detection

In order to perform the signal detection, inductive coils are employed. They obey Faraday's law of induction:

$$u(t) = \oint_{\partial S} \vec{E}(l) \cdot dl = -\frac{d}{dt} \int_S \vec{B}(\vec{r}, t) \cdot d\vec{A} \tag{2.4}$$

where $\vec{B}(\vec{r}, t)$ is the magnetic flux density in (T) at the position $\vec{r}$, integrated over the coil's surface S, with normal vector $d\vec{A}$ expressed in (m^2). The flux density $\vec{B}$ is composed of the excitation field flux density $\vec{B_E}$ and the flux density generated by the MNP's magnetization $\vec{B_M}$. The signal generated by the MNPs is given by:

$$u(t) = -\frac{d}{dt} \int_V \vec{M}(\vec{r}, t) \cdot \mathbf{R}(\vec{r}) d^3 r \tag{2.5}$$

where $\mathbf{R}(\vec{r})$ is the receive coil sensitivity in (T/A), which is equivalent to the magnetic flux density generated by the receive coil per unit current at the position $\vec{r}$, in other words:

$$\mathbf{R}(\vec{r}) = \frac{\vec{B}(I, \vec{r})}{I} = \frac{\mu_0}{4\pi} \int \frac{d\vec{l} \times (\vec{r} - \vec{l})}{\mid \vec{r} - \vec{l} \mid^2} \tag{2.6}$$

where $\vec{r}$ and $\vec{l}$ are the vectors pointing to the MNPs location and the coil's center, respectively. The sensitivity is conveniently employed for modeling the signal generation based on the principle of reciprocity [2]. Drive fields steer the FFP over the FOV, so the resulting applied field over the sample can be expressed as a function of the FFP trajectory $r(t)$:

$$\vec{H}(\vec{x},t) = \mathbf{G}(\vec{x})\left(\vec{x} - \vec{r}(t)\right), \tag{2.7}$$

where $\mathbf{G}$ is the static gradient field in (T/m). The time-dependent signal generated by the particles can then be expressed as a convolution of the time derivative of the Langevin function over the spatial distribution of MNPs in space:

$$\vec{s}(t) = m \int_{\mathbb{R}^3} \rho(\vec{x})\mathbf{R}(\vec{x}) \frac{d}{dt} \mathcal{L}\left(\frac{\mu_0 m ||\vec{H}(\vec{x},t)||}{kT}\right) \frac{\vec{H}(\vec{x},t)}{||\mathbf{H}(\vec{x},t)||} dx \tag{2.8}$$

Since multiple coils can be used for the signal detection, in general oriented in orthogonal directions for 2D and 3D measurements, the signal is then expressed as a vector-valued function. Equation 2.8 can be seen as a convolution that denotes the influence of the point spread function (PSF), here represented by the first derivative of the Langevin function, as shown in figure 2.2, over the sample being measured. As the FFP is rapidly steered over the FOV, the changes in magnetization at the FFP and in the surroundings of it are detected by the inductive coils and therefore images obtained by direct gridding of the measured signals (as shown in [14, 75, 76]) are blurred. The width of the PSF, and therefore the amount of blurring observed, is inversely proportional to both the gradient field strength and the particle's magnetic moment magnitude. The PSF blurring can be removed by deconvolution, which requires numerically solving a linear system of equations, as will be demonstrated in chapter 3.

2.4 Excitation field feedthrough and signal filtering

Equation 2.8 describes the signal induced by the MNPs over the receive coils. However, as described in 2.3, the flux density $\vec{B}$ over the receive coil is composed of the excitation field flux density $\vec{B_E}$ and the flux density generated by the MNP's magnetization $\vec{B_M}$. The signal induced by the drive field over the receive coil is then

$$\vec{s}_d(t) = -\frac{d}{dt} \int_S \vec{B}_E(\vec{r},t) \cdot d\vec{A} \tag{2.9}$$

which, for orthogonally-oriented drive and receive coils and uniform drive fields can be reduced to

$$\vec{s}_d(t) = -\frac{d}{dt} \vec{B}_E(t) \cdot A \tag{2.10}$$

Therefore, the measured signal is the result of the superposition of the signals induced by the particles and by the drive field. The signals due to the excitation field are, however, several orders of magnitude stronger than the signals due to the particles, and consequently the desired MNP signal is usually lost in the digitizing

process due to truncation. Therefore, the signal induced by the drive field must be removed before the signal is digitized, in order to allow the detection of the MNP response only.

One strategy could be the use of a compensation signal, but since the signals induced by the drive fields are several orders of magnitude bigger than the ones induced by the MNPs, implementing such compensation accurately is extremely difficult. An equivalent approach would be to simultaneously measure the background signal with a coil away from the particles (but still inside the uniform drive field region) and then subtract it from the full signal by connecting the coils in series with opposite polarities, in a gradiometric configuration, before digitizing the signal. Even though this strategy yields better results, this compensation method is also challenging in 3D, due to the complex coupling between the three drive fields and the receive coils [2].

The most common method for dealing with the drive field feedthrough is filtering the measured signals. Since the excitation fields are mostly sinusoidal, they have only one frequency component in the spectrum, the *base frequency*. The signals generated by the MNPs, on the other hand, are composed of several frequencies, or harmonics. Therefore, employing high-pass or band-stop analog filters would remove the base frequency and the remaining signal would contain only the MNP harmonics. The drawback of this method is that the MNP harmonic response also contains a component at the base frequency, and then a significant part of the signal is lost during the filtering process. Methods for dealing with the loss of the base frequency will be discussed in chapter 3.

2.5 Particle dynamics and relaxation

The process by which the MNPs align with the external field is not instantaneous as the ideal model suggests. Instead, it takes place in a finite time, and this *relaxation time* depends on a series of factors which include the core material and its volume, the hydrodynamic diameter (the diameter of the MNP's coating layer), and the viscosity of the media where the MNPs are immersed [2]. The alignment of the particles with the applied field follows two mechanisms. In the *Brownian relaxation*, the particles physically rotate to align with the field, and the characteristic relaxation time is given by:

$$\tau_B = \frac{3\eta V_H}{kT} \tag{2.11}$$

where η is the fluid viscosity in $(\mathrm{Pa} \cdot \mathrm{s})$ and V_H is the particle's hydrodynamic volume in (m^3) [77]. In the *Néel relaxation*, on the other hand, only the particle's magnetic

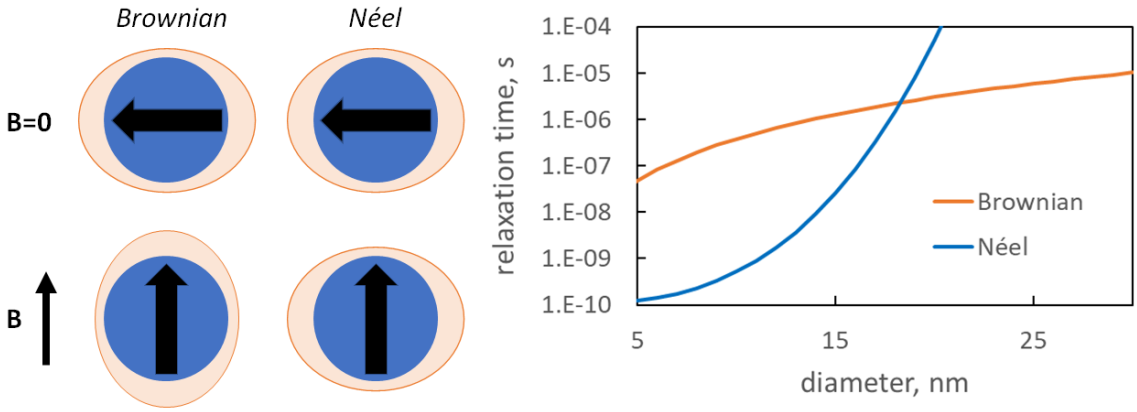

Figure 2.6: Differences between the *Brownian* (left) and *Néel* (right) relaxation mechanisms. The plot in the right shows how both relaxation times scale with different diameters for magnetite.

moment rotates to align with the field. In this case, the relaxation time is given by:

$$\tau_N = \tau_0 \cdot e^{KV_C/kT} \tag{2.12}$$

where τ_0 is the *attempt time* in (s), K is anisotropy constant in (J/m^3) and V_C is the particle's core volume in (m^3). Figure 2.6 illustrates how the two mechanisms work, and how the relaxation times relate to the particle size. Notice that both equations 2.11 and 2.12 show the temperature-dependence of the relaxation times, denoted by the thermal energy kT. In solid media, the particles are fixed in space, and only Néel relaxation is possible. In liquid media, both mechanisms take place simultaneously, and usually the fastest one prevails. When the relaxation times are of the same order of magnitude, the resulting relaxation can be expressed as a combination of both mechanisms, and the *effective relaxation time* can be expressed as:

$$\tau_{eff} = \frac{\tau_B \cdot \tau_N}{\tau_B + \tau_N} \tag{2.13}$$

Particle relaxation has a strong influence on the measured signals, and thus on the reconstructed images. Relaxation has been modeled for x-space image reconstructions as a convolution of the measured signal with a relaxation kernel [78], which in principle can be removed through deconvolution. A number of studies investigate the numerical modeling of both Néel and Brownian relaxation mechanisms, in order to incorporate them in model-based MPI signal generation and reconstruction methods [79–81]. It has been shown that relaxation shifts the PSF, and this is key for *multi-color* MPI [21] and T-MPI [72].

The MNP's dynamics properties are characterized by means of magnetic particle spectrometry (MPS). Sometimes referred as a "zero-dimensional" MPI technique, since it is not an imaging technique (even though some imaging studies were per-

formed with MPS, see [29]), it is more often used to perform characterization of MNPs to be used in MPI applications. The measurement is performed by applying an oscillating homogeneous field over the sample and then measuring the MNP magnetization response. The signals measured under different excitation frequencies allow determining the sample's dynamic properties.

Chapter 3

Image reconstruction

Image reconstruction is an essential step in magnetic particle imaging, since it allows translating the measured voltage signals into discretized representations of the spatial distribution of MNPs within a sample. Since the invention of MPI, a number of reconstruction methods have been developed. Those methods can be divided in algebraic and analytical, and/or in calibration-based and model-based.

Over the almost two decades of MPI development, two modeling and reconstruction methods became dominant within the MPI comunity. The *system function* method is an algebraic method that usually relies on the calibration of the measurement system, but can also be employed in a model-based approach. The *x-space* method, on the other hand, is an analytical and model-based method that relies on a simplification of the signal generation model, that allows a fast approximation of the MNP spatial distribution. The methods are discussed in more details in the following sections.

3.1 Modeling methods

3.1.1 System function MPI

This is the first approach employed by Gleich and Weizenecker [1, 3], and has been widely used by nearly all German institutes, where MPI research was more concentrated in its early days. The concentration reconstruction is based on the so-called *system function*, or *system matrix*, which is a matrix that contains the characteristic response of the measurement system to a a particular MNP sample. The image reconstruction is performed by finding the numerical solution of the following system of equations:

$$S \cdot c = v \tag{3.1}$$

where S is the $M \times N$ system function (or system matrix, or calibration matrix), c is the unknown concentration vector of size N, and v is the measured signal vector of size M. In a measurement sequence, the FFP is moved either mechanically or magnetically over the sample following a specific trajectory, while the induced signals are measured in regular intervals, such that the number of rows M of the calibration matrix represents the number of measurements per full scan, and the number of columns N represents the number of elements (voxels) of the 1D, 2D or 3D imaging grid.

Both the measurement vector v and the system matrix S may contain data generated by multiple receive coils – adding orthogonally-oriented receive coils improves the signal sensitivity in each coil's direction, at the expense of additional data and computational effort – in this case the data are concatenated in the same vector/matrix. Depending on the desired resolution and the measurement parameters, the system matrix can be extremely large, occupying several GB of data in the memory in some cases [53]. This is a major drawback when adopting this method, since the amount of available memory required to handle this data is considerably large, which is a limitation factor for end users. Additionally, the computational effort necessary to solve such massive systems of equations results in execution times that can range from minutes to hours. Since the measurement vector contains noise (as well as the system function, in the case of the calibration-based method, see ahead), solving the system for obtaining the particle distribution is an ill-posed problem [6], and regularization is required in order to achieve stable solutions (see ahead some reconstruction methods). The calibration matrix can be obtained by two methods, as follows.

Calibration-based

This is the method that results in the most accurate system functions and state-of-the-art reconstructions, and has been successfully employed by several authors [1–12, 73]. The determination of the system matrix consists of recording the signals generated by a small MNP reference sample, as it is placed at different positions within the FOV, representing the *voxels* of the image to be reconstructed. This sample is usually small enough to fit inside a voxel and concentrated enough to be considered an approximation of an impulse function (and this is why it is often referred as the *delta sample* in the literature [1, 3]). At each calibration position, a full measurement sequence is performed, scanning all the field of view (FOV) as the FFP is steered over it; the measured data is transformed from time to frequency domain through a Fourier transform (this allows for filtering undesired frequencies, such as the excitation frequency and higher harmonics with lower SNR), and the data are then stored as a column in the matrix. The process is repeated until all

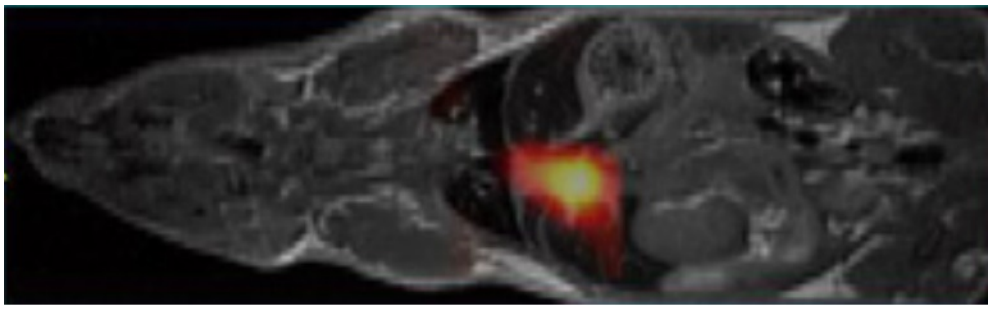

Figure 3.1: MPI measurement (color) of a living mouse injected with MNPs. Grayscale image is an MRI acquisition overlapped in order to identify the path of the MNPs inside the body [5]

positions are measured (i.e., all the system matrix's columns). The delta sample can be moved along the grid manually or by using a robot (the faster and most frequently used method).

The greatest advantage of this approach – responsible for the most accurate results – is the fact that the measured calibration matrix intrinsically accounts for the non-ideal behaviour of the whole measurement system, including imperfections or misalignment of components such as field coils and magnets, receive coils, electronics or even the particle's dynamic magnetic response, that could lead to any departure from the expected measurement result. Figure 3.1 exhibits the results of a measurement performed with this method. Another major advantage of the system function method is that it can be employed the same way using either FFP or FFL - each method requires a specific calibration, as the system's response would be different.

Among the disadvantages, the most significant one by far is the tedious calibration process. Consider, for example, a sampling sequence time of 1 second, followed by a repositioning procedure that takes 1 s in average. For a 10 x 10 x 10 3D grid, the calibration time would be about 30 minutes; for a 30 x 30 x 30 grid, the calibration is 15 hours; for a 100 x 100 x 100 grid, the calibration takes more than 23 days – clearly scaling is a major issue when adopting a calibration-based system function measurement method. In addition, the calibration matrix describes the characteristic response of the MPI system to a specific set of parameters that includes the coils, detectors, electronics, filters, FOV, scanning sequence and sampling, the MNP's dynamic response and even the sample's temperature. This means the system is "locked" for all these parameters, and every measurement must be performed in exactly the same way the calibration was done. One single change in any parameter would require a new system calibration in order to keep optimum accuracy.

Model-based

This method was proposed by Knopp et al [9, 10], based on a simulation study performed in [3], and is based on modeling the whole MPI system, including the magnetic fields (selection field, drive fields, focus fields etc), the magnetic response of the MNPs, as well as the receive chain and the system electronics (filters, amplifiers, ADCs etc). It is known to provide worse results in comparison to the calibration-based method, but it has, as the main advantages, the significantly reduced time to obtaining the system matrix, as well as the possibility of generating parts of the system function during the reconstruction, avoiding the storage of big matrices in the memory (at the expense of the computation time). Also, the model-based system function is not limited by the voxel size employed during measurements, which means it is straightforward to generate system functions with higher resolutions. The main drawback is the lack of information about the imperfections of the system, especially about the particle dynamic response, which is often modeled using the Langevin function – real particles do not behave exactly as described by this model. More accurate MNP behavior models would address this issue [24], and more recently a mathematical model for the MPI signal generation that accounts for both Néel and Brownian relaxation mechanisms has been proposed in [81].

In order to benefit from the advantages and overcome the challenges of both calibration-based and model-based methods, a hybrid solution was proposed by Gruettner et al [82]. The method consists of modeling the magnetic fields, since they can be more accurately modeled, and measuring the MNPs magnetization response in a separate step. The main advantages of this method is the fact that the calibration sample does not need to be moved over the FOV and it can be measured with higher SNR, since it is possible to use larger MNP samples even when modeling system matrices with high spatial resolutions. The results showed that the hybrid approach provided better results than the full model-based method, but the calibration-based system function method still delivers the most accurate reconstructions.

3.1.2 Multi-color MPI

First demonstrated by Rahmer et al. in 2015, the method is based on the fact that different particles, particles immersed in different medias with different viscosities, or even particles at different temperatures will exhibit different magnetization responses [21]. They were able to demonstrate that it is possible to perform a numerical reconstruction that allows detecting different particles within the same sample, and display them in a way that the user is able to distinguish them, usually by plotting them in different colors, hence the name *multi-color* MPI.

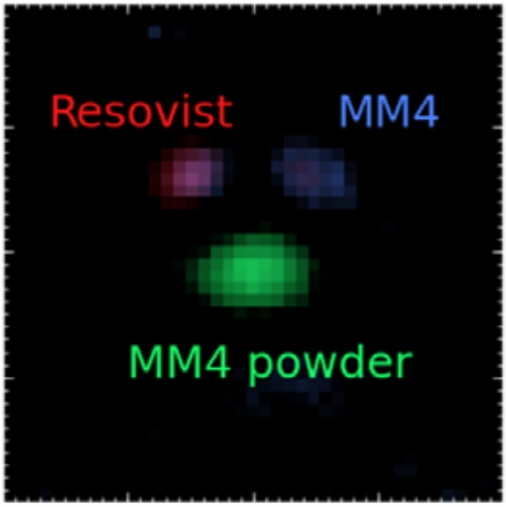

Figure 3.2: Multi-color MPI measurement. The same particle (MM4) was used both dry (green) and diluted in liquid suspension (blue) in order to demonstrate the different responses in different media. Adapted from Rahmer *et al.* (2015) [21].

The technique is based on the system function methodology. It relies on the principle of superposition of the MPI signals, and consists of generating one full system matrix per particle, and concatenating them for building an extended system of equations. Consider, for instance, that three different particles, $p1$, $p2$ and $p3$ are to be measured within a sample, so equation 3.1 can be rewritten as:

$$S_{p1} \cdot c_{p1} + S_{p2} \cdot c_{p2} + S_{p3} \cdot c_{p3} = v \tag{3.2}$$

or

$$\begin{vmatrix} S_{p1} & S_{p3} & S_{p3} \end{vmatrix} \cdot \begin{vmatrix} c_{p1} \\ c_{p2} \\ c_{p3} \end{vmatrix} = v \tag{3.3}$$

where S_{p1}, S_{p2} and S_{p3} are the particles' system functions, and c_{p1}, c_{p2} and c_{p3} are their individual concentrations. The solution of the inverse problem determines these three vectors based on the system functions and the measurement vector v. Figure 3.2 illustrates the result of a multi-color MPI measurement. The different dynamic responses of the different particles in different solutions result in different PSFs, and consequently calibration matrices with columns linearly independent with relation to other samples, which allows the solution of the extended system of equations.

Multi-color MPI temperature measurement

The System Function and Multi-color MPI methods were also employed for temperature measurements. In [34], the method was used for estimating both concentration

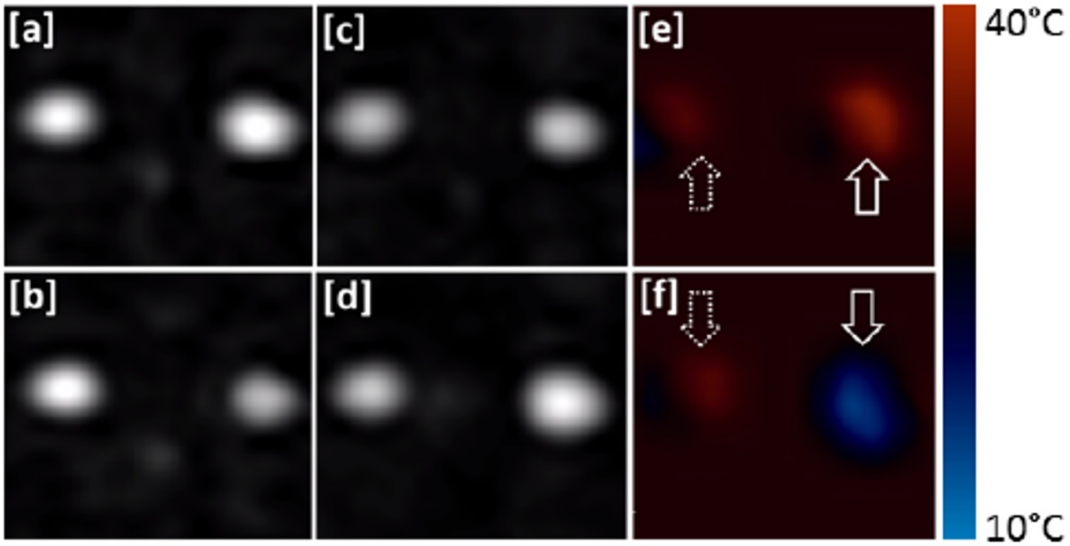

Figure 3.3: Multi-color MPI temperature measurement, where [a] and [b] represent the cold channel measurements, [c] and [d] the warm channel ones, and [e] and [f] are the temperature estimations for two different measurements. Adapted from [34].

and temperature. The procedure was similar to the one explained in 3.1.2, the difference was that instead of using different particles, two system matrices were acquired for the same particle sample at two different temperatures. The temperature then was determined by interpolating the reconstructed concentrations in the two channels, normalized by the total concentration i.e., the sum of both channels. Despite the lack of data for a quantitative analysis, the ratio of signal variation suggests a temperature precision of about 0.5 K. Figure 3.3 shows the results obtained with this method.

The system function method was also employed in [50], where it was demonstrated that a single system function can be used for temperature calculation, by using a sample with a known quasi-uniform MNP distribution, and assuming that inside the sample, any changes in the measured signals could be directly attributed to changes in temperature. The use of a single system matrix minimizes the calibration effort, as well as the computation time required for temperature calculation, in comparison to the multi-color approach. Temperatures then were determined from changes in the reconstructed concentrations, since it was known that the actual concentration was constant. The temporal and spatial temperature distributions were compared with an infrared camera, with a mean absolute deviation of about 1 K.

More recently, it has been shown that the multi-color T-MPI method has serious limitations on determining temperatures between the calibration points [72]. The extended system of equations can be solved to efficiently determine the amount of particles at the calibration temperatures, but cannot resolve it for intermediate ones.

Therefore, the temperature resolution of the method depends on the ability of having several calibration matrices more closely spaced in temperature, which would significantly increase the computational complexity of the reconstruction process. Additionally, this would require MNP samples extremely sensitive to temperature changes, in order to provide distinguishable PSFs and consequently linearly independent system matrices. This investigation is part of this work, and will be described in more details in 5.3.

3.1.3 X-Space

This analytical method was developed in 2010 by Goodwill and Conolly at the University of California Berkeley [14], initially for 1D reconstructions, but soon adapted for 2D and 3D [15]. By taking some simplifications such as linear gradient field, uniform drive fields and coil sensibilities, negligible relaxation time with relation to the scanning frequency, and adopting the superparamagnetic Langevin function as the particle's magnetic response model, the x-space method is formulated assuming that the total magnetization can be modeled by a convolution of the function describing the MNP concentration in space $\rho(x)$ with a point spread function (PSF), given by the first derivative of the Langevin equation $\dot{\mathcal{L}}[kGx]$ with relation to time, as shown in 2.3. The signal $s(t)$ generated by the receive coil with sensitivity $-B_1$ in the 1D approach is described by:

$$s(t) = B_1 \frac{d\Phi}{dt} = B_1 m \rho(x) * \dot{\mathcal{L}}[\xi G x]\big|_{x=x_s(t)} \xi G \dot{x}_s(t) \tag{3.4}$$

where $\xi = \mu_0 m/kT$ and $x_s(t)$ is the position of the FFP in the space. It is possible to notice that, in addition to the magnetic moment magnitude, which depends on the particle's characteristics, the signal intensity is proportional to the selection field's gradient and the FFP scan speed, as the voltage induced in the coils is proportional to the change in the magnetic flux caused by both the fast FFP movement and steep gradient, that magnetize and demagnetize the MNPs along the FFP scanning path. The x-space image is then given by:

$$IMG(x_s(t)) = \frac{s(t)}{B_1 m \xi G \dot{x}_s(t)} = \rho(x) * \dot{\mathcal{L}}[\xi G x]\big|_{x=x_s(t)} \tag{3.5}$$

Similar approaches are adopted for 2D and 3D measurements. It is important to note that the image contains information about the distribution of MNPs in space, but is still blurred by the PSF. The actual MNP concentration map may be obtained by deconvolution of x-space image with the PSF by, for example, Wiener filtering [15].

The main advantage of this method is that it doesn't require any previous cali-

bration to be run, as it analytically computes the magnetization of the system simply by weighting the signal with relation to the FFP speed and then gridding it to fit the data to the FFP position in space, which can be performed in short time. In addition to its simplicity, all the analysis is performed in time domain, which means no Fourier transform is required.

A major drawback of the method is the loss of the fundamental (or excitation, base, or drive) frequency component through filtering, as discussed in 2.4. In the system function method, this issue is naturally addressed by filtering the fundamental frequency during both in the calibration and the measurement, but for the x-space method it was shown that the loss of the base frequency is equivalent to having a dc offset in the MPI image [83, 84]. An algorithm for recovering this offset using prior information has been implemented, with reasonable results. It has also been shown in [43] that adopting a constant bias term in the formulation of a numerical reconstruction also allows recovering the offset caused by filtering.

Another great disadvantage of the X-Space method is that, similarly to other model-based methods, the analytical reconstruction does not account for the measurement system imperfections. The gradient field is assumed as being linear, the drive field is assumed as being uniform over the whole FOV, the receive coil sensitivity is assumed as being uniform as well, and the particle response is assumed as being adiabatic – meaning that the particle's magnetic moment vector aligns immediately with the applied field, without any energy loss or relaxation. Non-adiabatic behavior has also been studied and relaxation has been added to the x-space model, resulting in slightly better results [78, 85].

Even though the system function method is recognized as the more accurate one, is has been demonstrated in [86] that both system function and x-space methods are mathematically equivalent, since both methods are based on the same fundamental equation.

3.1.4 Alternative approaches

The biggest challenge for model-based approaches is accurately accounting for non-ideal behavior of the elements that compose the measurement system: active and passive coils, permanent magnets, base frequency filtering, particle magnetization and its dynamics. Most methods aim to compute the elements of the system matrix, which saves calibration time but potentially add systematic errors to the reconstructed concentration maps, in addition to the ones resulting from the algebraic reconstruction methods. Still, they do not address one of the biggest issues involving the system function method, which is the size of the calibration matrix.

The formulation of a forward MPI operator that allows direct deconvolution

without the use of the system matrix is highly desirable. A 1D convolution kernel derived from the conventional modeling, which could be extended to 3D, is shown in [75], while the formulation for a spatial model in terms of a MPI *core operator* and the corresponding reconstruction methods applicable to any arbitrary FFP trajectory are shown in [76]. This last approach was further improved in [43] by accounting for non-uniform receive coil sensitivity in the analytical model, as well as non-adiabatic MNP magnetization response through the inclusion of a relaxation kernel, and the recovery of the signal lost due to base frequency filtering.

Additional methods include AI approaches that makes use of neural networks. In [87] the authors propose a model-based image reconstruction method with additional learning during execution. The use of deep neural networks combined over publicly available calibration and measurement data (see [53]) is studied in [88]. Recent approaches also include the use of Bessel functions [89] and splines [90].

3.2 Inverse problem formulations: Tikhonov regularization, L1 norm and Total Least Squares

As discussed in section 3.1.1, since the measurement contains noise, regularization is required in order to obtain a more stable solution for the system function inverse problem. Equation 3.1 is then rewritten as:

$$S \cdot c = \tilde{v} \tag{3.6}$$

where $\tilde{v} = v + \eta$, for an additive noise η. The presence of noise requires the use of regularization techniques, such as Tikhonov regularization. In this case, the extended system of equations can be given by

$$\begin{vmatrix} S & \lambda I \end{vmatrix} \cdot \begin{vmatrix} c \\ r \end{vmatrix} = \tilde{v} \tag{3.7}$$

when using Craig's method, where I is the identity matrix and r is an auxiliary residual vector, or can also be given by

$$\begin{vmatrix} S \\ \lambda I \end{vmatrix} \cdot c = \begin{vmatrix} \tilde{v} \\ 0 \end{vmatrix} \tag{3.8}$$

in a traditional regularized least squares assembly. The least squares solution is then given by minimizing the functional:

$$\underset{c \geq 0}{\arg\min} \ \|S \cdot c - \tilde{v}\|_2^2 + \lambda \|c\|_2^2 \tag{3.9}$$

where λ is a regularization parameter, which is employed to balance noise and definition — if it is too low the solution will be noisy, whereas if it is too high the reconstructed image will be blurred, resulting in definition loss. It has been shown in [91] that both extended system of equations 3.7 and 3.8 are equivalent. Even though the regularization parameter is most often tuned manually, and checked by visual inspection of the reconstructed image, it has been shown that it can be automatically determined using the L-curve method [6].

The above solution includes a regularization term to penalize solutions with high Euclidean norm (or L2 norm), an indicative that the solution might be fitted to noise. An alternative regularization approach, based on an absolute norm (or L1 norm) penalty term is presented in [68], where a non-negative fused lasso model is employed:

$$\underset{c\geq 0}{\arg\min} \ \|S \cdot c - \tilde{v}\|_2^2 + \lambda \|c\|_1 + \beta \mathrm{TV}(c) \tag{3.10}$$

where $\mathrm{TV}(c)$ is a total variation term introduced to preserve the edges of the reconstructed volumes, resulting in images with reduced noise and sharper edges. The scaling $\beta > 0$ term is employed to control the weight of $\mathrm{TV}(c)$ in a similar way λ is employed to weight $\|c\|_1$.

Another L1 norm approach was proposed in [92], in which the authors instead propose replacing the least-squares term by a L1 norm term:

$$\underset{c\geq 0}{\arg\min} \ \|S \cdot c - \tilde{v}\|_{1,\epsilon} + \lambda \|c\|_1^2 \tag{3.11}$$

where ϵ is a very small smoothing term employed to keep the above equation differentiable. According to the authors, the L1 fitting is said to be statistically more robust to outliers than the L2 fitting.

The above regularization methods assume that only the measurement vector contains noise. However, in real-life applications the system matrix also contains noise – especially the calibration-based one. In order to account for potential model errors and noise in the system matrix, the Total Least Squares (TLS) method was proposed in [93]. In this approach, a deviation matrix δS is included in eq. 3.6:

$$(S + \delta S) \cdot c = \tilde{v} \tag{3.12}$$

The reconstruction requires minimizing the below Tikhonov-like functional for both δS and c:

$$\underset{c,\delta S}{\arg\min} \ \|(S + \delta S) \cdot c - \tilde{v}\|_2^2 + \|\delta S\|_F^2 + \lambda \|c\|_1 \tag{3.13}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The functional is iteratively minimized by

an alternating algorithm which computes, at each iteration, the new estimates for c and for δS.

3.3 Reconstruction algorithms

Over the relatively short lifetime of MPI since its inception, a number of algorithms has been investigated to provide fast and accurate image reconstructions. Among them, the most popular ones are the singular value decomposition (SVD), the Kaczmarz algorithm, and the conjugate gradient normal residual (CGNR) method. All these algorithms aim to find the solution of the Tikhonov regularization functional in 3.9, and they are discussed in more details in the following sections.

3.3.1 Kaczmarz algorithm

The main goal when choosing a reconstruction algorithm is to provide (if possible) real-time reconstructions, so speed and efficiency are of great importance. In this sense, the simplicity and fast convergence of the Kaczmarz algorithm granted it the position of most widely adopted MPI reconstruction algorithm. It is also known in the literature as Algebraic Reconstruction Technique – ART (even though ART can be considered a particular case of the Kaczmarz method for some authors), and it consists, for each row of the system matrix S, on projecting the current solution onto it, finding then the correction to derive an updated solution. One sweep over all the system matrix's rows is considered one Kaczmarz iteration, and the process is repeated until convergence is reached [94]. Applied to the regularized system of equations 3.7, the updated estimate $c^{(i+1)}$ is given by:

$$c^{(i+1)} = c^{(i)} + \frac{\tilde{v}_i - S_i \cdot c^{(i)} - \lambda \cdot r_i}{||S_i||^2 + \lambda^2} S_i^T \tag{3.14}$$

where S_i is a row of the system matrix, v_i is the corresponding element of the measured signal vector, and r_i is the corresponding element of an auxiliary vector. It is also possible to add constraints at the end of each update, such as non-negativity, in order to improve the accuracy of the reconstruction. A typical implementation of the Kaczmarz algorithm is shown in Algorithm 1. For nearly orthogonal rows, the solution converges very quickly, but it can be extremely slow for nearly parallel rows (which fortunately is not the case for MPI), with complexity $O(N^2)$, where N is the number of columns of the calibration matrix. One solution could be randomly choosing the rows to be projected to compute the new estimate, so sweep sequences would be different at each iteration, which has been proposed in [95]. Examples of the use of this algorithm can be found in [5, 10, 82].

Algorithm 1 Regularized Kaczmarz algorithm

$c^{(0)} \leftarrow 0$ ▷ Initialize solution vector
$r \leftarrow 0$ ▷ Initialize residual vector
for $i = 0 \rightarrow$M **do**
    $\beta_i \leftarrow \left(s_i - S_i \cdot c^{(i)} - \lambda \cdot r_i\right) / \left(||S_i||^2 + \lambda^2\right)$
    $c^{(i+1)} \leftarrow c^{(i)} + \beta_i \cdot S_i^T$ ▷ Update solution
    if $c^{(i+1)} < 0$ **then**
        $c^{(i+1)} \leftarrow 0$ ▷ Positivity constraint (optional)
    end if
    $r_i \leftarrow r_i + \beta_i \cdot \lambda$ ▷ Update residual vector
end for

As variants of the Kaczmarz algorithm, there are the Cimmino algorithm (or *simultaneous iterative reconstruction technique* - SIRT) and the block-ART or block-Cimmino algorithm [70, 94, 96]. The Cimmino algorithm averages the corrections obtained by the projections of all the rows of S before updating the solution, instead of updating after each projection, while the block-Cimmino runs the Cimmino algorithm over blocks that contains subsets of rows of S.

3.3.2 Conjugate Gradient Normal Residual

The Conjugate Gradient (CG) was developed in 1952 at the National Bureau of Standards (NBS) as an iterative solver applied to finding the solution of positive definite systems of equations [97], which is not the case of the regularized least squares problem defined in equation 3.8. In this situation, the CG method is applied to the normal equation:

$$S^T S c = S^T \tilde{v} \tag{3.15}$$

in the so-called conjugate gradient normal residual (CGNR) method. Also known as conjugate gradient least squares (CGLS) or conjugate gradient normal equation (CGNE), the algorithm works by iteratively finding a sequence of vectors $p^{(i)}$ that are mutually conjugate with relation to S, used as the search directions [94, 98]. The new estimate $c^{(i+1)}$ for the regularized least squares problem 3.8 is given by:

$$c^{(i+1)} = c^{(i)} + \alpha \cdot p^{(i)}, \tag{3.16}$$

where:

$$\alpha = \frac{||\tilde{S}^T \cdot r^{(i)}||^2}{||\tilde{S} \cdot p^{(i)}||^2} \tag{3.17}$$

$$p^{(i)} = \tilde{S}^T \cdot r^{(i)} + \frac{||\tilde{S}^T \cdot r^{(i)}||^2}{||\tilde{S}^T \cdot r^{(i-1)}||^2} p^{(i-1)} \tag{3.18}$$

$$r^{(i)} = s - \tilde{S} \cdot c^{(i)} \tag{3.19}$$

and $\tilde{S}$ is the augmented system matrix accounting for regularization:

$$\tilde{S} = \begin{vmatrix} S \\ \lambda I \end{vmatrix} \tag{3.20}$$

The CGNR method has complexity $O(N^2)$, where N is the number of columns of the calibration matrix, and is detailed in Algorithm 2. One of the biggest advantages of the method is the fact that the explicit calculation of $S^T S$ is unnecessary, saving memory space and computation time, as well as avoiding additional round-off errors. In addition, for optimum performance, the residuals do not need to be recalculated at every iteration, being updated together with the estimated solution (see lines 11 and 12 of Algorithm 2), which saves one matrix vector multiplication per iteration. Ideally, the method would converge in n iterations, where n is the largest dimension of S, but in real applications round-off errors slow it down. One disadvantage of the method is the fact that it relies on the conjugacy between search directions for proper convergence, which prevents the implementation of additional constraints, such as positivity. Convergence can be improved by the adoption of matrix preconditioning and/or the use of a weighting matrix [98, 99].

Algorithm 2 Regularized CGNR

1: $c^{(0)} \leftarrow 0$ ▷ Initialize solution vector
2: $r^{(0)} \leftarrow s - S \cdot c^{(0)}$ ▷ Initialize residual vector
3: $r_\lambda^{(0)} \leftarrow -\lambda \cdot c^{(0)}$ ▷ Initialize λ-residual vector
4: $z^{(0)} \leftarrow S^T \cdot r^{(0)} + \lambda \cdot r_\lambda^{(0)}$
5: $p^{(0)} \leftarrow z^{(0)}$ ▷ Define first search direction
6: **for** $i = 0 \rightarrow N_{iter}$ **do**
7: $w^{(i)} \leftarrow S \cdot p^{(i)}$
8: $w_\lambda^{(i)} \leftarrow \lambda \cdot p^{(i)}$
9: $\alpha^{(i)} \leftarrow ||z^{(i)}||^2 / (||w^{(i)}||^2 + ||w_\lambda^{(i)}||^2)$
10: $c^{(i+1)} \leftarrow c^{(i)} + \alpha^{(i)} \cdot p^{(i)}$
11: $r^{(i+1)} \leftarrow r^{(i)} - \alpha^{(i)} \cdot w^{(i)}$
12: $r_\lambda^{(i+1)} \leftarrow r_\lambda^{(i)} - \alpha^{(i)} \cdot w_\lambda^{(i)}$
13: $z^{(i+1)} \leftarrow S^T \cdot r^{(i+1)} + \lambda \cdot r_\lambda^{(i+1)}$
14: $\beta^{(i)} \leftarrow ||z^{(i+1)}||^2 / ||z^{(i)}||^2$
15: $p^{(i+1)} \leftarrow z^{(i+1)} + \beta^{(i)} \cdot p^{(i)}$
16: **end for**

3.3.3 Steepest Descent

The steepest descent method is an iterative algorithm widely employed in deep-learning applications [100]. It is a first-order method that uses the inverse of the residuals of the function (or the gradient) as search direction. Similarly to the approach employed with the conjugate gradient method, the steepest descent method is applied in this work to the normal equation (3.15), and when applied to minimize the regularized least squares problem (3.8), the next iteration is given by:

$$c^{(i+1)} = c^{(i)} + \alpha \cdot z^{(i)}, \tag{3.21}$$

where

$$\alpha = \frac{||z^{(i)}||^2}{||\tilde{S} \cdot z^{(i)}||^2}, \tag{3.22}$$

and

$$z^{(i)} = \tilde{S}^T \cdot (s - \tilde{S} \cdot c^{(i)}) \tag{3.23}$$

The regularized steepest descent method is shown in algorithm 3. For optimal performance, the residuals are updated together with the solution estimate, without the need of the calculation of new residuals, which saves computation time, as also implemented in the conjugate gradient method. If one wishes to enforce a constraint, such as non-negativity, the new residuals must be computed, adding a matrix-vector multiplication to the algorithm and slowing down its execution. The complexity of the steepest descent method is also $O(N^2)$.

Algorithm 3 Regularized Steepest descent

$c^{(0)} \leftarrow 0$ ▷ Initialize solution vector
$r^{(0)} \leftarrow s - S \cdot c^{(0)}$ ▷ Initialize residual vector
$r_\lambda^{(0)} \leftarrow -\lambda \cdot c^{(0)}$
for $i = 0 \rightarrow$N **do**
    $z^{(i)} \leftarrow S^T \cdot r^{(i)} + \lambda \cdot r_\lambda^{(i)}$ ▷ Update search direction
    $w^{(i)} \leftarrow S \cdot z^{(i)}$
    $w_\lambda^{(i)} \leftarrow \lambda \cdot z^{(i)}$
    $\alpha^{(i)} \leftarrow ||z^{(i)}||^2 / (||w^{(i)}||^2 + ||w_\lambda^{(i)}||^2)$
    $c^{(i+1)} \leftarrow c^{(i)} + \alpha^{(i)} \cdot z^{(i)}$
    $r^{(i+1)} \leftarrow r^{(i)} - \alpha^{(i)} \cdot w^{(i)}$
    $r_\lambda^{(i+1)} \leftarrow r_\lambda^{(i)} - \alpha \cdot w_\lambda^{(i)}$
end for

3.3.4 Singular Value Decomposition (SVD)

The SVD was one of the first methods proposed in the literature for MPI image reconstructions [6]. It is a direct solver, rather than an iterative method like Kaczmarz and CGNR, which works by decomposing the $M \times N$ system matrix into three matrices, U, Σ and V, where U $(M \times M)$ and V $(N \times N)$ are unitary orthogonal matrices, and Σ $(M \times N)$ is a rectangular diagonal matrix containing the singular values of S, or σ_i. These three matrices can be used to calculate the pseudoinverse (or the *Moore-Penrose* inverse) of S for the solution of systems of equations. The SVD would provide the exact solution for consistent system of equations, but it is extremely sensitive to noise in the input data. In this case, regularization can be applied, and then the approximate solution is given by:

$$c_\lambda = \sum_{i=0}^{M} \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \frac{U_i^T \cdot s}{\sigma_i} V_i \tag{3.24}$$

where $U_{.,i}$ is the ith column of U and $V_{.,i}$ is the ith column of V. The first term inside the sum is the filter factors f_i:

$$f_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \tag{3.25}$$

For $\sigma_i \gg \lambda$, $f_i \approx 1$, and for $\sigma_i \ll \lambda$, $f_i \approx 0$. In practice, the regularization parameter λ acts like a low-pass filter, allowing only large singular values at lower frequencies, in an approach similar to the truncaded SVD (TSVD) [94]. A simple implementation of the regularized SVD reconstruction is shown in algorithm 4.

The complexity of the SVD is $O(N^3)$, but fortunately the decomposition has to be performed only once for a single system function. Once it is done, the solution is calculated with complexity $O(N^2)$, for a given λ. Examples of the use of SVD can be found in [3, 6, 62].

Algorithm 4 Regularized SVD reconstruction

1: `Input matrices:` $U_{m\times m}$, $V_{n\times n}$, $\Sigma_{m\times n}$ ▷ SVD decomposition
2: $c^{(0)} \leftarrow 0$ ▷ Initialize solution vector
3: Begin reconstruction:
4: $p = U^T \cdot s$
5: **for** $i = 0 \rightarrow$M **do**
6: $\quad f_i = \sigma_i/(\sigma_i^2 + \lambda^2)$
7: $\quad p_i = p_i \cdot f_i$
8: **end for**
9: $c = V \cdot p$ ▷ Compute solution vector

Chapter 4

GPU parallel programming with CUDA

Most of the technological advances seen in the last decades are due mainly to the increase of processing capacity of computers. Since the early 80's personal computers, this improvement is a result of the increase of the processor operating frequencies, or *clocks*. Since then, the clock rates increased from a few MHz to about 4 GHz in the early 2000's, an acceleration of the order of thousands in processing speed. Until then, the increase in CPU processing power depended on two main factors: reducing the size of the transistors that compose the CPUs (and therefore increasing the number of transistors per chip) and increasing the clock speeds. Historically, the number of transistors per chip has doubled since the 1960's, in accordance with Moore's law [101, 102], but the same cannot be said about the CPU clocks. The acceleration of the CPU frequencies drastically declined in the early 2000's, as the CPU transistor sizes and overheating became major constraints on obtaining higher clocks [103], forcing the CPU manufacturers to explore other alternatives. As a consequence, the first dual- and multi-core architectures were released, fostering the development of applications using parallel approaches – something that until then was mostly restricted to large applications running in supercomputers, or to graphical processing executed in GPUs.

GPUs had a significant role in the development of parallel computing. In the early 1990's, the demand for applications with improved graphical performance increased significantly, mainly driven by the development of 3D games, which required rendering scenarios at high frame rates and demanded massive arithmetic operations to determine the color of each pixel on the screen. At that time, the employed parallelism was based on graphical *pipelines*, which could be configured but not programmed [104]. In 2001, in response to an increasing demand from developers, NVIDIA released the first GPU with programmable pipeline, and shortly after the developers realized that it could be used not only over graphical data, but also over

any kind of data that could benefit from the GPU parallel architecture. However, exploring the potential of the GPUs was not straightforward; the developers needed to be familiar with graphics libraries such as *OpenGL* or *DirectX* that allowed interacting with the GPUs, as well as *shading* languages like *GLSL* or *Shader*. These limitations were an obstacle to the acceptance of the GPUs as general parallel computing tools.

Finally, in 2006, NVIDIA releases the first GPU developed with the CUDA (*Compute Unified Device Architecture*) architecture, followed in 2007 by a programming language that extended the traditional C language (the *CUDA C* language) by defining a set of keywords that allowed accessing the GPU resources, a specific compiler for this extended language, and a driver to explore the GPU features. The CUDA architecture was developed to be applied to either graphics and general applications, and revolutionized the field of high-performance computing (HPC). After the release of CUDA, it was no longer necessary to be familiar with graphics libraries or shading languages to use GPUs for general purpose computing, which boosted the popularity of the architecture. These graphic cards, and the ones that followed this new paradigm became known as *General-Purpose Graphical Processing Units* (GPGPUs). Currently, NVIDIA's GPUs are employed in a wide range of applications, including data analysis and visualization, signal processing, database management and machine learning. Among the 500 fastest supercomputers in the world, 165 of them employ CUDA-enabled GPUs as accelerators, six of them among the top ten systems, according to the November/2023 release of the TOP500 list [105]. Currently, several other programming languages offer support for exploring the CUDA features, such as C++, Python, Fortran, Julia and Matlab.

4.1 The CUDA architecture

CUDA is based on the SIMT (Single Instruction, Multiple Thread) architecture, which is similar to the SIMD (Single Instruction, Multiple Data) one, and was designed to execute the same set of instructions by thousands of threads concurrently. While a regular CPU is highly specialized on data processing and execution control, a GPU is highly specialized on parallel computations [106]. Figure 4.1 illustrates how the CPU dedicates more transistors to execution control than the GPU. The computations over the pixels of an image are a good example of a highly parallel application suitable to be executed in a GPU.

The architecture is based on the concept of *Streaming Multiprocessors* (SMs), specialized processors composed of multiple cores. Each SM manages the creation and executions of *warps*, groups of 32 threads that run in parallel. The threads within a warp execute the same program simultaneously, but each thread has its own

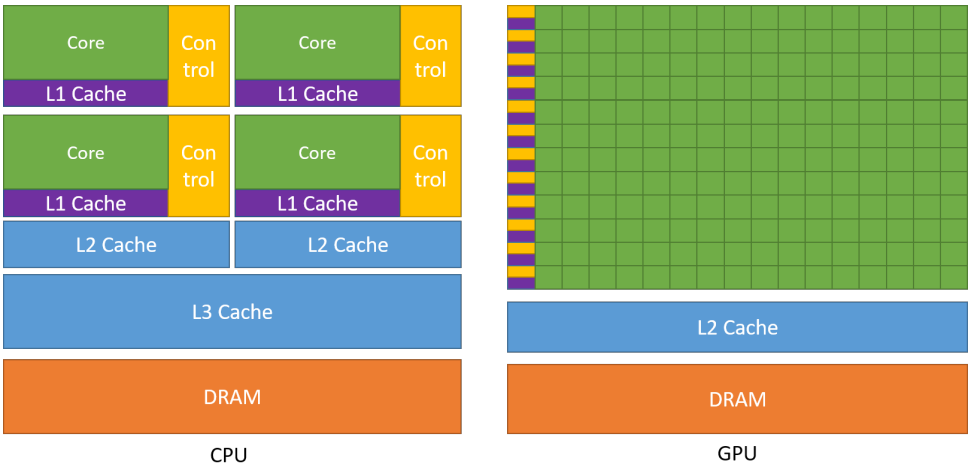

Figure 4.1: Comparison between the CPU (left) and GPU (right) architectures. Source: NVIDIA (2024) [106].

program counter and state registers, so they can run independently, even though the maximum performance is obtained when all the threads execute exactly the same instructions, so conditional branching should be avoided, if possible. The threads are sent to the SM for execution in *blocks*; the SM organize the threads within the block into warps and execute each one of them according to the available number of cores. The blocks can also be organized into *clusters*, which in turn are organized as *grids*. This organization ensures the scalability of the architecture, since GPUs with different number of SMs and cores can execute the same program, and also ensures an efficient memory usage by implementing different memory levels and access rules for each group.

A CUDA program is called a *kernel*, which is called during the execution of the main program to run a number of threads in parallel. The execution of a CUDA kernel is asynchronous, and the main program is not interrupted by the kernel call unless synchronization is explicitly requested. This heterogeneous programming model is advantageous because even though many tasks can be accelerated by the GPU, others are more efficiently executed in the CPU, so the workload can be distributed accordingly. Additionally, the asynchronous kernel execution can be combined with asynchronous memory transfers so different kernels can be executed simultaneously. The main CUDA program may contain one or more kernels that are invoked during its execution, which can be assigned to different *streams* and then also be executed concurrently.

4.2 Thread and memory hierarchy

The smallest instance of a CUDA program is a thread, which is executed in a CUDA core. Each core contains its own on-chip registers with the lowest access times, typically of the order of one clock cycle. Each thread has its own identification register, which can be accessed through the variable `threadIdx`. This identification is unique within the thread block, and the threads can be organized in one, two or three-dimensional blocks for programming convenience. In this case, the thread IDs are determined by the indexes `threadIdx.x`, `threadIdx.y`, `threadIdx.z` and the block dimensions, which are user-defined and can be accessed within the kernel through the `blockDim` variable. When the threads are organized as an one-dimensional array, the ID is simply given by `threadIdx.x`.

The threads are organized into blocks, which can also be organized as one, two or three-dimensional grids, and analogously, the indexes of the blocks can be accessed through the variables `blockIdx.x`, `blockIdx.y` and `blockIdx.z`. Therefore, a unique thread in a kernel call can be identified as a combination of the thread indices, block indexes and the block dimensions. For example, when the thread and blocks as organized as an one-dimensional array, the unique thread identification can be determined by `blockIdx.x * blockDim.x + threadId.x`. The threads within a block can all access an on-chip, high bandwidth and low latency shared memory, similar to a L1 cache memory. The shared memory is divided into equally-sized modules called *banks* that can be accessed simultaneously. The threads within a block are always executed in a single SM, so they can cooperate within each other through the shared memory and in-block synchronization.

The blocks can also be organized into *block clusters*. Analogous to the blocks and threads, the clusters can also be organized in one, two or three dimensions, and the blocks within a cluster are always executed in a single GPU processing cluster, or GPC. The thread blocks within the cluster have access to the distributed shared memory, which is the set containing the shared memory of all thread blocks that compose the cluster. Together with a built-in cluster synchronization capability, the threads within a cluster can also cooperate on a higher level.

All threads have access to the GPU global memory, which has lower bandwidth in comparison to the shared memory. The access to the global memory is optimized through a L2 cache, provided the locality of the data accesses is respected. Each thread has access to its own local memory, that resides in the device memory and have the same access times as the global memory. Finally, two additional read-only memory spaces are available, the constant and texture memory spaces. The constant and texture memories are cached in independent constant caches, for improved data throughput, being the texture cache optimized for 2D spatial locality.

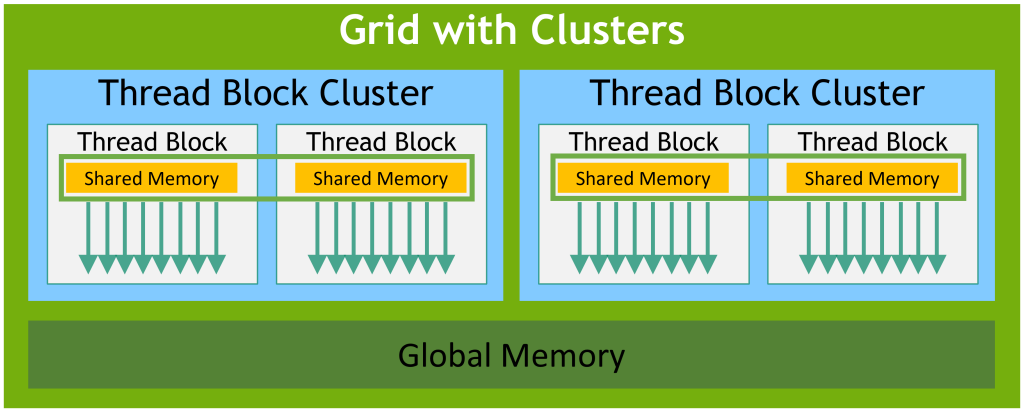

Figure 4.2: Threads and memory hierarchy in the CUDA architecture. The arrows in the image represent the threads. Adapted from NVIDIA (2024) [106].

Figure 4.2 illustrates the thread and memory hierarchy. The lifetime of the data in the registers, shared memory and distributed shared memory are the same of their respective threads, blocks and clusters. The global, constant and texture memory data are maintained between kernel launches, persisting during the whole application. The GPUs can also access and manipulate data allocated in the host (the CPU) main memory. This operation, however, is very expensive, with very high latency, and should be avoided if possible.

4.3 Best practices for programming with CUDA

CUDA is a powerful parallel architecture which can deliver, if properly used, orders of magnitude in execution speedup. Even though many applications can benefit from parallel implementations, it is important to keep in mind that some applications are not suitable to be executed in parallel, and will exhibit poorer performance (an example of such application is shown in 5.2.2). Often a sequential algorithm will have to be completely rewritten to be executed in parallel, and this can be done in several different ways. In general, algorithms containing mostly `for` loops are good candidates for being executed in the GPU, while others containing many `if` statements usually are more suitable for being executed in the CPU.

CUDA-enabled GPUs are composed of hundreds to thousands of cores. Therefore, optimal parallel performance is obtained when the GPU utilization is maximized. This means exposing the application to the parallelism of the system in such a way that all the components are kept busy most of the time. The granularity of the algorithm must be such that all the GPU cores are employed for the computation, so the more the problem can be broken into smaller and (possibly) independent steps the better is the performance gain.

Having the device idle should be avoided at all costs. Even though all the threads

within the SM are running exactly the same code, the threads can diverge if the code has conditional statements. In the case of conditional branching, the branches will be executed in sequence, so while a group of threads runs through one branch, the remaining threads will remain idle, which represents a significant impact on the application throughput. Therefore, conditional statements should be avoided as much as possible. Often conditional branching can be successfully replaced by logical operations.

Cooperation between threads often requires access to shared data and synchronization. A race condition is created when two or more threads try to access the same memory location, and memory fence functions help ordering these accesses. However this causes threads to remain idle while waiting for access to the resources. Similarly, synchronization points force threads within a block to remain idle while other threads don't reach the same point. Memory fences and synchronization points should be used only if strictly necessary, and should be avoided as much as possible. When the threads within a warp are idle, the SM warp scheduler replaces the warp by another one that is ready to be executed, so having enough warps to keep the SM busy while some are idle is a good strategy for hiding latency times.

Data transfers between CPU and GPU are a major bottleneck of most CUDA applications, and should be minimized for optimum performance. Sometimes it is worthwhile running segments of code that cannot be completely parallelized in the GPU rather than transferring the data to the CPU for a more efficient computation, since the cost of the transfer is significantly higher. Because the overhead associated to memory transfer functions calls can add up, batching many small data transfers into larger transfers is always preferred. On systems with a front-side-bus (FSB), it is sometimes advantageous to allocate space in the CPU memory as *pinned* or *page-locked host memory*, which prevents the host operating system from changing the position of the memory block and increases data transfer speeds. Additionally, when using mapped page-locked memory on the host, there is no need to allocate memory on the device and copy the data to it, the data transfers are implicitly performed when a kernel accesses the mapped memory. This is particularly advantageous in systems where the the CPU and GPU memory is shared, such as in some laptop models.

When threads access data from the global memory, the data locality must be considered to leverage from caching. Accesses on which sequential threads read data from contiguous memory addresses are said to be *coalesced*, and maximize the data instruction throughput by masking the high wait times for data from the global memory. Shared memory and registers should be prioritized as much as possible, since the access to them is significantly faster. On the other hand, using the shared memory adds more complexity to the algorithm, and the amount of available shared

memory is a limiting factor on defining the number of threads within a block. The developer should balance the accesses to each memory in order to maximize the overall performance of the system.

4.4 CUDA specialized libraries

In addition to the core CUDA library, NVIDIA developed a number of specialized libraries that efficiently explore the GPU architecture, and support the development of high-performance applications. This work employs two of these libraries to accelerate MPI image reconstruction algorithms, the *cuBLAS* and *cuSOLVER*. The cuBLAS library provides a GPU-accelerated implementation of basic linear algebra subroutines (BLAS) like matrix-matrix multiplications, dot products and norms, among several other useful functions [107]. The cuSOLVER library provides a collection of functions implementing linear algebra solvers using SVD, Cholesky, LU, and QR decompositions in parallel in the GPU [108]. Since both cuBLAS and cuSOLVER – and also the CUDA core – are proprietary libraries, their implementation is completely transparent to the user, and there is no control over the workload distribution within the GPU.

Chapter 5

Parallel MPI reconstruction strategies using CUDA

Ever since MPI was developed, image reconstruction was always a major focus of research. As a potential biomedical imaging method, the main goal was to provide the most accurate image reconstructions as fast as possible. Regardless of the chosen modeling method (system matrix or x-space), the complexity and the size of the optimization problem that involves obtaining high accuracy MPI images requires the adoption of signal processing and/or numerical techniques. Examples of such techniques include the reduction of the data set by analog filtering or row selection [63], data compression [60] and low-rank approximations [109]. However, until this work was developed and the first results were published in [70], no high-performance computing approach had been proposed in the literature for dealing with such large and complex problem. This work, therefore, is the first one to deal with the MPI image reconstruction by means of parallel algorithms running in GPUs.

The programs developed for this work were written using standard C/C++ language and the CUDA framework [106], for optimum performance in both serial and parallel implementations, allowing then a fair and reliable evaluation of speedups. For the development of the parallel algorithms, the *cuBLAS* library [107] was employed for most matrix and vector operations, while the *cuSOLVER* library [108] was used for the SVD factorization. A number of customized *kernels* were developed for specific applications, for which neither specialized libraries were suitable.

5.1 Preliminary MPI simulations

This work was carried out as part of the efforts of NIST's *Thermal MagIC* project, to develop the instrumentation and techniques for remotely measuring and controlling temperature using MNPs and MPI techniques. By the time the project started,

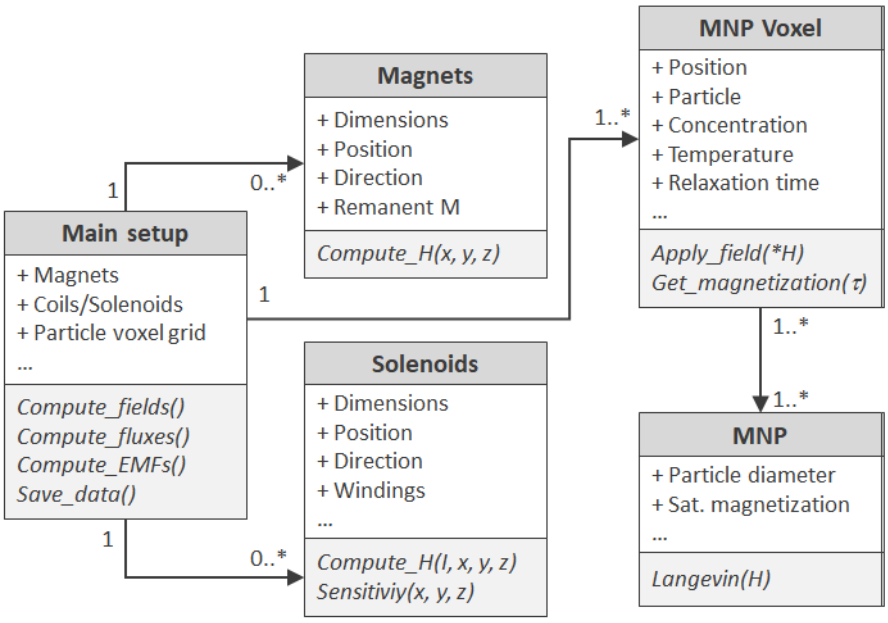

Figure 5.1: Simplified class diagram of the MPI simulation software.

no instrumentation was available for generating data, and the only data available was the *OpenMPIDataset*, generated and maintained by the Technical University of Hamburg [53, 110]. The data was generated to be used employing the system function method only – the calibration and measurement signals were acquired in frequency domain using a preclinical MPI scanner that measured the samples with a FFP following 2D and 3D Lissajous trajectories. At that time, the instrumentation design and the measurement strategies (FFP or FFL, system function or x-space, time domain or frequency domain) were not yet defined. Therefore, in order to allow the development and testing of different designs and strategies, and to support the development of image reconstruction strategies while the instrumentation was not ready, a simulation framework was developed to provide realistic MPI data [42].

The framework is able to simulate a whole MPI scanner by modeling the individual components that compose the scanner. An object-oriented programming approach was adopted, in which the individual components – permanent magnets, electromagnets, receive coils and particles – were modelled by specific classes. Figure 5.1 shows a simplified class diagram of the simulation software, illustrating the classes and how the communicate. Cylindrical permanent magnets and solenoids employed for generating the FFP and drive fields were modelled in terms of a generalized elliptic integral, as described in [111]. The fields generated by rectangular magnets used to generate FFLs were modelled using an approach adapted from [112]. Figure 5.2 illustrates the field generated by a pair of cylindrical magnets creating a a FFP, and the validation of the simulation against *COMSOL Multiphysics*, a commercial finite element analysis (FEA) software [113]. The same numerical model

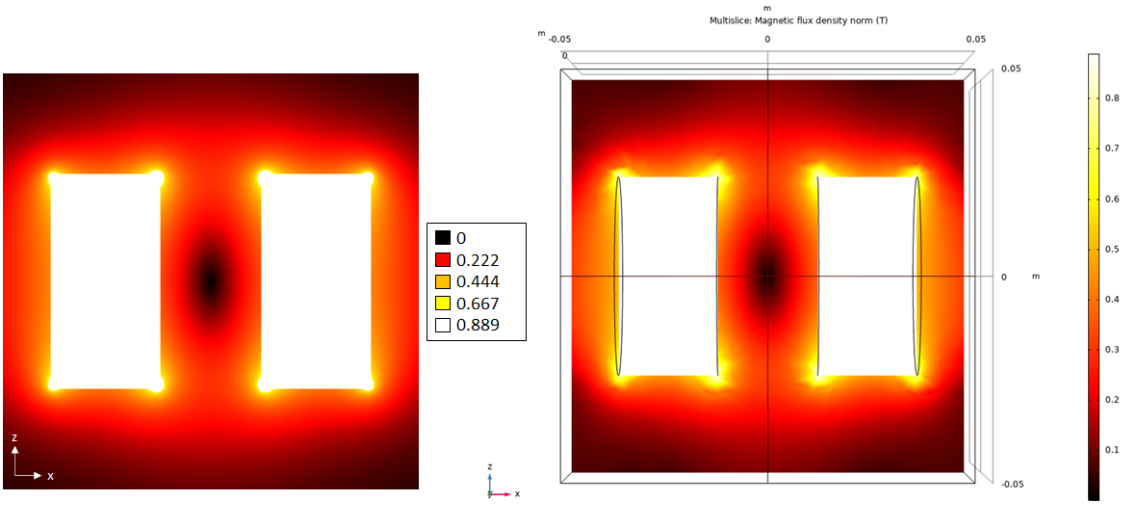

Figure 5.2: Cross section of a simulated FFP generated by a pair of cylindrical permanent magnets (left) and the same simulation executed with COMSOL (right). Figure adapted from [42].

employed for computing the fields generated by solenoids was used to compute the induced signals measured by the receive coils, following the principle of reciprocity [2]. Consequently, the space-dependent receive coil sensitivity $R(x)$ can be defined as the field the coil would produce at the position x when driven by an unitary current, and then be employed for either computing the field generated by a current I or the voltage induced by a time-varying magnetization $M(t)$ at the position x, in accordance to equation 2.6.

The particles were modelled to behave according to the superparamagnetic Langevin equation. Voxel objects hold the information related to the particles within them, their concentration, temperature and relaxation time, as well as the voxel positions relative to the center of the sample. Discrete arrays of voxel objects compose the grid that represents the simulated 1D, 2D or 3D MNP sample to be scanned. This structure allows each voxel to respond individually to an applied magnetic field, providing enough flexibility to simulate samples with different particles, particles at different concentrations, and also at different temperatures within the grid. Particle relaxation is implemented within each voxel by employing a magnetization array, which stores a time series containing the adiabatic magnetization values (i.e., without relaxation) given by the Langevin model for ideal nanoparticles. At every simulation time step, an external magnetic field – the sum of the fields generated by the permanent magnets and drive coils – is applied to the voxel, and the adiabatic magnetization resulting from this field is calculated and stored in the array. The non-adiabatic (or relaxed) magnetization is given by the convolution of a relaxation kernel over the adiabatic magnetization time series:

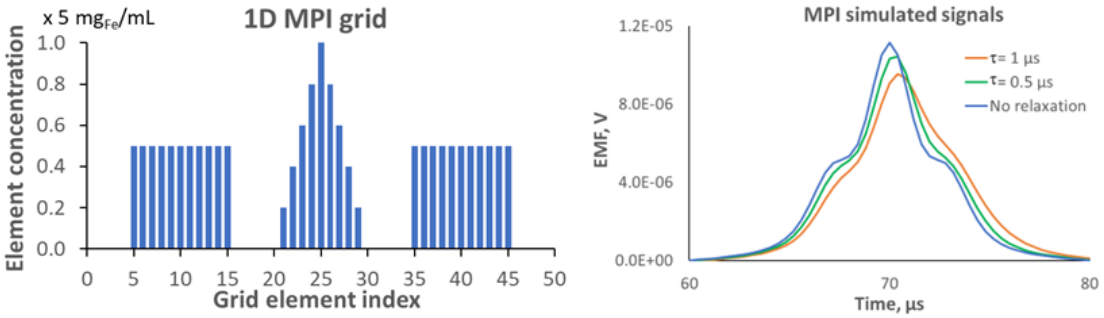

Figure 5.3: 1D MPI simulation. The simulated grid is shown in the left, is 10 mm long, and is composed of 51 voxels containing 20 nm magnetite particles at different concentrations. The plot at the right show the simulated signals for different relaxation times. Figure adapted from [42].

$$M(t) = M_{adiab}(t) * r(t) \tag{5.1}$$

where

$$r(t) = e^{-\frac{t}{\tau_{eff}}} \tag{5.2}$$

is the *relaxation kernel*. The size of the array depends on the sampling frequency, and is expected to store up to seven times the particle's relaxation time, which is equivalent to 99.9% of the exponential magnetization decay, so the size of the array is $\texttt{int}(7 \cdot \tau_{eff} \cdot f_S) + 1$. An auxiliary array stores the values of the relaxation kernel normalized by the sum of elements (so no magnitude is added or subtracted), and the computation of the non-adiabatic magnetization is then straightforward. Figure 5.3 shows the 1D simulated signals for different particle relaxation times. The receive coils compute the induced electromotive forces (emfs) by detecting the changes in the magnetic flux density resulting from the combination of the magnetization values returned by the voxels. In setups with more than one receive coil, such as 2D and 3D scanners that feature coils oriented in orthogonal directions, this process is repeated for each coil, since the detected magnetic fluxes will be different. The execution flow can be summarized as:

- For each time step t:
    1. For each voxel:
        - Compute field vectors over voxel
        - Compute magnetization vector
        - Compute contribution to the flux across the receive coil(s)
    2. Compute total magnetic flux for each coil
    3. Compute emf(s) as the time derivative of the magnetic flux

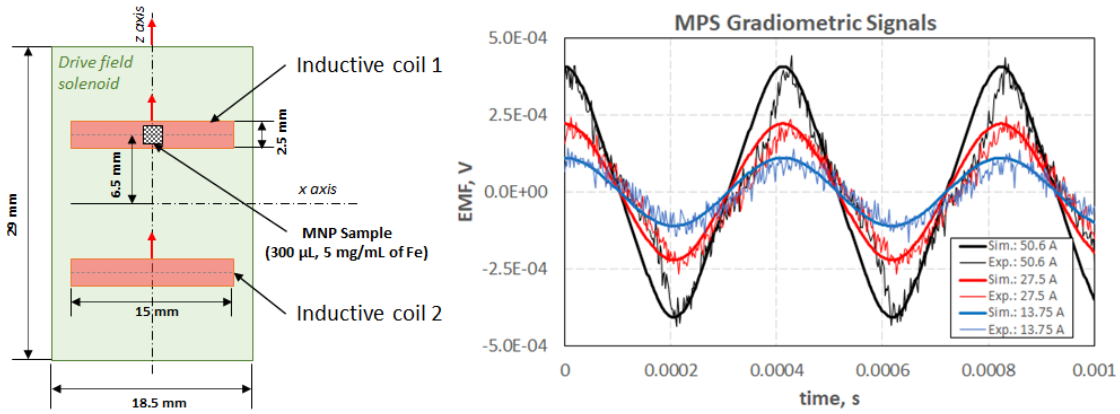

Figure 5.4: Validation of the simulation framework against a real MPS measurement [38]. The MPS setup with its components is shown on the left, and the comparison between simulated (thick lines) and measured (thin lines) is shown on the right. Figure adapted from [42].

The biggest advantage of the simulation software is that it allows simulating different setups, including 1D, 2D and 3D MPI scanners, and also MPS systems. The accuracy of the simulation software as a whole was validated by comparing the simulated and measured emf values for a gradiometric MPS setups, as described by [38]. Additionally, it allows accounting for non-ideal behavior of the elements that compose MPI scanners, such as non-uniform drive fields, non-linear gradient fields and space-dependent receive coil sensitivity, since the realistic modeling of the scanner components would produce such design-dependent features. The simulation framework provided valuable insights about the signal generation process, and supported the development of earlier versions of image reconstruction algorithms, as reported in [42, 43, 45].

5.2 2D MPI reconstructions

With a better understanding of the signal generation process, it was possible to start the investigation of MPI image reconstruction methods and potential algorithm optimizations in GPU. Regardless of the method chosen for the image reconstruction, optimal accuracy requires solving linear systems of equation of the form $K \cdot c = s$, where the matrix $K \in R^{m \times n}$ is analogous to the system matrix of equation 3.1, or to a sequence of PSFs given by equation 3.5. This matrix contains, in each row, the characteristic response of the MNP grid for a specific FFP position in space, given by the scanning trajectory as function of time.

In this evaluation, a model-based calibration matrix was employed, representing a FFP scanning a 2D sample following a Lissajous trajectory. The numerical

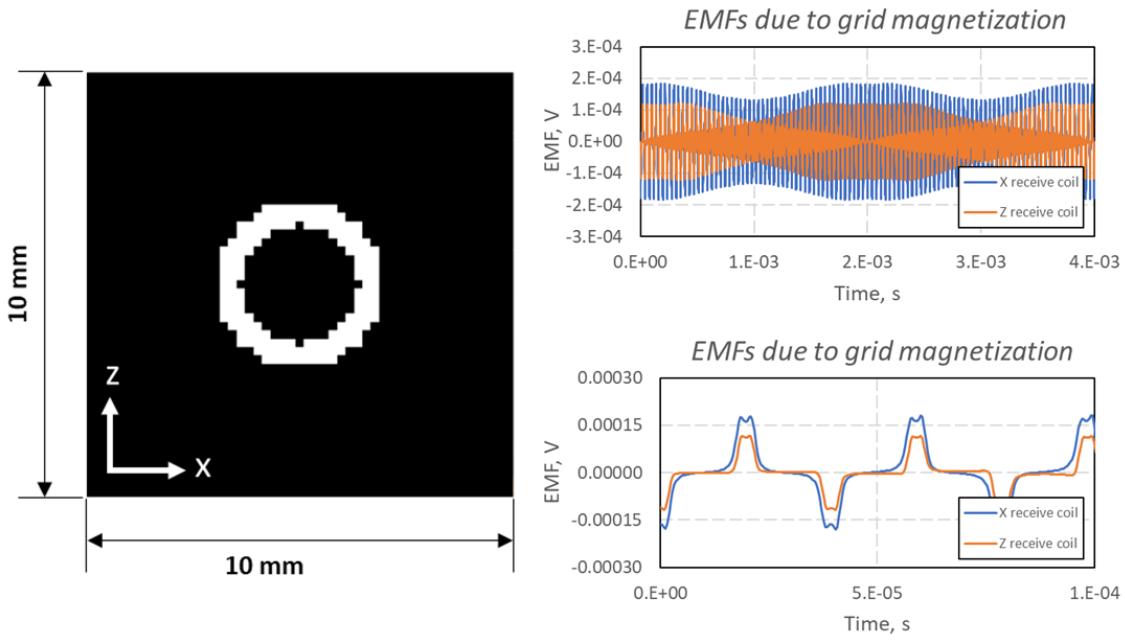

Figure 5.5: Validation of the 2D model-based calibration matrix. A circular 2D MNP phantom (left) was scanned by a FFP following a Lissajous trajectory to generate the signals shown in the top right plot. The plot in the bottom right is a zoom in time to illustrate the signal features resulting from the FFP moving over the phantom.

method implemented for the calculation of the calibration matrix was based of the method shown in [76], and has been validated against the rigorous simulation model described in section 5.1. The calculated calibration matrix was employed to simulate the signals that would be generated by two receive coils, aligned with the x and z directions, while scanning a simulated 2D phantom grid, as shown in figure 5.5.

The model-based calibration matrix was computed for a selection field gradient of 8 T/m (in the x direction), a receive coil sensitivity of 8.4×10^{-4} T/A, a scanning nominal frequency of 25 kHz and 20 nm magnetite particles. The phantom is a 51×51 grid containing particles at a concentration equivalent to 5 mg(Fe)/ml in a circular pattern. The plot in the bottom right side of figure 5.5 illustrates well the signal generation process; as the FFP steers over the circular phantom, the magnetization changes induced in the MNPs cause the measured signals to increase and create a small valley between the two main peaks. It is important to notice that the gradient field generated by cylindrical magnets (or electromagnets) is twice as strong in their symmetry axis directions (as can be implied from figure 5.2), so the resolution in the x direction is two times better than the resolution in the z direction, and then features in the z receive coil signal are less pronounced. The discrete representation of calibration matrix is given by:

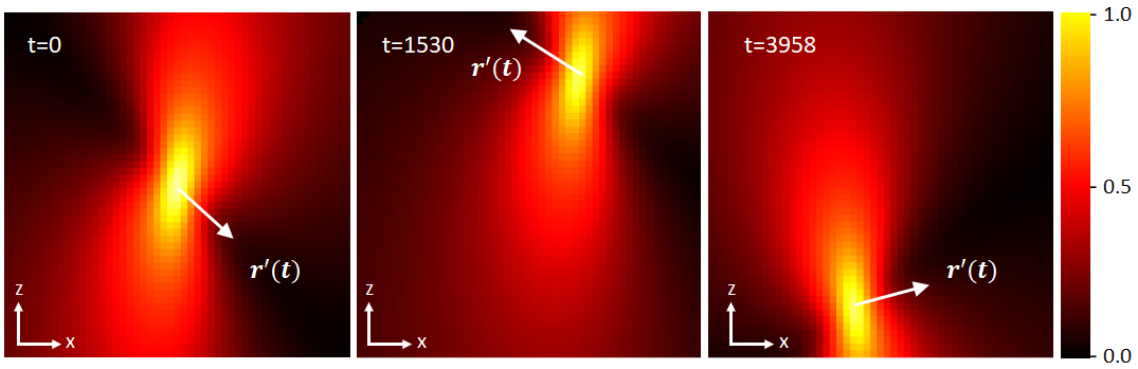

Figure 5.6: Plots of the magnitudes of a 2D scaled PSF (sPSF) following a Lissajous trajectory at t=0, t=1530 and t=3958 (arbitrary units), for a 51x51 voxels grid. The white arrow shows the direction of the FFP movement, and the colored area highlights its area of influence. The sPSF magnitude scale is shown on the right side, and is normalized by the maximum value.

$$\vec{K}_{i,j} = \mu_0 m v \mathbf{R}_j \frac{\vec{\mathcal{M}}\Big(\xi, \vec{H}(t_i + dt, \vec{x}_j)\Big) - \vec{\mathcal{M}}\Big(\xi, \vec{H}(t_i - dt, \vec{x}_j)\Big)}{2 \cdot dt} \tag{5.3}$$

where

$$\vec{H}(t_i, \vec{x}_j) = \mathbf{G}\big[\vec{r}(t_i) - \vec{x}_j\big] \tag{5.4}$$

is the field applied over the position x_j at the time t_i, based on the FFP position $r(t_i)$. $\mathcal{M}(x)$ is the magnetization for a given field magnitude, which can be given by either the ideal Langevin function $\mathcal{L}(x)$ from equation 2.2 or another model that accounts for particle dynamics effects, such as relaxation, more realistically (in section 5.3 it will be shown the importance of this for temperature reconstructions). The time interval dt is arbitrary, and is used to approximate the instantaneous change in magnetization at the time t_i. A good approximation would be obtained by setting dt as a fraction of 10 or 100 of the sampling time interval. The square matrix $\mathbf{R}_j$ represents the sensitivity of each receive coil over the voxel j, v is the voxel volume, and m is the particle's magnetic moment magnitude.

As can be seen in equation 5.3, each $K_{i,j}$ element is a vector, and it represents the characteristic rate of magnetization change in the sensing directions (as given by $\mathbf{R}_j$) as the FFP moves in the space. The calibration matrix is such that each line is then the PSF, represented by the derivative of the Langevin function (or the derivative of any other magnetization function), scaled by the FFP scanning speed, the receive coil sensitivity, the particle's magnetic moment magnitude and the voxel volume. Therefore, each line K_i of the calibration matrix can be seen as a *scaled point spread function*, or simply a sPSF, and representing a system matrix in terms of a scaled PSF is in perfect agreement with [86] in the sense that the system

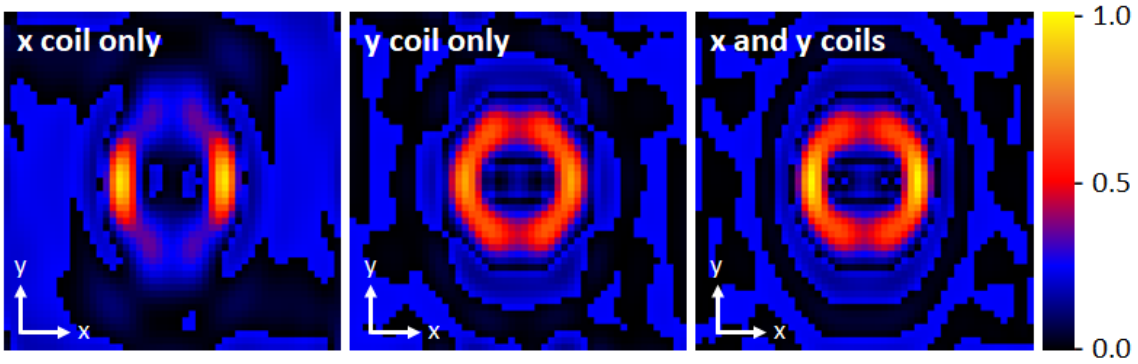

Figure 5.7: Reconstructions using only the x coil (left), only the y coil (center), and both x and y coils (right). The concentration scale is shown on the right side, and is normalized by the maximum value.

function and x-space methods are mathematically equivalent. Figure 5.6 shows plots of a 2D sPSF at different times, illustrating how its magnitude changes over space. The plots highlight the area of influence of the FFP as it steers over the sample – all the voxels covered by the sPSF will contribute to the measured signal, according to the magnitude of the sPSF and the concentration of particles within the voxel, being this the cause of the blurring in the images directly mapped from the signals and the reason why deconvolution is necessary to remove the PSF blurring. It is possible to notice that the sPSF is sharper in the direction of the FFP movement, and much broader in the orthogonal directions, meaning that the both the signal intensities and the resolution of the images taken with detection coils aligned with the FFP scanning directions is significantly improved. This is in perfect agreement with the stated in [15] with respect to the collinear and transverse components of the x-space 3D PSF, and also explains the use of orthogonally oriented receive coils when adopting the 2D or 3D Lissajous trajectories typically used in the system function method scans. Figure 5.7 illustrates the reconstructions of the same circular phantom using data measured by the x coil only, data from the y coil only, and using data from both coils, showing how using orthogonally oriented coils improves the reconstructions.

Once the numerical model for the signal generation has been defined and validated, the application of the model for image reconstructions started to be investigated. The first step was defining a new simulation phantom that contained more distinguishable features then a ring one. It should contain curves, corners and void spaces, and should also have an orientation in space, so any rotation would be noticeable. With these requirements in mind, the new phantom was set as being a 2D question mark (the '?' character), as illustrated in figure 5.8. The phantom grid has a 51×51 resolution, 5 mm of side, and is composed of 20 nm magnetite

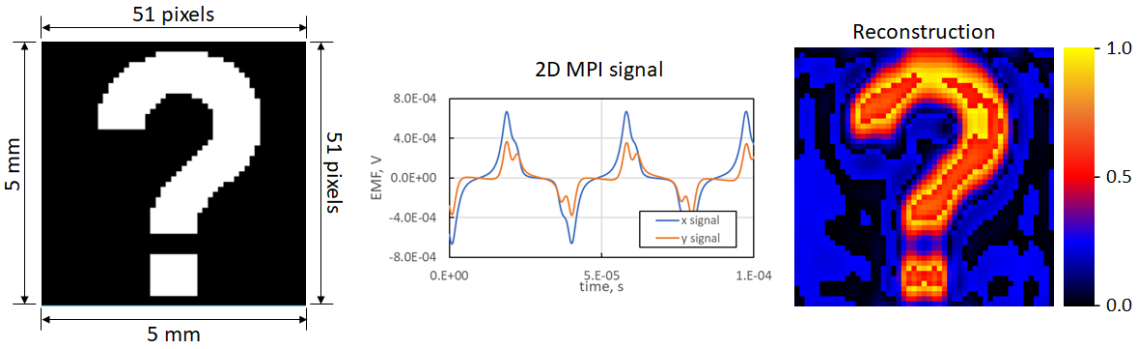

Figure 5.8: Illustration of the 2D question mark phantom grid adopted for the simulations and reconstructions (left). The simulated signals resulting from scanning the phantom (center) we used to numerically reconstruct the grid (right).

particles at a concentration equivalent to 5 mg(Fe)/ml in the area covering the '?' shape. The measurement of the phantom would be done by scanning a FFP over the sample following a 2D Lissajous trajectory. Table 5.1 provides a summary of all the parameters adopted for the simulations and reconstructions for this sample. The simulated signals are also shown in figure 5.8, together with the result of a reconstruction employing 100 iterations of the Kaczmarz algorithm.

For this preliminary reconstruction investigation, four sets of simulated signals were generated. The first one contained no noise, while the other three contained 5 μV, 50 μV and 500 μV of Gaussian white noise, that was added to the original undisturbed signal. Figure 5.9 shows a simplified pipeline of this simulated experiment, while Figure 5.10 shows the signals for these four configurations. Reconstructions were performed using the Kaczmarz algorithm (see algorithm 1) to solve the regularized system of equations defined in equation 3.7, for different regularization levels. Figure 5.11 shows the reconstruction results when no constraints are applied. The first and most striking conclusion can be taken from the reconstruction with no noise, and tells about the fast convergence of the Kaczmarz algorithm. In a single iteration of complexity $O(N^2)$ (composed of N projections and updates), the shape of the question mark is clearly noticeable, meaning that most of the reconstruction is already done. This is typical for the Kaczmarz algorithm when solving systems in which the matrix rows are nearly orthogonal, which is the case of the MPI calibration matrix. It is also evident how regularization acts like a low-pass filter, removing high

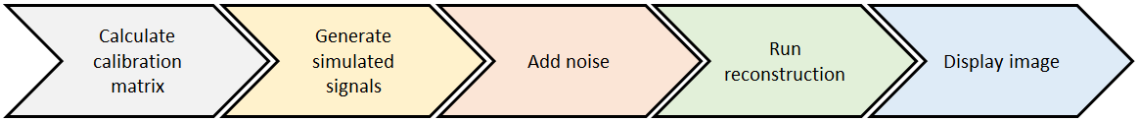

Figure 5.9: Simplified execution pipeline.

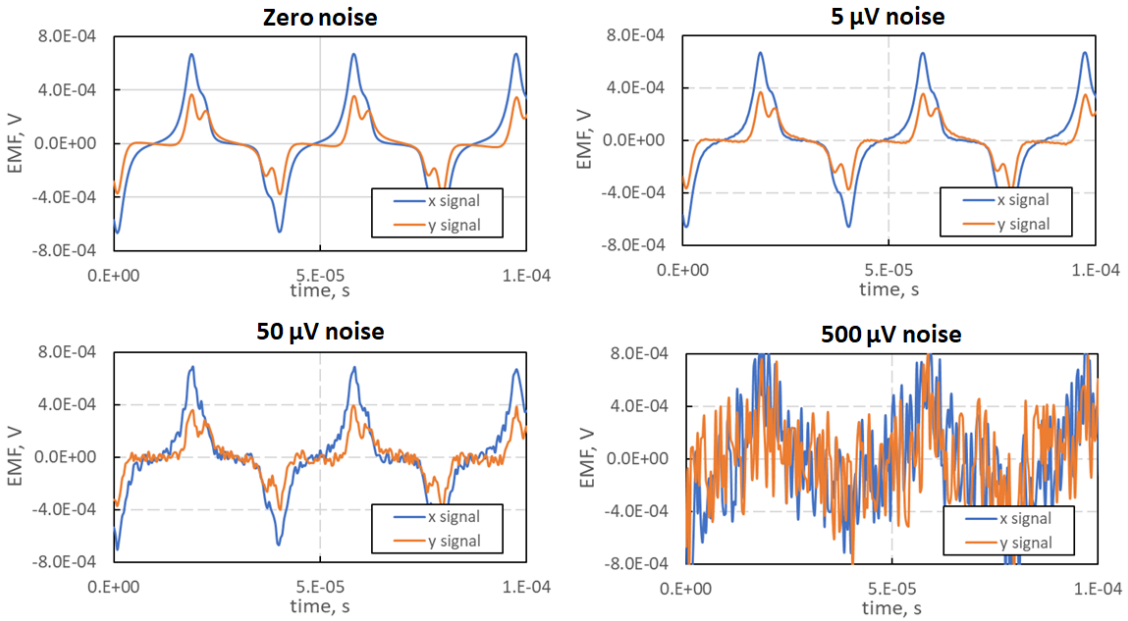

Figure 5.10: 2D MPI signals for the '?' phantom with different noise levels.

frequency features, such as corners and sharp edges, smoothing the reconstructed image. As the noise in the signal increases, it causes the algorithm to diverge with increasing number of iterations, so regularization is required to keep the solution stable. It is worth noting that, for small noise and large regularization values, the reconstruction is almost identical to the one obtained from an undisturbed signal, which means that it managed to suppress all the noise, but also removed part of the spatial information. With increasing noise, the required regularization to reach stable solutions also increases, as the level of details in the reconstructed images is reduced. This is evidenced by the fact that the reconstructions look all the same for 10^8 of regularization, regardless of the noise level.

Symbol	**Parameter**	**Value**	**Units**
d_C	Particle core diameter	20	nm
M_S	Particle saturation magnetization	450	kA/m
c	Particle concentration	$3.168 \cdot 10^{20}$	particles/m^3
T	Temperature	273	K
G	Gradient strength {x,y}	$\{-8, 4\}$	T/m
B	Drive field amplitude {x,y}	$\{40, 20\}$	mT
f_B	Base drive field frequency	250	Hz
f_M	Lissajous frequency multipliers {x,y}	$\{102, 101\}$	-
S_R	Sampling frequency	2.5	MHz
R	Receive coil sensitivity	$8.4 \cdot 10^{-4}$	T/A
I_R	Reconstructed image resolution	51×51	pixels

Table 5.1: 2D MPI simulation and reconstruction parameters

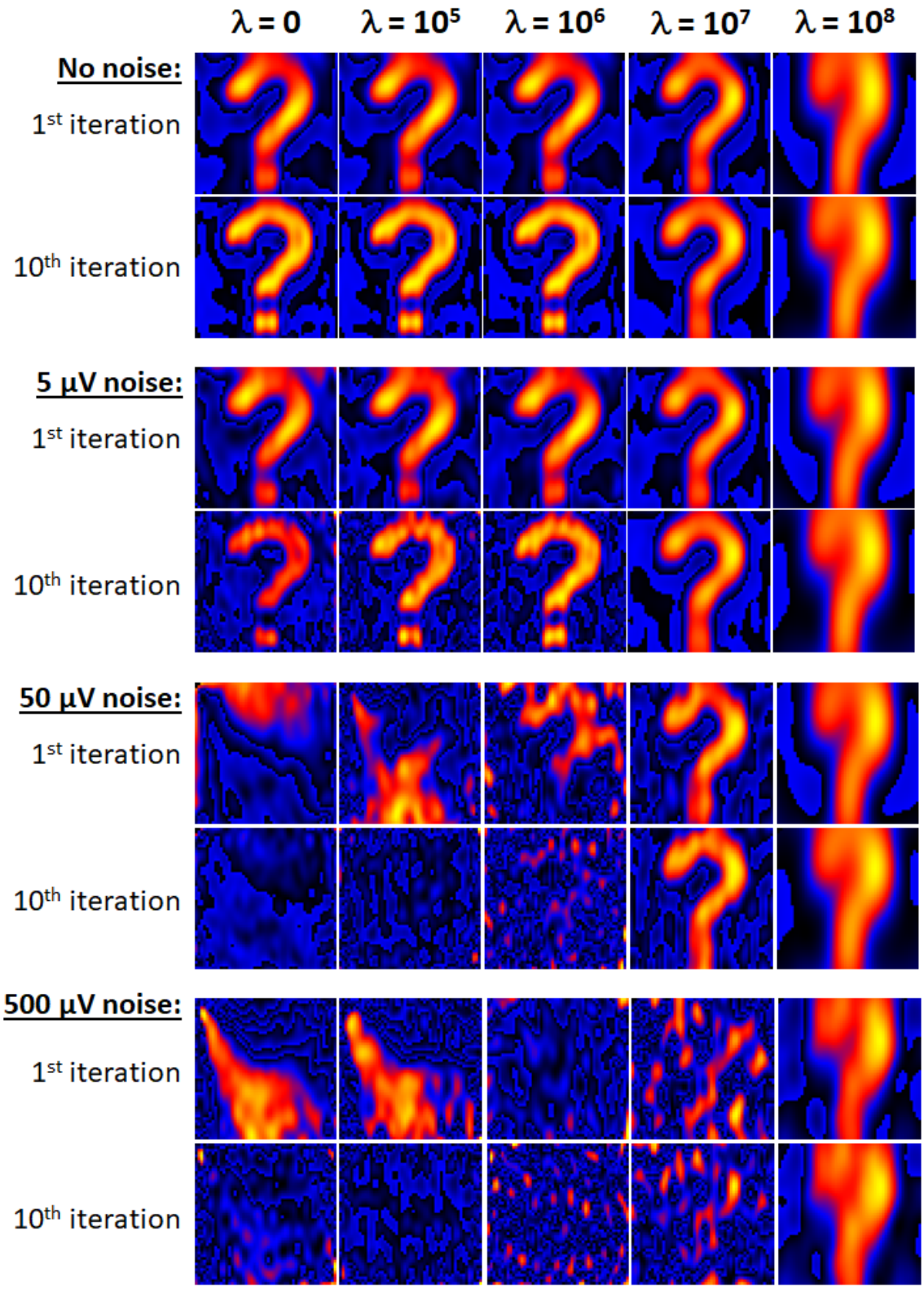

Figure 5.11: Reconstructions of signals with different noise levels using the Kaczmarz algorithm with no constraints for different regularization parameters.

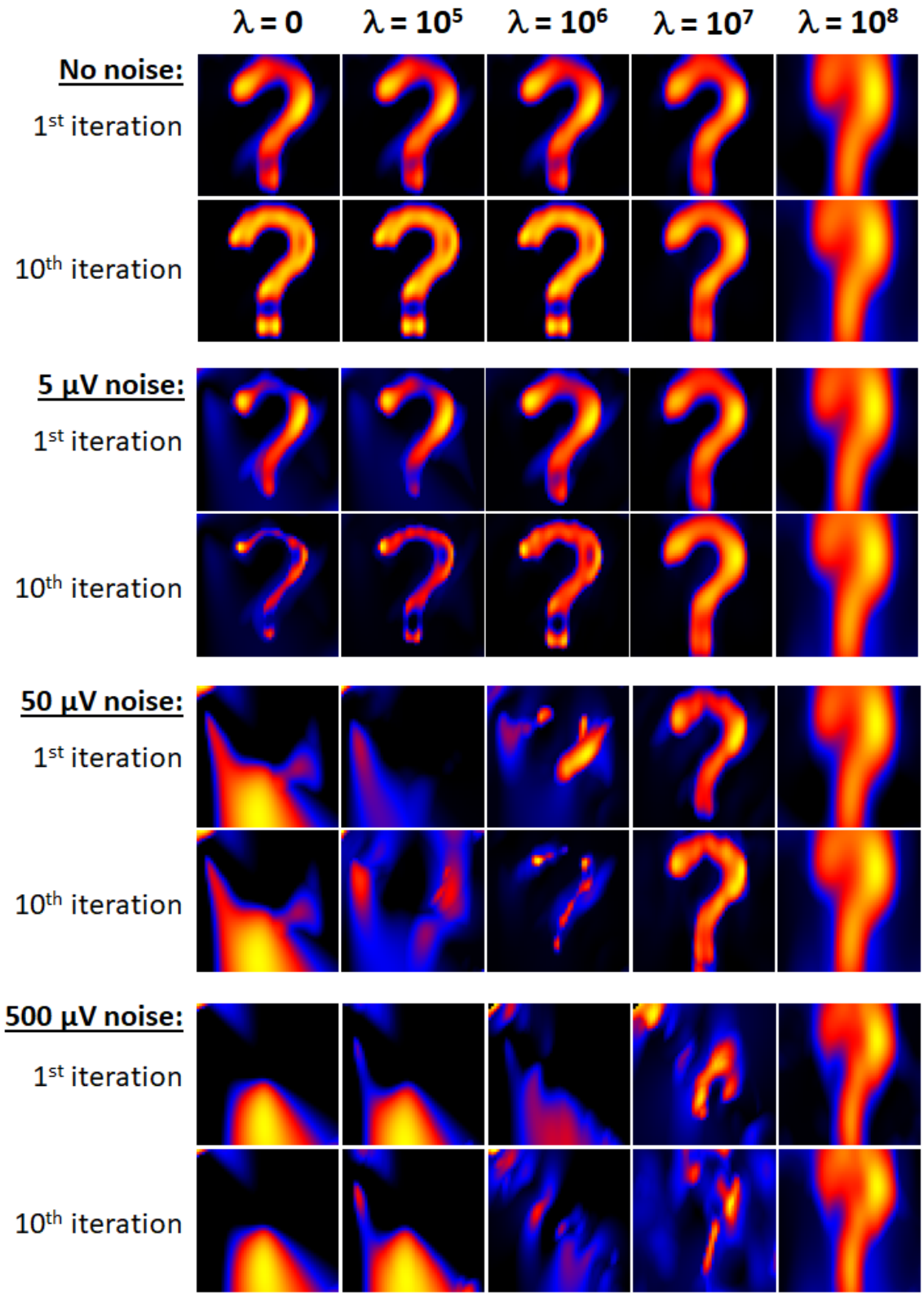

Figure 5.12: Reconstructions of signals with different noise levels using the Kaczmarz algorithm with positivity constraint for different regularization parameters.

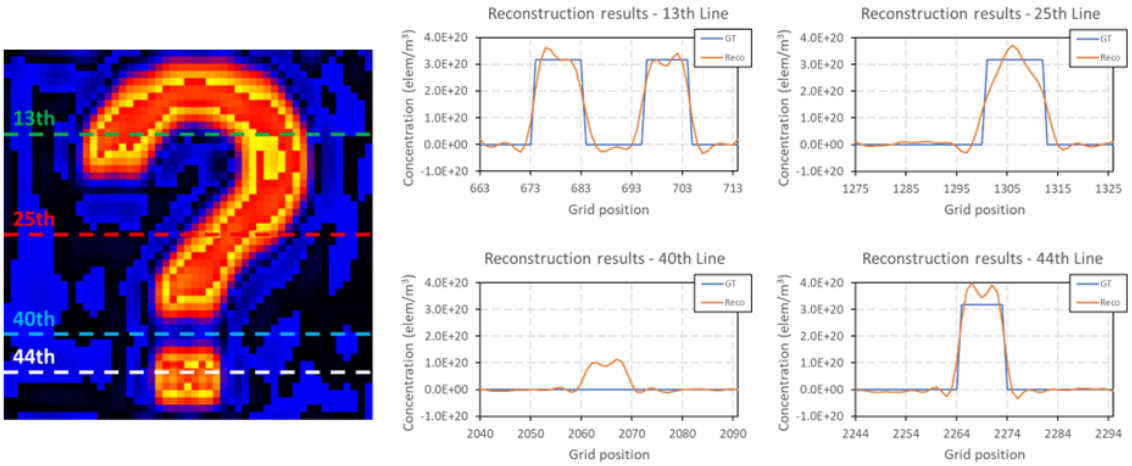

Figure 5.13: Comparison between the reconstructed image and the ground truth. The concentration values over the dotted lines in the reconstructed image on the left were compared to the real concentration values in the plots on the right.

Figure 5.12 shows the results when a positivity constraint is added to the reconstruction algorithm. It is clear how the constraint improved the accuracy of the reconstruction, and removed all the artifacts around the phantom. Since the Kaczmarz is a sequential row projection algorithm, for each iteration the algorithm perform N updates, and for each one of them the positivity of the solution is ensured (see line 7 of algorithm 1). As in the unconstrained resolutions, increasing noise levels progressively compromised the convergence of the algorithm, and required the use of larger regularization parameters, resulting in loss of definition in the reconstructed images.

The accuracy of the reconstruction was also evaluated, by comparing the resulting concentration values against the ground truth (i.e., the phantom grid). Figure 5.13 illustrates such comparison for a 51×51 reconstruction without positivity constraint, after 50 iterations of the Kaczmarz algorithm, against the ground truth. The comparison was done at a number of rows, so certain features such as the lower point, the void space between the point and the main body of the phantom, and the curvature at the top, could be properly evaluated. Even an unconstrained reconstruction provided reasonable results, with the phantom main features consistent with the ground truth. The void space between the point and the main body still showed some residuals, and this a result from the strong influence of nearby particles above and below that region in the measured signals. Such influence would require more iterations to be removed.

A further step on the evaluation of the reconstruction algorithm is to check for any bias in the results. Since the same model can be used to generate the signal and perform the reconstruction, it has to be checked for "inverse crimes"

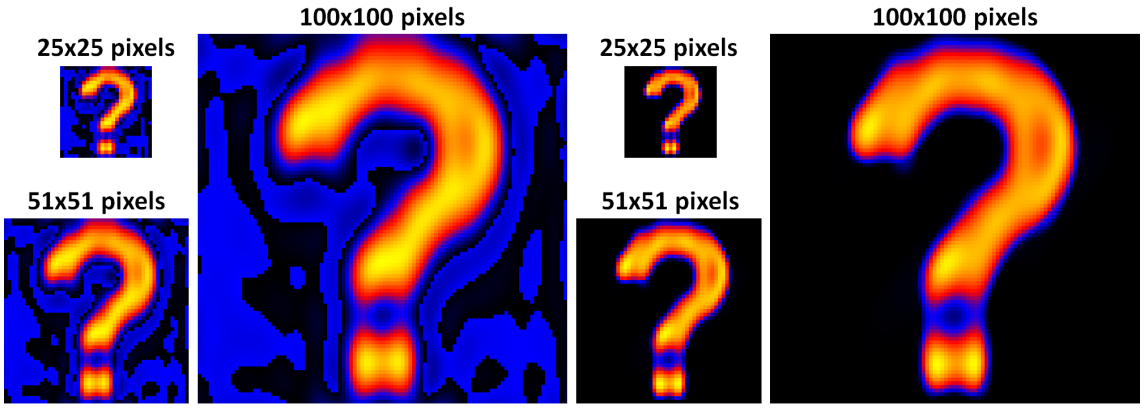

Figure 5.14: Reconstructions with different resolutions using the same dataset as input. Images on the left side were reconstructed with no constraint, while the ones on the right side were reconstructed with positivity constraint.

[114]. Besides adding random noise to the original signals, one way to avoid it is to perform the reconstruction using different resolutions. Figure 5.14 show the results of reconstructions with different resolutions using the same dataset as input, demonstrating the robusteness of the reconstruction.

5.2.1 Performance considerations

The two parameters that impact the most on the computational performance of the reconstructions are the measurement sampling rate and resolution of the reconstructed images. The sampling rate defines the total number of measurements, and consequently the number of rows of the calibration matrix, while the resolution of the image defines the number of columns of the calibration matrix. The number of measurements can be taken from the sampling rate S_R and the base excitation frequency f_B through the relation

$$n = S_R / f_B \tag{5.5}$$

This number of measurements is for a single receive coil, and adding more coils would increase the number of rows proportionally. For the 2D MPI reconstruction example illustrated in figure 5.8, assuming the parameters informed om table 5.1, the calibration matrix would have a size of $20,000 \times 2,601 = 52,020,000$, which is equivalent to 396 MB of double-precision floating point data, and would take about 1.3 s to run, for example. Increasing the resolution to 100×100 as shown in figure 5.14 would increase the size of the matrix in the memory to 1.4 GB, and would increase the execution time to about 4.9 s. Naturally, 3D reconstructions would increase even more the size of the calibration matrix, and consequently the execution times, as will be shown in the next sections. These parameters severely impact the

reconstruction performance, as the execution times increase proportionally to the size of the calibration matrix, so these parameters should be chosen carefully. Is is important to keep in mind that simply increasing the sampling rate (and thus the number of measurements) or the image resolution would not necessarily improve the quality of the reconstructions. Instead, it could compromise the orthogonality of the matrix rows and consequently the convergence of the reconstruction, specially if the Kaczmarz algorithm is the chosen one.

5.2.2 Parallel 2D reconstructions in GPU

Once the signal generation and the reconstruction algorithm were properly characterized, it was time to investigate potential parallel implementations of the MPI image reconstruction in GPU using CUDA. In this section, the development of parallel 2D reconstruction algorithms and the strategies adopted for evaluating these algorithms is described in details. A summary of the development and evaluation of the algorithms described in this section can be found in [70].

Parallel Kaczmarz algorithm

Since the Kaczmarz algorithm is the most popular one in the MPI community, it was the first and natural candidate for parallelization. However, as can be implied from algorithm 1 and already pointed out in [115], since the Kaczmarz algorithm operates one row at a time and each update depends on the previous values, the only operations for which parallelization could be explored were the dot products $S_i \cdot c^{(i)}$ and $\beta \cdot S_i^T$, as shown in lines 4 and 5 of the algorithm. Therefore, unless the problem is large enough to compensate the overhead of successively calling level-one BLAS (*Basic Linear Algebra Subprograms*) functions, it was very unlikely that any parallel implementation of the Kaczmarz algorithm would result in significant speedups. On the other hand, given the popularity of the Kaczmarz algorithm within the MPI community and the fact that the work described on [115] did not compare it against other algorithms, the decision during this research was, for completeness, to implement the parallel Kaczmarz algorithm and compare both serial and parallel implementations against the other algorithms studied in this work. In a sense, the serial Kaczmarz algorithm running in the CPU can be seen as the gold standard to be outmatched. In the parallel implementation of the Kaczmarz algorithm, the dot products in lines 4 and 5 of algorithm 1 were executed in parallel using Nvidia's *cuBLAS* library [107].

Parallel Cimmino algorithm

Since each row projection of the Kaczmarz algorithm computes one correction for the current solution estimate, one interesting parallel approach could be compute all the corrections in parallel and then average them for computing the new estimate. This is precisely the idea behind the Cimmino algorithm (or SIRT, see 3.3.1). In this parallel implementation, each thread would entirely compute the correction factor β_i in line 4 of algorithm 1 for its corresponding matrix row, being this computation implemented in a customized kernel. A second kernel computes the averages of the corrections and apply them to each voxel. The next estimate would then be given by:

$$c^{(i+1)} = c^{(i)} + \frac{1}{M} \sum_{j=0}^{M} \left(\frac{\tilde{v}_j - S_j \cdot c^{(i)}}{||S_j||^2} S_i^T \right) \tag{5.6}$$

These two kernels are called in sequence in every iteration, and while the first kernel is executed by M threads over the rows of K, the second kernel is executed by N threads over the elements of c. The Cimmino algorithm is known for always converging to the minimum least squares solution, even for noisy data.

Parallel block-Cimmino algorithm

As a hybrid method between Kaczmarz and Cimmino, the block-ART algorithm is meant to benefit from the fast-converging properties of the Kaczmarz algorithm and the parallel execution structure of the Cimmino. At each iteration, the calibration matrix is broken in blocks, and each block computes an new estimate by running the parallel Cimmino algorithm over the block. The blocks are executed serially, which compromises the parallelization (and thus the speedups), but increases the convergence rate. In the implementation adopted for this work, the number of rows processed in each block is exactly the number of GPU cores (128 in this case), for optimum throughput. In the limit where the number of rows per block is equal to one, the algorithm is reduced to the Kaczmarz, and in the limit where the number of rows is M, the algorithm is reduced to Cimmino. The opposite could also be done, with each block running Kaczmarz instead of Cimmino. In this case the blocks could be executed in parallel, potentially with equivalent results in terms of accuracy and performance.

Parallel CGNR

As can be observed by examining equations 3.17 and 3.16, and also algorithm 2, the CGNR algorithm works by executing multiple matrix-matrix, matrix-vector and

vector-vector operations. These are classical *BLAS* (*Basic Linear Algebra Subprograms*) operations that are already optimized in Nvidia's *cuBLAS* library [107]. Therefore, all the operations of algorithm 2 but the ones of lines 9 and 14 (which only compute scalars), were implemented in parallel using either *cuBLAS* or intrinsic CUDA functions. A serial version of the CGNR algorithm was also implemented for comparison purposes.

5.2.3 Experiments

All reconstruction algorithms were evaluated and compared to the serial Kaczmarz algorithm. Included in the evaluation are both serial and parallel Kaczmarz implementations, the parallel Cimmino algorithm, the parallel block-Cimmino algorithm, and both the serial and parallel implementation of the CGNR algorithm. All algorithms were employed to reconstruct a simulated signal, generated as described in section 5.2, using the hypothetical 51x51 element phantom grid with a "?" shape descibed in figure 5.8, scanned following a Lissajous FFP trajectory as detailed in table 5.1. As discussed in 5.2.1, the problem size (i.e., the size of the calibration matrix) is $20,000 \times 2,601 = 52,020,000$, equivalent to 396 MB of data, either in the main or GPU memory spaces.

For simplicity of the comparisons, all algorithms were tested using the same signal in time domain with no noise, no regularization or constraints, such as positivity. For performance comparisons, the adopted convention was that one iteration was considered the set of instructions operated over all the rows of the calibration matrix K, which means, for example, that the Kaczmarz algorithm would run M updates on c while the CGNR would perform only one. Also for performance evaluation, there was now filtering or row selection, meaning that all the rows of K were employed in the reconstructions. The accuracy and performance of the algorithms were assessed by analyzing the errors and execution times of all algorithms after 1, 10, 20, 30, 40 and 50 iterations. The experiments were executed using a laptop containing an Intel(R) i7-7700HQ processor, 32 GB of RAM memory, and a NVIDIA GeForce GTX 1060 GPU equipped with 10 *SMs* with 128 cores each, and 6 GB of dedicated RAM memory. The evaluation of the algorithms is explained in details in chapter 6.

5.3 3D *multi-color* T-MPI reconstructions

As part of the *Thermal MagIC* project, one of the objectives of this research was to identify methods for providing fast and accurate 3D temperature imaging using MPI techniques, or T-MPI [36]. One of the methods proposed in the literature for

temperature imaging is the so-called *multi-color* MPI [34], which has been presented in section 3.1.2. The method is based on calibrating the MPI system at multiple temperatures, obtaining then two or more calibration matrices. Understanding how this method can be effectively used for T-MPI is of great importance for the project, and is also a good opportunity for assessing parallel implementations of reconstruction algorithms, since the use of multiple calibration matrices increases the problem size, and consequently the execution times, proportionally. Therefore, this section will discuss in more details MPI reconstructions using the *multi-color* method as a tool for temperature mapping. A summary of the development and evaluation of the method described in this section, including the evaluation of parallel algorithms, can be found in [72].

Regardless of the algorithm adopted for the reconstruction, the method consists on finding the optimal solution of a system of equations with the form:

$$\begin{vmatrix} K_{T_1} & K_{T_2} \end{vmatrix} \cdot \begin{vmatrix} c_{T_1} \\ c_{T_2} \end{vmatrix} = s \tag{5.7}$$

where K_{T_1} and K_{T_2} are the calibration matrices obtained at the temperatures T_1 and T_2 during calibration, and c_{T_1} and c_{T_2} are the unknown concentration vectors at the temperatures T_1 and T_2, for example. The measurement system can be calibrated in as many temperatures as desired, keeping in mind that the number of calibration temperatures proportionally increases the size of the problem to be numerically solved. The method was developed to estimate both spatially-resolved particle concentration and temperature. The particle concentration map is simply given by the sum of the channels c_{T_1} and c_{T_2}, while the temperature map would be given by a linear interpolation of the concentrations given in the output channels. So, for 3 calibration points, for example, the temperature at each position would be given by

$$T_i = \frac{c_{i,T_1} \cdot T_1 + c_{i,T_2} \cdot T_2 + c_{i,T_3} \cdot T_3}{c_{i,T_1} + c_{i,T_2} + c_{i,T_3}} \tag{5.8}$$

where T_1, T_2 and T_3 are the calibration temperatures. Figure 5.15 illustrates the process; a simulated 3D phantom grid with particles at three different temperatures is scanned to generate signals on the x, y and z receive coils, these signals are used, together with the calibration matrices at these temperatures to compute the images at the channels corresponding to each calibration temperature. The final map is built in such way that the final particle concentration at each position is the sum of the concentrations in the three channels at this position, and is represented by the size of the polygons (each grid element is represented by a diamond-shaped polygon), and the temperature at each position is determined by equation 5.8 and

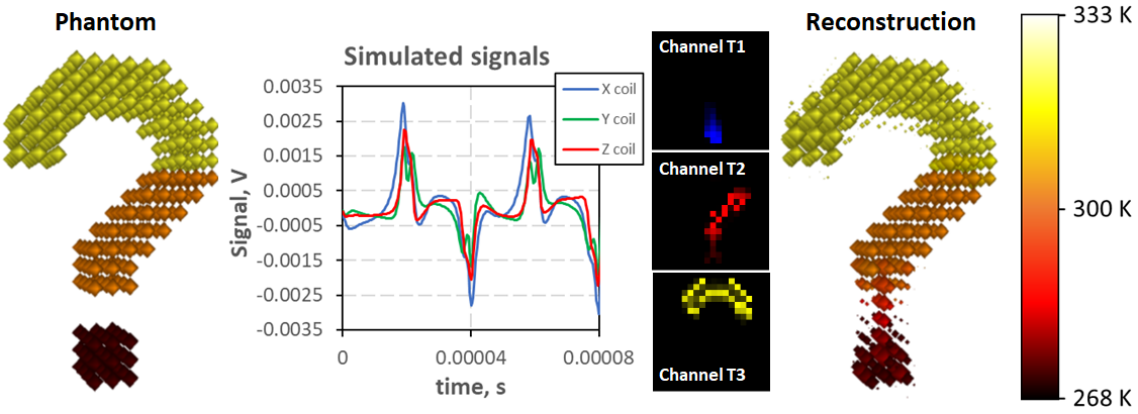

Figure 5.15: Illustration of a simulated multi-color T-MPI measurement, reproduced from [72]. The magnetization signals generated by the hypothetical 3D phantom with particles at three distinct temperatures are measured by the x, y and z pickup coils. These signals are used to reconstruct the concentrations at the calibration temperatures T_1, T_2 and T_3, and the concentrations of the three channels are used to compute the reconstructed concentration (represented by the size of the polygons) and temperatures (represented by the color of the polygons). The image resolution is 20x20x20 *voxels*, and the total FOV volume is (10x10x10) mm^3.

is represented by the color of the polygons, according to the scale at the right side of the figure. The simulation parameters are described in table 5.2.

To better understand this temperature reconstruction process, we need to revisit the work of Rahmer *et al.* [21], where the foundations of the *multi-color* method were established. The method was idealized to allow detecting different particles within a sample, and the output vectors would contain the concentration of each specific particle for which the system has been calibrated to detect. It relies on the fact that each particle type has a specific magnetization response, which depends on a number of factors, including the particle's saturation magnetization and its

Symbol	**Parameter**	**Value**	**Units**
d_C	Particle core diameter	20	nm
M_S	Particle saturation magnetization	450	kA/m
c	Particle concentration	$3.168 \cdot 10^{20}$	particles/m^3
T	Temperature range	273-323	K
G	Gradient strength {x,y,z}	$\{-8, 4, 4\}$	T/m
B	Drive field amplitude {x,y,z}	$\{40, 20, 20\}$	mT
f_B	Base drive field frequency	250	Hz
f_M	Frequency multipliers {x,y,z}	$\{101, 100, 99\}$	-
S_R	Sampling frequency	2.5	MHz
R	Receive coil sensitivity	$8.4 \cdot 10^{-4}$	T/A
I_R	Reconstructed image resolution	$20 \times 20 \times 20$	voxels

Table 5.2: 3D T-MPI simulation and reconstruction parameters

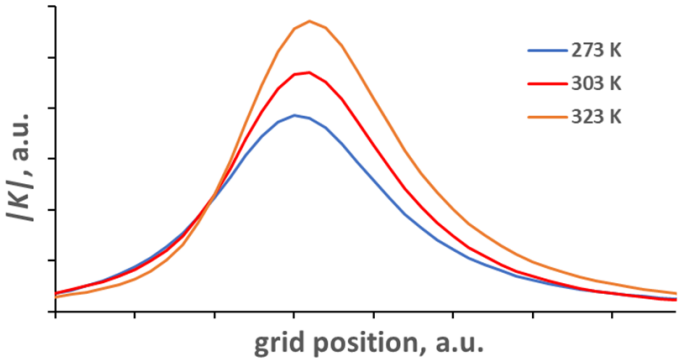

Figure 5.16: Temperature-dependent PSFs. Plots were taken from the same rows of the calibration matrices generated at three different temperatures when accounting for particle relaxation, and illustrate how temperature influences the particle responses. Reproduced from [72].

relaxation time. This combination of factors results in a unique PSF for a specific particle, and therefore a unique calibration matrix. This is key for the success of the method, since it results in calibration matrices with linearly independent rows, allowing then the solution of the extended system of equations.

When applied to T-MPI, the method therefore returns the particle concentration maps at the calibration temperatures. It relies also on the fact that a single particle type, at different temperatures, would exhibit different PSFs and therefore different calibration matrices, with nearly orthogonal rows. The factors that allow this distinction are mostly the temperature-dependent saturation magnetization and relaxation time. Figure 5.16 plots the magnitudes of the same row of three calibration matrices generated at three different temperatures. The lines show the temperature-dependent PSFs, and illustrate how temperature affects their shape in space. The alignment of the particles with an applied field is not instantaneous, and this time-dependent alignment, or relaxation, is governed by the Néel and Brownian mechanisms, which are strongly dependent on the temperature (see 2.5 for more details). According to them, lower temperatures result in longer relaxation times, and these relaxation times result in phase shifts in the measured signals, which in turn are represented as shifts of the PSF in space, since it is moving as it scans the sample.

5.3.1 Parallel 3D T-MPI reconstructions in GPU

The use of multiple calibration matrices has a strong impact on the reconstruction times due to the increased problem sizes. For reconstructions using the parameters of table 5.2, for example, the problem size size is 5.4 GB, and the reconstructions using the Kaczmarz algorithm could easily take from several minutes to a few hours, depending on the system running the reconstruction, the adopted programming

language and the chosen algorithm. Therefore, the *multi-color* T-MPI method is an excellent candidate for being accelerated in the GPU.

Based on the results obtained with the parallel 2D reconstructions, which will be presented in section 6.2, the only parallel algorithm analyzed in this scenario was the CGNR. The parallel CGNR algorithm running on the GPU was compared to the serial Kaczmarz algorithm running on the CPU. No changes were applied to the parallel CGNR implementation described in section 5.2.2.

5.3.2 Experiments

In order to investigate the feasibility of the *multi-color* T-MPI method and to assess the performance gains resulting from running the reconstructions in parallel in the GPU, both the serial Kaczmarz and the parallel CGNR algorithms were employed to reconstruct the simulated signals generated by two phantoms. Both phantoms have the shape of a question mark, as shown in figure 5.15, are composed of $20 \times 20 \times 20 = 8,000$ voxels containing 20 nm magnetite MNPs at a 5 mg/ml concentration. Table 5.2 summarizes the simulation and reconstruction parameters except for the sampling rate S_R, which was reduced to 625 kHz for assessing more realistic reconstruction times, which gives a total of $7,500$ samples per each one of the x, y and z coils, resulting in a problem size of $3 \times 8,000 \times 3 \times 7,500 = 5.4 \cdot 10^8$. One of the simulated phantoms had particles at the three calibration temperatures (273 K at the bottom, 303 K in the middle and 323 K at the top), and the other phantom had a linear vertical temperature gradient (the temperature increased from 273 K at the bottom to 323 K at the top). The simulated phantoms had $(10 \times 10 \times 10)$ mm^3 and voxels with 0.5 mm of size in each side.

The accuracy and performance of the algorithms were assessed by analyzing the errors and execution times of both algorithms after the first iteration and after every 10 iterations up to the 100th iteration. Also for simplicity of the comparisons, both algorithms were tested using the same simulated signals in time domain with no filtering, no noise, and no regularization was applied. These experiments were executed using the same laptop described in 5.2.3. The results of the evaluation are detailed in chapter 6.

5.4 3D MPI reconstructions

The investigations described in sections 5.2 and 5.3 confirmed the potential of parallel algorithms, in particular the CGNR algorithm, for accelerating MPI image reconstructions. However, the previous experiments still lacked important evaluations for demonstrating their feasibility in real life applications. First of all, it was

still necessary to evaluate the impact of noise of over the accuracy of the parallel reconstruction methods. Real MPI signals contain a certain degree of noise, which depends on the measurement system, and therefore, this section describe the evaluation of parallel MPI reconstruction algorithms executed over signals in the presence of different levels of noise.

As mentioned in section 2.4, one of the biggest challenges in MPI is the drive frequency feedthrough, and a common strategy for dealing with it is filtering the fundamental excitation frequency. So far this evaluation dealt only with unfiltered signals in time domain, and even though the methods proposed in this study are mathematically suitable for reconstructing signals in both time and frequency domains, the impact of filtering on the parallel reconstructions was still unknown. Therefore, this section also describes the evaluation of parallel MPI reconstruction algorithms when applied for reconstructing real measurement signals provided by the `OpenMPIData` dataset [110]. The data in this set is in frequency domain, contain noise, and has been filtered for removing the base frequency signals. The evaluation of the reconstructions performed using this dataset then allowed for a complete assessment of the parallel algorithms implemented in this work, and for demonstrating the full potential of the proposed methods. A summary of the development and evaluation of parallel strategies for real-time 3D MPI reconstructions can be found in [116].

5.4.1 Parallel 3D reconstructions in GPU

For this investigation, the steepest descent and the SVD were included as potential parallel alternatives to the serial Kaczmarz algorithm, as follows.

Parallel CGNR

The only difference between the CGNR implementation used in this evaluation in comparison to the one described in 5.2.2 is the addition of regularization to the minimization problem. This required adding vectors for dealing with the lambda-residuals, as can be seen in lines 3, 4, 8, 9, 12 and 13 of algorithm 2. Once more, all matrix-matrix, matrix-vector and vector-vector operations were implemented using the *cuBLAS* library.

Preliminary attempts to add a positivity constraint to the CGNR execution were unsuccessful. In addition to the need of computing a new residual vector, which increased the computation time, the loss of conjugacy between the iterations search vectors would severely compromise the convergence of the algorithm, and therefore was not included.

Parallel Steepest Descent

The steepest descent method was included in this investigation due to its simplicity. It was expected the reduced set of operations would yield a faster execution. Similarly to the CGNR implementation, all linear algebra operations were implemented using the *cuBLAS* library. Differently from the CGNR method, the steepest descent allows implementing a positivity constraint after each iteration, but it also required computing a new residual vector, which increased the execution times.

Parallel SVD

It is well known in the MPI literature that the truncated SVD method delivers accurate reconstructions, always converging to the least squares solution when a proper regularization parameter is chosen. However, since the decomposition if the calibration matrix requires N^3 operations, the executions times are so long that it would be better to use the Kaczmarz algorithm or CGNR instead of the SVD. However, as noted in [62, 63], once the calibration matrix is decomposed, storing the Moore-Penrose pseudoinverse (or its components) in the memory would allow faster reconstructions, since only a few matrix-vector operations would be required. The decomposition, on the other hand, requires a significant amount of free memory to be performed, which is a limitation factor for bigger calibration matrices, and is also extremely time-consuming (it was reported as being more than 600× slower than CGNR or Kaczmarz in [99]).

A parallel implementation of the SVD method was also investigated in this section. The parallel decomposition of the calibration matrix is executed using Nvidia's *cuSolver* library [108]. Even though the decomposition is not fast enough to allow its use in real time, the execution times ranged from 0.63 s to 100 s in this study, being reasonable for practical applications (it could be computed prior to the scan). Once the calibration matrix is already decomposed and stored in the GPU memory, the parallel reconstruction if performed by computing the filter factors presented in equation 3.25 and line 6 of algorithm 4 using a custom kernel, then computing the product $U^T \cdot s$ of line 4 of the algorithm, and computing the element-wise multiplication of line 7 with another custom kernel. The solution vector c is finally computed by another product, as shown in line 9 of the algorithm. Both matrix-vector products shown in algorithm 4 were implemented using *cuBLAS* functions.

5.4.2 Parallel computation of the calibration matrix

The computation of the calibration matrix for model-based reconstructions requires computing, for each line, the scaled PSF as function of the FFP position, FFP speed and receive coil sensitivity. It requires, for each time interval, to determine

the FFP position and speed, and then derive the fields and induced magnetizations in order to compute the calibration signals for each grid position, corresponding to the calibration matrix columns. This is also an expensive operation, with complexity $O(N^2)$. The cost of computing the calibration matrix usually requires it to be calculated once and then be constantly stored in the memory during the whole execution. Any change in the measurement parameters, including field magnitudes and frequencies, sampling rates and even the sample temperature, would require the calculation of a new calibration matrix for keeping the reconstruction accuracy.

To address this limitation, a simple parallel implementation of the computation of the calibration matrix is proposed in this work. A custom kernel was implemented to let M threads simultaneously compute all K_i rows of the calibration matrix, as shown in equation 5.3. For each row, a thread computes the FFP position based on the scanning parameters (drive field amplitudes and frequencies, gradient field strengths and sampling rate) and then determine the magnitudes of the scaled PSF for each grid position in sequence.

5.4.3 Experiments

The experiments with parallel 3D MPI reconstructions were divided into two groups. The first groups of experiments assessed the performance of the algorithms while operating over simulated data. The second group of experiments assessed the accuracy of the proposed reconstruction algorithms when applied over real measurement data. The results of these experiments will be detailed in chapter 6.

Simulated data

The experiments performed with the parallel 3D MPI reconstruction algorithms were designed to assess the accuracy and the performance of the proposed algorithms in the presence of noise. Similarly to what has been done in sections 5.2 and 5.3, the algorithms were employed to reconstruct simulated signals generated by a 3D phantom consisted of $20 \times 20 \times 20$ voxels grid containing 20 nm magnetite particles covering the shape of a question mark, as shown in figure 5.17. The signals generated by the particles of this phantom were measured when scanned by a simulated 3D FFP following a Lissajous trajectory. Table 5.3 summarizes the parameters adopted for these simulations. The data of both the calibration matrix and the measurement signals were generated and processed in time domain. No filtering or row selection were used in the reconstructions, since the goal of this evaluation was to assess the performance of the algorithms while handling large datasets.

For the purpose of this evaluation, this work defines the *problem size* as the size of the calibration matrix. Therefore, the problem size is equal to $3 \times m \times n$, where m

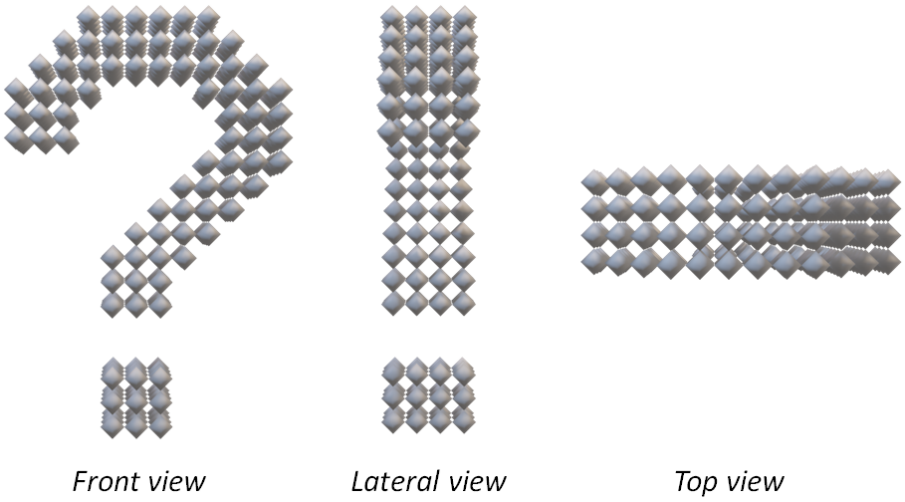

Figure 5.17: Simulated phantom grid adopted for generating the signals used in the assessment of the parallel reconstruction algorithms. Reproduced from [116].

is the number of rows of the calibration matrix (i.e., the total number of samples), n is the number of columns of the the calibration matrix (i.e., the resolution, or number of voxels, of the reconstructed grid), and 3 stands for the number of receive coils, one for each of the x, y and z axis. The scalability of the parallel algorithms – the ability of dealing with larger datasets while maintaining the performance – was assessed by simulating signals and reconstructing images for different sampling rates and resolutions, which also contributed to not incurring in the systematic error of reconstructing the data generated by the same model (or avoiding the "inverse crimes" [114]). The sampling rates adopted were 312.5 kHz, 625 kHz, 1.25 MHz and 2.5 MHz, which result in $1,250$, $2,500$, $5,000$ and $10,000$ samples per receive coil, respectively. The resolutions of the reconstructions were 10x10x10, 15x15x15, 20x20x20, 25x25x25 and 30x30x30 voxels. The problem sizes then ranged from 3.75×10^6 to 8.1×10^8, more than two orders of magnitude. Gaussian noise was added to the simulated signals, and the adopted noise levels were 5 μV (or 0.1 % of the signal peak, or -60 dB) and 50 μV (or 1 % or -40 dB). The regularization parameters adopted were 5×10^5 for the signals with 0.1 % of noise and 10^6 for the signals with 1 % of noise, and they were chosen by analyzing the reconstruction errors and by visual inspection of the reconstructed images.

The reconstruction errors and execution times were taken for all algorithms after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations for the serial Kaczmarz algorithm, with and without positivity constraint, running in the CPU in a single thread, and for the parallel CGNR and steepest descent running in the GPU. Since the SVD is a direct solver that only needs to be executed once and the calibration matrix is previously decomposed and stored in the GPU memory, its execution data were taken after a single execution. The number of iterations was chosen to ensure the

Symbol	Parameter	Value	Units
d_C	Particle core diameter	20	nm
M_S	Particle saturation magnetization	450	kA/m
c	Particle concentration (max)	$3.168 \cdot 10^{20}$	particles/m^3
T	Temperature	295	K
G	Gradient strength {x,y,z}	$\{-8, 4, 4\}$	T/m
B	Drive field amplitude {x,y,z}	$\{40, 20, 20\}$	mT
f_B	Base drive field frequency	250	Hz
f_M	Frequency multipliers {x,y,z}	$\{101, 100, 99\}$	-
t_S	Scan time	4	ms
S_R	Sampling frequency	1.25	MHz
R	Receive coil sensitivity	$8.38 \cdot 10^{-4}$	T/A
S	Sample grid size	{10,10,10}	mm
I_R	Reconstructed image resolution	$20 \times 20 \times 20$	voxels

Table 5.3: 3D MPI simulation and reconstruction parameters

convergence of the algorithms as much as possible. The times of the serial and parallel calculation of the calibration matrix were also taken for a single execution per simulation. The simulations were performed using a desktop configured with an Intel Xeon W-2145 CPU, 32 GB of RAM memory, and a NVIDIA Quadro RTX 4000 GPU containing 36 streaming multiprocessors with 64 cores each, totaling $2,304$ CUDA cores, and 8 GB of dedicated GPU global RAM memory.

Real measurement data

All experiments in this study were performed over simulated data so far. Even though the formulations, the methods and algorithms implemented were well supported by the MPI theory and extensive literature, the accuracy of the reconstruction algorithms needed, for completeness of this study, to be evaluated when running over real measurement data.

The `OpenMPIData` [110] is a public online dataset containing real MPI measurement data, widely adopted in the literature for validating experimental MPI reconstruction algorithms. The dataset was generated by using a commercial MPI scanner to measure the signals generated by three MNP phantoms: the *shape phantom*, the *resolution phantom* and the *concentration phantom*, as shown in figure 5.18. The measurement data asociated with these three phantoms can be used to assess different aspects of the image reconstructions. The shape phantom data allows quickly identifying when the reconstruction algorithm is working properly, the resolution phantom allows assessing the definition of the reconstructions, and the concentration phantom allows identifying the detection limit of the reconstructions. The dataset also include the calibration matrices for the scanner and the measured signals for different configurations and reconstruction resolutions.

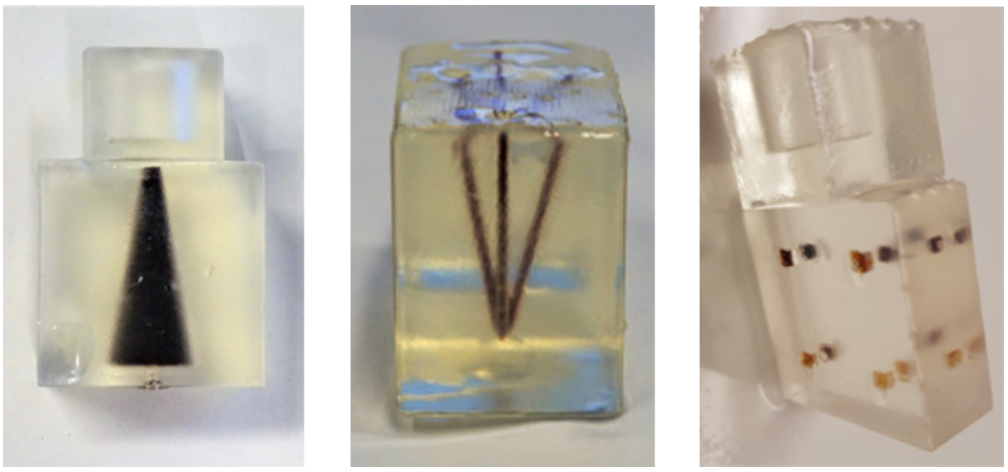

Figure 5.18: OpenMPIData [110] phantoms: Shape phantom (left), resolution phantom (center) and concentration phantom (right).

The datasets studied in this work corresponded to the experiment number 3 (see [110] for a list of all experiments and their parameters), and consisted of the calibration matrix and the signals measured with the three phantoms for a 38x38x19 mm field of view and a 19x19x19 resolution. The calibration matrix and the measured signals were digitally filtered to remove frequencies below 80 kHz, i.e., only the rows at frequencies higher than the cutoff frequency were used. Therefore the reduced calibration matrix occupied 3.86 GB of memory, which corresponds to $75,615$ rows, which includes data of all x, y and z receive coils, and $6,859$ columns, resulting in a problem size of 5.18×10^8. This problem size lies within the range of sizes studied with simulated data and confirmed that the parameters adopted for the simulations were reasonable. However, the calibration matrix was too large for the parallel SVD to be executed in the GPU – it requires an additional memory space to store temporary values during the decomposition – so the SVD reconstruction was evaluated using the calibration data and signal from the x coil only.

Each measurement dataset containing the measured signals is composed of $1,000$ repetitions of the same scan sequence. The noise level of the measured signals was estimated by taking the averages of the standard deviations of the repetitions of each element of the signal vectors. For the shape phantom, the noise was estimated as being approximately equal to 0.14% of the peak signal, or $-56.8\ dB$, while for the resolution and concentration phantoms the estimated noise levels were 0.32% ($-49.9\ dB$) and 0.36% ($-48.8\ dB$), respectively, which also lies within the range of noises adopted for the simulation experiments.

For the reconstructions with all phantoms, a regularization parameter $\lambda = 3.5 \cdot 10^4$ was adopted, based on the study of the reconstructions errors and also by visual inspection of the reconstructed images. The methodology adopted for the experiments using real measured data was the same adopted for the exper-

iments with the simulated data, and the reconstruction errors and execution times were taken for all algorithms after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations for the iterative algorithms, and after a single execution for the SVD reconstruction. All algorithms were assessed using the calibration matrix and signals from the x coil only, while the serial Kaczmarz algorithm, the paralle steepest descent and the parallel CGNR were also assessed using data from all three coils. The experiments were also performed using the desktop described in section 5.4.3.

Chapter 6

Results and discussion

6.1 Evaluation metrics

For the assessment of the parallel algorithms implemented in this work, a number of metrics were adopted. The metrics adopted for assessing the quality of the reconstructions were used in two spaces (or domains). In *signal space*, the reconstruction residuals ($s - K \cdot c$) were analyzed, and the input signals were used as reference for the analysis. In *image space*, on the other hand, a reference image (or the ground truth) was used for the comparison. The metrics will be discussed as follows.

6.1.1 Mean Squared Error - MSE

The Mean Squared Error (MSE) is given by the following expression:

$$MSE(x, y) = \frac{1}{N} \sum_{i=0}^{N} (x_i - y_i)^2 \tag{6.1}$$

where x is the vector being assessed and y is the reference vector. The MSE can be used either to analyze either the reconstruction residuals in signal space or the differences between the reconstructed images and the ground truth (when it is known) in image space. An alternative way of expressing the MSE is by taking it with relation to a reference error. The *Relative* MSE (or RMSE) can also be seen as the MSE with relation to the initial solution estimate, when it is set zero (MSE_0), and is given by

$$RMSE(x, y) = \frac{MSE(x, y)}{MSE_0(y)} = \frac{MSE(x, y)}{\sum_{i=0}^{N} (y_i)^2} \tag{6.2}$$

6.1.2 Peak Signal-to-Noise Ratio - PSNR

The Peak Signal-to-Noise Ratio (PSNR) is usually expressed in decibels (dB) and is given by

$$PSNR(x, y) = 20 \cdot \log_{10}\left(\frac{\max(x)}{\sqrt{MSE(x, y)}}\right) \tag{6.3}$$

where $\max(x)$ is the peak value of x. In this work the PSNR is used to analyze only the reconstruction residuals in signal space.

6.1.3 Structural Similarity Index Measure - SSIM

The Structural Similarity Index Measure (SSIM) is a very popular method developed for comparing images [117]. It gives an index ranging from -1 to 1, where 1 means that the two images are identical, 0 means that they are completely different, and -1 means that they are opposite. The index is given by:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6.4}$$

where μ_x is the average of the values of the image x, μ_y is the average of the values of the reference image y, σ_x^2 is the variance of the values of x, σ_y^2 is the variance of the values of y, σ_{xy} is the covariance between x and y, $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$, where $k_1 = -0.01$, $k_2 = 0.03$, and L is the range of the values of the image pixels – often determined as $2^{\#Bits}$, but can also be real-valued. Given the nature of the index, the SSIM is used to analyze the reconstruction residuals in image space only.

6.1.4 Speedup

The Speedup is simply the ratio between two execution times. When used to determine the acceleration rate of parallel algorithms with relation to their serial counterparts, it is given by:

$$S_P = \frac{t_{serial}}{t_{parallel}} \tag{6.5}$$

where t_{serial} is the execution time of the serial algorithm and $t_{parallel}$ is the execution time of the parallel algorithm. The speedup can be used for determining other two metrics, the *Efficiency*:

$$E_P = \frac{S_P}{n_p} \tag{6.6}$$

and the *Cost*:

$$C_P = t_{parallel} \cdot n_p \tag{6.7}$$

where n_p is the number of processors. The efficiency gives a measure of the speedup relative to the number of processors, and is helpful for comparing results using different architectures with different number of cores. The cost, on the other hand, gives

a measure of the total time spent on the computation, and is useful for comparing the performance of serial and parallel algorithms.

6.2 Parallel 2D MPI results

The main goal of the experiments with 2D MPI reconstructions was to investigate the feasibility of parallel implementations of the reconstruction algorithms and their potential for speeding up the reconstructions. The results of these experiments are shown below.

6.2.1 Convergence and accuracy

Figure 6.1 shows the reconstructions of the 2D MPI signal using the serial Kaczmarz algorithm, the parallel Kaczmarz, the parallel Cimmino algorithm, the parallel block-Cimmino, and the serial and parallel implementations of the CGNR algorithm. The Kaczmarz algorithm shows excellent convergence, and the shape of the phantom is clearly noticeable after only a single iteration. As expected, the results are identical for both serial and parallel versions of Kaczmarz. The serial and parallel versions of the CGNR algorithm also show identical results as expected, but their convergence is worse than Kaczmarz's at the beginning, but it improves significantly with further iterations. The convergence of the Cimmino algorithm is very poor, and even after 50 iterations only a blurred contour of the phantom is visible. Finally, the block-Cimmino algorithm shows a better convergence than Cimmino, but still worse than Kaczmarz and CGNR.

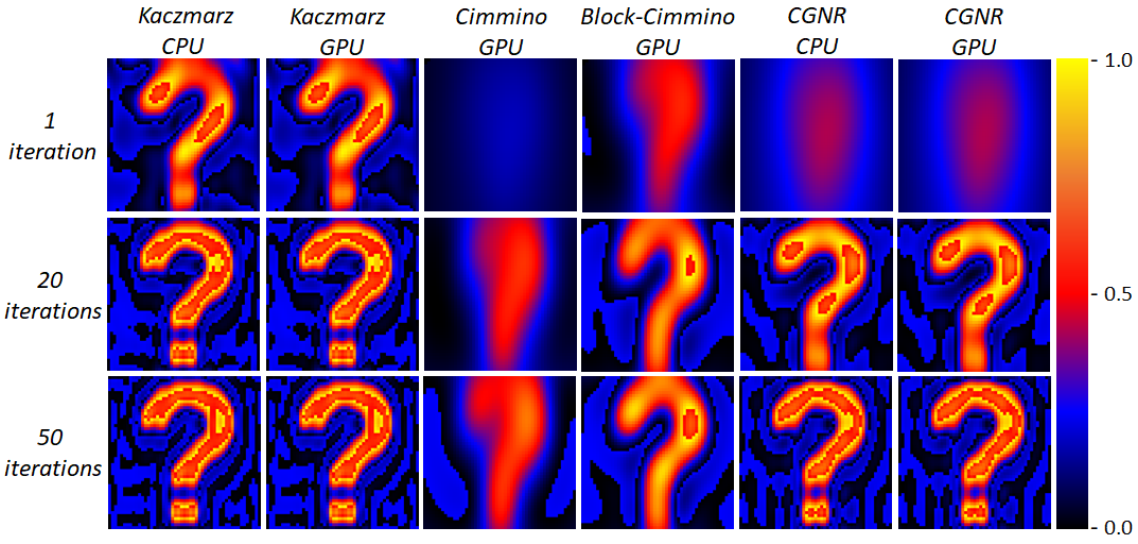

Figure 6.1: 2D MPI reconstruction results after 1 iteration (top line), after 20 iterations (middle line) and after 50 iterations. Reproduced from [70].

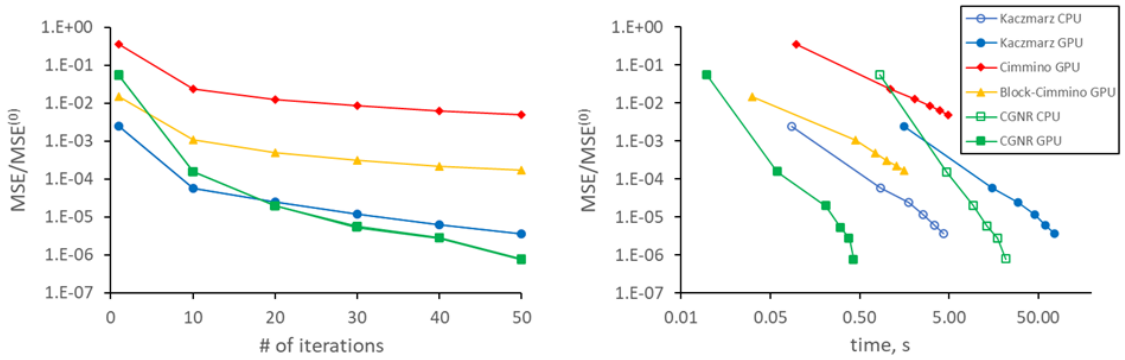

Figure 6.2: Convergence of the 2D MPI reconstruction algorithms in terms of number of iterations (left) and execution time (right). Reproduced from [70].

The left plot of figure 6.2 illustrates the convergence rate of the algorithms studied in this experiment in terms of the RMSEs with relation to the imput signal as funtion of the number of iterations. The Kaczmarz algorithm converges very quickly until about the 10th iteration, and then it slows down. The CGNR algorithm, on the other hand, shows a more consistent convergence rate, and after 20 iterations it surpasses the Kaczmarz algorithm. The plot also confirms the poor convergence of the Cimmino algorithm, with RMSE values nearly four orders of magnitude higher than the Kaczmarz and CGNR, while the block-Cimmino algorithm showed RMSE values about two order of magnitude lower than the Cimmino algorithm, but still about two orders of magnitude higher than CGNR and Kaczmarz. While the averaging of the projections would ensures the solution of the least squares minimization problem [115], it excessively smooths the convergence, compromising its performance.

6.2.2 Performance

Table 6.1 summarizes the execution times of the algorithms studied in this experiment, as well as their speedups with relation to the serial Kaczmarz algorithm executed in the CPU (KACZ CPU), and the reported times correspond to the average of five independent executions. The first conclusion is that the parallel Kaczmarz algorithm executed in the GPU (KACZ GPU) is on average 17.4× slower than the serial Kaczmarz algorithm executed in the CPU. As already discussed in 5.2.2, due to the nature of the algorithm, only a limited number of operations could be executed in parallel, compromising its performance. The Cimmino algorithm (CIMM GPU) was 1.2× slower than the serial Kaczmarz, on average, and the block-Cimmino (BLK-CIMM GPU) was 2.5× faster, with none of them showing improvements in terms of reconstruction accuracy. However, while the serial CGNR (CGNR CPU) was on average 5.6× slower than the serial Kaczmarz algorithm, the parallel CGNR (CGNR GPU) accelerated its serial counterpart by 58.2×, resulting in an average

		KACZ CPU	KACZ GPU	CIMM GPU	BLK-CIMM GPU	CGNR CPU	CGNR GPU
1 iter.	time, ms	86	1,566	96	31	843	9
	S_p	-	0.06	0.89	2.7	0.10	8.9
10 iter.	time, ms	862	15,091	1,108	452	4,677	59
	S_p	-	0.06	0.78	1.9	0.18	14,4
20 iter.	time, ms	1,761	29,918	2,074	744	9,921	211
	S_p	-	0.06	0.85	2.4	0.19	8.3
30 iter.	time, ms	2,576	45,281	3,038	1,006	13,277	307
	S_p	-	0.06	0.85	2.6	0.19	8.4
40 iter.	time, ms	3,438	60,164	3,966	1,288	17,588	382
	S_p	-	0.06	0.86	2.7	0.20	9.0
50 iter.	time, ms	4,333	75,454	4,913	1,559	21,827	429
	S_p	-	0.06	0.88	2.8	0.20	10.1

Table 6.1: Execution times and speedups S_p of the 2D reconstruction algorithms. The speedups are relative to the execution time of the serial Kaczmarz algorithm.

speedup of 9.9× with relation to the serial Kaczmarz algorithm, while showing superior accuracy.

Since the algorithms converge differently, a comparison based only on the execution times of a single iteration is not an appropriate way of estimating the actual speedups of the reconstruction algorithms, and the quality of the reconstructions should also be taken into account. The right plot of figure 6.2 shows the RMSE values of the algorithms studied in this experiment as function of time. Clearly the parallel CGNR is the most efficient algorithm, reaching a RMSE lower than 10^{-4} even before the Kaczmarz algorithm finishes its very first iteration. Considering the time spent by the algorithms to reach a RMSE lower than 10^{-5}, the parallel CGNR is 11.2× faster than the serial Kaczmarz, a speedup bigger than the average per-iteration acceleration. It happens because, even though the Kaczmarz algorithm converges very quickly in the first iterations, its convergence rate slows down after 10 iterations, as shown the left plot of figure 6.2. Even when accounting for the reconstruction accuracy, the parallel Kaczmarz, the parallel Cimmino, the parallel block-Cimmino and the serial CGNR showed a worse performance when compared to the serial Kaczmarz algorithm.

The results with the parallel CGNR algorithm were very promising, and encouraged the pursue of parallel implementations for 3D image reconstructions. The results of this first investigation on parallel MPI reconstruction algorithms were published in [70], and have already been confirmed by another research group, as can be seen in [71].

6.3 Parallel 3D T-MPI results

The main goal of the experiments with parallel 3D T-MPI reconstructions was to test the feasibility of the multi-color method for temperature imaging, while investigating the speedups of the parallel CGNR algorithm for larger 3D MPI datasets. The results of these experiments are shown in the next pages.

6.3.1 Convergence and accuracy

Figure 6.3 shows the results of the 3D T-MPI reconstructions for two different signals. On the left side, the reconstruction of the signals simulated for a phantom at exactly the three calibration temperatures (T_1, T_2 and T_3) is shown. On the right side is the reconstruction of the signals simulated for a phantom under a vertical temperature gradient. The ground truth phantoms on the left side of their respective reconstructions illustrate the actual temperature profiles. The three 2D color images below the 3D images are the central cross sections of the reconstructions for each one of the T_1, T_2 and T_3 reconstruction channels. As shown in 5.3, the resulting particle concentration is given by the sum of the concentrations in the three channels, while the temperature is given by an interpolation of these values, according to equation 5.7. Both reconstructions were performed using the same parameters, including the calibration matrix, reconstruction algorithm (Kaczmarz) and the number of iterations – 100, in the case illustrated in the figure.

As can be seen in the image, the multi-color T-MPI method worked well when the particles are at the calibration temperatures, and the reconstructed image shows clearly the distinct regions, in special the top region. The images at the three channels confirm that the method successfully detected the particles at the three temperatures, as expected. The accuracy in both concentration and temperature is lower in the bottom of the image, and this might be attributed to the fact that the difference between the PSFs at 273 K and 303 K is smaller than the difference between the PSFs at 303 K and 323 K, which favors a better distinction between the two higher temperatures (see image 5.16). Also, the magnitude of the PSF is smaller at 273 K, and this could contribute to a slower convergence in that region.

However, the same cannot be said when some particles were not exactly at the calibration temperatures. The reconstruction at the right side of figure 6.3 shows that the method failed to converge properly for the phantom under a temperature gradient, even if the gradient lies within the calibration temperatures. The concentration at some points in the central region of the image (represented by the size of the polygons) are significantly higher than expected. Additionally, the temperatures of the reconstructed image do not match the temperatures of the phantom. The cross section images at the three channels support this, showing that the reconstruction

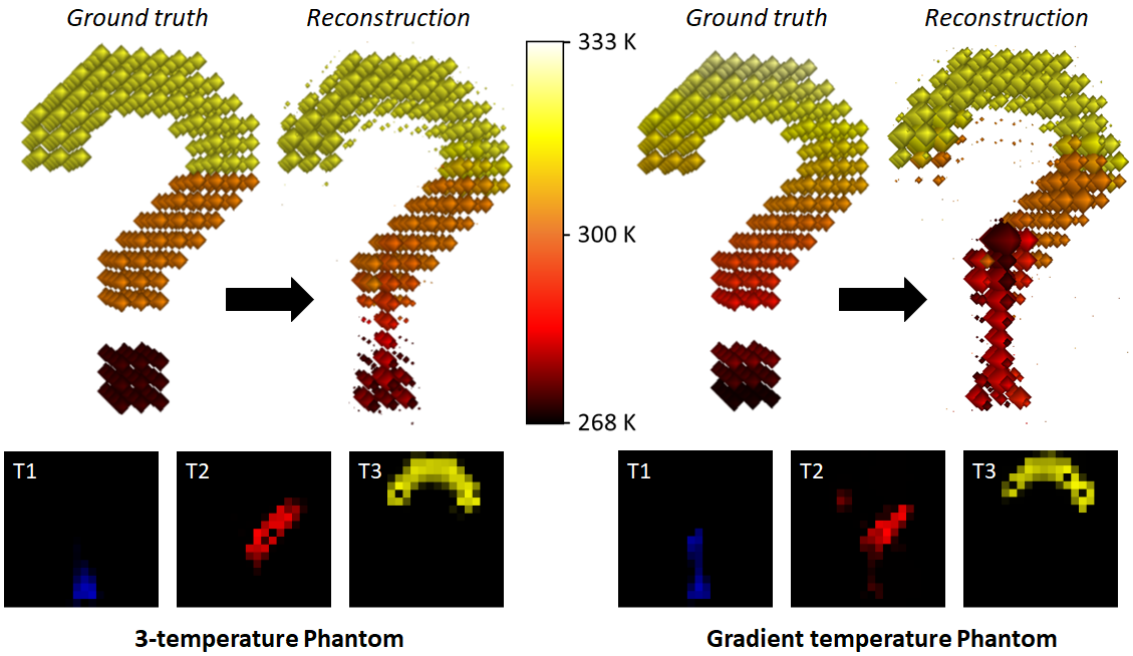

Figure 6.3: T-MPI reconstruction results with the phantom with the particles at the three calibration temperatures (left) and with the phantom with a vertical temperature gradient (right). The images in the bottom are cross sections of the central portion of the 3D image for the channels T_1, T_2 and T_3. The size of the polygons is proportional to the particle concentration in that specific voxel, while the color is proportional to the temperature at that position, according to the color scale in the center of the image. Adapted from [70].

failed to return images of a temperature gradient.

These results suggest that the multi-color MPI method has limited interpolation capabilities, and it finds support in the fundamentals of the multi-color MPI method [21]. The method was developed to determine the concentration of particles of different kinds, as defined by their calibration matrices. In other words, the method returns how many particles of each kind are present in a specific sample. When applied to T-MPI, the method then determines how many particles are at each of the calibration temperatures only, and the results above suggest that the method cannot resolve intermediate temperatures. What gives the method the ability of detecting particles at different temperatures are the different PSF, which means that the temperature range and resolution of the method depends on having several calibration matrices closely spaced in temperature, which is not computationally feasible, since the size of the calibration matrices can easily occupy several GB of memory space. Moreover, the different calibration matrices have to be nearly orthogonal to allow the algorithm to converge, which requires particles very sensitive to temperature changes to allow finer resolutions. An alternative to the multi-color method could be using the recently reported harmonic PSF approach [118] for 3D

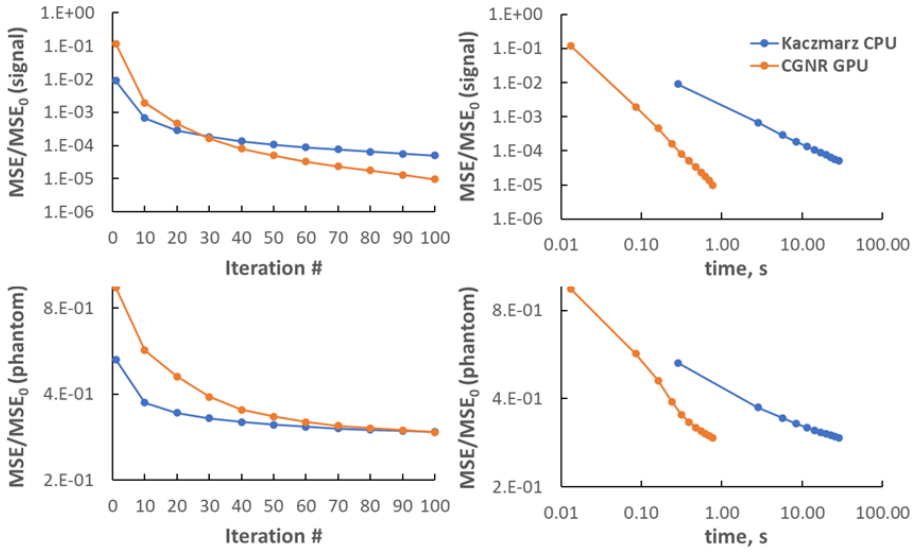

Figure 6.4: Convergences of the serial Kaczmarz algorithm and the parallel CGNR employed for 3D T-MPI reconstructions. The plots show the RMSE values with relation to the simulated signals (top) and with relation to the ground truth (bottom), as function of the number of iterations (left) and time (right). Reproduced from [72].

temperature imaging and it should be investigated.

The left plots of figure 6.4 illustrates the convergence of the serial Kaczmarz and the parallel CGNR algorithms, denoted by their RMSE values with relation to the simulated signal (top) and with relation to the ground truth (bottom), as function of the number of iterations. Similarly to what was seen with the 2D MPI (see 6.2), The Kaczmarz algorithm converges very quickly in the first iterations, but the convergence slows down after the 10th iteration. The convergence of the CGNR is slower than Kaczmarz's in the first iterations, but is more consistent and eventually the CGNR reaches the Kaczmarz. In signal space, it happens after 30 iterations, but in image space it happens only after 100 iterations. One explanation for this is that since the system of equations has $22,500$ rows and $24,000$ columns, it is underdetermined and therefore has multiple solutions that minimize the residuals $(s - K \cdot c)$ and produce similar RMSE values in image space. The differences between the results in different spaces will be further investigated in the next section, when the final results with 3D MPI will be shown and discussed in details.

6.3.2 Performance

The right plots of figure 6.4 illustrate the convergences of the serial Kaczmarz algorithm and the parallel CGNR, denoted by their RMSE values with relation to the simulated signals (top plots) and with relation to the ground truth (bottom plots), as function of time. In time domain is possible to better visualize the performance

gains resulting from the adoption of the parallel CGNR algorithm. By the time the serial Kaczmarz finishes its first iteration, the CGNR nearly finished more than 30 iterations, reaching RMSE values nearly two orders of magnitude lower than Kaczmarz's in signal space, and about 50% lower in image space. The average speedup of the parallel CGNR algorithm with relation to the serial Kaczmarz is 36.6×, and confirms the potential of the algorithm for providing more efficient MPI or (T-MPI) reconstructions. The results of this investigation were published in [72].

6.4 Parallel 3D MPI results

The experiments with the parallel CGNR algorithm for the reconstruction of 2D concentration images and 3D concentration and temperature maps demonstrated the potential of GPU-accelerated strategies for speeding up MPI image reconstructions. At this point of the research, the main goal was to fully demonstrate that the parallel algorithms are suitable for providing real-time reconstructions while ensuring that the accuracy of these reconstructions remain in optimum levels under real life measurement conditions. The experiments included simulated signals, as well as real measured signals provided by the `OpenMPIData` dataset [110]. The results of the experiments with these data are shown in the next sections.

6.4.1 Simulated data

Convergence and accuracy

Figure 6.5 illustrates the convergence of the 3D MPI reconstruction algorithms in signal space, denoted by the RMSEs on the left two plots and by the PSNRs on the right two plots, as function of the number of iterations. The algorithms were employed for the reconstruction of the simulated signals with a 20x20x20 resolution, and to these signals, 5 μV (0.1%, top two plots) and 50 μV (1%, bottom two plots) of Gaussian noise were added to reproduce real measurement conditions and assess the accuravy of the regularized reconstruction algorithms. As already seen with the results with 2D MPI and 3D T-MPI, the Kaczmarz algorithm showed very good convergence in the first iterations – after a single iteration, the RMSE of the unconstrained Kaczmarz algorithm (KACZ_CPU, blue circles) dropped to less than 10^{-2} (or a PSNR higher than 30 dB), and after 10 iterations it reached less than 10^{-3} (or a PSNR higher than 40 dB), even for 1% of noise. The convergence of the constrained Kaczmarz algorithm (KACZ_PC_CPU, blue triangles) was slightly worse in its first iterations, starting with higher RMSE and lower PSNR values than the unconstrained Kaczmarz, but after 20 iterations it reached better values. The parallel CGNR algorithm (CGNR_GPU, green squares), also as shown in the results

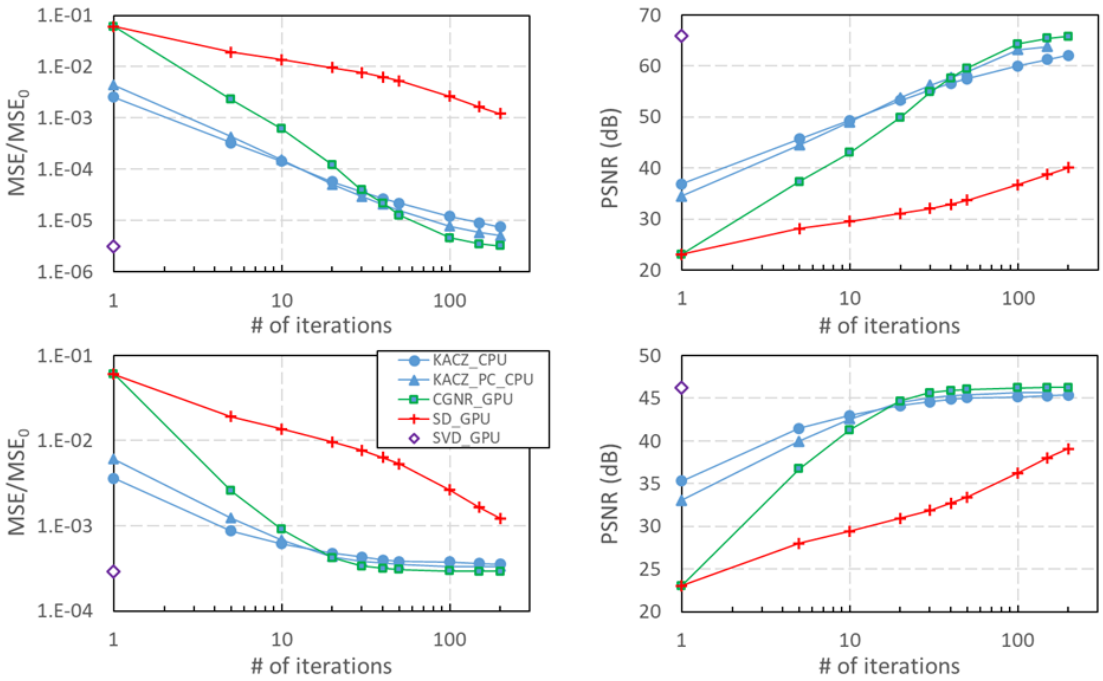

Figure 6.5: Convergence of the 3D MPI reconstruction algorithms in signal space as function of the number of iterations. The plots show the RMSE values (left) and PSNR values (right) of the reconstructions with relation to the measured signals, for 5 μV (0.1% or -60 dB, top) and 50 μV (1% or -40 dB, bottom) of noise. The points correspond to values after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations.

with 2D MPI and 3D T-MPI, showed a modest convergence in its first iteration, but consistently decreased its RMSE values and increased its PSNR values, and eventually surpassed the Kaczmarz algorithm after about 40 iterations. The steepest descent algorithm (SD_GPU, red crosses), on the other hand, showed very poor convergence, and even after 200 iterations it did not manage to reach the accuracy of the Kaczmarz algorithm after only 5 iterations, or the accuracy of the CGNR algorithm after 10 iterations. The reconstructions employing SVD (SVD_GPU, open purple diamonds) resulted in better metrics (lower RMSEs and higher PSNRs) regardless of the noise level. Since the SVD is a direct solver, only one iteration of it is shown in the plots.

One advantage of assessing the reconstruction algorithms using simulated data is the fact the ground truth (i.e., the actual space-dependent MNP concentration) is known, which allows the evaluation of the reconstructions also in image space. Figure 6.6 illustrates the convergence of the reconstruction algorithms in image space, denoted by the RMSE values on the left plot and the SSIM values in the right plot, as function of the number of iterations. The plots correspond to the reconstructions of the signal with 1% of noise generated by the 20x20x20 phantom of figure 5.17, which was used as ground truth for the calculations of the RMSE and

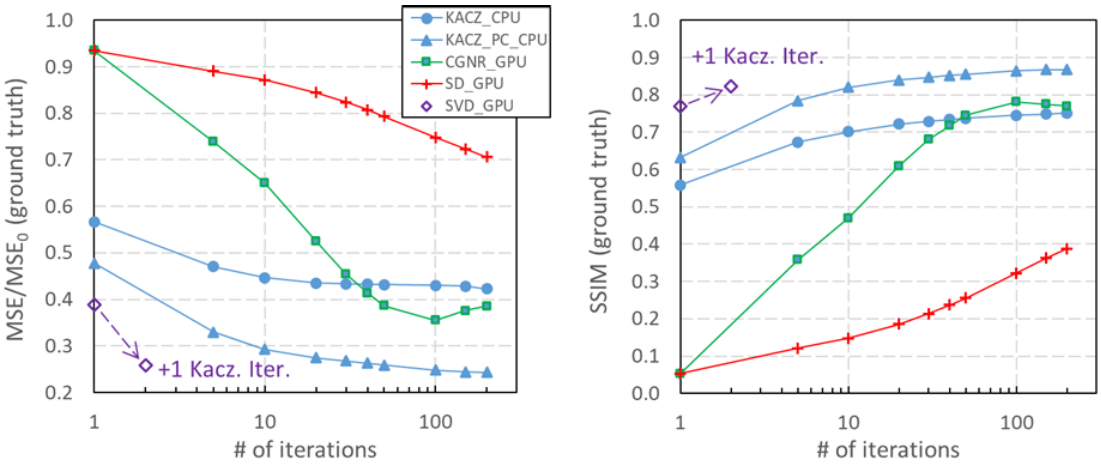

Figure 6.6: Convergence of the 3D MPI reconstruction algorithms in image space as function of the number of iterations. The plots show the RMSE values (left) and SSIM values (right) with relation to the ground truth for the reconstructions of the signal with 1% of noise. The points correspond to values after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations.

SSIM values. In image space, the reconstructions using the constrained Kaczmarz algorithm obtained better scores than all other algorithms, including the SVD, which has shown the best results in signal space. This suggests that even though the SVD converges to the minimum least squares solution, the inclusion of a constraint to the system of equations leads to more accurate solution. Figure 6.7 shows the 3D reconstructions with all algorithms, and also illustrates the improved accuracy of the constrained Kaczmarz reconstruction in comparison to the unconstrained ones.

It has been found in the course of this study that adding one iteration of the constrained Kaczmarz algorithm can improve the accuracy of the unconstrained reconstructions. As can be see in Figure 6.6, adding a single constrained Kaczmarz iteration after the SVD reconstruction can significantly reduce the RMSE and increase the SSIM to values close to the values obtained by the full constrained Kaczmarz reconstruction after 200 iterations, even closer than the values obtained with the unconstrained Kaczmarz algorithm. Figure 6.7(f) also illustrates how adding a single constrained Kaczmarz iteration can improve the quality of the reconstruction, removing unwanted artifacts around the main volume. The same approach can be adopted for the unconstrained CGNR reconstruction, with the same results. It is also interesting to notice that all unconstrained reconstructions converge to the same images, including the Kaczmarz one – the reconstruction with the steepest descent algorithm would as well, presumably, if the number of iterations were large enough.

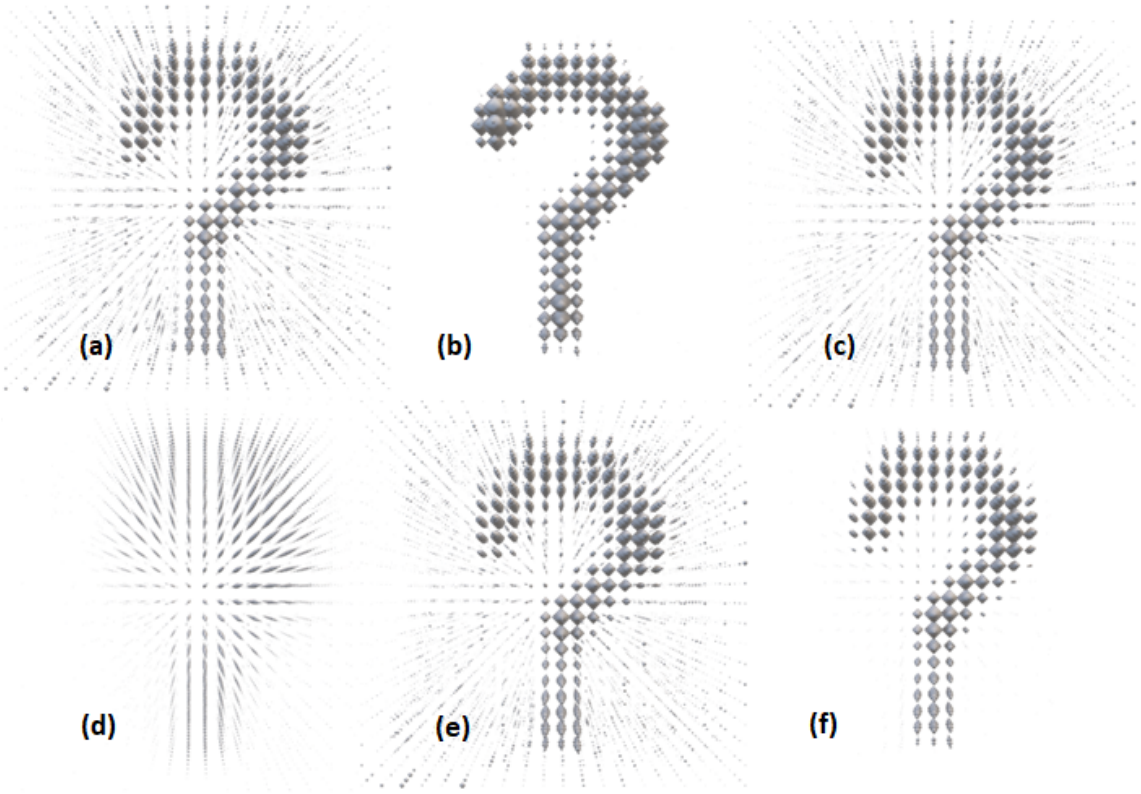

Figure 6.7: 3D reconstructions after 200 iterations for a signal with 1% of noise. The images correspond to (a) the serial unconstrained Kaczmarz algorithm, (b) the serial Kaczmarz algorithm with positivity constraint, (c) the parallel CGNR algorithm, (d) the parallel steepest descent algorithm, (e) the parallel SVD reconstruction, and (f) the parallel SVD reconstruction followed by one iteration of the Kaczmarz algorithm with positivity constraint. Reproduced from [116].

Performance

Figures 6.8 and 6.9 illustrate again the convergence of the 3D MPI reconstruction algorithms studied in this work, but now as function of the execution time instead of the number of iterations. This analysis is necessary because the algorithms have their own execution times, and their performance should then be measured in terms of convergence per time unit. In time domain, it is clear the performance gain resulting from the adoption of the parallel algorithms, in special the CGNR and SVD. By the time serial Kaczmarz algorithm finished its very first iteration, more than 30 iterations of the parallel CGNR algorithm had already been executed, resulting in significantly better RMSE and PSNR values in comparison to Kaczmarz's results. The Kaczmarz's scores after one iteration that took 184 ms are comparable to the ones of CGNR after five iterations and 39 ms, a speedup of about 3.8×. After 10 iterations of the Kaczmarz algorithm (a number of iterations typically used in the literature), the accuracy of the reconstruction would be comparable to the one of

$\mathbf{MSE/MSE_0}$	**Kaczmarz (CPU)**	**CGNR (GPU)**	**SVD (GPU)**
10^{-3}	345 ms	59 ms (5.8×)	6.2 ms (55.6×)
10^{-4}	2.7 s	147 ms (18.4×)	6.2 ms (431×)
10^{-5}	21 s	362 ms (58.0×)	6.2 ms (3,337×)

Table 6.2: 3D MPI reconstruction times as function of the desired acuracy

the CGNR algorithm after 20 iterations. In this case the speedup of the CGNR reconstruction would be about 13×. If all 200 iterations are taken into account, the parallel CGNR algorithm is about 30× faster than the serial Kaczmarz. The parallel steepest descent showed nearest the same execution times as the parallel CGNR did, but unfortunately with much poorer convergence.

The parallel SVD-based reconstruction took on average 6.2 ms to be executed, nearly 30× faster than one single iteration of the serial Kaczmarz algorithm, and resulted in significantly better metrics, with relation to both the input signal and the ground truth. Since the SVD is a direct solver, a proper way of comparing its performance to the ones of iterative solvers is by means of taking the times to reach a certain error level. Having this in mind, fitting the linear intervals of the top left plot of figure 6.8 for the serial Kaczmarz algorithm and the parallel CGNR (the blue and green dashed lines, respectively) reveals that, for 10^{-3} of desired RMSE, the serial Kaczmarz algorithm would take 345 ms while the parallel CGNR would take 59 ms, as speedup of 5.8×. The parallel SVD would therefore run 55.6× faster than the serial Kaczmarz. For a RMSE of 10^{-4}, the parallel CGNR would run 18.4× faster and the parallel SVD would run 431× faster, and for a RMSE of 10^{-5}, the speedups of the parallel CGNR and SVD reconstructions would be 58× and 3,337×, respectively. Table 6.2 shows the executions times for reaching the desired accuracies.

Even though the constrained Kaczmarz algorithm is less efficient than the parallel CGNR and SVD, its improved accuracy certainly would justify its use. A shown in 6.4.1, it is possible to improve the accuracy of the CGNR and SVD reconstructions by adding a single iteration of the constrained Kaczmarz algorithm at the end of the execution. The plots in figure 6.9 show the improvements in the RMSE and SSIM values with relation to the ground truth resulting from the addition of one constrained Kaczmarz iteration after the SVD reconstruction, as well as the increase in the reconstruction time. In this case, the average execution time would increase to 298 ms, meaning that this approach would still lead to speedups of 1.2×, 9.1× and 70× with relation to the Kaczmarz reconstruction, when the desired RMSE levels were 10^{-3}, 10^{-4} and 10^{-5}, respectively.

Finally, the parallel calculation of the calibration matrix revealed, for the simulated 20x20x20, an average execution time of 472 ms against 7.6 s taken by the serial

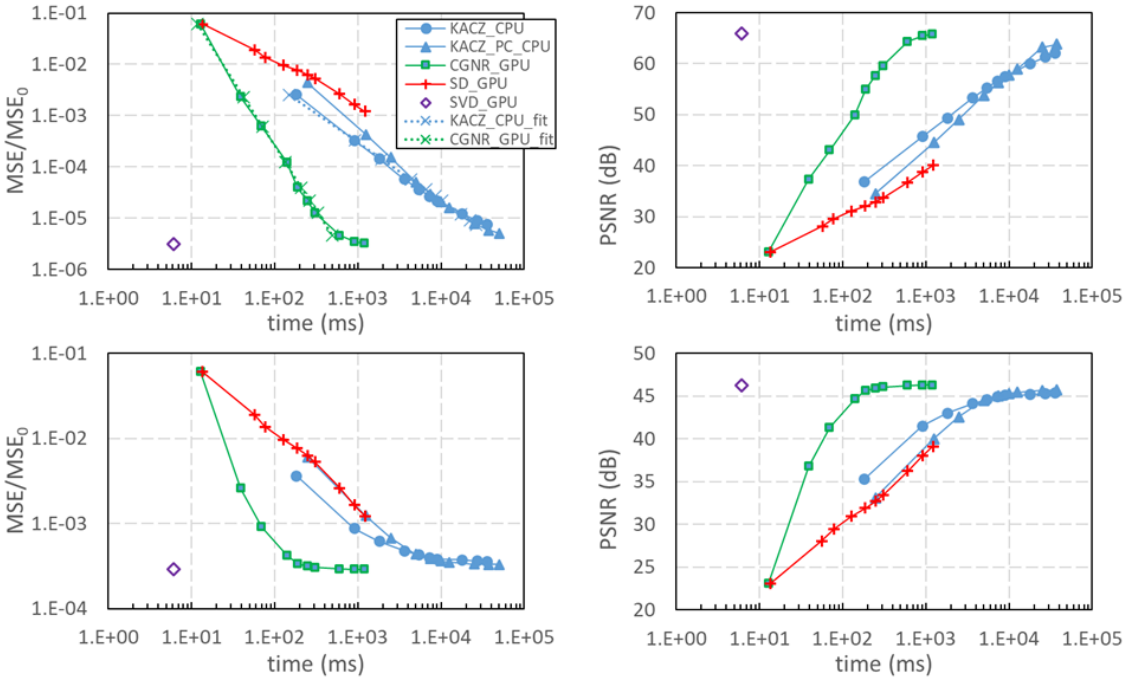

Figure 6.8: Convergence of the 3D MPI reconstruction algorithms in signal space as function of the execution time. The plots show the RMSE values (left) and PSNR values (right) of the reconstructions with relation to the measured signals as functions of time, for 5 μV (0.1% or -60 dB, top) and 50 μV (1% or -40 dB, bottom) of noise. The points correspond to values after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations. The blue and green dashed lines in the top left plot are the fits of the relative MSEs of the Kaczmarz algorithm and CGNR, respectively, for performance analysis. Adapted from [116].

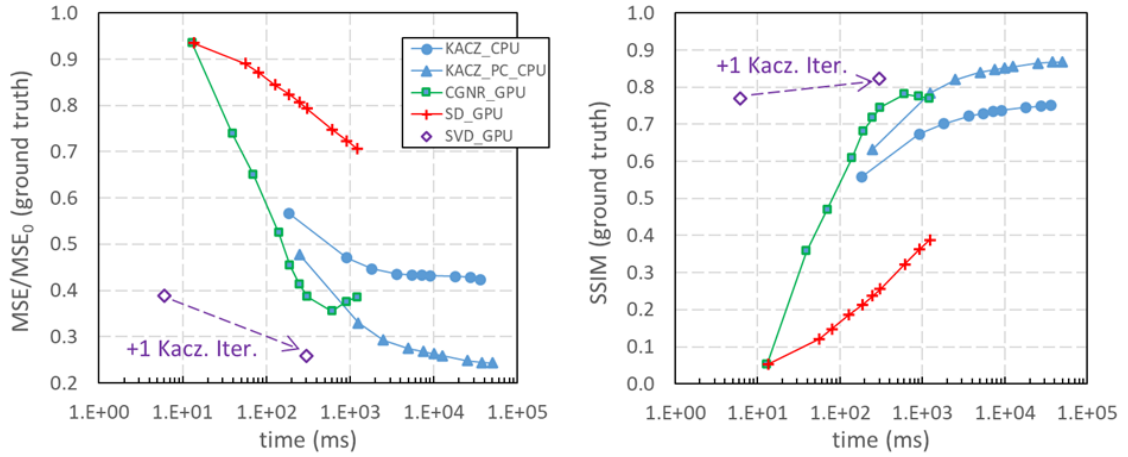

Figure 6.9: Convergence of the 3D MPI reconstruction algorithms in image space as function of the execution time. The plots show the RMSE values (left) and PSNR values (right) of the reconstructions with relation to the ground truth for the reconstructions of the signal with 1% of noise. The points correspond to values after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations. Adapted from [116].

implementation, a speedup of about 16×. This execution time is comparable to the one of the parallel CGNR reconstruction, and suggests the possibility of calculating the calibration matrix (or parts of it) in real time, without the need of storing it in the system memory.

Scalability and speedups of the parallel algorithms

The plot in figure 6.10 illustrates the scalability of the parallel implementation of the reconstruction algorithms, as well as the parallel implementation of the calculation of the calibration matrix, by plotting their execution times as function of the problem size (i.e., the size of the calibration matrix), given by the number of measured samples times the number of voxels of the reconstructed image. For the iterative algorithms, the times were the averages of the execution times after 200 iterations. For the parallel SVD reconstruction and the calculation of the calibration matrix, the times were averages of the execution times after a single execution. Given the poor convergence of the parallel steepest descent algorithm, its execution times would be meaningless and were not included in this figure. Since the parallel SVD decomposition occupies a significant amount of memory space – it required 5.4 GB of space for decomposing the calibration matrix for a 20x20x20 reconstruction and a sampling rate of 1.25 MHz, for example – for bigger problem sizes the GPU would run out of memory and then the parallel SVD evaluation was limited to a smaller subset of problem sizes. It is possible to see clearly how the parallel CGNR and SVD reconstructions consistently outperformed the serial Kaczmarz algorithm, regardless of the problem size, and the same can be said about the parallel calculation of the calibration matrix with relation to its serial counterpart.

It is possible to notice in the figure that while the serial algorithms scale perfectly linearly with the problem size, there is a certain non-linearity in how the parallel algorithms scale with the program size, but still close enough to be assumed as linear. For the parallel calculation of the calibration matrix, it can be easily correlated to the algorithm structure. The orange X's in the plot can be separated in groups of five points, each group corresponding to a different sampling rate, or number of rows of the calibration matrix. Within each group, the points correspond to different image resolutions, or number of columns of the calibration matrix. The parallel algorithm for the calculation of the calibration matrix, as described in 5.4.2 is such that each kernel computes one row of it. Therefore, for a single sampling rate (or number of rows), the parallel algorithm scales linearly with the image resolution because the elements within the row are calculated sequentially by its corresponding thread. Additionally, increasing the sampling rate also increases the degree of parallelism (number of threads), and therefore the execution speedups. That is why some points in figure 6.10 seem to have better performance than others with similar problem sizes

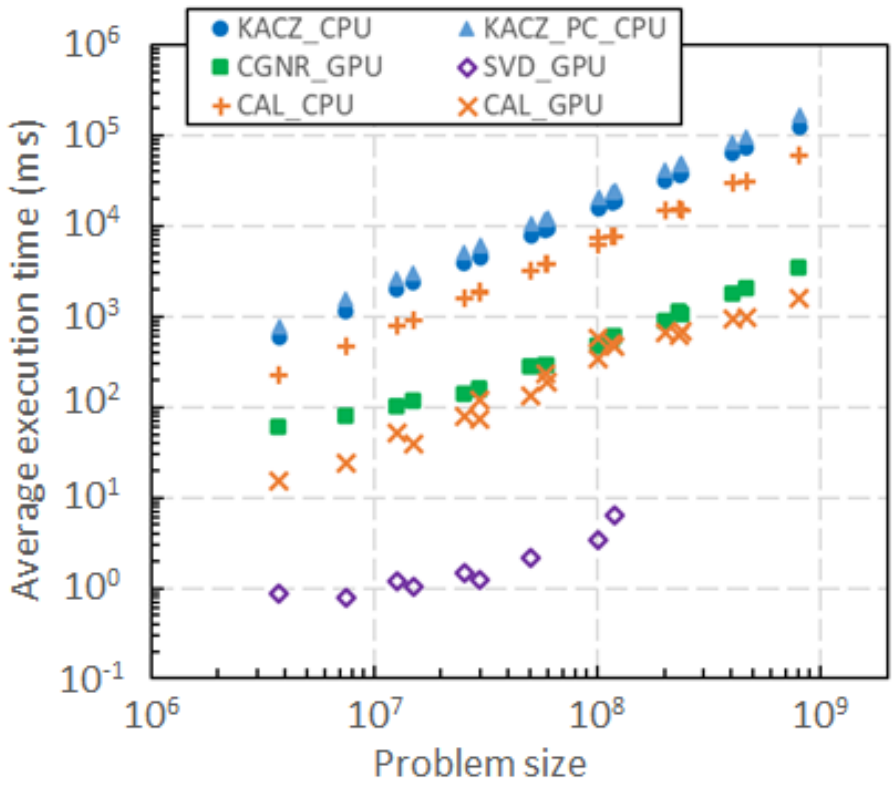

Figure 6.10: Scalability of the parallel implementations expressed as the execution times of the algorithms implemented in this work as function of the problem size. Reproduced from [116].

(see the clusters at $3 \cdot 10^7$ and 10^8, for example). Since the parallel CGNR and SVD reconstructions were executed using the CUBLAS function calls, there is no control on the workload distribution in the GPU, and identifying a pattern is not as straightforward, but it is still possible to notice points with approximate problem sizes and different performances. Table 6.3 shows the combinations of sampling rates and image resolutions that produced the problem sizes studied in this work, and table 6.4 summarizes the speedups of the parallel algorithms. It is important to note that the table shows speedups for the prallel SVD reconstruction that are larger than the number of GPU cores (the NVIDIA Quadro RTX 4000 has $2,304$ CUDA cores), and it is only possible because the execution times of the parallel SVD reconstruction are being compared to the the ones of the serial Kaczmarz algorithm. Since different algorithms are being compared, speedups larger than the number of cores are acceptable.

It has been shown in the literature that previously storing the pseudoinverse of the calibration matrix in the memory resulted in smaller reconstruction times [63]. The results of that work, however, were limited to problem sizes between 10^5 and 10^6. The present work extended this range by two orders of magnitude, and included parallel algorithms executed in the GPU using CUDA to deliver reconstructions

		Sampling rate			
		312.5 kHz	625 kHz	1.25 MHz	2.5 MHz
Resolution	10x10x10	3.8×10^6	7.5×10^6	1.5×10^7	3.0×10^7
	15x15x15	1.3×10^7	2.5×10^7	5.1×10^7	1.0×10^8
	20x20x20	3.0×10^7	6.0×10^7	1.2×10^8	2.4×10^8
	25x25x25	5.9×10^7	1.2×10^8	2.3×10^8	4.7×10^8
	30x30x30	1.0×10^8	2.0×10^8	4.1×10^8	8.1×10^8

Table 6.3: Problem size matrix. The matrix shows the problem sizes (number of elements of the calibration matrix) as function of the sampling rate and the image resolution.

at rates between 156 and $1,256$ frames per second. This approach is very useful in situations in which the user needs images a high frame rates and resolution is not critical, as in search scans, for example. Then, when higher resolutions are required for revealing details of the sample under investigation, the parallel CGNR can be used, with reconstructions times about 30× faster than the serial Kaczmarz algorithm, on average.

The overall results show that it is possible to calculate the calibration matrix and run a reconstruction using the CGNR algorithm in parallel in the GPU in much less time than the time spent to run the reconstruction using the serial Kaczmarz algorithm with the calibration matrix previously stored in the memory. This is a significant finding, and one implication of it is that it would no longer be necessary to previously store the calibration matrix, allowing for reconstructions with increased sampling rates and/or image resolutions with no memory space limitations. As an example, taking the simulation and reconstruction parameters adopted for the analysis in 6.4.1 (1.25 MHz of sampling rate and 20x20x20 of reconstruction resolution), the problem size would be 1.2×10^8. For a 0.1% of noise level and a minimum reconstruction RMSE with relation to the input signal of 10^{-4}, the serial Kaczmarz algorithm would need 15 iterations and take about 2.7 s to be executed. The parallel CGNR would need 22 iterations and take about 140 ms, being 19× faster then Kaczmarz. The calculation of the calibration matrix would take 470 ms, so the total time spent to calculate the matrix and reconstruct with CGNR would

Algorithm	**Minimum**	**Maximum**	**Average**
Calibration matrix[a]	12.7×	37.2×	21.8×
CGNR[b]	9.6×	36.8×	29.2×
SVD[b]	883×	$6,101\times$	$3,482\times$
Calibration matrix + CGNR[b]	7.7×	25.1×	18.1×

[a] Speedups relative to the serial computation of the calibration matrix

[b] Speedups relative to the serial Kaczmarz algorithm

Table 6.4: Speedups of the parallel algorithms

be 610 ms, 4.4× faster than Kaczmarz. Finally, if one wishes to add a single constrained Kaczmarz iteration, the total execution time would be 810 ms, 3.3× faster than the serial Kaczmarz algorithm and showing improved accuracy without the need of storing the calibration matrix in the memory.

One additional application of a real-time calculation of the calibration matrix would be being be possible to change the scanning and reconstruction parameters during execution time. This would allow the user, for example, to start a scan with a broader field of view, and change the drive field amplitudes to focus in some point of interest, i.e., zooming in to reveal more details of the sample. It could also be possible to partially compute the calibration matrix to reconstruct only parts of the field of view, perhaps combining it with CUDA streams [106] to allow for the concurrent computation of the calibration matrix and the image reconstruction for even better performance. This can be applied to any parameters, including the sampling rate and the resolution of the images to be reconstructed. Another great potential application of this feature is the fast computation of the temperature-dependent calibration matrices using the new method for temperature reconstruction employing the harmonic PSFs method[41]. The potential applications of the real-time calculation of the MPI calibration matrix are countless, and the implementation presented in this work is a proof of concept for further development in the future.

6.4.2 Real measurement data

The final stage of the present study was to demonstrate that the parallel reconstruction algorithms developed in this work are indeed suitable for real life applications. This was accomplished by running the reconstruction algorithms using the data provided by the `OpenMPIData` dataset [110].

As detailed in 5.4.3, the problem size of these reconstructions is 5.18×10^8, when accounting for the filtering of all frequency components below 80 kHz, and the resulting calibration matrix was too large to allow running the SVD decomposition in the GPU. For this reason, the SVD reconstruction was performed using the data from a single coil only instead of all three coils. To allow for a proper comparison between the reconstruction algorithms, the Kaczmarz, CGNR and SVD reconstructions were executed using data from a single coil, and the Kaczmarz and CGNR reconstructions were also executed using the data from all the three coils.

Accuracy

Preliminary reconstruction images and data revealed that the Kaczmarz algorithm with positivity constraint showed very poor convergence in comparison to the unconstrained algorithms. Table 6.5 shows the RMSE and PSNR values of the re-

construction algorithms when applied to the shape phantom. The results for the Kaczmarz algorithm, with and without positivity constraint, and for the CGNR algorithm correspond to 100 iterations. The CGNR and SVD were also assessed after adding a single iteration of the constrained Kaczmarz algorithm. Clearly all the constrained reconstructions resulted in very poor metrics in comparison to the unconstrained ones. To explain this, it is interesting to revisit the work of Lu et al. in [84], in which the authors describe the impact of the baseline filtering (the removal of the frequency components correspondent to drive field frequency to remove the feedthrough, see 2.4) as a DC offset in the reconstructed MPI image. This means that part of the reconstructed images would have negative concentration values due to this offset. Therefore, applying a positivity constraint during the reconstruction of a filtered signal would actually force the algorithm to converge to the wrong solutions and for this reason, only unconstrained algorithms were analysed in this section. A possible solution is applying the method described in [43] to recover the information corresponding to the base frequency, and this should be investigated in future studies.

Figure 6.11 shows the RMSE and PSNR values with relation to the measured signals for the shape, resolution and concentration phantoms. The plots show that both parallel algorithms show better accuracy than the serial (and unconstrained) Kaczmarz algorithm in all three scenarios. The best metrics were obtained with the shape phantom, and this can be explained by the lowest noise level (-56.8 dB, see 5.4.3) of the signals obtained with this phantom among the three of them. The reconstructions with the concentration phantom obtained the worst results, and this also can be explained by the noise level of the signals measured with this phantom. The parallel CGNR algorithm converged to the optimum solution after only 50 iterations, regardless of the noise level of the signal, while the serial Kaczmarz algorithm failed to converge in most cases, despite the usual good convergence rate after the first iteration.

Figure 6.12 shows the cross sections of the 3D images reconstructed from the

Algorithm	**RMSE**	**PSNR**
Kaczmarz	1.1×10^{-3}	66.0 dB
Kaczmarz_PC	5.7×10^{0}	29.0 dB
CGNR	4.1×10^{-4}	70.5 dB
CGNR + 1× Kaczmarz_PC	1.1×10^{1}	26.0 dB
SVD	4.1×10^{-4}	70.5 dB
SVD + 1× Kaczmarz_PC	1.0×10^{1}	26.4 dB

Table 6.5: RMSEs and PSNR values of the reconstruction algorithms with (Kaczmarz_PC) and without positivity constrain applied to the `OpenMPIData` dataset signal corresponding to the shape phantom.

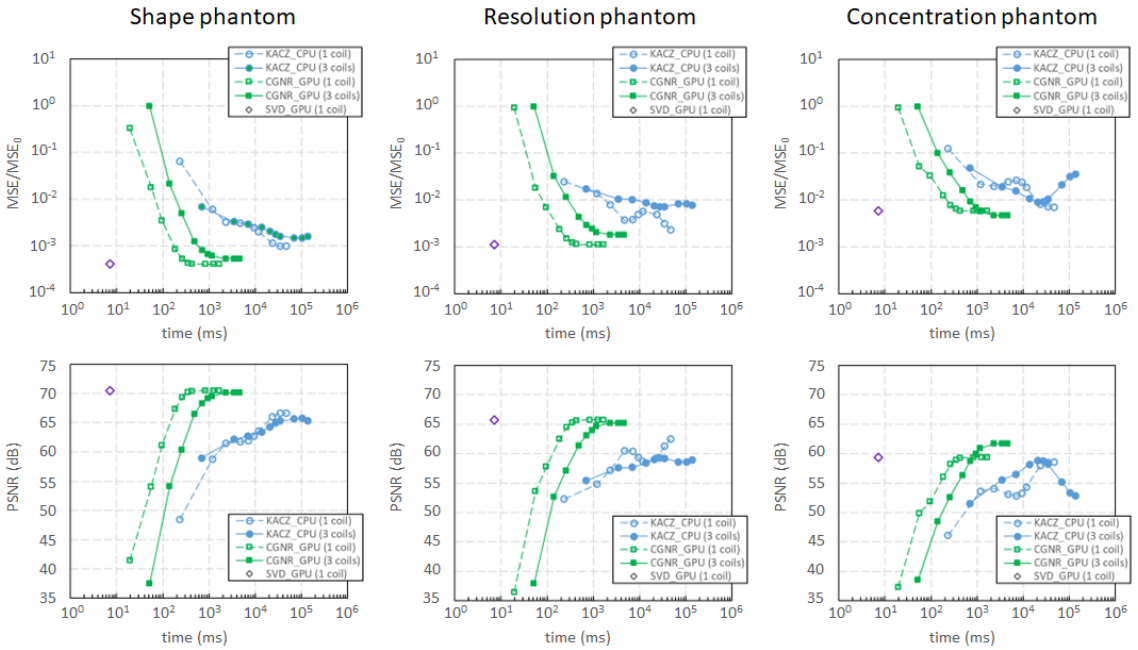

Figure 6.11: RMSE (top) and PSNR (bottom) values of the reconstruction of the `OpenMPIData` signals as function of time, for the shape phantom (left), resolution phantom (center) and concentration phantom (right). Points correspond to the values after 1, 5, 10, 20, 30, 40, 50, 100, 150 and 200 iterations. Reproduced from [116].

signals measured with the three phantoms, as well as the cross sections of the reconstructions presented on the `OpenMPIData` website [110]. It is possible to see that the images agree almost perfectly, and that small differences could be attributed probably to different regularization parameters used during the reconstruction, since the values adopted in [110] were not informed. Finally, figure 6.13 shows the 3D reconstructions of the shape, resolution and concentration phantoms in perspective. The comparison of these three images with the pictures of their corresponding phantoms, shown in figure 5.18, shows that the reconstruction of the shape phantom was the most successful one, followed by the reconstruction of the resolution phantom and then the one of the concentration phantom. For the latter, it is only possible to see the points with more particles, which is also true for the `OpenMPIData` reconstructions shown in figure 6.12 and can be attributed to the lower signal to noise ratio of the signals measured with this phantom.

Performance

Figure 6.11 also tells the performance of the reconstruction algorithms when applied to the `OpenMPIData` dataset. Both parallel CGNR and SVD outperform the serial Kaczmarz algorithm in all cases, regardless of the phantom being reconstructed or

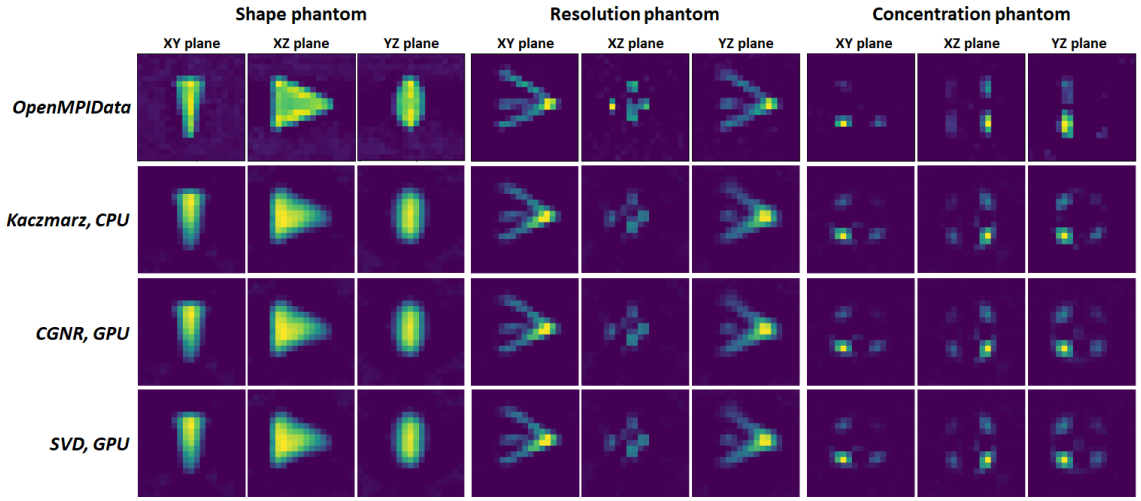

Figure 6.12: Cross sections of the reconstructions of the shape (left), resolution (center) and concentration (right) phantoms presented by the `OpenMPIData` dataset in comparison to the ones obtained using the algorithms studied in this work. Reproduced from [116].

whether data from one or three coils were used for the reconstruction. In all scenarios, the parallel CGNR managed to converge before the serial Kaczmarz algorithm completed its fifth iteration, a remarkable performance, in special after noticing that the Kaczmarz algorithm , in most cases, could not reach the solution even after 200 iterations.

In order to estimate the speedups of the parallel algorithms while reconstructing the `OpenMPIData` data, the same procedure adopted for the simulated results was employed, and the linear interval of RMSE plots of figure 6.11 were fitted to provide estimates of the execution times for a desired RMSE value. The results of this

		Desired RMSE		
Phantom	**Algorithm**	10^{-2}	5×10^{-3}	10^{-3}
Shape	CGNR (1 coil)	17.8	31.5	118
	SVD (1 coil)	210	502	3,820
	CGNR (3 coils)	0.9	4.6	*
Resolution	CGNR (1 coil)	17.6	54.9	-
	SVD (1 coil)	297	1,266	-
	CGNR (3 coils)	18.0	*	-
Concentration	CGNR (1 coil)	82.2	-	-
	SVD (1 coil)	2,879	-	-
	CGNR (3 coils)	22.8	-	-

* Serial Kaczmarz algorithm did not reach this RMSE level

\- None of the algorithms reached this RMSE level

Table 6.6: Speedups of the parallel algorithms applied to the `OpenMPIData` dataset, relative to the serial Kaczmarz algorithm.

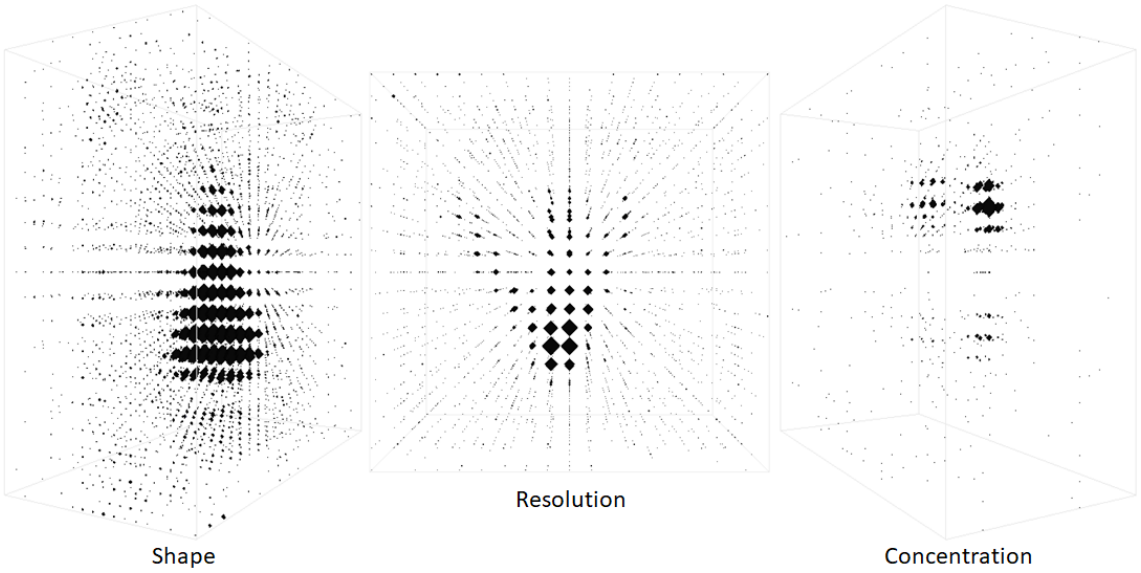

Figure 6.13: 3D reconstructions of the shape (left), resolution (center) and concentration (right) phantoms shown in perspective.

analysis is summarized in table 6.6. The results shown in the table demonstrate that the speedups of the parallel algorithms depend on the desired accuracy, in particular when the SNR is high and more accurate reconstructions are possible.

The overall results of the assessment of the performance of the parallel MPI reconstruction algorithms applied for simulated and real measurement data support the conclusion that the parallel algorithms are more efficient than the serial Kcazmarz algorithm even for small data sets, which means that they can be combined to other data reduction approaches, such as row selection or filtering, to obtain even better results.

In comparison to the previous parallel implementations, as presented in 6.2.2 and 6.3.2, the results presented in this section managed to significantly improve the performance of the parallel MPI reconstruction algorithms, obtaining speedups of up to three orders of magnitude. The methods presented in this work, and published in [70, 72, 116] are the first to actually accelerate single-frame MPI reconstructions by executing the algorithms in parallel in GPUs, allowing for real-time, high-resolution and high-accuracy MPI image reconstructions. It also important to keep in mind that the results presented in this work were obtained using an average GPU (a Nvidia Quadro RTX 4000), so much higher speedups are expected to be obtained with higher performance boards. The adoption of GPUs for MPI image reconstructions is a considerably low investment in comparison to the whole measurement setup, and a simple solution for obtaining faster and more accurate reconstructions.

Chapter 7

Conclusions

This thesis was focused on the development of GPU-accelerated parallel algorithms for speeding up MPI image reconstructions using the CUDA framework. The main results of this endeavour were the world's first parallel algorithms for MPI image reconstructions and the world's first parallel algorithm for the calculation of the calibration matrix. Even though the preliminary results of this project already motivated the development of other GPU-accelerated methods for high-performance MPI reconstructions (see [71]), the methods presented in this document remain, at the date of the publication of this document, as the only parallel strategies for speeding up the reconstruction of single MPI frames towards real-time imaging.

To reach these goals, a number of studies were performed and the improvements resulting from them contributed to the fulfilment of the main goals of this research. It started with the basic modeling of the MPI scanning and signal generation process, and the many factors that impact on the accuracy of technique – magnetic fields, particle magnetization response, particle dynamics, discrete sampling, filtering, temperature, among others. This allowed the development of a software that performed a rigorous simulation of MPI scanners working in one, two and three dimensions, and the signals generated by simulation suite allowed understanding better the capabilities and limitations of MPI systems. The results of this development were presented in [42]. The signals generated by the rigorous simulation suite allowed the validation of an analytical MPI signal generation model employed in the first MPI image reconstruction algorithms implemented at NIST, and contributed to the development of a new method for recovering the signal loss due to base frequency filtering [43], the investigation of the impact of the sampling frequency on the reconstruction accuracy [44] and world's first method for estimating uncertainties in MPI reconstructions [45].

In the sequence, the first implementations of parallel MPI image reconstruction algorithms executed in GPU were investigated. Among the algorithms studied were the Kaczmarz, Cimmino, block-Cimmino and CGNR. While the parallel Kaczmarz

showed very poor performance and the parallel Cimmino and block-Cimmino showed very poor convergence, the parallel CGNR algorithm showed excellent results and a performance 10× superior to the one of the serial Kaczmarz algorithm executed in the CPU. However, this study was still limited to 2D images, reconstructed from simulated signals without noise, and no regularization scheme was included in the reconstruction algorithms. Nevertheless, these results, published in [70], represent the first GPU-accelerated parallel MPI image reconstruction method proposed in the literature.

As part of the *Thermal MagIC* project developed at NIST [36], the research presented in this work is part of a greater effort to develop the techniques for 3D temperature mapping using MPI techniques. Therefore, the next study investigated the suitability of the parallel algorithms recently developed to speeding up 3D T-MPI reconstructions using the *multi-color* MPI method. It was found that, even though the parallel CGNR algorithm outperformed the serial Kaczmarz algorithm by a factor of 30, the *multi-color* method itself showed difficulties to reconstruct regions where the temperatures are different from the calibration temperature, revealing limited interpolation capabilities. The results of this study were published in [72].

The next step was to apply the techniques developed in the previous studies to signals of simulated 3D scans in the presence of noise, to better approximate real MPI measurement conditions. To allow for the reconstruction of noisy signals, Tikhonov regularization was integrated to the parallel CGNR algorithm. In addition to the new regularized algorithm, two new reconstruction approaches were investigated, employing the steepest descent algorithm and singular value decomposition (SVD), and a novel parallel algorithm for the calculation of the calibration matrix was implemented. The results of the tests with the parallel algorithms showed that both CGNR and SVD managed to reconstruct noisy signals with good accuracy, but the convergence speed of the steepest descent algorithm was significantly poorer in comparison to the other algorithms. In terms of performance, it was found that the CGNR algorithm can be up to 58× faster than the serial Kaczmarz algorithm, and that the parallel SVD reconstruction can be up to $3,337\times$ faster, when accounting for the required accuracy of the reconstructions. It was also shown that the better the required accuracy, the better is the speedup of the parallel reconstruction algorithms. The parallel calculation of the calibration matrix was accomplished through the development of a custom CUDA kernel in which each thread computed one row of the calibration matrix. The execution times were up to 37× faster than the ones of its serial counterpart running in the CPU, and it was shown that now it is possible to calculate the calibration matrix and run the image reconstruction in less time than the time taken to run the serial reconstruction with the Kaczmarz algorithm in the CPU. In this case, the speedups could be as high as 25×. Investigations of the

parallel algorithms running for signals obtained with different sampling rates and reconstructed with different resolutions revealed that the speedups scale well with the size of the problem.

The final stage of this investigation was the validation of the proposed methods using real measurement data. It was accomplished reconstructing images from the signals provided by the `OpenMPIData` dataset, widely used by MPI researchers for the validation and comparison of reconstruction algorithms. The results not only confirmed that both the parallel CGNR and SVD reconstructions outperform the serial Kaczmarz algorithm by one to three orders of magnitude in execution speed, but also revealed that the CGNR algorithm converged much more consistently than the Kaczmarz algorithm for all three measurement datasets with different noise levels. Additionally, the reconstruction of real, noisy and filtered signals also revealed that the adoption of a positivity constraint in the reconstruction algorithms led to incorrect solutions, and the reason for this is the fact that filtering the signal removes important information and is reflected in a DC offset in the reconstructed images. The results of this study were published in [116].

The main individual contributions of this work can be summarized as follows:

1. This work demonstrated that the performance of the parallel Kaczmarz algorithm for MPI reconstructions is worse than the performance of its serial implementation. The main reason for this is the structure of the algorithm, which relies in successive row projections in order to converge to the solution. The operations involved in these projections are mostly dot products, and the parallelization of such operations do not yield significant performance gains.

2. This work proposed the first parallel implementation of MPI image reconstructions using the CGNR algorithm using the cuBLAS library, and demonstrated that it resulted in execution times up to 40x faster when compared to the serial Kaczmarz algorithm executed in the CPU, in a per-iteration comparison. When accounting for the difference convergence rates between different algorithms, the different noise levels of the signals and the desired level of reconstruction accuracy, the parallel CGNR algorithm obtained execution times up to 58× faster for simulated data and up to 118× faster for real measured data. these results were possible because the majority of the operations that compose the CGNR algorithm are matrix multiplications, which significantly benefited from the parallel architecture of GPUs and the implementation of the cuBLAS library.

3. This work proposed the first parallel MPI image reconstruction approach based on singular value decomposition (SVD). In this approach, the MPI calibration matrix is decomposed in the GPU memory using the cuSolver library prior

to the execution. The matrices that compose the pseudoinverse of the calibration matrix are then used to obtain the reconstructed image by means of parallel matrix multiplication operations in the GPU using the cuBLAS library and custom kernels that allow obtaining the solution of the regularized least squares minimization problem. Using this approach, it was possible to accelerate the MPI image reconstruction by up to about $3,000\times$ for the reconstructions using simulated data and up to about $3,800\times$ for reconstructions using real measurement data, when accounting for the signal noise and the required accuracy level. Even though the SVD decomposition is not part of the reconstruction routine, its execution times are small enough to let it be part of the initialization of the main measurement program.

4. This work demonstrated that the parallel methods implemented in this study are suitable for being used with real, noisy and filtered measurement data, through the execution of the reconstruction algorithms using the signals provided by the `OpenMPIData` dataset for three different phantoms.

5. This work demonstrated that the speedups resulting from the use of the parallel algorithms proposed here are directly proportional to the desired accuracy level, in special when the noise level of the signal is low and greater accuracy is possible.

6. This work demonstrated that the proposed methods are suitable to reconstructing signals in both time and frequency domains.

7. This work presented for the first time a parallel algorithm for the calculation of the MPI calibration matrix. With this approach, it was possible to speed up this calculation by up to $37\times$, depending on the problem size. It was also demonstrated in this work, for the first time, that it is possible to calculate the calibration matrix and run the parallel CGNR reconstruction in much less time than running the serial Kaczmarz algorithm in the CPU. The real-time calculation of the calibration matrix is a significant achievement, which can potentially eliminate the need of storing the calibration matrix in the memory and allow changing the scanning and reconstruction parameters during execution.

8. This work demonstrated the limitations of the *multi-color* T-MPI method for resolving interpolated temperatures (temperatures in between the temperatures used for the calibration of the T-MPI system). The solution of the extended system of equations provides the estimated number of particles at each calibration temperature, thus the method is more well suited to be employed as a detector (i.e., a thermostat) rather than as a thermometer.

9. This work demonstrated that when the MPI signal is filtered to remove the drive field contribution at the fundamental excitation frequency, the solution of the MPI system of equations is shifted by a DC offset, and therefore applying a positivity constraint in the reconstruction algorithm leads to incorrect solutions.

The results show in this thesis demonstrates that the main goals of this research were fully achieved. GPU-accelerated parallel algorithms specially tailored for MPI image reconstructions can already be considered a reality now, and it is expected that the principles introduced in this work will be applied in future studies and practical MPI applications. Real-time MPI reconstructions, as proved possible in this work, can have a huge impact in biomedical imaging, the main goal of the technique.

The promising results shown in this thesis do not exhaust all the possibilities, and the algorithms can be further improved to deliver fast and more accurate reconstructions. The adoption of preconditioned calibration matrices can potentially improve the convergence of CGNR reconstructions, consequently improving the accuracy and/or reducing the number of iterations necessary for convergence, and shall be investigated. Additionally, different regularization schemes, including total least squares, L1 regularization and total variance, for example, can also provide good results and should also be studied. For the real-time calculation of the calibration matrix, the use of CUDA Streams for the concurrent computation of the matrix and the image reconstruction could yield significant performance gains and represents an interesting topic for further research. Still related to performance, the use of multiple GPUs, and the use of tensor cores for matrix multiplications could also be explored in the future.

The techniques developed in this work can potentially impact future studies on the harmonic-based temperature reconstructions based on the harmonic PSF principle introduced by NIST as part of the *Thermal MagIC* project. The parallel algorithms presented in this thesis have the potential of speeding up the calculation of temperature-dependent harmonic PSFs, allowing for high-performance and high-accuracy 3D temperature imaging.

Finally, during the development of this project, it was identified the opportunity of speeding up Monte Carlo simulations of MNPs, in which the Néel and Brownian relaxation mechanisms of the particles are activated to reveal details of the particles' dynamic responses to various applied fields. The adoption of parallel algorithms in GPU can substantially improve the performance of these simulations, and may result in better model-based calibration matrices for MPI reconstructions.

References

[1] GLEICH, B., WEIZENECKER, J. "Tomographic imaging using the nonlinear response of magnetic particles", *Nature*, v. 435, pp. 1214–1217, 2005.

[2] KNOPP, T., BUZUG, T. M. *Magnetic Particle Imaging*. Springer, 2012.

[3] WEIZENECKER, J., BORGET, J; GLEICH, B. "A simulation study on the resolution and sensitivity of magnetic particle imaging", *Physics in Medicine and Biology*, v. 52, pp. 6363–6374, 2007. doi: doi:10.1088/0031-9155/52/21/001. Available at: <stacks.iop.org/PMB/52/6363>.

[4] GLEICH, B., WEIZENECKER, J., BORGET, J. "Experimental results on fast 2D-encoded magnetic particle imaging", *Physics in Medicine and Biology*, v. 53, pp. N81–N84, 2008.

[5] WEIZENECKER, J., GLEICH, B., RAHMER, J., et al. "Three-dimensional real-time in vivo magnetic particle imaging", *Physics in Medicine and Biology*, v. 54, pp. L1–L10, 2009.

[6] KNOPP, T., BIEDERER, S., SATTEL, T., et al. "Singular Value Analysis for Magnetic Particle Imaging". In: *IEEE Nuclear Science Symposium*. IEEE, 2008.

[7] KNOPP, T., BIEDERER, S., SATTEL, T., et al. "Trajectory analysis for magnetic particle imaging", *Physics in Medicine and Biology*, v. 54, pp. 385–397, 2009.

[8] SATTEL, T. F., KNOPP, T., BIEDERER, S., et al. "Single-sided device for Magnetic particle imaging", *Journal of Physics D: Applied Physics*, v. 42, pp. 5pp, 2009. doi: doi:10.1088/0022-3727/42/2/022001.

[9] KNOPP, T., SATTEL, T. F., BIEDERER, S., et al. "Model-Based Reconstruction for Magnetic Particle Imaging", *IEEE Transactions on Medical Imaging*, v. 29, pp. 12–18, 2010. doi: 10.1109/TMI.2009.2021612.

[10] KNOPP, T., BIEDERER, S., SATTEL, T. F. "2D model-based reconstruction for magnetic particle imaging", *Medical Physics*, v. 37, n. 2, pp. 485–491, 2010. doi: 10.1118/1.3271258.

[11] WEIZENECKER, J., GLEICH, B., BORGERT, J. "Magnetic particle imaging using a field free line", *Journal of Physics D: Applied Physics*, v. 41, n. 105009, pp. 3pp, 2008. doi: 10.1088/0022-3727/41/10/105009.

[12] KNOPP, T., ERBE, M., SATTEL, T. F., et al. "Generation of a static magnetic field-free line using two Maxwell coil pairs", *Applied Physics letter*, v. 97, pp. 3pp, 2010.

[13] GOODWILL, P. W., SCOTT, G. C., STANG, P. P., et al. "Narrowband Magnetic Particle Imaging", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 28, n. 8, pp. 1231–1237, 2009.

[14] GOODWILL, P. W., CONOLLY, S. M. "The X-Space Formulation of the Magnetic Particle Imaging Process: 1-D Signal, Resolution, Bandwidth, SNR, SAR, and Magnetostimulation", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 29, n. 11, pp. 1851–1859, 2010. doi: 10.1109/TMI.2010.2052284.

[15] GOODWILL, P. W., CONOLLY, S. M. "Multidimensional X-Space Magnetic Particle Imaging", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 30, n. 9, pp. 1581–1590, 2011.

[16] GOODWILL, P. W., KONKLE, J. J., ZHENG, B., et al. "Projection X-Space Magnetic Particle Imaging", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 31, n. 5, pp. 1076–1085, 2012.

[17] KONKLE, J. J., GOODWILL, P. W., CARRASCO-ZEVALLOS, O. M., et al. "Projection Reconstruction Magnetic Particle Imaging", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 32, n. 2, pp. 338–347, 2013.

[18] VOGEL, P., RÜCKERT, M. A., KLAUER, P., et al. "Traveling Wave Magnetic Particle Imaging", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 33, n. 2, pp. 400–407, 2014.

[19] VOGEL, P., RÜCKERT, M. A., JAKOB, P. M., et al. "μMPI—Initial Experiments With an Ultrahigh Resolution MPI", *IEEE TRANSACTIONS ON MAGNETICS*, v. 51, n. 2, pp. 4pp, 2015.

[20] VOGEL, P., RÜCKERT, M. A., KEMP, S. J., et al. "Micro-Traveling Wave Magnetic Particle Imaging—Sub-Millimeter Resolution With Optimized

Tracer LS-008", *IEEE TRANSACTIONS ON MAGNETICS*, v. 55, n. 10, pp. 7pp, 2019.

[21] RAHMER, J., HALKOLA, A., GLEICH, B., et al. "First experimental evidence of the feasibility of multi-color magnetic particle imaging", *Physics in Medicine and Biology*, v. 60, pp. 1775–1791, 2015.

[22] LIEBL, M., STEINHOFF, U., WIEKHORST, F., et al. "Quantitative imaging of magnetic nanoparticles by magnetorelaxometry with multiple excitation coils", *Physics in Medicine and Biology*, v. 59, pp. 6607–6620, 2014.

[23] VOGEL, P., LOTHER, S., RÜCKERT, M. A., et al. "MRI Meets MPI: A Bimodal MPI-MRI Tomograph", *IEEE Transactions on Medical Imaging*, v. 33, n. 10, pp. 1954–1959, 2014. doi: 10.1109/TMI.2014.2327515.

[24] KNOPP, T., GDANIEC, N., MODDEL, M. "Magnetic particle imaging: from prooof of principle to preclinical applications", *Physics in Medicine and Biology*, v. 62, pp. R124–R178, 2017.

[25] NOTHNAGEL, N., RAHMER, J., GLEICH, B., et al. "Steering of Magnetic Devices With a Magnetic Particle Imaging System", *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, v. 63, n. 11, pp. 2286–2293, 2016.

[26] TAY, Z. W., SAVLIWALA, S., HENSLEY, D. W., et al. "Superferromagnetic Nanoparticles Enable Order-of-Magnitude Resolution & Sensitivity Gain in Magnetic Particle Imaging", *Small Methods*, v. 5, n. 11, pp. 2100796, 2021. doi: 10.1002/smtd.202100796.

[27] FUNG, K. L. B., COLSON, C., BRYAN, J., et al. "First Superferromagnetic Remanence Characterization and Scan Optimization for Super-Resolution Magnetic Particle Imaging", *Nano Letters*, v. 23, n. 5, pp. 1717–1725, 2023. doi: 10.1021/acs.nanolett.2c04404. Available at: <`https://doi.org/10.1021/acs.nanolett.2c04404`>. PMID: 36821385.

[28] COLSON, C., FUNG, K. B., BRYAN, J., et al. "Evidence that SPIO Chain Formation is Essential for High-Resolution MPI", *bioRxiv*, 2022. doi: 10.1101/2022.11.27.518061. Available at: <`https://www.biorxiv.org/content/early/2022/11/28/2022.11.27.518061`>.

[29] ZHONG, J., SCHILLING, M., LUDWIG, F. "Magnetic nanoparticle temperature imaging with a scanning magnetic particle spectrometer", *Measurement Science and Technology*, v. 29, pp. 9pp, 2018.

[30] WEAVER, J. B., RAUWERDINK, A. M., HANSEN, E. W. "Magnetic nanoparticle temperature estimation", *Medical Physics*, v. 36, n. 5, pp. 1822–1829, 2009.

[31] RAUWERDINK, A. M., HANSEN, E. W., WEAVER, J. B. "Nanoparticle temperature estimation in combined ac and dc magnetic fields", *Phys Med Biol.*, v. 54, n. 19, pp. 6pp, 2009. doi: 10.1088/0031-9155/54/19/L01.

[32] ZHONG, J., LIU, W., DU, Z., et al. "A noninvasive, remote and precise method for temperature and concentration estimation using magnetic nanoparticles", *Nanotechnology*, v. 29, pp. 9pp, 2012.

[33] ZHONG, J., LIU, W., KONG, L., et al. "A new approach for highly accurate, remote temperature probing using magnetic nanoparticles", *Scientific Reports*, v. 4, pp. 5pp, 2014.

[34] STEHNING, C., GLEICH, B., RAHMER, J. "Simultaneous magnetic particle imaging (MPI) and temperature mapping using multi-color MPI", *International Journal on Magnetic Particle Imaging*, v. 2, n. 2, pp. 6pp, 2016.

[35] WELLS, J., PAYSEN, H., KOSCH, O., et al. "Temperature dependence in magnetic particle imaging", *AIP Advances*, v. 8, n. 056703, pp. 7pp, 2018.

[36] NIST. "Thermal MagIC: New NIST Project to Build Nano-Thermometers Could Revolutionize Temperature Imaging". 2020. Available at: <`https://www.nist.gov/news-events/news/2020/10/thermal-magic-new-nist-project-build-nano-thermometers-could-revolutionize` Acessed in: 12/06/2020.

[37] BUI, T. Q., BIACCHI, A. J., WOODS, S. I. "Towards bimagnetic nanoparticle thermometry", *International Journal on Magnetic Particle Imaging*, v. 6, n. 2, pp. 3, 2020. doi: 10.18416/ijmpi.2020.2009010.

[38] BUI, T. Q., TEW, W. L., WOODS, S. I. "AC magnetometry with active stabilization and harmonic suppression for magnetic nanoparticle spectroscopy and thermometry", *Journal of Applied Physics*, v. 128, n. 224901, pp. 9, 2020. doi: 10.1063/5.0031451.

[39] BUI, T. Q., BIACCHI, A. J., DENNIS, C. L., et al. "Advanced characterization of magnetization dynamics in iron oxide magnetic nanoparticle tracers", *Applied Physics letters*, v. 120, n. 012407, 2022. doi: 10.1063/5.0077016.

[40] ABEL, F. M., CORREA, E. L., BIACCHI, A. J., et al. "Thermosensitivity through Exchange Coupling in Ferrimagnetic/Antiferromagnetic Nano-Objects for Magnetic-Based Thermometry", *ACS Applied Materials & Interfaces*, v. 15, n. 10, pp. 13439–13448, 2023. doi: 10.1021/acsami.2c19673.

[41] BUI, T. Q., HENN, M.-A., TEW, W. L., et al. "Harmonic dependence of thermal magnetic particle imaging", *Scientific Reports*, v. 13, n. 15762, 2023. doi: 10.1038/s41598-023-42620-1.

[42] QUELHAS, K. N., HENN, M.-A., BUI, T. Q., et al. "Flexible Software for Rigorous Simulations of Magnetic Particle Imaging Systems", *International Journal on Magnetic Particle Imaging*, v. 8, n. 1 Suppl 1, pp. 3, 2022. doi: 10.18416/IJMPI.2022.2203081.

[43] HENN, M.-A., QUELHAS, K. N., BUI, T. Q., et al. "Improving Model-Based MPI Image Reconstructions: Baseline Recovery, Receive Coil Sensitivity, Relaxation and Uncertainty Estimation", *International Journal on Magnetic Particle Imaging*, v. 8, n. 1, 2022. doi: 10.18416/IJMPI.2022.2208001.

[44] HENN, M.-A., QUELHAS, K. N., WOODS, S. I. "Investigating the Influence of Sampling Frequency on X-Space MPI Image Reconstructions", *International Journal on Magnetic Particle Imaging*, v. 9, n. 1 Suppl 1, 2023. doi: 10.18416/IJMPI.2023.2303042.

[45] HENN, M.-A., QUELHAS, K. N., WOODS, S. I. "Uncertainty estimation for 2D magnetic particle imaging", *International Journal on Magnetic Particle Imaging*, v. 8, n. 1 Suppl 1, pp. 3, 2022. doi: 10.18416/IJMPI.2022.2203082.

[46] PERREARD, I. M., REEVES, D. B., ZHANG, X., et al. "Temperature of the magnetic nanoparticle microenvironment: estimation from relaxation times", *Phys. Med. Biol*, v. 59, pp. 1109–1119, 2014. doi: 10.1088/0031-9155/59/5/1109.

[47] ZHOU, M., ZHONG, J., LIU, W., et al. "Study of Magnetic Nanoparticle Spectrum for Magnetic Nanothermometry", *IEEE TRANSACTIONS ON MAGNETICS*, v. 51, n. 9, pp. 6pp, 2015. doi: 10.1109/TMAG.2015.2434322.

[48] GARAIO, E., COLLANTES, J.-M., GARCIA, J. A., et al. "Harmonic phases of the nanoparticle magnetization: An intrinsic temperature probe", *AP-*

PLIED PHYSICS LETTERS, v. 107, n. 123103, pp. 5pp, 2015. doi: 10.1063/1.4931457.

[49] HE, L., LIU, W., XIE, Q., et al. "A fast and remote magnetonanothermometry for a liquid environment", *Measurement Science and Technology*, v. 27, n. 025901, pp. 6pp, 2016. doi: 10.1088/0957-0233/27/2/025901.

[50] SALAMON, J., DIECKHOFF, J., KAUL, M. G., et al. "Visualization of spatial and temporal temperature distributions with magnetic particle imaging for liver tumor ablation therapy", *Scientific Reports*, v. 10, n. 7480, pp. 11pp, 2020. doi: 10.1038/s41598-020-64280-1.

[51] KNOPP, T., VIERECK, T., BRINGOUT, G., et al. "MDF: Magnetic Particle Imaging Data Format", *ArXiv*, p. 16, 2020. Available at: <`https://arxiv.org/abs/1602.06072`>.

[52] KNOPP, T., SZWARGULSKI, P., GRIESE, F., et al. "MPIReco.jl: Julia Package for Image Reconstruction in MPI", *International Journal on Magnetic Particle Imaging*, v. 5, n. 1-2, pp. 9, 2019. doi: 10.18416/ijmpi.2019.1907001.

[53] KNOPP, T., SZWARGULSKI, P., GRIESE, F., et al. "OpenMPIData: An initiative for freely accessible magnetic particle imaging data", *Data in brief*, v. 28, n. 104971, pp. 7, 2020. doi: https://doi.org/10.1016/j.dib.2019.104971.

[54] MATTINGLY, E., MASON, E. E., HERB, K., et al. "OS-MPI: an open-source magnetic particle imaging project", *International Journal on Magnetic Particle Imaging*, v. 6, n. 2, pp. 3, 2020. doi: 10.18416/ijmpi.2020.2009059.

[55] MASON, E. E., COOLEY, C. Z., CAULEY, S. F., et al. "Design analysis of an MPI human functional brain scanner", *International Journal on Magnetic Particle Imaging IJMPI*, v. 3, n. 1, 2017. doi: 10.18416/ijmpi.2017.1703008.

[56] GRAESER, M., THIEBEN, F., SZWARGULSKI, P., et al. "Human-sized magnetic particle imaging for brain applications", *Nature Communications*, v. 10, n. 1936, 2019. doi: 10.1038/s41467-019-09704-x.

[57] VOGEL, P., RÜCKERT, M. A., GREINER, C., et al. "iMPI: portable human-sized magnetic particle imaging scanner for real-time endovascular interventions", *Scientific Reports*, v. 13, n. 10472, 2023. doi: 10.1038/s41598-023-37351-2.

[58] WELLS, J. "Recent advances and prospects for accelerated development in magnetic particle imaging", *International Journal on Magnetic Particle Imaging*, v. 6, n. 2, pp. 3, 2021. doi: 10.18416/ijmpi.2021.2106001.

[59] SCHULZ, V. "Recent impetus in magnetic particle imaging", *International Journal on Magnetic Particle Imaging*, v. 7, n. 1, pp. 2, 2022. doi: 10.18416/ijmpi.2022.2201001.

[60] LAMPE, J., BASSOY, C., RAHMER, J., et al. "Fast reconstruction in magnetic particle imaging", *Physics in Medicine and Biology*, v. 57, pp. 1113–1134, 2012. doi: 10.1088/0031-9155/57/4/1113.

[61] KNOPP, T., HOFMANN, M. "Online reconstruction of 3D magnetic particle imaging data", *Physics in Medicine and Biology*, v. 61, pp. N257–N267, 2016.

[62] VOGEL, P., KAMPF, T., RÜCKERT, M. A., et al. "Flexible and Dynamic Patch Reconstruction for Traveling Wave Magnetic Particle Imaging", *International Journal on Magnetic Particle Imaging*, v. 2, n. 2, 2016. doi: 10.18416/ijmpi.2016.1611001.

[63] VOGEL, P., HERZ, S., KAMPF, T., et al. "Low Latency Real-time Reconstruction for MPI Systems", *International Journal on Magnetic Particle Imaging*, v. 3, n. 2, pp. 8, 2017. doi: 10.18416/ijmpi.2017.1707002.

[64] EKLUND, A., DUFORT, P., FORSBERG, D., et al. "Medical image processing on the GPU – Past, present and future", *Medical Image Analysis*, v. 17, pp. 1073–1094, 2013. doi: http://dx.doi.org/10.1016/j.media.2013.05.008.

[65] STONE, S., HALDAR, J., TSAO, S., et al. "Accelerating advanced MRI reconstructions on GPUs", *J. Parallel Distrib. Comput.*, v. 68, pp. 1307–1318, 2008. doi: 10.1016/j.jpdc.2008.05.013.

[66] LI, J., SUN, J., SONG, Y., et al. "Accelerating MRI reconstruction via three-dimensional dual-dictionary learning using CUDA", *J Supercomput*, v. 71, pp. 2381–2396, 2015. doi: 10.1007/s11227-015-1386-z.

[67] A.FLORES, L., VIDAL, V., MAYO, P., et al. "Parallel CT image reconstruction based on GPUs", *Radiation Physics and Chemistry*, v. 95, pp. 247–250, 2014. doi: http://dx.doi.org/10.1016/j.radphyschem.2013.03.011.

[68] STORATH, M., BRANDT, C., HOFMANN, M., et al. "Edge Preserving and Noise Reducing Reconstruction for Magnetic Particle Imaging", *IEEE*

TRANSACTIONS ON MEDICAL IMAGING, v. 36, n. 1, pp. 74–85, 2017. doi: 10.1109/TMI.2016.2593954.

[69] CHEN, X., JIANG, Z., HAN, X., et al. "The Reconstruction of Magnetic Particle Imaging: Current Approaches Based on the System Matrix", *Diagnostics*, v. 11, n. 773, pp. 18, 2021. doi: 10.3390/diagnostics11050773.

[70] QUELHAS, K. N., HENN, M.-A., FARIAS, R. C., et al. "Parallel MPI image reconstructions in GPU using CUDA", *International Journal on Magnetic Particle Imaging*, v. 9, n. 1 Suppl 1, 2023. doi: https://doi.org/10.18416/ijmpi.2023.2303043.

[71] SHEN, Y., ZHANG, L., SHANG, Y., et al. "An Adaptive Multi-Frame Parallel Iterative Method for Accelerating Real-time Magnetic Particle Imaging Reconstruction", *Physics in Medicine and Biology*, v. 68, n. 245016, 2023. doi: 10.1088/1361-6560/ad078d.

[72] QUELHAS, K. N., HENN, M.-A., FARIAS, R., et al. "Parallel 3D Temperature Image Reconstruction Using Multi-Color Magnetic Particle Imaging (MPI)", *Temperature: Its Measurement and Control in Science and Industry*, 2024 (in press).

[73] RAHMER, J., WEIZENECKER, J., GLEICH, B., et al. "Signal encoding in magnetic particle imaging: properties of the system function", *BMC Medical Imaging*, v. 9, n. 4, pp. 21pp, 2009. doi: 10.1186/1471-2342-9-4.

[74] PANAGIOTOPOULOS, N., DUSCHKA, R. L., AHLBORG, M., et al. "Magnetic particle imaging: current developments and future directions and future directions", *International Journal of Nanomedicine*, v. 10, pp. 3097–3114, 2015.

[75] SCHOMBERG, H. "Magnetic Particle Imaging: Model and Reconstruction", *Int. Symp. on Biomedical Imaging: from Nano to Macro*, pp. pp 992–5, 2010. doi: 10.1109/ISBI.2010.5490155.

[76] MÄRZ, T., WEINMANN, A. "Model-based reconstruction for magnetic particle imaging in 2D and 3D", *arXiv preprint arXiv:1605.08095*, 2016.

[77] REEVES, D. B., WEAVER, J. B. "Simulations of magnetic nanoparticle Brownian motion", *Journal of Applied Physics*, v. 112, n. 124311, 2012. doi: 10.1063/1.4770322.

[78] CROFT, L. R., GOODWILL, P. W., CONOLLY, S. M. "Relaxation in X-Space Magnetic Particle Imaging", *IEEE Trans Med Imaging*, v. 31, n. 12, pp. 2335–2342, 2012. doi: doi:10.1109/TMI.2012.2217979.

[79] REEVES, D. B., WEAVER, J. B. "Approaches for modeling magnetic nanoparticle dynamics", *Critical Reviews in Biomedical Engineering*, v. 42, n. 1, pp. 85–93, 2014. doi: 10.1615/critrevbiomedeng.2014010845.

[80] GARCÍA-PALACIOS, J. L., LÁZARO, F. J. "Langevin-dynamics study of the dynamical properties of small magnetic particles", *Physical Review B*, v. 58, n. 14937, 1998. doi: 10.1103/PhysRevB.58.14937.

[81] KLUTH, T. "Mathematical models for magnetic particle imaging", *Inverse Problems*, v. 34, n. 8, 2018. doi: 10.1088/1361-6420/aac535.

[82] GRUETTNER, M., GRAESER, M., BIEDERER, S., et al. "1D-Image Reconstruction for Magnetic Particle Imaging Using a Hybrid System Function". In: *IEEE Nuclear Science Symposium Conference Record*. IEEE, 2011.

[83] LU, K., GOODWILL, P., ZHENG, B., et al. "The Impact of Filtering Direct-Feedthrough on the x-Space Theory of Magnetic Particle Imaging". In: *SPIE Medical Imaging*, 2011. doi: 10.1117/12.878446.

[84] LU, K., GOODWILL, P. W., SARITAS, E. U., et al. "Linearity and Shift Invariance for Quantitative Magnetic Particle Imaging", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, v. 32, n. 9, pp. 1565–1575, 2013.

[85] DHAVALIKAR, R., HENSLEY, D., MALDONADO-CAMARGO, L., et al. "Finite magnetic relaxation in x-space magnetic particle imaging: comparison of measurements and ferrohydrodynamic models", *Journal of Physics D: Applied Physics*, v. 49, n. 305002, pp. 7, 2016. doi: 10.1088/0022-3727/49/30/30500.

[86] GRÜTTNER, M., KNOPP, T., FRANKE, J., et al. "On the formulation of the image reconstruction problem in magnetic particle imaging", *Biomedizinische Technik/Biomedical Engineering*, v. 58, n. 6, pp. 583–591, 2013. doi: 10.1515/bmt-2012-0063.

[87] HATSUDA, T., TAKAGI, T., MATSUHISA, A., et al. "Basic Study of Image Reconstruction Method Using Neural Networks with Additional Learning for Magnetic Particle Imaging", *International Journal on Magnetic Particle Imaging*, v. 2, n. 2, pp. 5pp, 2015. doi: 10.18416/ijmpi.2016.1611002.

[88] DITTMER, S., KLUTH, T., HENRIKSEN, M. T. R., et al. "Deep Image Prior for 3D Magnetic Particle Imaging: A Quantitative Comparison of Regularization Techniques on Open MPI Dataset", *International Journal on Magnetic Particle Imaging*, v. 7, n. 1, pp. 19pp, 2021. doi: 10.18416/ijmpi.2021.2103001.

[89] MAASS, M., DROIGK, C., MERTINS, A. "MPI reconstruction using Bessel functions", *International Journal on Magnetic Particle Imaging*, v. 6, n. 2, pp. 3, 2020. doi: 10.18416/ijmpi.2020.2009017.

[90] SCHMIDT1, C., BRANDT, C. "Dynamic concentration reconstruction for magnetic particle imaging using splines", *International Journal on Magnetic Particle Imaging*, v. 6, n. 2, pp. 3, 2020. doi: 10.18416/ijmpi.2020.2009049.

[91] SAUNDERS, M. A. "Solution of sparse rectangular systems using LSQR and CRAIG", *BIT Numerical Mathematics*, v. 35, pp. 588–604, 1995. doi: 10.1007/BF01739829.

[92] KLUTH, T., JINB, B. "L1 data fitting for robust reconstruction in magnetic particle imaging: quantitative evaluation on Open MPI dataset", *International Journal on Magnetic Particle Imaging*, v. 6, n. 2, pp. 23, 2020. doi: 10.18416/ijmpi.2020.2012001.

[93] KLUTH, T., , MAASS, P. "Model uncertainty in magnetic particle imaging: Nonlinear problem formulation and model-based sparse reconstruction", *International Journal on Magnetic Particle Imaging*, v. 3, n. 2, pp. 10, 2017. doi: 10.18416/ijmpi.2017.170004.

[94] ASTER, R. C., BORCHERS, B., THURBER, C. H. *Parameter Estimation and Inverse Problems*. Elsevier, 2013.

[95] STROHMER, T., VERSHYNIN, R. "A randomized solver for linear systems with exponential convergence". In: *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, 2006. doi: 10.1007/11830924_45.

[96] CENSOR, Y., ZENIOS, S. S. *Parallel Optimization: Theory, Algorithms, and Applications*. New York, Oxford University Press, 1997.

[97] HESTENES, M. R., STIEFEL, E. "Methods of Conjugate Gradients for Solving Linear Systems", *Journal of Research of the National Bureau of Standards*, v. 49, n. 6, 1952.

[98] SHEWCHUK, J. R. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Technical report, Carnegie Mellon University, Pittsburgh PA, United States, 1994.

[99] KNOPP, T., RAHMER, J., SATTEL, T. F., et al. "Weighted iterative reconstruction for magnetic particle imaging", *PHYSICS IN MEDICINE AND BIOLOGY*, v. 55, pp. 1577–1589, 2010. doi: 10.1088/0031-9155/55/6/003. Available at: <stacks.iop.org/PMB/55/1577>.

[100] SOUMARE, H., BENKAHLA, A., GMATI, N. "Deep learning regularization techniques to genomics data", *Array*, v. 11, n. 100068, 2021. doi: 10.1016/j.array.2021.100068.

[101] MOORE, G. E. "Cramming more components onto integrated circuits", *Electronics*, v. 38, n. 8, pp. 4, 1965.

[102] MOORE, G. E. "Progress In Digital Integrated Electronics", *IEEE Text Speech*, 1975.

[103] SANDERS, J., KANDROT, E. *CUDA by example : an introduction to general-purpose GPU programming*. USA, Addison-Wesley, 2011. ISBN: 978-0-13-138768-3.

[104] KIRK, D. B., MEI HWU, W. *Programming Massively Parallel Processors: A Hands-on Approach*. USA, Elsevier, 2010. ISBN: 978-0-12-381472-2.

[105] "TOP500 the list, November 2023". 2023. Available at: <https://www.top500.org/lists/top500/2023/11/>.

[106] NVIDIA. "CUDA C++ Programming Guide Release 12.3". 2024. Available at: <https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf>.

[107] NVIDIA. "cuBLAS Release 12.2". 2023. Available at: <https://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf>.

[108] NVIDIA. "cuSOLVER Library". 2023. Available at: <https://docs.nvidia.com/cuda/pdf/CUSOLVER_Library.pdf>.

[109] KLUTH, T., JIN, B. "Enhanced reconstruction in magnetic particle imaging by whitening and randomized SVD approximation", *Physics in Medicine and Biology*, v. 64, n. 12, 2019. doi: 10.1088/1361-6560/ab1a4f.

[110] TECHNISCHE UNIVERSITÄT HAMBURG. "Open MPI Data". 2020. Available at: <https://magneticparticleimaging.github.io/OpenMPIData.jl/latest/index.html#Introduction-1>.

[111] DERBY, N., OLBERT, S. "Cylindrical magnets and ideal solenoids", *American Journal of Physics*, v. 78,, n. 229, pp. 229–235, 2009. doi: https://doi.org/10.1119/1.3256157.

[112] CAMACHO, J. M., SOSA, V. "Alternative method to calculate the magnetic field of permanent magnets with azimuthal symmetry", *Revista Mexicana de Física E*, v. 59, pp. 8–17, 2013.

[113] COMSOL. "COMSOL Multiphysics Reference Manual". 2019. Available at: <https://doc.comsol.com/5.5/doc/com.comsol.help.comsol/COMSOL_ReferenceManual.pdf>.

[114] WIRGIN, A. "The inverse crime". 2018. arXiv preprint math-ph/0401050.

[115] ELBLE, J. M., SAHINIDIS, N. V., VOUZIS, P. "GPU computing with Kaczmarz's and other iterative algorithms for linear systems", *Parallel Computing*, v. 36, pp. 215–231, 2009. doi: 10.1016/j.parco.2009.12.003.

[116] QUELHAS, K. N., HENN, M.-A., FARIAS, R., et al. "GPU-accelerated parallel image reconstruction strategies for magnetic particle imaging", *Physics in Medicine and Biology*, v. 69, n. 13, 2024. doi: 10.1088/1361-6560/ad5510.

[117] WANG, Z., BOVIK, A., SHEIKH, H., et al. "Image quality assessment: from error visibility to structural similarity", *IEEE Transactions on Image Processing*, v. 13, n. 4, pp. 600–612, 2004. doi: 10.1109/TIP.2003.819861.

[118] BUI, T. Q., HENN, M.-A., TEW, W. L., et al. "Harmonic dependence of thermal magnetic particle imaging", *Scientific Reports*, v. 13, n. 15762, 2023. doi: 10.1038/s41598-023-42620-1.