Université Catholique de Louvain

-

Centre d'étude de l'énergie nucléaire

Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM)

PhD Thesis

Simulation of a Radio-Frequency-Quadrupole with the Method of Moments (MoM)

Author:
Christopher Raucy

Thesis advisors:
Christophe Craeye
Jean-François Remacle

Jury members:
Dirk Vandeplassche
Danielle Vanhoenacker-Janvier
Francesco Andriulli
Willem Kleeven
David Bol

Contents

Symbols and Acronyms

ϵ_{ijk}	The tensor of permutation
$\cdot$	Tensor contraction or a scalar product on a Euclidean space.
$\overline{x}$	The over line means a vector related to the Euclidean geometric space.
$\times$	Cartesian product or a cross product
∇	The gradient operator
$\overline{E}$	Electric field
$\overline{H}$	Magnetic field
$\overline{J}_m$	Equivalent magnetic current
$\overline{J}_e$	Equivalent electric current
$\overline{J}$	Free current density
ρ_m	The equivalent magnetic charge density
ρ_e	The equivalent electric charge density
ρ	The free charge density
ϵ	The permittivity of the concern medium
μ	The permeability of the concern medium
$\equiv$	Definition
$ker(A)$	Null-space or kernel of an operator A

$$||x||_2 \equiv \sqrt{\sum_{i=1}^{N} x_i x_i^*} \qquad \text{with } x \in \mathbb{C}^N$$

$$[\overline{J}] \equiv T^{-1}(\overline{J}) \qquad \text{with } \overline{J} \in Vec(S)$$

SCK-CEN	Belgian Nuclear Research Centre
CST	Computer Simulation Technology
LINAC	Linear particle accelerator
MBF	Macro Basis Functions
MoM	Method of Moment
RFQ	Radio-Frequency Quadrupole
AC	Acceleration Cavity
ADS	Accelerator Driven System
ASM	Array Scanning Method

CRC	Centre de Ressources du Cyclotron
CW	Continuous Wave
EM	ElectroMagnetic field
EPL	Ecole Polytechnique de Louvain
HF	High Frequency
IAP	Instituts für Angewandte Physik (in Frankfort)
ICTEAM	Institute of Information and Communication Technologies, Electronics and Applied Mathematics
IMAP	Materials and process engineering
IMMC	Institute of Mechanics, Materials and Civil Engineering
IS	Ion Source.
KUL	Katholieke Universiteit Leuven
LEBT	Low Energy Beam Transport
SC	Space Charge
SKA	Square Kilometre Array
PEC	Perfect Electric Conductor
EFIE	Electric Field Integral Equation
MFIE	Magnetic Field Integral Equation
FMM	Fast Multipole Method
MLFMA	Multi-Level Fast Multipole Algorithm
GPU	Graphic Processor Unit
CPU	Central Processor Unit
RAM	Random Access Memory
SRAM	Static Random Access Memory
DRAM	Dynamic Random Access Memory
SDRAM	Synchronous Dynamic Random Access Memory
DDR SDRAM	Double Data Rate Synchronous Dynamic Random Access Memory
OS	Operating System
FNFMMS	Fast Near Field MoM Solver

Chapter 1

Introduction

The aim of this thesis is to attempt to reduce the computational time of present solvers for the simulation of accelerator cavities that constitute LINear ACcelerators (LINAC) while conserving the accuracy of a full wave solver. Cavity has to be understood in a large sense namely any RF resonators that constitutes a LINAC and not only closed resonators. Reducing the computational time may offer to possibility to perform numerical optimizations of these cavities. A Radio Frequency Quadrupole (RFQ) is an accelerating cavity placed at the very beginning of a LINAC. It is a very critical component of an accelerator because it affects the quality and stability of the beam in all the further sections of the accelerator. Unfortunately, the conditions of stability of the beam inside a RFQ are very sensitive to the geometrical parameters, to the input beam parameters and to the RF source parameters. Therefore, the design of a RFQ must be performed as carefully as possible. In general, it is validated with the help of several independent solvers. In this thesis, the RFQ will be the cavity of test for our solver. In particular, the reference RFQ design used in this thesis is the MYRRHA RFQ, designed at the University of Goethe (Frankfurt).

The Myrrha project [1], [2] consists of building a challenging Accelerator Driven System (ADS) [3], in particular to study the transmutation of nuclear waste. One of the biggest issue with nuclear plant is the long life of radioactive activity of the wastes [4],[5]. At the present time, the wastes are buried deep underground in clay. For instance the HADES [2] project aims at investigating the burying of wastes at 220 meters underground in clay. However, this solution is not without risks. Indeed, earth quakes and underground flooding might at any time spread radiocative isotopes in nature. An ADS is a sub-critical core stimulated by an external source of fast neutrons. Fast neutrons can be produced by projecting fast ions onto a target containing heavy atoms, which in turn ejects fast neutrons. Fast ions

can, for instance, be obtained with the help of a linear particle accelerator (LINAC) [6] fed by an Ion Source (IS).

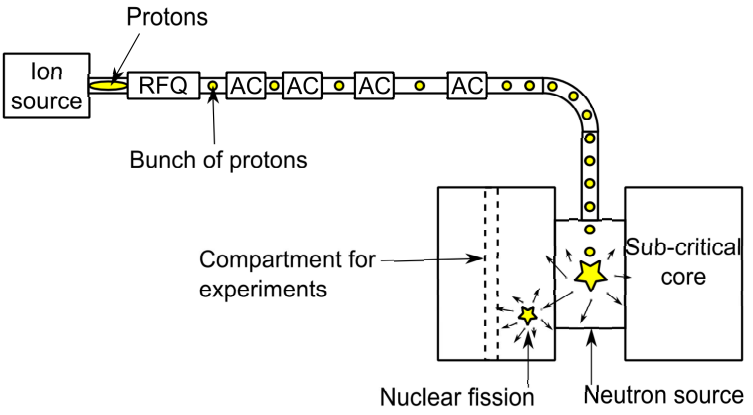


Figure 1.1: ADS system.

In the framework of the Myrrha project, the ions chosen are protons. Fig. 1.1 shows the three parts of an ADS system. The LINAC on the top of the figure is made of many different cavities. Most of the cavities are designed to accelerate the beam of protons. At the very beginning of the accelerator, one special cavity, known as the Radio-Frequency Quadrupole (RFQ) [6], performs the initial bunching (a bunch is set of particles gather together as a group) and acceleration while guaranteeing a strong focusing. The design of the Myrrha's LINAC [7] is challenging in terms of performances. Indeed, it shall deliver a Continuous Wave (CW) proton beam up to 4 mA with an energy of 600 MeV. At the beginning of the accelerator such as in the RFQ, focusing particles at low speed and high density is a challenging problem because the magnetic field produced by the particles is too low to compensate the repulsive electrostatic forces. The electrostatic repulsion is known as the the Space Charge(SC) [8] effects in the accelerator community. Presently, the optimisation of the RFQ accounts for the SC effects in an approximate manner only. Accurate simulations with complete coupling of the electromagnetic and dynamic aspects remains a challenge, owing to the complexity of the non-linear problem. The sub-critical core of MYRRHA is very sensitive to beam interruptions. If they are longer than three seconds, thermal shocks can appear inside the reactor [1]. These shocks affect the cladding materials of the reactor and might cause some operational issues. A fault tolerance scheme has been defined to avoid such failures. The first step in this scheme is to guarantee a high reliability of the components.

LINAC	Cyclotron
Large space requirement (few hundred meters long) but light	Compact but heavy
Expensive	Cheaper
Less efficient power conversion	More efficient power conversion
Modularity provides redundancy	No intrinsic redundancy
Upgradable in energy	Difficult to upgrade in energy
Straightforward beam extraction	Difficult extraction and related beam losses
Capable of high beam current (100 mA)	Modest beam current capability (5 mA)

Table 1.1: Comparison between LINAC and cyclotron capabilities

A Linear Accelerator (LINAC) [6], [9], [10], [11] consists of accelerating charged particles such as protons on a straight line path. Such an accelerator differs from a synchrotron, cyclotron or cyclic accelerators where particles follow a circular or elliptic trajectory. In a synchrotron and a cyclic accelerator, particles must perform many loops before achieving the requested energy. A continuous injection is not possible for these accelerator since the magnetic field in the focusing dipoles and quadrupoles is continuously adapted to the energy of the beam. Therefore, these accelerators are used for high energy applications and low current throughput. A cyclotron allows a continuous injection but suffers from the relativist effects at high energy. In short, the focusing in the transversal plane is lost at high energy. Similarly, LINACs are used for middle energy applications with high current throughput since particles can be continuously injected at the input of the accelerator. Tab. 1.1 enumerates several differences between cylotrons and LINACs.

A LINAC is made of several stages each of them with a specific task. The first few stages of the accelerator consist of an ion source, a low energy beam transport section (LEBT) and a Radio-Frequency-Quadrupole (RFQ) [6], [10]. The ion source produces charged particles that are injected in the LEBT at stable constant energy but with a wide direction velocity distribution. The goal of the LEBT is to transport the particles to the RFQ thanks to a high magnetic field that focuses the beam and to focus the beam so that it matches the RFQ input requirements. The RFQ has three

important tasks:

- Focusing the beam
- Bunching the beam
- Accelerating the beam

The focusing is done with the help of a quasi-electrostatic fields based on a Paul Trap [12]. The total transversal energy of the particles remains practically constant in a RFQ. The acceleration is achieved by curving the electrodes in the direction of propagation of the beam. The accelerator being based on a RF technology, the beam must be bunched (i.e grouped into packets). The RFQ bunches the beam in the first few sections of the RFQ before any significant acceleration. The following paragraphs detail each of these three steps.

One assumes all along this thesis that the electric field in the accelerator can be decomposed as the sum of a source field and a self field. The source field corresponds to the electric field produced by the RF source when the accelerator is unloaded (i.e without beam load). It is computed with a full wave solver. The self field is the electric field scattered by the particles i.e by the space charge. The self field might change the impedance seen by the RF source if the space charge is too strong. In turn, this could affect the source field. A RF regulation is required to guarantee the stability of the source field. The thesis is divided in two parts corresponding respectively to the computation of the source field and to the computation of the beam dynamics accounting for the space charge. The magnetic field can be neglected as shown further in the thesis.

In this paragraph, the full wave solver used in this thesis is introduced. More details will be provided in the chapter devoted to the analysis of this method. The Method of Moments (MoM) [13], [14] is a Boundary Element Method (BEM) that can be used to simulate electromagnetic fields scattered by objects placed in linear media. The strength of the MoM lies partially in the fact that the solver is based on the Maxwell's equations solution for a Dirac current distribution [15], [16]. Because of the linearity of the Maxwell's equations, the linearity of the constitutive equations and with the help of the equivalence theorem, the fields can be obtained by the convolution of the latter solution with the current distribution in the media. The equivalence theorem allows one to obtain equivalent currents at the boundaries of a given medium such that the electromagnetic fields verify the Maxwell's equations and the constitutive equations of the given medium everywhere in space i.e for all $\mathbb{R}^3$. In other words, with the equivalence theorem, one only needs to solve the Maxwell's equations as if there was only

a single medium. The equivalent currents themselves depend on the local electromagnetic fields at the boundaries. The convolution is called the scattering operator and it is a integro-differential equation. The MYRRHA RFQ can be assumed as a perfect electric conductor (PEC) in free-space. Since one deals with PEC, the currents can be approximated as surface currents. The equivalent currents are directly linked to the real current on a PEC. In particular, the magnetic current is equal to zero while the electric current matches the real current. Therefore, structure involving only PEC implies a convolution that contains only surface integrals. The scattering operator involves unkown currents. The MoM consists in imposing the boundary conditions between different media using the scattering operator proper to each medium in order to invert the scattering operators and obtain the current distributions in each medium. The current distributions are approximated by linear combinations of primitive current distributions called the basis functions. The boundary conditions are imposed by computing a local average field using some functions placed on the surface of the scatterer and called testing function. In the latter case, the testing procedure is called a Galerkin test. Because of the linearity of the scattering operators, one ends up with a linear system of equations to be solved. There are several MoM solvers which differ from one another by the scattering operator used. There exist two scattering operators, one for the electric field called in this thesis the Electric Field Scattering Operator (EFSO) and one for the magnetic field called in this thesis the Magnetic Field Scattering Operator (MFSO). They depend both of magnetic and electric currents. The parallel component of the electric field must vanish on the PEC. This implies that the magnetic current is equal to zero. Therefore, a natural choice is to use the EFSO and enforcing the said boundary conditions. The resulting integral equation in this case is so-called the Electric Field Integral Equation (EFIE) [13], [16] and is the most widely used integral equation in the antenna community. One could also use the MFSO and impose the boundary conditions on the magnetic field. In this case the integral equation is the so-called Magnetif Field Integral Equation (MFIE) [13], [16]. The EFIE and MFIE offer different numerical stability. Sometimes, both might by used to solve PEC problems for numerical stability reasons (Xavier thesis REF). One of the advantages of the MoM in comparison to Volume Element Methods (VEM) is that the number of unknowns is relatively small. Indeed, for VEM one needs to mesh in 3D the volume where the electromagnetic fields have to be calculated. These methods are practically only used for closed structures since the boundary conditions for open structures are imposed at infinity. Therefore, another advantage of the MoM is the ability of the solver to solve free-space problems and in particular antenna problems. If the number of unkowns is relatively small, the MoM produced however a

dense system of equations. Hence, fast solvers for sparse matrices produced by a VEM solver cannot be used to solve a MoM problem. Although an RFQ is a cavity, the method applied in this thesis is the EFIE used for scattering problems. The RFQ is simulated without its enclosure shield making it a scatterer.

The main activity of the antenna group at UCL concerns the development of fast solvers based on the MoM and in particular the EFIE. The laboratory is specialized in antenna coupling [17]. Most of the MoM algorithms can be separated in two categories. The first one consists in computing the interactions between the basis functions and the testing functions. In general, the set of testing functions are identical to the set of basis functions. One calls this step the filling up of the impedance matrix and its complexity is $O(N^2)$ with N being the number of basis functions. The second step consists of solving the MoM system of equations. If one computes a full matrix inversion, the complexity of this step is $O(N^3)$. The laboratory has been using several methods to reduce the complexity of these two steps. The simulation of a RFQ with the MoM involves two big challenges. First, due to the small structure size relative to the wavelength the EFIE undergoes the so-called Low-Frequency Breakdown (LBF) [18], [19],[20], [21], [22] which leads to an ill system of equations i.e. a system of equations with a bad condition number. A bad condition number implies in practice a very poor convergence of iterative solvers and accuracy problem either in the image space (currents) of in the source space (average field). Second, the basis functions being small in comparison to the wavelength, typical fast methods such as the Multi-Level Fast Multipole Algorithm (MLFMA) [23] cannot be used to reduce the complexity of the system solving.

The beam dynamics are calculated separately from the MoM solver. Of course, the source field obtained with the MoM is required by the beam dynamics solver. Other solvers based on different methods could be used to simulate the source field such as for instance a solver proposed by the Computer Simulation Technology (CST). The beam dynamics solver consists in integrating the equation of motion and evaluate at each step of the integration the interaction between particles. Different approaches can be used to simulate the beam motion. For instance, a continuous distribution of the beam could be assumed and plasma solvers based on the well known Vlasov equations [24] could be used. Another well known approach consists in assuming that particles can be grouped together into several macro-particles, each macro-particle having a charge and a mass respectively equal to the sum of the charges and masses of all the particles constituting it. For each macro-particle the equations of motion must be solved. This is the approach that will be used in this thesis. The challenge of the solver lies in its ability

to calculate the macro-particles interactions with a reasonable complexity. Several physical hypotheses will be adopted in order to obtain a reasonable complexity. In order to further improve the computation time of the solver a Graphic Processor Unit (GPU) will be used. Implementing a solver on a GPU is itself a complicated and tricky work. Indeed, the implementation must account for the hardware design of the GPU in order to be efficient. There are several levels of memories in a GPU and the transfers of data between them must be well controlled in order to come reasonably close to the theoretical performances of the GPU.

This thesis is subdivided in different chapters. First, a general introduction about what is a RFQ is presented in chapter 2. Then, the MoM solver challenges are investigated in different chapters. A chapter reviewing the MoM and giving all the prerequisite notations and results is presented in chapter 3. The MoM challenges are then investigated in the chapter 4. A short review about fast solvers based on iterative solvers is first presented. Then, the Low-Frequency Breakdown (LFB) and the matrix-vector product evaluation issue are both studied and solved in separate sections. The beam dynamics solver is addressed in different chapters. First, three chapters reviewing the physics of particle dynamics, explaining the functioning of a GPU and recalling main results about ordinary differential equations are presented. The beam dynamics solver and all the numerical results are presented in chapter 6.

Chapter 2

Radio-Frequency-Quadrupole

2.1 The principle of the RFQ

The principle of the RFQ [25] is based on the well known electrostatic quadrupole. It is made of four rods each of them set at a given potential which conformed to the quadrupole symmetry. In this section the electrostatic quadrupole is recalled and the modifications to obtain an RFQ from it are explained. Before getting into the heart of the subject, let us define a few useful variables. The direction of propagation of the beam is in the direction $-\hat{x}_3$. The perpendicular plane to the direction $\hat{x}_3$ is spanned by two perpendicular vectors $\hat{x}_1$ and $\hat{x}_2$. These vectors are represented by respectively the axes Z, X, Y.

In order to focus the beam, one needs an electric field which tends to keep the particles around the ideal path. Intuitively, this could be achieved with a linear force such as in an harmonic oscillator. In the following paragraphs one will show how to obtain such a field. By design, one considers the case of a quasi-static field in vacuum (the charge of the beam itself being neglected). In this case, the fields must satisfy the static Maxwell's equations in vacuum

$$\nabla \times \overline{E} = 0 \tag{2.1}$$
$$\nabla \times \overline{H} = 0 \tag{2.2}$$
$$\nabla \cdot \overline{E} = 0 \tag{2.3}$$
$$\nabla \cdot \overline{H} = 0 \tag{2.4}$$

where $\overline{E}$ is the electric field and $\overline{H}$ is the magnetic field.

There is a more compact way to rewrite Maxwell's equations. Since $\nabla \times \overline{E} = 0$ the electric field is equal to the gradient of a potential $\overline{E} = -\nabla\phi$ (this is a consequence of the Helmholtz decomposition theorem or the so-called Poincaré lemma [26], [27]). Then applying the divergence, one gets the well known equation of Poisson in vacuum $\nabla^2\phi = 0$. By symmetry, one gets the same expression for the magnetic field.

As proposed previously, let us try to inject a linear expression of the electric field in the static Maxwell's equations.

$$E = k_1 x_1 \hat{x}_1 + k_2 x_2 \hat{x}_2 \tag{2.5}$$

with $k_1, k_2 \in \mathbb{R}$ two constants.
Let us first inject the field in the curl

$$\nabla \times \overline{E} = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} \frac{\partial E_i}{\partial x_j}\epsilon_{jik}\hat{x}_k \tag{2.6}$$

$$\nabla \times \overline{E} = \sum_{k=1}^{3}\left(\frac{\partial E_1}{\partial x_1}\epsilon_{11k} + \frac{\partial E_2}{\partial x_2}\epsilon_{22k}\right)\hat{x}_k = 0 \tag{2.7}$$

Where ϵ_{ijk} is the Levi-Civita symbol. The field trivially satisfies the curl. Then let us inject the electric field in the divergence

$$\begin{aligned} \nabla \cdot \overline{E} &= \sum_{i=1}^{3} \frac{\partial E_i}{\partial x_i} \\ \nabla \cdot \overline{E} &= \frac{\partial E_1}{\partial x_1} + \frac{\partial E_2}{\partial x_2} = 0 \\ \nabla \cdot \overline{E} &= k_1 + k_2 = 0 \end{aligned} \tag{2.8}$$

This implies that $k_1 = -k_2 \equiv k$. The eq. 2.8 means that it is not possible to have a focusing field in both planes simultaneously. Therefore, as it will be discussed later in the RFQ, the fields must oscillate over time in order to reverse the focusing and defocusing effects. But, in order to keep the approximation of quasi-static fields, the oscillation frequency must be low enough in order to have a wavelength that is long with respect to the dimensions of the cross section of the RFQ.

From the equations of the fields, it is easy to get the expression of the potential

$$\begin{aligned}\phi(x_1,x_2,x_3) =& \phi(0,0,0) + \int_{(0,0,0)}^{(x_1,0,0)} E_1 dx_1' + \int_{(x_1,0,0)}^{(x_1,x_2,0)} E_2 dx_2' \\ &+ \int_{(x_1,x_2,0)}^{(x_1,x_2,x_3)} E_3 dx_3' \qquad (2.9)\\ \phi(x_1,x_2,x_3) =& \phi(0,0,0) + \int_{(0,0,0)}^{(x_1,0,0)} k x_1' dx_1' - \int_{(x_1,0,0)}^{(x_1,x_2,0)} k x_2' dx_2' \\ &+ \int_{(x_1,x_2,0)}^{(x_1,x_2,x_3)} 0 dx_3' \qquad (2.10)\\ \phi(x_1,x_2,x_3) =& \phi(0,0,0) + \frac{k}{2}x_1^2 - \frac{k}{2}x_2^2 \qquad (2.11)\end{aligned}$$

A perfectly conducting medium has the particular property to have a constant potential value everywhere inside the conductor and on its surface only perpendicular electric field may exist. Indeed, let be $x : \mathbb{R} \to \mathbb{R}^3$ a regular path lying on a constant potential, one has the following property

$$\begin{aligned}\nabla\phi &= -\overline{E} \qquad (2.12)\\ \frac{d\phi(x(l))}{dl} &= \frac{\partial\phi}{\partial x_i}\frac{dx_i}{dl} = -\overline{E}\frac{dx_i}{dl} = 0 \qquad (2.13)\end{aligned}$$

This suggests that a perfect conductors might be placed to generate the electrostatic field. This is motivated for bounded domain by the uniqueness theorem of Poisson's equation [15]. In appendix, the unicity theorem is recall (Th. 1) and demonstrated for close structure.

The equipotential paths of the potential 2.11 are hyperbolic. A parametric representation of this equation is given by

$$\begin{aligned}x_1 &= A\sinh(l)\\ x_2 &= B\cosh(l)\end{aligned} \qquad (2.14)$$

where $l \in \mathbb{R}$ is the parametric variable. Let us inject these parametric equations in the potential 2.11

$$\frac{2}{k}\phi(\overline{x}(l)) = A^2\sinh(l)^2 - B^2\cosh(l)^2 \qquad (2.15)$$

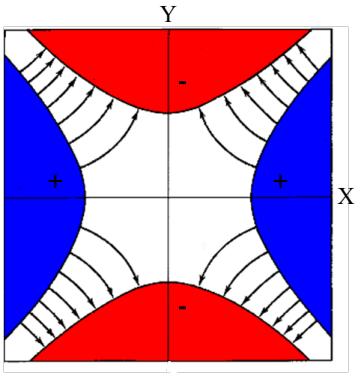


Figure 2.1: RFQ - electrodes

Recalling that $\cosh(l)^2 - \sinh(l)^2 = 1$, imposing $A = B$ leads to

$$\frac{2}{k}\phi(\overline{x}(l)) = A^2 \tag{2.16}$$

Four hyperboles must be placed around the center in order to guarantee a good distribution of the electric fields as shown Fig. 2.1. In reality, the electrodes must have a finite dimension. This implies that the electric field has a different expression than the ideal one. However, it can be shown that if the shape of the electrodes are matching the ideal shape near the region of interest the ideal electric field remains a very good approximation.

As it has been said previously, the potential of the electrodes must oscillate over time. Indeed, the alternated focusing is required in order to be able to obtain a global focusing effect in both planes [28]. Fig. 2.2 shows the principle of the alterating voltage in order to swap the focusing and defocusing effects. At time t_0, a bunch of particles is focused in the plance YZ while it is defocused in the plane XZ. Then at the time t_1, it is the other way around. The dynamics equations obtained in the transverse plane XY is the famous Hill differential equation. It will be investigated in detail in the chapter devoted to numerical analysis and differential equations (Ch. **??**).

The RFQ has another task than focusing the beam; it should accelerate the beam. In order to accelerate the beam, a certain field in the $-\hat{x}_3$ direction must be created. A simple way to do so, is to curve the electrodes in the $\hat{x}_3$ direction. Without getting into the details, Fig. 2.3 shows the internal

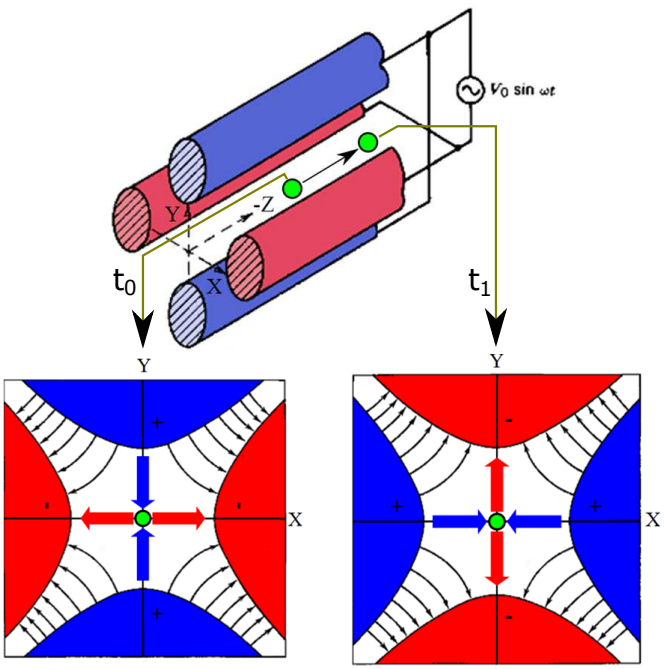


Figure 2.2: RFQ - Inversion principle.

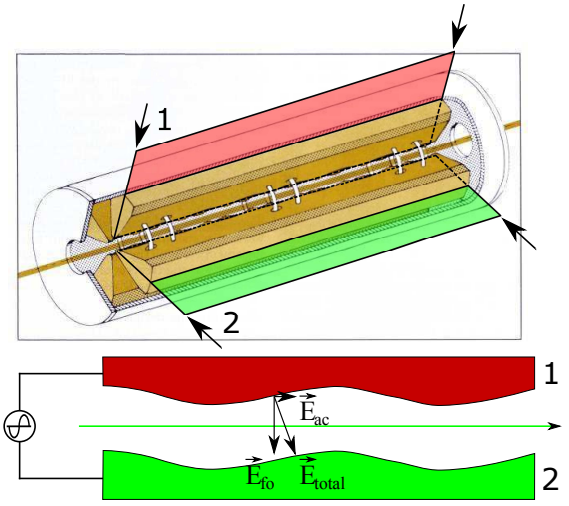


Figure 2.3: RFQ - Implementation of the acceleration task

geometry of the RFQ with the accelerating field. On the picture, the accelerating field is shown in the right direction. Since the RFQ must be supplied with an alternative power source, the accelerating field will oscillate with time as well. Therefore, the beam of particles must be bunched and the bunches must be synchronized with the oscillating field as prerequisite for acceleration. The continuous beam of particles is injected at the entrance of the RFQ and then some particles of the beam are accelerated and some others are decelerated. Thus, bunches appear naturally at the beginning of the RFQ. There are two principal ways to build an RFQ: The four-Vane RFQ and the four-Rod RFQ. The four-Vane RFQ is rather used for higher power density designs since the heat extraction for such a structure is easier. It is based on four weakly coupled cavities as shown on top Fig. 2.3. It can be designed for a wide range of frequencies (from 50Mhz to 500Mhz) but preferably for higher frequencies than a four-Rod RFQ since it tends to take a huge amount of space (copper) at lower frequencies in comparison to a four-rod RFQ. For Myrrha's LINAC the four-rod RFQ will be used.

For a four-rod RFQ, the rods are supported by the so-called stems. The stems are made of copper and are connected all together through a ground plane. They transport current that flows between rods of different polarities. Fig. 2.4 shows the current flow in a four-Rod RFQ.

The rods and the stems form a resonant circuit. The stems can be

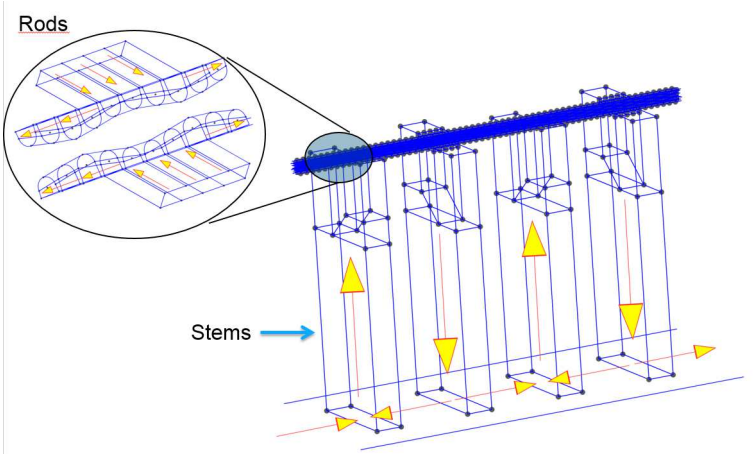


Figure 2.4: Four-rod RFQ stems

modelled as equivalent inductances while the four rods can be modelled locally as equivalent capacitors. Since there are many stems (40 stems for Myrrha's RFQ), the RFQ behaves approximatively as a transmission line. The positions and dimensions of the stems and rods determine the resonant frequency of the system.

The frequency of Myrrha's RFQ is set to 176.1Mhz. Its nominal current is 5mA, the input beam energy is 30keV and it shall deliver an output energy beam of 1.5MeV. The accelerator is 4m long.

2.2 The Space Charge issue

In the previous section, the charge of particles was fully neglected in the RFQ. However, for a high-current LINAC, the effect of the Space Charge (SC) [29] can strongly affect the distribution of the fields. Nowadays, the effect of SC are not included in the simulation of the field directly. The problem is solved in two steps. The first step is the computation of the field due to the external RF power supply called the source field without any beam load in the accelerator. This field is assumed dominant in the RFQ. In the Maxwell's equations, only equivalent current distributions appear since there is no beam load.

$$\nabla \times \overline{E}_s = -\mu \frac{\partial \overline{H}_s}{\partial t} + \overline{J}_m \tag{2.17}$$

$$\nabla \times \overline{H}_s = -\epsilon \frac{\partial \overline{E}_s}{\partial t} + \overline{J}_e \tag{2.18}$$

$$\nabla \cdot \overline{E}_s = \rho_e \tag{2.19}$$

$$\nabla \cdot \overline{H}_s = \rho_m \tag{2.20}$$

These equations can be solved with the Method of Moments (MoM) a Boundary Element Method (BEM) presented in the chapter 3.

Then, the second step consists in solving locally the static Maxwell's equations at each integration step of the equations of motion. The static Maxwell's equation are given by

$$\nabla \times \overline{E}_l = 0 \tag{2.21}$$

$$\nabla \times \overline{H}_l = \overline{J} \tag{2.22}$$

$$\nabla \cdot \overline{E}_l = \rho \tag{2.23}$$

$$\nabla \cdot \overline{H}_l = 0 \tag{2.24}$$

where the subscript l stands for local. The idea is to take into account the Coulombian repulsion force and the magnetic force as shown Fig. 2.5. A given bunch of particles creates a static electric field due to its charge and a quasi-static magnetic field due to its charge in motion. Hence, a force appears on the surrounding bunches due to these fields.

The force undergone by a given particle is given by the Lorentz force

$$\overline{F} = q(\overline{E}_t + \overline{v} \times \overline{B}) \tag{2.25}$$

where the subscript t stands for total. For a given bunch, the total field is given by the superposition of the source fields and the static fields due to some neighbour bunches.

$$\overline{E}_t = \overline{E}_s + \overline{E}_l \tag{2.26}$$

$$\overline{H}_t = \overline{H}_s + \overline{H}_l \tag{2.27}$$

In the assumption of quasi-static field for the SC consideration, the problem of the SC at low speed can be understood easily. Let us take

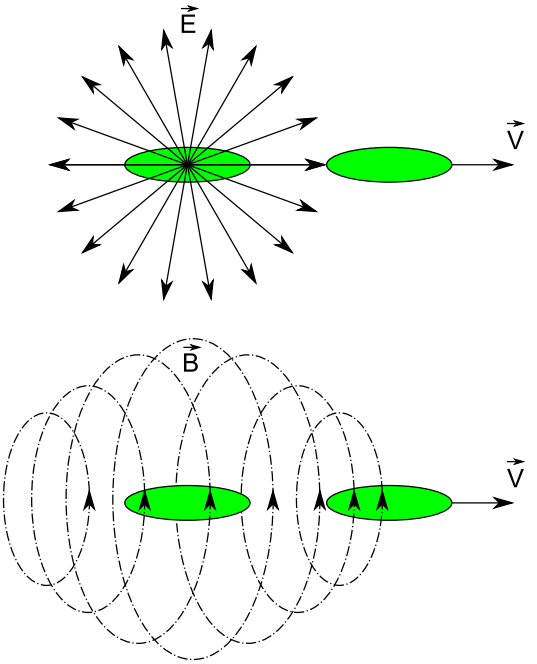


Figure 2.5: Local static electric field and quasi-static magnetic field approximation

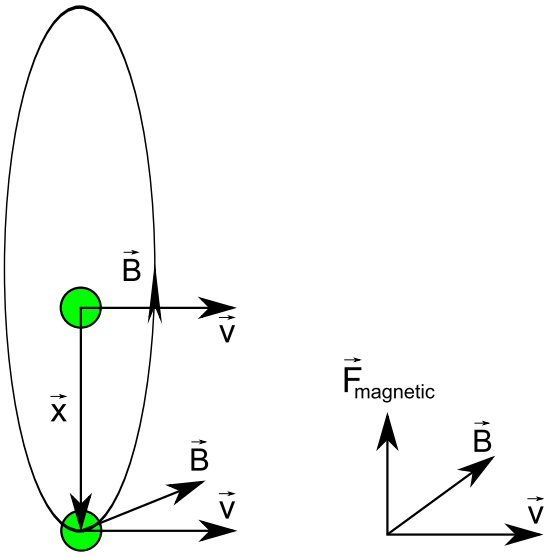


Figure 2.6: Magnetic field due to space charge.

two bunches of particles with an identical speed as shown Fig. 2.6. The first bunch (top bunch) creates a magnetic field which is felt by the second bunch. The expression of the magnetic field is given by

$$\overline{B} = \frac{\mu_0}{4\pi}\frac{q\overline{v} \times \overline{x}}{x^3} \tag{2.28}$$

The force applied on the second bunch due to the magnetic field is obtained by the Lorentz force

$$\overline{F}_{magnetic} = q\overline{v} \times \overline{B} = -\frac{\mu}{4\pi}\frac{q^2v^2}{x^2}\hat{x} \tag{2.29}$$

Hence, the total force accounting for the Coulombian force is equal to

$$\overline{F} = (\frac{1}{\epsilon} - \mu_0 v^2)\frac{q^2}{4\pi x^2}\hat{x} \tag{2.30}$$

Clearly, if the speed is low, the Coulombian effect is less compensated by the magnetic force. Unfortunately, in a RFQ the energy of the particles is low and the effects of the magnetic field are negligible in comparision to the Coulombian field.

Chapter 3

Method of Moments

3.1 Introduction

In this chapter the Method of Moments (MoM) principle is explained in details. The pure mathematical aspects such as the regularities of the functions, uniqueness and existence of solutions are not discussed here but can be found in following book [16]. The book covers all the theory regarding the uniqueness and the existence of the solution of the Helmoltz equation within the appropriate Sobolev space. The integro-differential solution is also analysed in details. As stated at the very beginning of the book, the surface operators are presented in a very simplify form in order to avoid all the complicated settings of differential geometry. If one wants to go futher, additional readings about differential geometry can be found in [30],[26]. In the following sections, the algebraic operations performed to obtain the integro-differential operators of the MoM are shown. The objects in these equations may be functions in a smooth functional space, functions in a Lebesgue or Sobolev space or even distributions. Again, we send back the reader to the references here-above for more details about these objects.

The MoM is a Boundary Element Method (BEM). It is based on the impulse response of the Helmholtz equation in $\mathbb{R}^3$ enforced with the Sommerfeld radiation condition. This implies that all the media must have a linear electromagnetic behaviour. In order to obtain a Helmholtz equation valid on $\mathbb{R}^3$, the equivalence theorem is used [13]. It allows to shadow some media by using equivalent currents on the surfaces surrounding them. The equivalence theorem can be demonstrated with the help of distributions.

3.2 Electric Field Integral Equation

Thanks to the linearity of the Maxwell's equations, the equivalence theorem and the theory of distribution one may express the electric and magnetic field everywhere in space as a integro-differential equations. For perfect conductors only, the electric field expression is given by the following formula

$$\overline{E}(\overline{r}) = -\mu(j\omega - \frac{1}{j\omega\epsilon}\nabla\nabla\cdot)\int_S \overline{J}(\overline{r}')G(\overline{r}',\overline{r})d\overline{r}' \tag{3.1}$$

where

- $G : \mathbb{R}^3 \to \mathbb{C}, \overline{r} \to \frac{e^{jk||\overline{r}'-\overline{r}||}}{||\overline{r}'-\overline{r}||}$ the Green's function
- $\overline{E} : \mathbb{R}^3 \to \mathbb{C}^3$ the harmonic electric field
- $Vec(S)$ the space of vector fields (or one-form) on S
- $\overline{J} : Vec(S) \to \mathbb{C}^3$ the harmonic current distribution on the surfaces of the perfect conductors
- S the perfect conductor boundary surfaces assumed as a piecewise compact orientable C^k manifold $(k > 1)$
- μ is the permeability of the medium
- ω is the impulsion frequency
- ϵ is the permittivity of the medium

The Method of Moments [13], [14] consists of inverting this integro-differential equation in order to obtain the current distribution. In turn, the current distribution can be used to calculate the fields scattered everywhere in space. The idea is simply to approximate the current distribution as a linear combination of basis functions and to find the best coefficients in a sense that must be defined. Let $\{\overline{J}_i\}_{i\in[1,...,N]}$ be the set of basis functions and $E = span < \overline{J}_i, ..., \overline{J}_N >$ be the vector space spanned by these basis functions. Any approximated current on the structure is given by $\overline{J} = \sum_{i=1}^{N} \alpha_i \overline{J}_i$. Let $T : \mathbb{C}^N \to Vec(S), \alpha \to \sum_{i=1}^{N} \alpha_i \overline{J}_i$ be the linear application that provides the current distribution on $Vec(S)$ of a given set of coefficients $x \in \mathbb{C}$. It is trivial to show that this operator is an isomorphism and will be extensively used in this thesis. Let $\overline{J} \in E$ be a current, the inverse operation of T is denoted by $[overline J] = T^{-1}(\overline{J})$ and will be used intensively as well.

Let be $L : Vec(S) \to Vec(S)$ the integro-differential operator that gives the electric field vector field (i.e the parallel component of the electric field to the embedded surface) for a given current distribution on S. Let be $||\ ||$:

$Vec(S) \rightarrow Vec(S)$ the norm given by $||x|| = \sqrt{< x|x >}$ where the scalar product is defined on the piecewise manifold by $< | >: Vec(S) \times Vec(S) \rightarrow \mathbb{C}, (\overline{f}, \overline{g}) \rightarrow \int_S \overline{f} \cdot \overline{g}^* dS$. Let be $\overline{E}_s : Vec(S) \rightarrow Vec(S)$ the source field (in general scattered by far away sources). The total vector field on S is the sum of the source field and the field scattered by the current distribution. The total field must be equal to zero on a perfect conductor. Since the current is approximated by a finite number of basis functions, the boundary condition cannot be fulfilled for every point on the surface. However, the boundary condition can be approximated by the minimization of the norm

$$\min_{\overline{J} \in Vec(S)} ||L(\overline{J}) + \overline{E}_s|| \tag{3.2}$$

Because the norm is defined from a scalar product and since the current distribution is approximated by a linear combination of basis functions it is straightforward to show that the minimization of the norm is equivalent to

$$\sum_{k=1}^{N} \alpha_k < \overline{J}_k|\overline{J}_l >= - < \overline{E}_s|\overline{J}_l >, \ \forall l \in [1, ..., N] \tag{3.3}$$

where $\{\alpha_i\}_{[1,...,N]} \subset \mathbb{C}$ is a finite sequence of complex coefficients, $\{\overline{J}_j\}_{[1,...,N]}$ a finite sequence of basis functions on $Vec(S)$. The matrix defined by $Z_{kl} \equiv < \overline{J}_k|\overline{J}_l >$ is called the impedance matrix.

After some additional analysis [13], the impedance matrix is given by

$$Z_{kl} = j\mu c \int_{\partial S} \int_{\partial S} k < \overline{J}_k|\overline{J}_l > -\frac{1}{k}[\nabla \cdot \overline{J}_k][\nabla \cdot \overline{J}_l] G(\overline{r}', \overline{r}) d\overline{r}' d\overline{r} \tag{3.4}$$

Fig. 3.1 summarizes the EFIE in a single figure.

The basis functions used for the approximation of the current distribution are the so-called Rao-Wilton-Glisson basis functions(RWG) [13]. The definition is given by the following equations for a basis function at index n

$$\overline{J}_n \equiv \frac{L_n}{2S_-}(\overline{r} - \overline{v}_-) \tag{3.5}$$

$$\overline{J}_n \equiv \frac{L_n}{2S_+}(\overline{v}_+ - \overline{r}) \tag{3.6}$$

where the variables are defined on Fig. 3.2

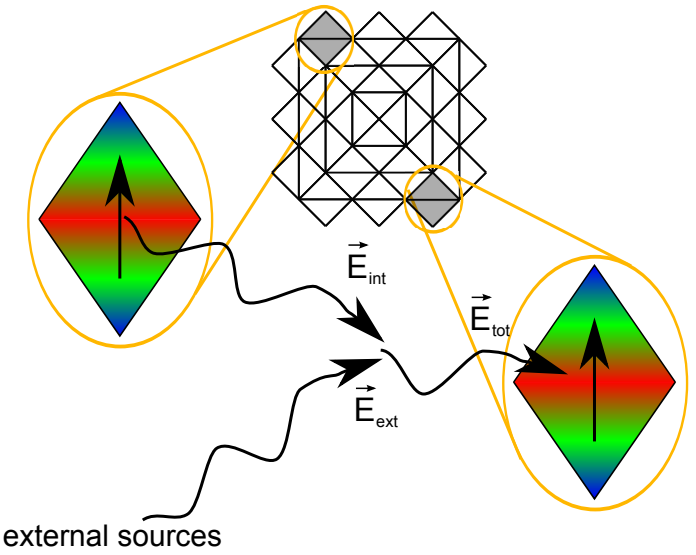


Figure 3.1: The Electric Field Integral Equation in picture

The RWG basis functions are div-conforming (i.e the divergence is finite) which is required for the EFIE. Unfortunately, they are not curl-conforming which is an important point to understand for the low-frequency breakdown section.

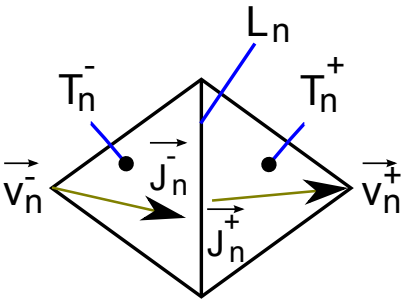


Figure 3.2: The Rao-Wilton-Glisson basis function

Chapter 4

Iterative solvers

4.1 Introduction

In this chapter an iterative solver is proposed to solve the MoM system of equations. The General Minimal RESidual method (GMRES) is the iterative solver used in this research and will be recalled in section 4.2. The EFIE suffers from the so-called Low-Frequency Breakdown (LFB) which causes GMRES to converge very slowly. In order to improve the convergence of GMRES, a preconditioner proposed by [20] is used. The theory related to this preconditioner is explained in section 4.3. Computing the whole EFIE impedance matrix is of complexity $O(N^2)$ with N the number of basis functions. This complexity is acceptable for a single simulation of the accelerator but becomes less acceptable in the context of a numerical optimization of the accelerator. Indeed, for each new modification of the geometry the impedance matrix must be recalculated (at least partially). GMRES, as most iterative techniques, requires at each step the multiplication of a column vector to the left by the matrix of the system of equations. This is generally considered computationally too expensive, and several methods have been developed to obtain this product without actually computing the matrix itself [31], [32], [33], [34], [35], [34], [36]. However these methods cannot be used when the structure becomes too small in comparison to the wavelength (< 2 wavelengths) which is the case with our accelerator. In order to improve the computation time of the solver a new method is proposed in section 4.4. The complexity of this new method contains still a term in $O(N^2)$ but with a much smaller constant than the one in the full matrix calculation.

4.2 General Minimal Residual Method

Direct inversion of the linear MoM system of equations is of complexity $O(N^3)$. In order to improve the complexity, iterative solvers are commonly used for MoM solution. Unfortunately, the MoM impedance matrix does not have any nice common property, namely it is not a diagonal dominant matrix nor a positive defined matrix. Hence, several iterative solvers such as the conjugate gradient cannot be applied for MoM solution [37],[38]. A natural choice when the matrix does not seem to have any nice property is the Krylov iterative solver family a particular case of projection methods [38], [37]. Since the final goal of our simulations is to obtain an accurate electromagnetic field, GMRES [39] is retained. Indeed, GMRES consists in minimizing the residual error int the image space of the matrix. Hereunder, the GMRES method is recalled in a way that consists in introducing the method by fulfilling our need namely obtaining an accurate electric field.

Let $A \in \mathbb{C}^{N\times N}$ be an invertible matrix. One wants to solve $Ax = b$ where $x, b \in \mathbb{C}^N$. Let assume that one has a first guess of the solution that one calls $x_0 \in \mathbb{C}^N$. The initial residual error is defined by

$$r_0 = b - Ax_0 \tag{4.1}$$

Now, the question is what one wants to approximate? One could try to approximate either the vector b or the vector x. Once this question is answered, another question comes directly, how do one measure the approximation i.e what norm should one use?

Since one wants to obtain an accurate simulation of the electric field in the accelerator, the minimization is performed in the image space (i.e on b). The accuracy of the currents might be important if the ohmic losses have to be evaluated. An upper bound of this accuracy can be obtained with the help of the condition number of the matrix A calculated for the operator norm. Let us recall the definition of the operator norm

$$||A|| \equiv \sup_{||x||_2=1} ||Ax||_2 = \max_{||x||_2=1} ||Ax||_2 \tag{4.2}$$

where the last equality comes from the fact that one deals with finite dimensional vector space i.e $B[0,1]\backslash B(0,1)$ is compact and A is continuous. Now, let us assume that b is approximated by $\tilde{b}$ one has by definition

$$||x - \tilde{x}||_2 \leq ||A^{-1}||||b - \tilde{b}||_2 \tag{4.3}$$

with $\tilde{x} \equiv A^{-1}\tilde{b}$. Again, by definition one has

$$||A||||x||_2 \geq ||b||_2 \tag{4.4}$$

A bound for the relative current error can be obtained by combining Eq. 4.3 and Eq. 4.4 as follows

$$\frac{||x - \tilde{x}||_2}{||x||_2} \leq ||A^{-1}||||A|| \frac{||b - \tilde{b}||_2}{||b||} \tag{4.5}$$

The condition number is defined by $k(A) \equiv ||A^{-1}||A||$. The Eq. 4.5 becomes

$$\frac{||x - \tilde{x}||_2}{||x||_2} \leq k(A) \frac{||b - \tilde{b}||_2}{||b||} \tag{4.6}$$

A small condition number guaranty a good current estimation. If the condition number is large, no conclusion can be made based on the bound given by Eq. 4.6. It is important to stress that the norm two of the current vector and the excitation vector do not represent the norm 2 of the physical current and the physical excitation on the manifold.

Let us assume a search subspace K of $\mathbb{C}^N$ and an initial guess solution $x_0 \in \mathbb{C}^N$, the solution can be expressed as follows

$$x = x_0 + \alpha, \alpha \in K \tag{4.7}$$

The approximation is expressed as

$$\min_{\alpha \in K} ||b - Ax||_2 = \min_{\alpha \in K} ||r_0 - A\alpha||_2 \tag{4.8}$$

The norm two is the canonic norm induced by the following scalar product

$$< x|y > \equiv \sum_{i=1}^{N} x_i^* y_i \text{ with } x, y \in \mathbb{C}^N \tag{4.9}$$

where $*$ means conjugate. One can prove easily that the minimization is equivalent to find $\alpha \in \mathbb{C}^N$ such that

$$< \beta | r_0 - A\alpha >= 0, \forall \beta \in AK \tag{4.10}$$

i.e the residual error is orthogonal to the space AK. The question now is how to choose the vector space K. The generation of new vectors must ensure that if the generation process stops growing the vector space K then the exact solution must be inside the space. Indeed, if this condition is not satisfied then one might never reach the minimum threshold error chosen. This condition can be fulfilled be trying to generate an invariant subspace ($AK = K$) where r_0 is inside this invariant subspace. Indeed, in such situation, since A is injective one has $\exists \alpha \in K$ such that $A\alpha = r_0$. A natural way to generate such a space is to build the following subspace $K_m(A, r_0) =< r_0, Ar_0, ..., A^{m-1}r_0 >$ known as the Krylov subspace. If the space stops growing at a certain iteration k, it means by definition that $AK_k \subset K_k$ since the vector $A^k r_0$ is a linear combination of the former vectors. By injectivity, one has $dim(AK_k) = dim(K_k)$ and therefore $AK_k = K_k$.

Now, one can reformulate the problem considering the Krylov subspace K_m. Any solution is given by the following equation

$$\alpha = \sum_{i=0}^{m-1} \alpha_i A^i r_0 \tag{4.11}$$

The orthogonality condition becomes

$$< A^k r_0 | r_0 - A \sum_{i=0}^{m-1} \alpha_i A^i r_0 > \quad = \quad 0, \forall k \in [1, ..., m] \tag{4.12}$$

$$\Leftrightarrow \sum_{i=1}^{m} < A^k r_0 | A^i r_0 > \alpha_{i-1} \quad = \quad < A^k r_0 | r_0 > \forall k \in [1, ..., m] \tag{4.13}$$

In order to solve this equation numerically, one should be careful to the floating point errors. A vectors $A^i r_0, i \in [1, ..., m]$ could be almost linearly dependent. By almost linearly dependent one means that the absolute value of the scalar product between two normalized vectors spanning the Krylov subspace is close to one. One way to improve and ensure the numerical linear independence of basis vectors is to orthogonalize them with the canonic scalar product. If the orthogonalization fails, the iterative solver is stopped.

The orthogonalization can be performed in a smart way i.e let's try to obtain an orthonormal basis $v_0, ..., v_{m-1}$ such that the expression Av_k is simply expressed as linear combination of v_l obtained in the orthogonalization process. The important observation that one should do is the following: Let assume $v_0, ..., v_{m-1}$ an orthonormal basis of K_m obtained with Gram-Schmidt i.e

$$v_k \equiv \frac{x_k}{||x_k||} \tag{4.14}$$

$$x_k \equiv A^k r_0 - \sum_{i=0}^{k-1} < v_i | A^k r_0 > v_i \tag{4.15}$$

then, it is strictly equivalent to replace $A^k r_0$ by the vector Av_{k-1}. Indeed, let us inject Av_{k-1} (giving up the normalization) in the expression of x_k replacing the original $A^k r_0$

$$x'_k = Av_{k-1} - \sum_{i=0}^{k-1} < v_i | Av_{k-1} > v_i \tag{4.16}$$

$$= A^k r_0 - \sum_{j=0}^{k-2} < v_j | A^k r_0 > Av_j$$

$$- \sum_{i=0}^{k-1} < v_i | A^k r_0 - \sum_{j=0}^{k-2} < v_j | A^k r_0 > Av_j > v_i \tag{4.17}$$

$$= A^k r_0 - \sum_{j=0}^{k-2} < v_j | A^k r_0 > Av_j$$

$$- \sum_{i=0}^{k-1} < v_i | A^k r_0 > v_i + \sum_{i=0}^{k-1} \sum_{j=0}^{k-2} < v_j | A^k r_0 >< v_i | Av_j > v_i \tag{4.18}$$

The double sum can be rewritten as follows

$$\sum_{i=0}^{k-1} \sum_{j=0}^{k-2} < v_j | A^k r_0 >< v_i | Av_j > v_i = \sum_{j=0}^{k-2} < v_j | A^k r_0 > \sum_{i=0}^{k-1} < v_i | Av_j > v_i \tag{4.19}$$

Since $Av_j \in K_m(A, r_0) = sev < v_0, ..., v_{m-1} >$, one has $\sum_{i=0}^{k-1} < v_i | Av_j > v_i = Av_j$. Injecting this result in x'_k one gets

$$x'_k = A^k r_0 - \sum_{i=0}^{k-1} < v_i | A^k r_0 > v_i = x_k \tag{4.20}$$

Therefore, the equation becomes

$$v_k = \frac{Av_{k-1} - \sum_{i=0}^{k-1} < v_i | Av_{k-1} > v_i}{||Av_{k-1} - \sum_{i=0}^{k-1} < v_i | Av_{k-1} > v_i||} \tag{4.21}$$

The expression of Av_k is therefore given by

$$Av_k = ||Av_k - \sum_{i=0}^{k} < v_i | Av_k > v_i || v_k + \sum_{i=0}^{k} < v_i | Av_k > v_i \tag{4.22}$$

One defines the (modified) Hessenberg matrix by

$$H_{k,l} = < v_k | Av_l > \tag{4.23}$$

$$H_{l+1,l} = ||Av_l - \sum_{i=0}^{l} < v_i | Av_l > v_i|| \tag{4.24}$$

One has therefore $AV_m = V_{m+1}H$ where V_m is the orthonormalized basis of the krylov space K_m.

Let us rewrite the GMRES equation in term of Hessenberg matrix. Let be $\alpha = \sum_{i=0}^{m-1} \alpha_i v_i$, one has

$$\sum_{i=0}^{m-1} < v_k | Av_i > \alpha_i = < v_k | r_0 > \forall k \in [1, ..., m]$$

In matrix form, one has

$$(AV_m)^a AV_m \alpha = (AV_m)^a r_0 \tag{4.25}$$

$$\Leftrightarrow \quad (V_{m+1}H)^a V_{m+1} H\alpha = (V_{m+1}H)^a r_0 \tag{4.26}$$

$$\Leftrightarrow \quad H^a V_{m+1}]^a V_{m+1} H\alpha = (V_{m+1}H)^a r_0 \tag{4.27}$$

$$\Leftrightarrow \quad H^a H\alpha = H^a(:,1)||r_0|| \tag{4.28}$$

The derivation of error bounds to evaluate the convergence of GMRES is a complicated subject for general matrices (non-normal matrices) [40]. Under the hypothesis of an self-adjoint operator there exists an efficient error bound [41]. In the following paper [42], different GMRES bounds are compared and the non-normal matrix case is addressed as well.

4.3 Low-Frequency-Breakdown and preconditioner

4.3.1 Introduction

The EFIE suffers from what is commonly called a low-frequency-breakdown. Let us recall the EFIE equation [13]

$$Z_{kl} = j\mu c \int_S \int_S k < \overline{J}_k | \overline{J}_l > -\frac{1}{k}[\nabla \cdot \overline{J}_k][\nabla \cdot \overline{J}_l] G(\overline{r}', \overline{r}) d\overline{r}' d\overline{r} \qquad (4.29)$$

One may notice two terms in this equation; one which is proportional to k and one inversely proportional to k. Let us define two bilinear forms corresponding respectively to these two terms

$$\Sigma(\overline{J}_k, \overline{J}_l) \equiv \int_S \int_S k < \overline{J}_k | \overline{J}_l > G(\overline{r}', \overline{r}) d\overline{r}' d\overline{r} \qquad (4.30)$$

$$\Gamma(\overline{J}_k, \overline{J}_l) \equiv \int_S \int_S [\nabla \cdot \overline{J}_k][\nabla \cdot \overline{J}_l] \; G(\overline{r}', \overline{r}) d\overline{r}' d\overline{r} \qquad (4.31)$$

The EFIE becomes therefore

$$Z_{kl} = j\mu c(k\Sigma(\overline{J}_k, \overline{J}_l) + \frac{1}{k}\Gamma(\overline{J}_k, \overline{J}_l)) \qquad (4.32)$$

From the Γ and Σ forms, one may define for a given set of testing functions two operators that we call Γ and Σ as well since the identification is unambiguous

$$\Gamma : Vec(S) \to \mathbb{C}^N : \overline{J} \to (\Gamma(\overline{J}_1, \overline{J}), ..., \Gamma(\overline{J}_N, \overline{J})) \qquad (4.33)$$

$$\Sigma : Vec(S) \to \mathbb{C}^N : \overline{J} \to (\Sigma(\overline{J}_1, \overline{J}), ..., \Sigma(\overline{J}_N, \overline{J})) \qquad (4.34)$$

When the frequency increases, the Σ operator becomes dominant while when the frequency decreases the Γ operator is dominant. When the frequency is low, the magnetic field and the electric field tends to uncouple. The Σ operator corresponds to the contribution of the magnetic field while the Γ operator corresponds to the contribution of the electric field in this limit case. The magnetic field does not contribute to the electric field at low frequency. Since the EFIE is based on the test of the electric field on S, the Σ operator tends to vanish (but not entirely) in front of the Γ operator.

Let us investigate a bit closer the operator Γ. A first very important observation is that the operator is not injective. Indeed, let us take the very simple example of current distribution as shown Fig. 4.1, where 8 RWGs are combined to form a quasi-circulating current.

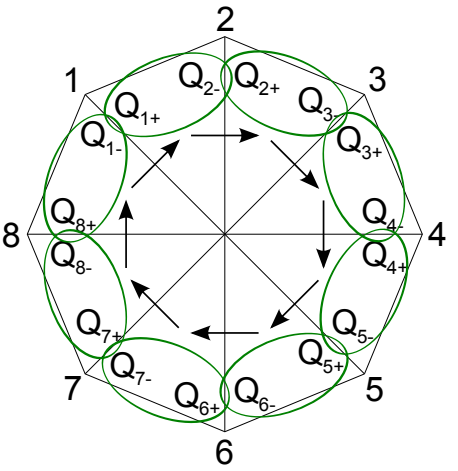


Figure 4.1: Kernel of the Γ operator

All the elements are isometric. By definition of the RWG basis functions, it is easy to build a current distribution such that the divergence of the field is equal to zero everywhere. Physically, it means that the harmonic charge density is equal to zero everywhere. Using the notation of Fig. 3.2, one can easily obtain that

$$\nabla \cdot \overline{J}_n^- = \frac{L_n}{2S_-^n} \tag{4.35}$$

$$\nabla \cdot \overline{J}_n^+ = -\frac{L_n}{2S_+^n} \tag{4.36}$$

In passing, the global charge of a current distribution approximate by linear combination of RWG is always equal to zero.

The arrows show the orientation of the basis functions. Since the triangles are all isometric one gets

$$\nabla \cdot \overline{J}_n^- = \frac{L}{2S} \tag{4.37}$$

$$\nabla \cdot \overline{J}_n^+ = -\frac{L}{2S} \tag{4.38}$$

It is easy to see that the following current distribution $\overline{J} = \sum_{i=1}^{8} \overline{J}_i$ ensures a divergence free function everywhere. The current representing a loop and for which the divergence is equal to zero is called a current loop. It is cleared by definition that $\Gamma(\overline{J}_k, \overline{J}) = 0$, where $\overline{J}_k$ is any test function i.e $\overline{J} \in ker(\Gamma)$.

The Helmholtz decomposition theorem (known as the lemma of Poincaré [26], [27]) ensures that any vector field can be written as the unique sum of a curl-free vector field and div-free vector field[1]. The curl-free and div-free vector fields are furthermore orthogonal in the sense of the following scalar product

$$\int_S C(\overline{J}) \cdot D(\overline{J}) d\overline{r} \tag{4.39}$$

where $C(\overline{J})$ is the curl-free component of $\overline{J}$ and $D(\overline{J}$ is the div-free component of the current.

By definition, divergence-free currents are in the kernel of the Γ operator. It is relatively easy to see that the div-free current can only be a loop current. Indeed, a non-closed current line made of RWG would induce a non-zero divergence at the tail and head of the current line. Reciprocally, a loop current is always div-free. The set of div-free current is therefore completely identify for RWG basis functions. It is straightforward to see that it is a vector space. A natural question is, is-there a set a basis functions that spans the div-free space? Based on the linear characteristic of the RWG and the Helmholtz decomposition theorem, it is possible to find such a set of basis functions. Let $L \subset span < \overline{RWG}_1, ..., \overline{RWG}_n >$ be the sub-space of div-free basis functions and $\overline{J}_l \in L$ then on each triangle of the mesh one has $\nabla \cdot \overline{J}_l = 0$. It means, that on closed oriented manifold one can find a scalar function $M \in Scal(S)$ a [18] which is a piecewise linear function[2] such that

$$\overline{J}_l = \nabla_t \times \hat{n} M \tag{4.40}$$

where $\hat{n}$ is the normal vector to the oriented piecewise manifold S and ∇_t is the covariant gradient operator on S. Choosing a set of basis functions that spans M allows one to generate basis functions for the L sub-space. A basic and simplest representation of M is to employ the scalar linear

[1] The hypothesis of the Poincaré lemma are not discussed here but a very important attention must be addressed concerning the regularity of the vector fields.

[2] This is imposed by the regularity of RWG functions

Lagrange (or nodal interpolating) basis, i.e the piecewise linear functions Λ_k that has a unit value on the node k and zero value on the direct neighbouring nodes of the node k. Now, the basis functions for L can be generated thanks to the ones for M

$$\overline{L}_k = \nabla_t \times \hat{n}\Lambda_k \tag{4.41}$$

For compact manifold with boundaries and eventually with holes, the value of M at the nodes on boundaries must be constant. This is because the current cannot flow outside the geometry. Therefore, special M functions must be used for the generation of basis functions for the boundaries and the Λ_k functions can only be used for inner nodes of the mesh. After some analysis, it turns out that the difference between two loop currents around the same hole loop is equal to a linear combinations of some inner loops. The number of loop basis functions is therefore equal to the number of inner nodes plus the number of holes. Additional details can be found in the following paper [18]. Without details, for general compact manifolds, the number of basis functions is equal to the number of nodes minus one, plus the number of handles. The inner node loops $\overline{J}_k$ are called primary loops, while the handle and hole loops are simply called handle loops.

Since the Γ operator is not injective, the Σ operator will never be completely numerically negligible even at very low frequency. One can show that the MoM impedance matrix is ill conditioned at low frequency. Let be $\overline{J} \in \ker(\Gamma)$, one has

$$< \overline{J}_k | L(\overline{J}) >= j\mu c(k\Sigma(\overline{J}_k, \overline{J}) + \frac{1}{k}\Gamma(\overline{J}_k, \overline{J})) = j\mu ck\Sigma(\overline{J}_k, \overline{J}) \tag{4.42}$$

One ends up with a vector in $\mathbb{C}^N$ proportional to k. Let be $\overline{J} \in \ker^{\perp}(\Gamma)$, one has

$$< \overline{J}_k | L(\overline{J}) >= j\mu c(k\Sigma(\overline{J}_k, \overline{J}) + \frac{1}{k}\Gamma(\overline{J}_k, \overline{J})) \tag{4.43}$$

For a given set of RWG basis functions, it turns out that when $k \to 0$ some image vectors tend to zero while some other tend to infinity. This implies a condition number of the impedance matrix that grows to infinity as k tends toward zero.

4.3.2 Loop-Tree and Loop-Star

At low frequency, GMRES has a poor convergence when applied to the EFIE because of the ill conditioned system of equations. One identified the reason of the bad conditioning of the impedance Matrix; some vectors in the image space are proportional to k while some are (neglecting k) proportional to $\frac{1}{k}$. The natural idea to solve this problem is to use the space L of loop basis functions and the basis functions that spans a complementary space to L to rescale them accordingly before using them for discretization of the EFIE.

Let $E = span < \overline{RWG}_1, ..., \overline{RWG}_N >$ be the vector space of all the current distribution generated by a linear combination of RWG basis functions. Let $L = span < \overline{L}_1, ..., \overline{L}_M > \subset E$ be the sub-vector space of loop currents generated with M loop basis functions and let $S = span < \overline{S}_1, ..., \overline{S}_K > \subset E$ be a complementary space of L generated by the k basis functions such as $E = L \bigoplus S$. Let

$$
\begin{aligned}
Z_{kl}^{ll} &\equiv < \overline{L}_k | L(\overline{L}_l) > && (4.44)\\
Z_{kl}^{ls} &\equiv < \overline{L}_k | L(\overline{S}_l) > && (4.45)\\
Z_{kl}^{sl} &\equiv < \overline{S}_k | L(\overline{L}_l) > && (4.46)\\
Z_{kl}^{ss} &\equiv < \overline{S}_k | L(\overline{S}_l) > && (4.47)
\end{aligned}
$$

be the matrix blocks of the impedance matrix such that

$$
Z \equiv \begin{pmatrix} Z^{ll} & Z^{ls} \\ Z^{sl} & Z^{ss} \end{pmatrix} \qquad (4.48)
$$

One has

$$
\begin{aligned}
Z_{kl}^{ll} &\propto k && (4.49)\\
Z_{kl}^{ls} &\propto k && (4.50)\\
Z_{kl}^{sl} &\propto k && (4.51)\\
Z_{kl}^{ss} &\propto k\Sigma(\overline{S}_k, \overline{S}_l) + \frac{1}{k}\Gamma(\overline{S}_k, \overline{S}_l) && (4.52)
\end{aligned}
$$

In order to improve the condition number of the impedance matrix, the following rescaling is applied on the basis functions

$$\overline{L}'_k = \frac{1}{\sqrt{k}}\overline{L}_k \tag{4.53}$$
$$\overline{S}'_k = \sqrt{k}\overline{S}_k \tag{4.54}$$
$$\tag{4.55}$$

The matrix blocks become therefore

$$Z^{ll}_{kl} \propto 1 \tag{4.56}$$
$$Z^{ls}_{kl} \propto k \tag{4.57}$$
$$Z^{sl}_{kl} \propto k \tag{4.58}$$
$$Z^{ss}_{kl} \propto k^2\Sigma(\overline{S}_k, \overline{S}_l) + \Gamma(\overline{S}_k, \overline{S}_l) \tag{4.59}$$

The question is now what kind of complementary space S should one uses, and what kind of basis functions for L and S space should be used. Two complementary spaces will be investigated in the two next sub-sections. It is important to understand that the method proposed here does not cure the undesirable scaling of the matrix condition number with h (the average size of the basis functions) [20],[19].

4.3.2.1 Loop-Tree

A first natural choice for the complementary space would be the span of the set of all the RWG basis functions in which one removes on distinct RWG per current loop. In order to find a set of distinct RWG for which each RWG belongs to a loop, a tree or several trees are built. The trees pass through all the nodes of the mesh. Each edge, corresponds to one RWG to be removed.

The Loop-Tree [21], [22], [43], [19] algorithm consists in
- Finding all the primary loops and handle loops in order to generate the L subspace.
- Taking the set of all the RWGs and removing one RWG per basis function loop. This resulting set constitutes the basis functions for S. This is done thanks to a tree or several trees passing through all the nodes of the mesh.
- Discretizing the impedance matrix with the basis functions that spans these two complementary spaces.
- Building the preconditioner for the discretized impedance matrix.
- Solving the system of equations iteratively.

For a manifold without holes and handles, primary loops can be built easily. A Tree (or several trees if the geometry is disconnected) that pass through all the nodes of the mesh allows one to select one RWG that should be removed from the set of all RWG basis functions. Fig. 4.2 shows a compact, simply connected manifold S in which the loop-tree algorithm is used. The red lines represent the tree and the green arrows represent the RWG functions that must be removed from the set of all the RWG basis functions used to span the S space.

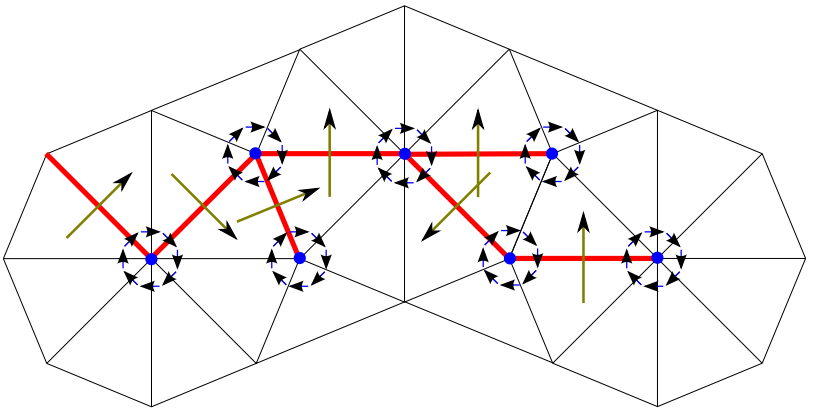


Figure 4.2: Loop-Tree method on a simple connected, compact surface S

The difficulty with this method is the generation of the handle loops. Another method [20] is propose in the next section to avoid loops generation.

4.3.3 Loop-Star and Quasi-Helmholtz projectors

For piecewise manifolds with holes and/or handles one must add the loops around the handles and holes to have a complete set of basis functions for the L space. Instead of trying to compute the loops, one considers $L^{\perp}$ the orthogonal space of L for the canonical scalar product on $\mathbb{C}^N$, N being the number of RWGs. The idea comes from the Helmholtz decomposition theorem; as explained previously, it states that each current distribution can be uniquely decomposed as the some of a div-free function and a curl-free function. But, because of the RWG definition, it is impossible to be curl-conforming. Therefore, it is impossible to build curl-free current with RWG. The Helmholtz decomposition theorem cannot be resorted using exclusively RWGs. However, a similar orthogonal property can be recover in the space of coefficients i.e in $T^{-1}(L)$ and $T^{-1}(S)$.

Let us investigate the remainder of the current distribution defined by $\overline{J}_s \equiv \overline{J} - \overline{J}_l$ where $\overline{J}_l \in L$. A first property of the remainder is that its divergence cannot vanish everywhere on the surface. Furthermore, since the RWGs are piecewise linear, the divergence is constant on each triangle cells [18]. Let p_i be the constant unit function on the triangle indexed i, then one has

$$\nabla \cdot \sum_{i=1}^{N} x_i \overline{RWG}_i = \sum_{j=1}^{N_c} y_j p_j \tag{4.60}$$

where N_c is the number of triangles (or cells). The coefficients y_j can be calculated simply by testing the divergence of the current distribution $\overline{J}_s$ on the corresponding constant function p_i as follows

$$y_j = \sum_{i=1}^{N} x_i < p_j | \nabla \cdot \overline{RWG}_i > \tag{4.61}$$

Let us define the charge matrix

$$Q_{ji} \equiv < p_j | \nabla \cdot \overline{RWG}_i > \tag{4.62}$$

$Q \in \mathbb{C}^{N_c \times N}$. Let us assume that the structure is made of k disconnected surfaces and let $N_{c,i}$ be the number of cells for the connected domain $i \in [1, .., k]$. Let $\sigma : [1, ..., k] \times \mathbb{N} \to \mathbb{N}$ be the function that give the cell index for a corresponding index of this cell in its domain. Let $I_i : \mathbb{C}^{N_c} \to \mathbb{C}^{N_i}$ be a matrix defined by

$$I_{ij,k} = 1 \text{ if } \sigma(k, i) = j \tag{4.63}$$

Since the global charge must be equal to zero on each disconnected surfaces, each line of each matrix $I_i Q$ is linearly dependent on the others and $rank(I_i Q) = N_{c,i} - 1$. Indeed, on a connected surface, it possible to set the charge of any cell except one that will take the opposite sign of the total charge on the other cells. Since $rank(Q) = \sum_{i=1}^{k} N_{c,i}$ one has $rank(Q) = \sum_{i=1}^{k} N_{c,i} - k$.

It is straightforward to see that

$$ker(Q) = L \tag{4.64}$$

On the other end, by definition of S, one has

$$Qy \neq 0 \text{ with } y \in T(S) \tag{4.65}$$

and Q is injective on $T(S)$ and therefore $rank(Q) = dim(S)$. In passing, since $N = dim(L) + dim(S)$ one has $N = dim(E) = M + \sum_{i=1}^{k} N_{c,k} - k$ [18]. This suggests to take Q^t as the coefficients of the functions that can be used to span the complementary space S. More precisely, for each connected domain i, this suggests to take $N_{c,i} - 1$ lines as coefficients of the basis functions to span the space S. Let Σ be the matrix defined by these lines but transposed such that a column of Σ correspond to a line of coefficients of a star function in Q.

Let $\overline{J} \in L$ be any loop current, then the Eq. 4.64 is equivalent to

$$Q[\overline{J}] = 0 \tag{4.66}$$

Eq. 4.66 shows that the orthogonal property of the Helmholtz theorem is recovered in the coefficient space. This implies that the complementary space $T(S)$ generated by the lines of Q is orthogonal (in the coefficient space) to loop subspace $T(L)$. This orthogonality in the coefficient space can be very useful. Indeed, as it was stressed the difficulty with the loop-tree method is the generation of handle loops. On the contrary, the generation of star functions is relatively straightforward. Thanks to the orthogonality property, it is possible to calculate the unique component of a given current distribution in S and L based only of the star matrix. Let $x \in \mathbb{C}^N$ be a current and let be $x_s \in T(S)$ and $x_l \in T(L)$ the unique vectors such that $x = x_s + x_l$. For any vector $z \in T(S)$, one has $z = 0 \Leftrightarrow < z|s_i >= 0 \forall i \in [1, ..., K]$ where $S = span < s_1, ..., s_k >$. This is easy to show by linearity of the scalar product. Using this property and the fact the $T(L)$ and $T(S)$ are orthogonal, one has

$$x - x_l - x_s = 0 \tag{4.67}$$
$$\Leftrightarrow \quad < x - x_l - x_s|s_i > = 0 \tag{4.68}$$
$$\Leftrightarrow \quad < x - x_l|s_i > =< x_s|s_i > \tag{4.69}$$
$$\Leftrightarrow \quad < x|s_i > =< x_s|s_i > \tag{4.70}$$

Now, let us assume that s_i is a basis for $T(S)$ and let be $\{\alpha_i\}_{i\in[1,...,K]}$ the unique coefficients such that

$$x_s = \sum_{j=1}^{K} \alpha_i s_i \tag{4.71}$$

Injecting this expression in the later equation one gets

$$< x|s_i >= \sum_{j=1}^{K} \alpha_j < s_j|s_i > \tag{4.72}$$

Let $G_{ji} \equiv < s_j|s_i >$ the Gram matrix of the basis functions s_i (which is invertible since s_i is a basis). The vector x_s is given by

$$x_s = \sum_{j=1}^{K} s_j \sum_{i=1}^{K} G_{ji}^{-1} < x|s_i > \tag{4.73}$$

The operator defined by

$$P_s : T(E) \to T(S), x \to \sum_{j=1}^{K} s_j \sum_{i=1}^{K} G_{ji}^{-1} < x|s_i > \tag{4.74}$$

and is called the orthogonal projector of $T(S)$. Similarly, the orthogonal projector of $T(L)$ is given by $P_l \equiv I - P_s$. The loop current is given by $x_l = x - P_s(x)$. Let us take s_i the columns of Σ, the projector becomes in a matrix form

$$P_s \equiv \Sigma(\Sigma^t\Sigma)^{-1}\Sigma^t \tag{4.75}$$
$$P_l \equiv I - P_s \tag{4.76}$$

Now, one can apply the rescaling factors respectively on the loops and the stars. The preconditioner [20] is therefore given by

$$P = \sqrt{k}P_s + \frac{1}{\sqrt{k}}(I - P_s) \tag{4.77}$$

The MoM system of equations becomes

$$P^t Z P[\overline{J}] = P^t e \tag{4.78}$$

If the loops don't need to be calculated, an inverse should be computed. Indeed, one must calculate $(\Sigma\Sigma^t)^{-1}$. However, the matrix $\Sigma\Sigma^t$ is sparse and is a graph Laplacian matrix. The preconditioner is used in an iterative scheme and therefore one only wants to evaluate the product of a given vector by this inverse matrix. A fast iterative solver can be used as well to calculate the result. It can be shown that the solution can be calculated with a linear complexity with the help of an Algebraic MultiGrid (AGM) solvers [20], [44], [45], [46].

Before finishing this section, it is important to notice that the spectrum of a projector takes only two values: zero and one. The basis function size and the size of the problem have no influence on the spectrum of our two projectors. It means that the loop and star rescaling have respectively the same amplitude impact for all the loop basis functions and all the star basis functions.

4.4 Fast Near Field MoM Solver

4.4.1 EFIE at low frequency

The complexity of a direct solution of system of equations is given by $O(N^3)$. An the other hand, computing a full EFIE impedance matrix is of complexity $O(N^2)$. With these complexities, EFIE problems become very quickly infeasible as the number of unknown grows. Several solutions were used to overcome these difficulties. In particular, iterative solver such as GMRES [41], [39] are now widely used since they have shown a very good complexity regarding the direct inversion of the MoM system of equations. Furthermore, the construction of a Krylov subspace does not require the knowledge of the impedance matrix. On the contrary, only the evaluation of the impedance matrix product a given vector has to be calculated for each iteration. This observation has led the community to conceive several fast solvers. There are two categories of fast methods: The Kernel-dependent methods and the Kernel-free methods.

In the kernel-dependent methods, one famous method is the well known Fast Multipole Method (FMM) [31], [32], [33], [34], [35] of complexity $O(N^{1.5})$. An updated version of the FMM is the so-called and well-known Multi-Level Fast Multipole Algorithm (MLFMA) [23]. The MLFMA method is a direct improvement of the FMM which consists in subdividing the domain into an oct-tree of parents and children domains. Using some tricks, the complexity of the aggregation operations and the disaggregation operations can be reduced. The computation of the product of the impedance

matrix with a given vector is achieved with the impressive complexity of $O(Nlog(N))$. However, the basis and test functions must be placed apart from each other at a minimum distance of a one wavelength. For basis and test functions separated of a distance smaller than that, the FFM-MLFMA undergoes a breakdown [34], [36]. At low frequency, the translation operator computation becomes unstable i.e it is said that the translator operator suffers from a numerical breakdown when the domains become smaller than one wavelength. A classic computation of the impedance matrix for such a close basis and test functions must be performed, at least if using only the classic formulation of the FFM-FLFMA. When the size of the basis and test functions becomes relatively small (smaller than a hundredth of a wavelength), the computation of the impedance matrix becomes cumbersome because of its complexity in $O(N^2)$. Several authors have been trying to stabilize the FMM-FLFMA breakdown in the past decade [34], [36], [47], [48]. However, all these methods suffer from uncontrolled approximations and sometimes even from inconsistent hypotheses such as in [36] where two conditions contradict each other and require a tedious empirical tuning as clearly stated in the paper.

For low-frequency problems, several authors have focused on Kernel-free methods. Basically, they rather focus on the algebraic property of the rank deficiency of the off-diagonal blocks of the impedance matrix itself than the kernel used to calculate it. These methods are said to be kernel free because they can be applied to any matrix that contains rank-deficient off diagonal blocks. For instance, in [49] a multi-level SVD method is used to improve the computation time of the matrix vector products. However, if the iterative solver converges faster because of the improved complexity of the matrix-vector products, the full impedance matrix is still required. A method called Adaptive Cross Approximation (ACA) used for asymptotically smooth kernel is also well known in the literature [50]. Several methods has been developed based on the ACA and in particular the Multilevel Adaptive Cross Approximation (MLACA) method [51]. This method is based on the observation that a scatterer emitting in testing regions of same solid angles (as seen from the source) have the same degree of freedom i.e the same impedance matrix rank. The matrix is recursively approximated using an oct-tree style method accounting for the former observations with the solid angle and using the ACA-SVD method at each level. The ACA-SVD method consists in applying a low-cost SVD on an ACA decomposition to futher reduce the two low rank matrices obtained with the ACA. However, although this method provides a matrix-vector multiplication in $O(Nlog(N))$ the computation complexity of the matrix compression (pre-precessing) is in $O(N^2lof(N))$. The kernel-free methods suffer from an

important default: the control over the accuracy of the solution. The process that consists in stopping the columns and lines generation is somewhat empirical. In particular, it is relatively easy to imagine a PEC shape that breaks down the ACA as used with the MoM.

In this thesis, an alternative method is proposed in order to reduce the computation time due to a full EFIE impedance matrix computation. Moreover, the error is completely under controlled. The method is however not kernel-free.

4.4.2 Separable EFIE at low frequency

4.4.2.1 Introduction

The idea resides in the observation that for basis functions much smaller than the wavelength ($< \frac{\lambda}{1000}$) the Green's function variations are relatively small. Therefore, sampling the Green's function for each integration point is useless in terms of accuracy and expensive in terms of computational cost. An alternative approach consists in using a Taylor expansion of the Green's function for source and test points given on a regular grid and in combining this approximation with a SVD compression technique. Using a regular grid of only few different sizes allows one to pre-calculate the SVD compressions once and for all. It will be shown in the following sections that the method allows the separability of the integral.

4.4.2.2 Local separable Green's function approximation on a predefined grid

First, let us recall and define some notations. Let $Z : E \rightarrow F$ be a linear operator between two Hermitian vector spaces E and F, and let us consider the following problem $Z\alpha = e$ that is solved with GMRES. Le α_0 be an initial solution guess and let $r_0 = A\alpha_0 - e$ be the initial residual error, then the Krylov's subspace of order n corresponding to this operator is defined by

$$K_n(Z) \equiv span(r_0, Z(r_0), ..., Z^{n-1}(r_0)) \tag{4.79}$$

Let E be a subspace spanned by the RWG basis functions

$$E \equiv span(\overline{J}_1, ..., \overline{J}_N) \tag{4.80}$$

where $\overline{J}_i$ is a RWG basis function. The test functions used in the EFIE are identical to the basis functions i.e it is a Galerkin test. The space domain of a basis function is denoted by $S_i \subset \mathbb{R}^3$ with $i \in [1, ..., N]$. The characteristic function $\Pi_B : \mathbb{R}^3 \to [0, 1]$ with $B \subset \mathbb{R}^3$ is defined by

$$\Pi_B(\overline{x}) = \begin{cases} 1 \text{ if } \overline{x} \in B \\ 0 \text{ otherwise} \end{cases} \tag{4.81}$$

$$\tag{4.82}$$

The constants j,μ,c, correspond respectively to the imaginary number $\sqrt{-1}$, the permeability of free space and the speed of light. $F(\mathbb{K}^n, \mathbb{K}_2^m)$ denotes the vector space of all the functions where the domain lies in $\mathbb{K}^n$ and the image values lie in $\mathbb{K}_2^m$ where K, K_2 are fields (here, either $\mathbb{R}$ or $\mathbb{C}$). The green's function is defined by

$$g : \mathbb{R}^3 \backslash \{0\} \to \mathbb{C}, \overline{x} \to \frac{e^{jk||\overline{x}||}}{||\overline{x}||} \tag{4.83}$$

Let us first consider the following simple example: the calculation of an entry of the EFIE impedance matrix. The numerical computation of such an entry can be formulated as follows [13]

$$\begin{aligned} Z_{ba} &\equiv < \overline{B}_b | \overline{E}(\overline{B}_a) \\ &= j\mu c \sum_{l=1}^{4} \int_{S_a} \int_{S_b} f_l(\overline{B}_b)(\overline{r}') f_l(\overline{B}_a)(\overline{r}) g(\overline{r}' - \overline{r}) d\overline{r} d\overline{r}' \end{aligned} \tag{4.84}$$

with $f_l : F(\mathbb{R}^3, \mathbb{C}^3) \to F(\mathbb{R}^3, \mathbb{C})$ some linear functions defined by

$$\overline{B} \to f_1(\overline{B}) \equiv \sqrt{k} B_1 \tag{4.85}$$

$$\overline{B} \to f_2(\overline{B}) \equiv \sqrt{k} B_2 \tag{4.86}$$

$$\overline{B} \to f_3(\overline{B}) \equiv \sqrt{k} B_3 \tag{4.87}$$

$$\overline{B} \to f_4(\overline{B}) \equiv \frac{j}{\sqrt{k}} \nabla \cdot \overline{B} \tag{4.88}$$

The integral cannot be separated into two independent integrals because of the Green's function. Fig. 4.3 illustrates the numerical integration procedure; the Green's function is sampled for pairs of integration points on the test function and on the basis function. Several quadrature can be used depending on the desired accuracy.

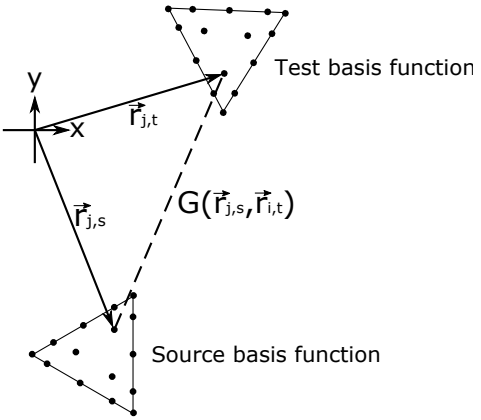


Figure 4.3: Brute force impedance matrix entry computation: Sampling of the Green's function.

But, since the basis functions are supposed to be really small in comparison to the wavelength, the Green's function is not fluctuating a lot over the integration domain assuming the function enough far away of each other to avoid the singularity. Therefore, one may think of approximating the Green's function by a simple constant over the two domains. This simple assumption would lead to a separable integral at the level of a given pair of test and basis functions. Let us take two non-overlapping local regions of space $A, B \subset \mathbb{R}^3$ placed relatively far away from each other to avoid the strong variation of the Green's function due to the $\frac{1}{r}$ singularity. Let us assume that the Green's function between these two local regions can be approximate as

$$g_a(\overline{y} - \overline{x}) \approx g(\overline{R}_b - \overline{R}_a)\Pi_A(\overline{x})\Pi_B(\overline{y}) \tag{4.89}$$

where $\overline{R}_a \in A$ and $\overline{R}_b \in B$ are two sampling points. Let us take a testing function in region B and a basis function in region A, and let us calculate the EFIE entry of the impedance matrix using approximation 4.89. One ends up with

$$Z_{ba} \approx j\mu c \sum_{l=1}^{4} \int_S \int_S f_l(\overline{B}_b)(\overline{r}') f_l(\overline{B}_a)(\overline{r}) \\ g(\overline{R}_b - \overline{R}_a)\Pi_A(\overline{r})\Pi_B(\overline{r}')d\overline{r}d\overline{r}' \tag{4.90}$$

$$\approx j\mu c \sum_{l=1}^{4} \int_{S_b} \int_{S_a} f_l(\overline{B}_b)(\overline{r}') f_l(\overline{B}_a)(\overline{r}) g(\overline{R}_b - \overline{R}_a)d\overline{r}d\overline{r}' \tag{4.91}$$

Using the Fubini theorem one gets

$$Z_{ba} \approx j\mu c g(\overline{R}_b - \overline{R}_a) \sum_{l=1}^{4} [\int_{S_b} f_l(\overline{B}_b)(\overline{r})d\overline{r}][\int_{S_a} f_l(\overline{B}_a)(\overline{r})d\overline{r}] \tag{4.92}$$

In an iterative solver spirit, one wants to evaluate the product of the impedance matrix with a given vector. For the EFIE it means that one wants to test the electric field produced by a given current distribution on all the testing functions such as

$$(Z[\overline{J}])_k = \sum_{i=1}^{N} < \overline{B}_k|\overline{E}(\overline{B}_i) > \alpha_i =< \overline{B}_k|\overline{E}(\overline{J}) > \tag{4.93}$$

Therefore, from now on let us consider a more general situation with a test current distribution denoted by $\overline{T}$ in the local region B and a source current distribution denoted by $\overline{B}$ in the local region A.

This simple constant approximation is not satisfactory in the sense that the local Green's function error cannot be controlled efficiently. Indeed, the only way to control the error is to reduce the size of the local regions A and B which makes the method very limiting and a posteriori very inefficient. One must find a trick to preserve the local separability of the problem while improving the accuracy. The Taylor expansion can be used to achieve this goal. Indeed, the Taylor expansion expresses a given function as the sum of terms containing only monomial functions of the variables. The Taylor expansion of order n of the Green's function around a given point $\overline{X}$ is given by

$$g \approx \sum_{|\alpha|=0}^{n} \frac{1}{\alpha!} \frac{\partial^\alpha g}{\partial \overline{x}^\alpha}_{|\overline{X}} \overline{x}^\alpha \tag{4.94}$$

with $\alpha \in \mathbb{N}^3$, $\alpha! = \alpha_1\alpha_2\alpha_3$, $\overline{x}^\alpha = x_1^{\alpha_1}x_2^{\alpha_2}x_3^{\alpha_3}$. Now let us consider the approximation of the (6 variables) Green's function with this Taylor's expansion for two points $\overline{R}_a \in A$ and $\overline{R}_b \in B$, one ends with with

$$g_a(\overline{r}' - \overline{r}) \approx \sum_{|\alpha|=0}^{n} K^\alpha (\overline{r}' - \overline{r})^\alpha \Pi_B(\overline{r}')\Pi_A(\overline{r}) \tag{4.95}$$

where $K^\alpha \equiv \frac{1}{\alpha!}\frac{\partial^\alpha G_3}{\partial \overline{x}^\alpha}|_{\overline{R}_2 - \overline{R}_1}$. Because the difference between two vectors is a linear operation, injecting such an operation in the Taylor expansion does not harm the separability. Indeed, let us consider the following term $(r_i' - r_i)^{\alpha_1}$. Thanks to the Newton binomial formula this can be rewritten as follows

$$\begin{aligned}
(r_i' - r_i)^{\alpha_i} &= \sum_{k_i=0}^{\alpha_i} \begin{pmatrix} \alpha_i \\ k_i \end{pmatrix} (r_i')^{\alpha_i - k_i}(-r_i')^{k_i} && (4.96)\\
&= \sum_{k_i=0}^{\alpha_i} \begin{pmatrix} \alpha_i \\ k_i \end{pmatrix} (-1)^{k_i}(r_i')^{\alpha_i - k_i}(r_i)^{k_i} && (4.97)\\
& && (4.98)
\end{aligned}$$

Therefore, the Green's function approximation becomes

$$g(\overline{r}' - \overline{r}) \approx \sum_{|\alpha|=0}^{n} K^\alpha \sum_{k=0}^{\alpha} B^{\alpha,k} \prod_{i=1}^{3} (r_i')^{(\alpha_i - k_i)}(r_i)^{k_i} \Pi_B(\overline{r}')\Pi_A(\overline{r}) \tag{4.99}$$

with $\sum_{k=0}^{\alpha} \equiv \sum_{k_1=0}^{\alpha_1}\sum_{k_2=0}^{\alpha_2}\sum_{k_3=0}^{\alpha_3}$ and $B^{\alpha,k} \equiv \prod_{i=1}^{3} \begin{pmatrix} \alpha_i \\ k_i \end{pmatrix} (-1)^{k_i}$. Injecting this expression in the EFIE equation for a test current $\overline{T}$ into region A and a source current $\overline{J}$ in region B, one ends up with

$$< \overline{T}|\overline{E}(\overline{J}) > = j\mu c \sum_{l=1}^{4} \int_S \int_S \Big[\sum_{|\alpha|=0}^{n} K^{\alpha} \sum_{k=0}^{\alpha} B^{\alpha,k} \prod_{i=1}^{3} \Pi_B(\overline{r}') \Pi_A(\overline{r}) f_l(\overline{T})(\overline{r}') f_l(\overline{J})(\overline{r})(r_i)^{(\alpha_i - k_i)} (r_i')^{k_i} d\overline{r} d\overline{r}' \Big] \quad (4.100)$$

$$= j\mu c \sum_{l=1}^{4} \sum_{|\alpha|=0}^{n} K^{\alpha} \sum_{k=0}^{\alpha} B^{\alpha,k} \int_S f_l(\overline{T})(\overline{r})) \prod_{i=1}^{3} (r_i)^{k_i} \Pi_B(\overline{r}) d\overline{r} \int_S f_l(\overline{B})(\overline{r}) \prod_{i=1}^{3} (r_i)^{(\alpha_i - k_i)} \Pi_A(\overline{r}) d\overline{r} \quad (4.101)$$

In order to simplify this expression, let us define this following function that corresponds to the different moments of a given current $\overline{T}$ in region A

$$M_{l,k,A}(\overline{T}) \equiv \int_S f_l(\overline{T})(\overline{r}) \prod_{i=1}^{3} (r_i)^{k_i} \Pi_A(\overline{r}) d\overline{r} \quad (4.102)$$

where $k = (k_1, k_2, k_3)$. Now, the previous equation can be rewritten as follows

$$< \overline{T}|\overline{E}(\overline{J}) > = j\mu c \sum_{l=1}^{4} \sum_{|\alpha|=0}^{N} \sum_{k=0}^{\alpha} K^{\alpha} B^{\alpha,k} M_{l,k,A}(\overline{T}) M_{l,\alpha-k,B}(\overline{J}) \quad (4.103)$$

The local regions A and B correspond to respectively a local source region and a local test region, which must be separated by a reasonable distance in order to limit the bad behaviour of the Taylor expansion close to the singularity. Therefore, a dead region must be defined around a source region in which the brute force method is applied. Obviously, the dead region includes the source region. Therefore, in order to apply the approximation, the geometry of the whole MoM problem must be split into several local source domains, to which correspond respectively several local test domains. The field scattered by the current in the source domain is tested in the test region, where the approximation remains valid. The rest of the geometry where the approximation is not valid is called the dead zone. For this region, the impedance matrix has to be calculated.

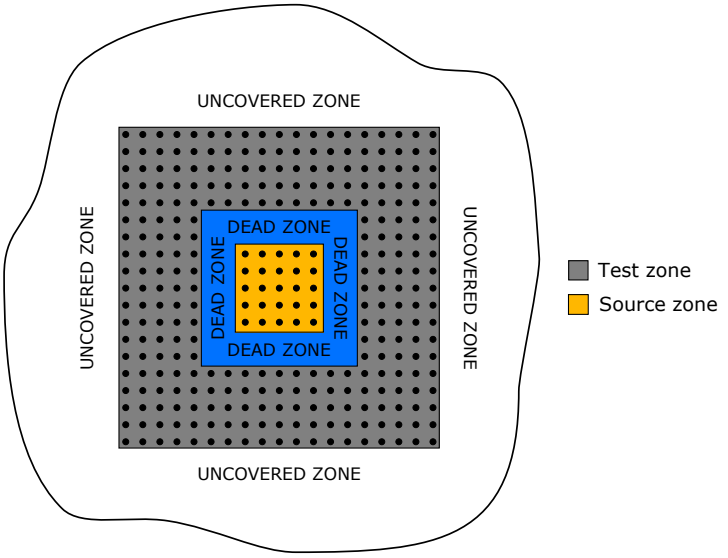


Figure 4.4: Definition of the different domains.

In order to make the solver more efficient, one defines a source domain by the union of several local source domains. Of course, doing so extends the dead region of a given local source domain which means more brute force computations. This is compensated by an improvement of the computation of Eq.4.103, as shown further.

Fig. 4.4 shows the source domain, the test domain and the dead zone domain definitions used here. The domains are rectangular parallelepipeds. The uncovered region is the region where the MLFMA can be applied and therefore there is no need to use the near-field Green's function approximation. The test zone is marked in gray and corresponds to whole the region covered by the Green's function approximation for any source current present in the source zone marked in green. The blue zone is the dead zone where the impedance matrix must be calculated in a classic way (brute force). The source domain and the test domain are composed of respectively the union of local test domains and local source domains. A local domain is defined by a rectangle parallelepiped. A single sampled point is placed at the center of each local domain. Fig. 4.4 shows all the sampling points used in the source domain and the test domain.

Let us consider a source region S and a test region T. Let us define the source sampling points by the function $\overline{S}_o : [1, ..., O] \rightarrow \mathbb{R}^3$ and the

test sampling points by the function $\overline{T}_p : [1, ..., P] \rightarrow \mathbb{R}^3$. Let us define the rectangular parallelepiped regions around each source sampling point by $B_o : [1, ..., O] \rightarrow P(\mathbb{R})$ and the rectangular parallelepiped regions around each test sampling point by $A_p : [1, ..., P] \rightarrow P(\mathbb{R})$ where $P(\mathbb{R})$ is the set of parts of $\mathbb{R}^3$. Any field radiated by a current distribution $\overline{S}$ in the source region and tested over a test current $\overline{T}$ in the test region can be reformulated as follows

$$< \overline{T}, \overline{E}(\overline{S}) > = < \sum_{p=1}^{P} \Pi_{A_p} \overline{T}, \overline{E}(\sum_{o=1}^{O} \Pi_{B_o} \overline{S}) > \quad (4.104)$$

$$= \sum_{p=1}^{P} \sum_{o=1}^{O} < \Pi_{A_p} \overline{T}, \overline{E}(\Pi_{B_o} \overline{S}) > \quad (4.105)$$

Now, the final result of our former section can be used. Let us define $G_{p,o}^{\alpha} = \frac{1}{\alpha!} \frac{\partial^{\alpha} g}{\partial x^{\alpha}}_{|\overline{T}_p - \overline{S}_o}$. One has

$$< \overline{T}, \overline{E}(\overline{S}) > \approx j\mu c \sum_{o=1}^{O} \sum_{p=1}^{P} \sum_{l=1}^{4} \sum_{|\alpha|=0}^{n} G_{p,o}^{\alpha} \sum_{k=0}^{\alpha} B^{\alpha,k} M_{l,k,A_p}(\overline{T}) M_{l,\alpha-k,B_o}(\overline{S}) \quad (4.106)$$

Since the number of points $M + P$ is far larger than n, let us swap the sums and focus on the double sum $\sum_{o=1}^{O} \sum_{p=1}^{P}$

$$< \overline{T}, \overline{E}(\overline{S}) > \approx j\mu c \sum_{l=1}^{4} \sum_{|\alpha|=0}^{n} \sum_{k=0}^{\alpha} B^{\alpha,k} \sum_{p=1}^{P} M_{l,k,A_p}(\overline{T}) \sum_{o=1}^{O} G_{p,o}^{\alpha} M_{l,\alpha-k,B_o}(\overline{S}) \quad (4.107)$$

In order to speed up the evaluation of this expression, a truncated SVD decomposition [52] can be performed on the matrix $[G^{\alpha}]_{po} = G_{p,o}^{\alpha}$. The size of the matrix is reduced by selecting the singular values larger than certain given threshold. The SVD decomposition theorem states that any matrix $M \in \mathbb{C}^{N \times M}$ can be decomposed into a product of three matrices $M = USV$ with $U \in \mathbb{C}^{N \times N}$, $S \in \mathbb{R}^{N \times M}$, $V \in \mathbb{C}^{M \times M}$ and where U, V are two normal matrices and S a diagonal real positive matrix. It can be re-interpreted

through an alternative formulation. Let $\{c_i\}_{i\in[1,...,N]} \in \mathbb{C}^N$ be the set of columns of U. Let $\{l_k\}_{k\in[1,...,M]} \in \mathbb{C}^M$ be the set of lines of V. The SVD decomposition is equivalent to

$$M = \sum_{i=1}^{\min(N,M)} c_i S_{ii} < l_i| > \tag{4.108}$$

where $< l_i| >: \mathbb{C}^M \to \mathbb{C}$ is the linear form built from the canonical Hermitian product namely

$$< x|y > \equiv \sum_{i=1}^{N} a_i b_i^* \text{ with } x, y \in \mathbb{C}^N \tag{4.109}$$

The vectors c_i and l_i constitute respectively an orthonormal basis of $\mathbb{C}^N$ and $\mathbb{C}^M$ for the canonical Hermitian product. Let us approximate the matrix by selecting the P singular values such that $S_{ii} > \epsilon$ where $\epsilon > 0, \epsilon \in \mathbb{R}$ is a given threshold value. The approximate matrix is given by the following expression

$$M_a \equiv \sum_{i=1}^{P} c_i S_{ii} < l_i| > \tag{4.110}$$

This expression can be rewritten as the product of two matrices. Let us define $U_r \equiv [c_1, ..., c_p] \in \mathbb{C}^{M\times P}$ and $V_r = [S_{11}l_1; ...; S_{PP}l_P] \in \mathbb{C}^{P\times N}$ then the approximate matrix reads

$$M_a = U_r V_r \tag{4.111}$$

An upper bound for the error undergone in the image space can be easily calculated. $\forall v \in \mathbb{C}^M$ one has

$$||(M_a - M)v||_2 = || \sum_{i=P+1}^{\min(N,M)} c_i S_{ii} < l_i|v > ||_2 \tag{4.112}$$

$$= \sum_{i=P+1}^{\min(N,M)} S_{ii}^2 < l_i|v >^* < l_i|v > \tag{4.113}$$

$$\leq \sum_{i=P+1}^{\min(N,M)} S_{ii}^2 ||v||_2^2 \tag{4.114}$$

with $||.||_2 : \mathbb{C}^N \to \mathbb{R}^+, x \to ||x|| = \sqrt{\sum_{i=1}^{N} x_i x_i^*}$ the canonical norm.

The SVD decomposition is of complexity $O(N^3)$. However, it is important to notice that the compression of matrices G^α needs to be performed only once for any geometry and any frequency. This will become clearer in the next section. The matrices can be rewritten as follows $G^\alpha = U_r^\alpha V_r^\alpha$ where U_r^α and V_r^α are the reduced form produced by the SVD reduction method explained aforesaid. The equation 4.107 with the SVD reduction becomes

$$< \overline{T}, \overline{E}(\overline{S}) >= j\mu c \sum_{l=1}^{4} \sum_{|\alpha|=0}^{n} \sum_{k=0}^{\alpha} B^{\alpha,k} M_{l,k}^t(\overline{T}) U_r^\alpha V_r^\alpha M_{l,\alpha-k}^s(\overline{S}) \quad (4.115)$$

where $M_{l,k}^t(\overline{T})_o \equiv M_{l,k,A_o}(\overline{T})$ and $M_{l,k}^s(\overline{S})_o \equiv M_{l,k,B_o}(\overline{S})$.

4.4.2.3 Global separable Green's function approximation on a predefined grid

In this section, the Local Near-Field Interaction Method (LNFIM) described in the former section is used to perform of full EFIE simulation. At each iteration step, the iterative solver requires the test of a given current distribution on every testing functions. In order to use the LNFIM, the geometry is divided into different source regions. The source region size is predefined in order to be able to pre-calculated the compressed matrices. A uniform grid that covers the structure is first generated. The space covered by all the source regions is contained inside a compact subset $D \subset \mathbb{R}^3$. The grid is generated in the way that each source domain contains the same amount of sampling points. Around each sampling point, a rectangular parallelepiped defines the local region of validity of the Green's function approximation as explained in the former section and as shown Fig. 4.5. Each local region of validity of the Green's function approximation (the region of space around a sampling point) is indexed by a unique and global index. Let $D_r : [1, ..., R] \to P(\mathbb{R}^3)$ be the function that represents these regions where $[1, ..., R] \subset \mathbb{N}$ and $P(\mathbb{R}^3)$ is the set of parts of $\mathbb{R}^3$. This global index is also used to point to the center of the region which is a by definition a rectangular parallelepiped. Let $\overline{R}_k : [1, ..., R] \to \mathbb{R}^3$ be the center of these regions. Each source domain is given by $S_{g,s} : [1, ..., S] \to P(\mathbb{R}^3)$ where $[1, ..., S] \subset \mathbb{N}$, $S < R$. For each source domain corresponds a test domain and a dead zone domain given respectively by $T_{g,s} : [1, ..., S] \to \mathbb{R}^3$ and $DZ_{g,s} : [1, ..., S] \to \mathbb{R}^3$. One has the following property associated to these functions

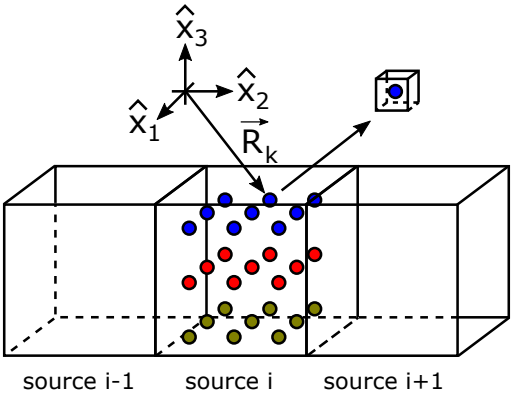


Figure 4.5: The source domains with their sampling points

$$\mu(S_{g,s_1} \bigcap S_{g,s_2}) = 0 \qquad \text{if } s_1 \neq s_2 \tag{4.116}$$

$$\bigcup_{s\in[1,\ldots,S]} S_{g,s} = D \tag{4.117}$$

$$\mu(D_{r_1} \bigcap D_{r_2}) = 0 \qquad \text{if } r_1 \neq r_2 \tag{4.118}$$

$$\forall S_{g,s}, \exists A \subset \mathbb{N} \text{ such that } S_{g,s} = \cup_{r\in A} D_r \tag{4.119}$$

$$\forall T_{g,s}, \exists A \subset \mathbb{N} \text{ such that } T_{g,s} = \cup_{r\in A} D_r \tag{4.120}$$

$$\forall DZ_{g,s}, \exists A \subset \mathbb{N} \text{ such that } DZ_{g,s} = \cup_{r\in A} D_r \tag{4.121}$$

where $\mu : P(\mathbb{R}^3) \to \mathbb{R}$ is the Lebesgue measure. Now, let us rewrite the EFIE equation

$$< \overline{T}|\overline{E}(\overline{S}) > = < \overline{T}|\overline{E}(\sum_{s=1}^{S} \Pi_{S_{g,s}} \overline{S}) > \tag{4.122}$$

$$= \sum_{s=1}^{S} < \overline{T}|\overline{E}(\Pi_{S_{g,s}} \overline{S}) > \tag{4.123}$$

$$= \sum_{s=1}^{S} < \Pi_{(T_{g,s} \cup DZ_{g,s})} \overline{T}|\overline{E}(\Pi_{S_{g,s}} \overline{S}) > \tag{4.124}$$

$$= \sum_{s=1}^{S} < \Pi_{T_{g,s}} \overline{T}|\overline{E}(\Pi_{S_{g,s}} \overline{S}) > + \sum_{s=1}^{S} < \Pi_{DZ_{g,s}} \overline{T}|\overline{E}(\Pi_{S_{g,s}} \overline{S}) > \tag{4.125}$$

Let us defined two bilinear functions

$$I_{s,1}(\overline{T}, \overline{S}) = < (\Pi_{T_{g,s}} \overline{T}|\Pi_{S_{g,s}} \overline{E}(\overline{S}) > \tag{4.126}$$

$$I_{s,2}(\overline{T}, \overline{S}) = < (\Pi_{DZ_{g,s}} \overline{T}|\Pi_{S_{g,s}} \overline{E}(\overline{S}) > \tag{4.127}$$

$$I_{s,1}(\overline{T}, \overline{S}) = \sum_{s \in [1,\ldots,S]} \sum_{l=1}^{4} \int_S \int_S f_l(\overline{T})(\overline{r}') f_l(\overline{S})(\overline{r}) G(\overline{r}', \overline{r}) S_{g,s}(\overline{r}) T_{g,s}(\overline{r}') d\overline{r} d\overline{r}' \tag{4.128}$$

$$I_{s,2}(\overline{T}, \overline{S}) = \sum_{s \in [1,\ldots,S]} \sum_{l=1}^{4} \int_S \int_S f_l(\overline{T})(\overline{r}') f_l(\overline{S})(\overline{r}) G(\overline{r}', \overline{r}) S_{g,s}(\overline{r}) DZ_{g,s}(\overline{r}') d\overline{r} d\overline{r}' \tag{4.129}$$

The first integral is computed with the fast near filed interaction method, while the second integral is computed with the classic brute force scheme.

The current distribution is given by a linear combination of RWG basis functions. For obvious reasons, cutting a basis function between two integration domains cannot be performed easily. By assumption, the basis function are smallers than a domain D_r (there exists a sphere that contains the basis function and that can be contained in any domain D_r). Therefore it seems

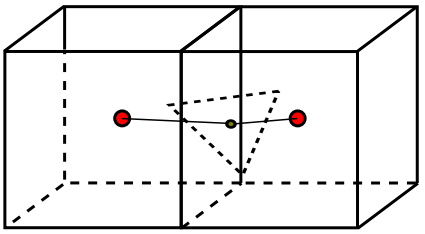


Figure 4.6: Association of the half basis function to a region

reasonable to associate a basis function to a unique domain. In order to improve this procedure, instead of working directly on RWG, we decided to work on the half RWG basis functions i.e currents defined on a triangles. Each half basis function (each triangle) is associated uniquely to domain D_r. Each triangle is associated to the region D_r for which its barycentric center is the closest to the barycentric center of region. This is illustrated Fig. 4.6. The half basis function are denoted by $\overline{H}_i : [1, ..., N_h] \rightarrow \mathbb{R}^3$ and any current distribution is given by $\overline{H} = \sum_{i=1}^{N_h} \beta_i \overline{H}_i$. By linearity, one can write

$$< \overline{H}_v | \overline{E}(\sum_{i=1}^{N_h} \beta_i \overline{H}_i) > = \sum_{i=1}^{N_h} < \overline{H}_v | \overline{E}(\overline{H}_i) > \beta_i \tag{4.130}$$

$$= \sum_{s=1}^{S} \sum_{i=1}^{N_h} (I_{s,1}(\overline{H}_v, \overline{H}_i) + I_{s,2}(\overline{H}_v, \overline{H}_i))\beta_i \tag{4.131}$$

Let us focus back on the first integration $I_{s,1}$. Each set of domains $(S_{g,s}, T_{g,s}, DZ_{g,s})$ correspond to a translation of the source,test and dead zone model presented in the former section. A mapping between the two can be performed. Let us define the matrix $I_{S,s} : \mathbb{C}^R \rightarrow \mathbb{C}^O$ the linear application that sends each point of the source domain $S_{g,s}$ to the corresponding point of the source domain of the model. Similarly, we define $I_{T,s} : \mathbb{C}^R \rightarrow \mathbb{C}^P$ the linear application that sends each point of the test domain $T_{g,s}$ to the test domain of the model. Let $M_{l,k,r,i} \equiv M_{l,k,D_r}(\overline{J}_i) = \int_S f_l(\overline{J}_i)(\overline{r}) \prod_{m=1}^{3} (r_m)^{k_m} \Pi_{D_r}(\overline{r}) d\overline{r}$ be the global moments vector. The product $I_{s,1}\beta$ can be expressed as

$$I_{s,1}\beta = j\mu c \sum_{l=1}^{4} \sum_{|\alpha|=0}^{n} \sum_{k=0}^{\alpha} B^{\alpha,k} M_{g,l,k} I_{T,s}^{t} U_3^{\alpha} V_3^{\alpha} I_{S,s} M_{g,l,\alpha-k} \beta \tag{4.132}$$

where $I_{S,s}$ and $I_{T,s}$ are the matrix form of the respectively $I_{S,s}$ and $I_{T,s}$ for their canonical basis. The matrices $I_{S,s}$, $M_{l,\alpha-k}$ and $I_{T,s}^{t}$ are sparse matrices. The number of non-zero elements in $M_{l,\alpha-k}$ is equal to N. The matrices U_3^{α} and V_3^{α} are the compressed matrices that remain of small fixed dimension. The contribution of all source domains to the matrix-vector product is given by

$$\sum_{s=1}^{S} I_{s,1}\beta = j\mu c \sum_{l=1}^{4} \sum_{|\alpha|=0}^{n} \sum_{k=0}^{\alpha} B^{\alpha,k} M_{g,l,k} \sum_{s=1}^{S} I_{T,s}^{t} U_3^{\alpha} V_3^{\alpha} I_{S,s} M_{l,\alpha-k} \beta \tag{4.133}$$

The second integral requires the brute force computation of the impedance matrix. However, the number of half basis and test functions in the source and dead-zone domain remains small in comparison to the overall number of half basis and test functions.

4.4.2.4 Complexity of the method

As shown in the former section, the matrix-vector product of the impedance matrix with a source current is given by

$$< \overline{H}_v | \overline{E}(\sum_{i=1}^{N_h} \beta_u \overline{H}_i) > = \sum_{i=1}^{N_h} < \overline{H}_v | \overline{E}(\overline{H}_i) > \beta_i \tag{4.134}$$

$$= \sum_{s=1}^{S} \sum_{i=1}^{N_h} (I_{s,1}(\overline{H}_v, \overline{H}_i) + I_{s,2}(\overline{H}_v, \overline{H}_i))\beta_i \tag{4.135}$$

It is assumed that the source, dead-zone and test domains are of a fixed size. The complexity under this general hypothesis is evaluated for two different assumptions. First, let us assume that the minimum size of any mesh cell is of fixed size. In this situation, the number of domains grows in the worse case as $O(N_h)$ with N_h being the number of half basis functions

(i.e twice the number of RWG basis functions). Let us investigate the first integral namely

$$\sum_{s=1}^{S}[I_{s,1}][\beta] = j\mu c\sum_{l=1}^{4}\sum_{|\alpha|=0}^{n}\sum_{k=0}^{\alpha}B^{\alpha,k} M_{l,k}\sum_{s=1}^{S}I_{T,s}^{t}U_3^{\alpha}V_3^{\alpha}I_{S,s}M_{l,\alpha-k}\beta \tag{4.136}$$

Let us define the intermediate results of this equation and calculate their complexities

$$\beta_{l,\alpha,k} \equiv M_{l,\alpha-k}\beta \qquad O(N_h) \tag{4.137}$$
$$\beta_{s,l,\alpha,k} \equiv I_{S,s}\beta_{l,\alpha,k} \qquad O(1) \tag{4.138}$$
$$\beta_{s,l,\alpha,k,1} \equiv V_3^{\alpha}\beta_{s,l,\alpha,k} \qquad O(1) \tag{4.139}$$
$$\beta_{s,l,\alpha,k,2} \equiv U_3^{\alpha}\beta_{s,l,\alpha,k,1} \qquad O(1) \tag{4.140}$$
$$\beta_{s,l,\alpha,k,3} \equiv I_{T,s}^{t}\beta_{s,l,\alpha,k,2} \qquad O(1) \tag{4.141}$$
$$\gamma_{l,\alpha,k} \equiv \sum_{s=1}^{S}\beta_{s,l,\alpha,k,3} \qquad O(S)O(N_h) \tag{4.142}$$
$$\gamma_{l,\alpha,k,2} \equiv B^{\alpha,k}M_{l,k}\gamma_{l,\alpha,k} \qquad O(N_h) \tag{4.143}$$
$$\gamma_{l,\alpha,k,3} \equiv j\mu c\sum_{l=1}^{4}\sum_{|\alpha|=0}^{n}\sum_{k=0}^{\alpha}\gamma_{l,\alpha,k,2} \qquad O(N_h)O(f(n)) \tag{4.144}$$

The number of terms used in the Taylor series is directly a function of the desired accuracy. However, this number is independent of the number of unknowns. Therefore, the complexity of the computation of this integral is given by $(O(N_h)+4O(1)O(S)+O(N_h)O(S)+O(N_h))O(f(n)) = O(N_h)O(S)O(f(n)) = O(N_h^2)$. Let us assume that the number of unknowns per source domain is simply given by $n_s \equiv \frac{N_h}{S}$. Since the number of domains is assumed to grow linearly with the number of unknowns, the number of basis functions in each source domain is constant. This reasoning holds for the dead-zone region. Therefore, the complexity for the brute force integral is given by $O(N_h)$. Therefore, the overall method is of quadratic complexity $O(N_h^2)$.

Now, let us assume that the volume containing the whole geometry is fixed. In this case, the number of source domains is fixed. The complexity for the integral given by Eq. 4.136 is given by $(O(N_h) + 4O(1)O(S) +$

$O(N_h)O(S) + O(N_h))O(f(n)) = O(N_h)O(S)O(f(n)) = O(N_h)$. The number of unknowns per source domain is still given by $n_s \equiv \frac{N_h}{S}$ by since S is constant the complexity of the brute force for the dead-zone domains is $O(N_h^2)$. Again, the method under this assumption is of quadratic complexity $O(N_h^2)$.

The method is stuck with a quadratic complexity because of the fixed size of the source domains. Changing the source domain size implies a variability in the SVD compression which might be different for each approximation matrix associated to a specific Taylor order. However, if one can prove that if the SVD compression remains the same under a homothetic transformation then a multilevel scheme could be considered. Using a multilevel scheme still requires an efficient computation of the moments for different grid sizes.

4.5 Preconditioner based on Macro Basis Function

The Macro Basis Function (MBF) [53], [54], [55], [56], [57], [58] concept consists simply in isolating a section of the geometry, that one calls macro domain, and calculate a set of MoM solutions of this isolated geometry for different excitations. This set of solutions constitutes the new set of basis functions for the isolated section that are used to solve the whole MoM problem. Recalling that the low-frequency-breakdown preconditionner presented in section 4.3 does not improve the condition number due to the variation of basis functions size. Solvers based on MBFs suffer less from this issue if only few basis functions per domain are required and if the domains are relatively large in comparison to the geometry.

Because the geometry is connected, the macro domains must overlap each-other in order to ensure the continuity of current between macro domains. In order to generate macro currents, an external excitation should be provided for each of them. The principle of the so-called primary and secondary currents approached is to re-use the macro currents calculated as a source scatterer for the other macro domains. The first current calculated is called the primary current. It is generated by the macro domain that contains the source field. If the source field cannot be isolated in a given macro domain, several primary currents can be generated in several macro domains. Fig. 4.7 illustrates the primary and secondary currents generation process.

The MBF solver have been tested on the accelerator but many MBFs had to be generated in each domains in order to have a low relative residual error.

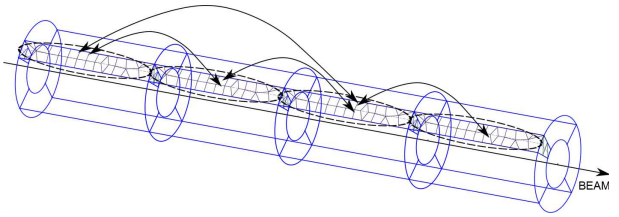


Figure 4.7: Macro currents generation based on the primary and secondary currents method

Therefore, this solver has not been investigated any further. We believe the problem comes from the connected structure. MBFs have shown a very good behaviour in an array of antenna made of identical elements when the elements are separated of a distance larger than a fifth of wavelength.

4.6 Numerical results

4.6.1 Accelerator brute force simulations

The accelerator has been solved using the MoM and a direct inversion of the impedance matrix. The number of unknowns for the whole accelerator is 70000. The impedance matrix takes 78400 MB which is too large for most of the computers. Therefore, a block inversion solver swapping data with the disk has been used. It has been implemented in way that limits the transfers of data between the DRAM and the disk.

Fig. 4.8 shows the whole accelerator viewed from the top. The particles come in from the right. The feeding is performed with a direct bias at the injection level. In practice, the accelerator is fed with the help a coupler. Unfortunately, no geometry design file of this coupler has been provided. Therefore, we ended up with the direct bias alternative that seems reasonable in terms of time of implementation and of physical accuracy. The frequency of the simulation is 176 MHz and the voltage applied between two neighbouring rods is 30 kV. The results presented here concern only the electromagnetic fields. Simulations related to beam dynamics are presented in the chapter devoted to it.

Fig. 4.9 shows the current distribution in direction $-\hat{x}_3$ in the rods. One can clearly see that two opposite rods have a current flowing in the same direction. This is consistent with the fact that two opposite rods are at the same potential and with the symmetry of the design. Fig. 4.10 shows the electric field in a cross section of the accelerator. The quadrupole symmetry

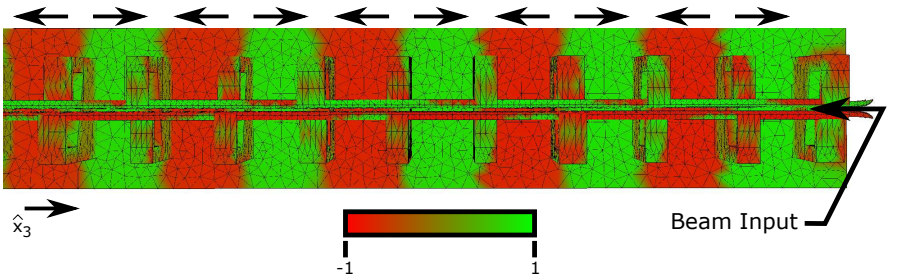


Figure 4.8: Current distribution in $\hat{x}_3$ direction for a section of the accelerator.

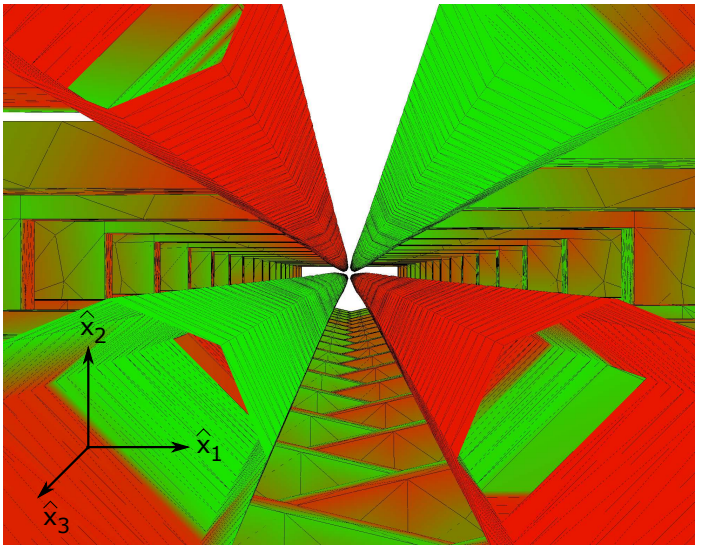


Figure 4.9: Current distribution in $\hat{x}_3$ direction in the rods.

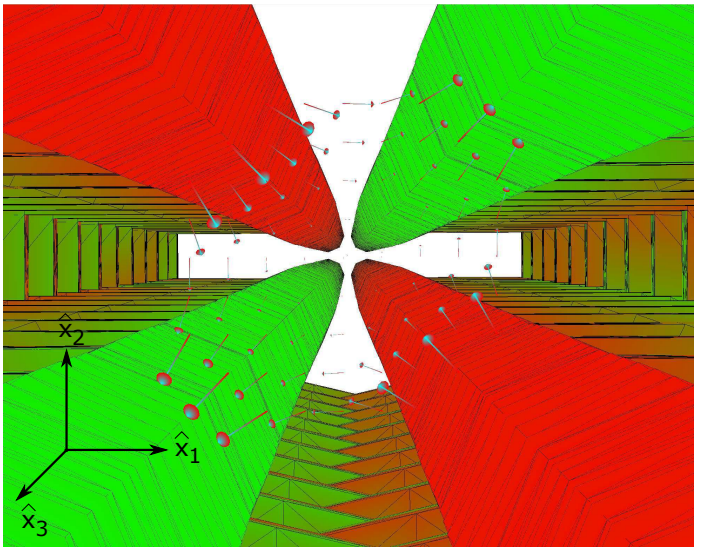


Figure 4.10: Electric field in a cross section of the accelerator.

is clearly visible in the electric field.

4.6.2 Low-Frequency preconditioner

The preconditioner has been tested without the help a the fast near field solver. Since the impedance matrix of the whole accelerator is too big to fit in DRAM and because swapping the block of the whole matrix with the disk for a GMRES solution takes a cumbersome amount of time, a section of the accelerator has been used to test it. The frequency of the simultion is 176MHz and the size of the section considered here is around 10 cm. Fig. 4.11 shows the mesh of the accelerator used for this test. The solution has been calculated for the EFIE with and without the preconditioner. For both computations, the GMRES convergence has been recorded and compared. The relative residual accuracy chosen in both simulations is 10^{-6}. Fig. 4.12 shows the convergence rate of these two computations. Clearly, GMRES converges much faster when the EFIE is combined with the preconditioner. Finally, Fig. 4.13 shows the current distribution in the direction $\hat{x}_3$ (recalling that $-\hat{x}_3$ is the direction of acceleration of the beam). The current amplitude is normalized such that it lives in the interval $[-1, 1]$. The pure red color represents a normalized current of amplitude one in $-\hat{x}_3$ while the pure green color represents a normalized current of amplitude one in $\hat{x}_3$. Clearly, one can see the current flowing between two rods and two stems of

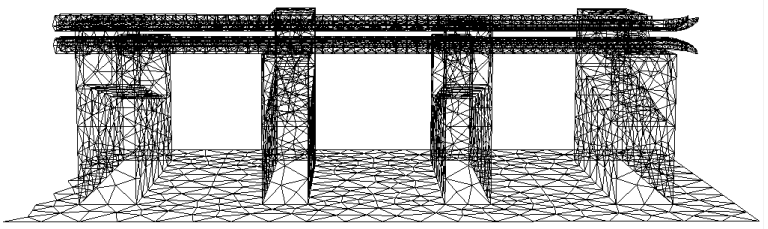


Figure 4.11: Accelerator mesh used for the low-frequency preconditioner test.

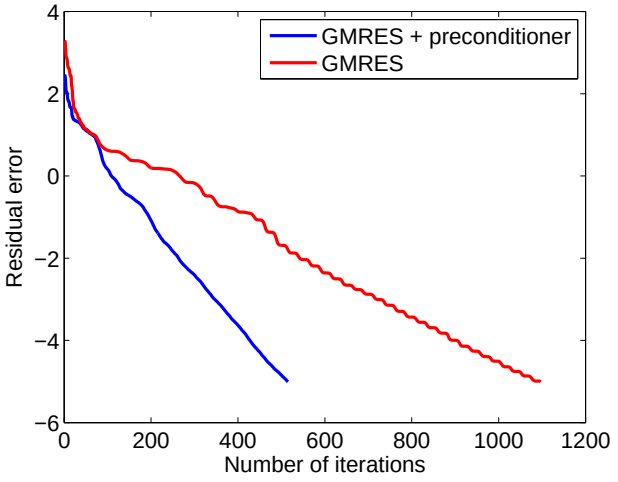


Figure 4.12: Comparison of the GMRES convergence rate between the non-preconditioned impedance matrix and the preconditioned impedance matrix.

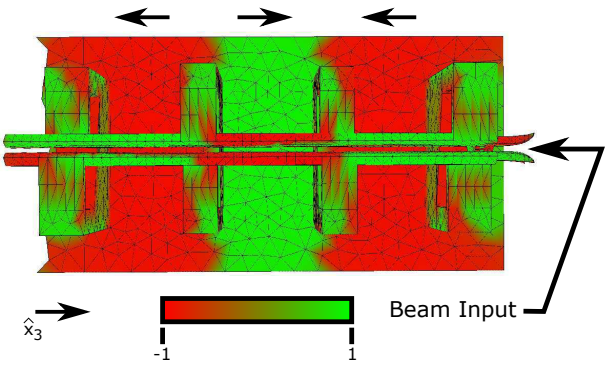


Figure 4.13: Component of the current distribution in $\hat{x}_3$ at a given time obtained with the Low-Frequency-Breakdown preconditioner.

opposite signs.

4.6.3 Fast Near field fast MoM solver simulations

In this section, simulation results using the Fast Near Field MoM Solver (FNFMMS) combined with the low-frequency preconditioner are presented. The method has been tested on a sphere of radius one and for a frequency of 176MHz. The excitation is provided by a plane wave irradiating the whole structure. The method has been tested for different combinations of source and dead-zone domains sizes. The test domain size was chosen such that the whole sphere was covered by the sampling points of the grid for any source domain. Two distances between sampling points have been tested namely a tenth of the wavelength and a twentieth of the wavelength. A Taylor order up to 3 was used. The different domain size given as multiple of grid step will be denoted by XxYxZ, where X represents the source domain size, Y the dead-zone domain size and Z the test domain size.

19000 RWG basis functions were used in this simulation. The spectrum of the impedance matrix could be still evaluated and is presented in Fig. 4.14. One can clearly see the step in the spectrum due to the relatively small overall object size in comparison to the wavelength. Therefore, it is expected that the low-frequency preconditioner is going to be of a great help to improve the convergence speed of GMRES. In order to improve the convergence speed of the solver when it is used with Taylor approximation of order n, the solution for a Taylor approximation at order $n-1$ is used as an initial guess solution. The accuracy of the solution for a Taylor approximation of order $n-1$ should be chosen smaller than the one for the

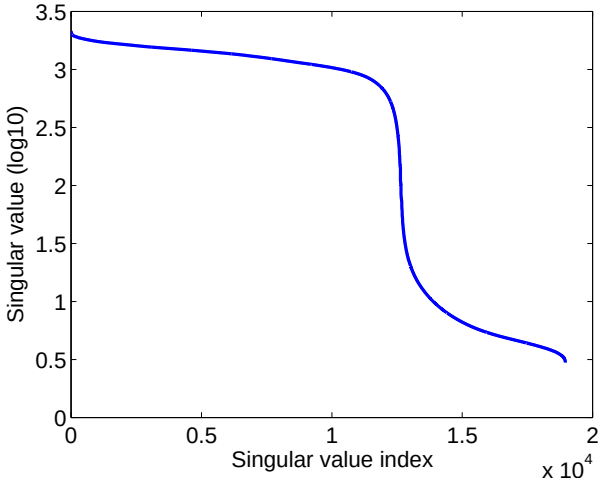


Figure 4.14: Spectrum of the impedance matrix.

solution for a Taylor approximation of order n as the results of simulations presented hereunder suggest.

The relative current error per basis functions will be presented in the following subsections. It is defined by

$$error \equiv \frac{|\alpha_i - \alpha'_i|}{\max_{i\in[1,\ldots,N]}(|\alpha_i|)} \tag{4.145}$$

where α_i are the coefficients of the brute force solution, α'_i the coefficients of the solution calculated with the FNFMMS solver (with or without the low-frequency preconditioner) and N the number of RWG basis functions. The brute force method is calculated as follows: The impedance matrix is obtained with a direct method of complexity $O(N^2)$ and the solution is obtained with GMRES. Similarly, the relative excitation error per basis functions will be also presented. It is defined by

$$error \equiv \frac{|\beta_i - \beta'_i|}{\max_{i\in[1,\ldots,N]}(|\beta_i|)} \tag{4.146}$$

where $\beta = Z\alpha$, $\beta' = Z\alpha'$, $\beta_i = (\beta)_i$, $\beta'_i = (\beta')_i$ and Z the brute force impedance matrix. It is important to keep in mind that this error can be presented because an accurate impedance matrix has been calculated with a brute force method.

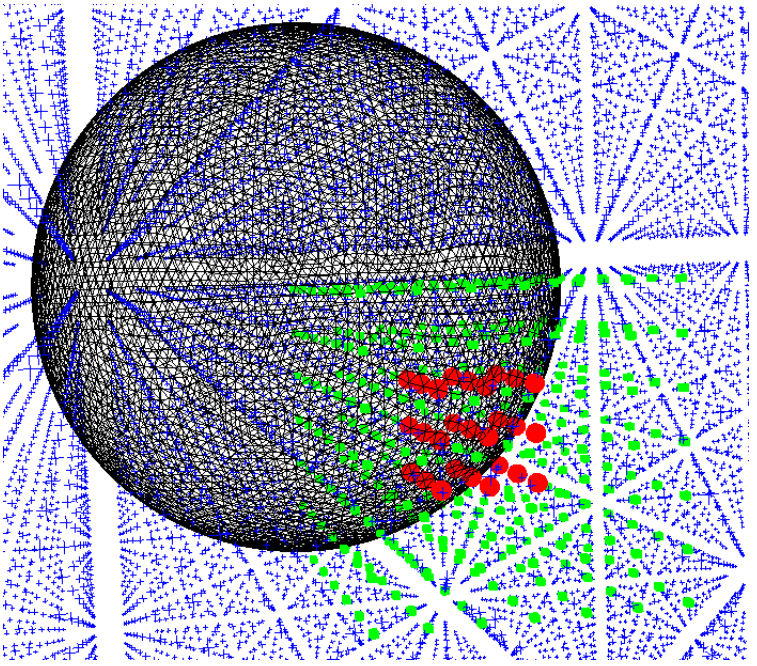


Figure 4.15: Presentation of a source domain (red) and its associated dead-zone (green) and test domains (blue).

4.6.3.1 $\frac{\lambda}{10}$ grid step, Taylor order 3, SVD compression 10^{-4} and grid size 3x7x31

Fig. 4.15 shows a given source domain in red and its associated dead-zone domain in green and the test domain in blue. The number of source domains require to cover the whole structure is 74.

The FNFMMS has been tested with and without the low-frequency preconditioner. Fig. 4.16 shows the GMRES convergences for the different orders of Taylor approximation for the Green's function between grid points. The accuracy of the relative residual error is chosen identical for the different orders. The relative error accuracy was set to 10^{-5}. Fig. 4.17 shows the relative current error per basis function index. The relative error on the excitation is presented Fig. 4.18. The same simulations were performed using the low-frequency preconditioner. Fig. 4.19 shows the GMRES convergences. Clearly, GMRES converges at any Taylor order much faster that when the system of equations is solved without any preconditioner. Fig. 4.20 and Fig. 4.21 respectively show the relative current error and the rela-

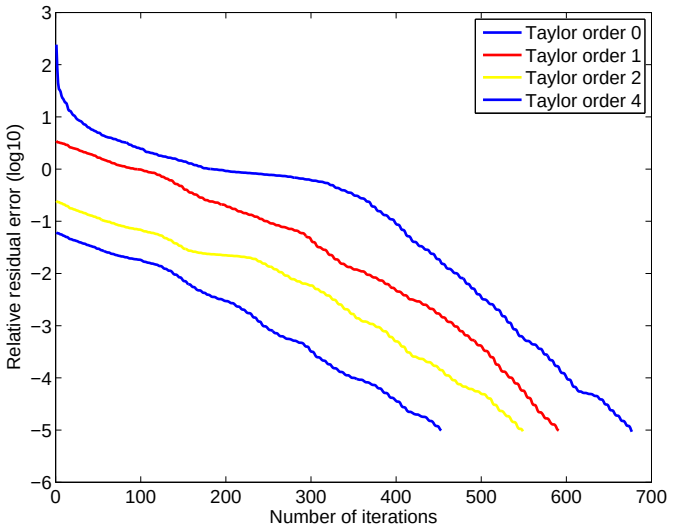


Figure 4.16: Convergence of GMRES without the low-frequency preconditioner - The accuracy for each order is set to the same value of 10^{-5}.

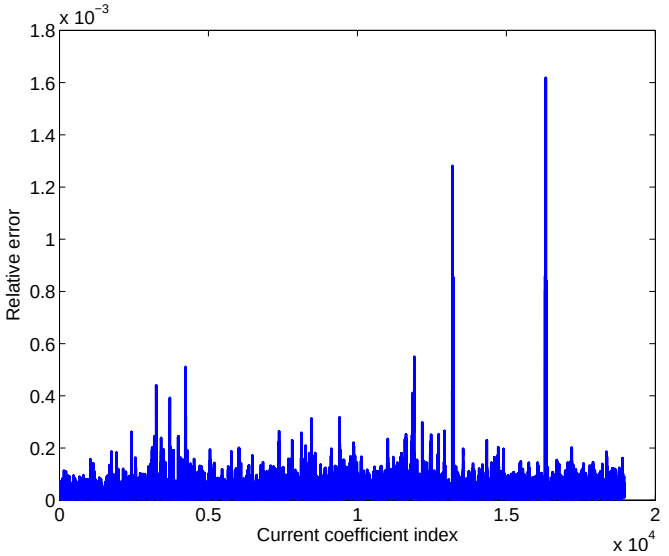


Figure 4.17: Relative current error between the brute force solution and the FNFMMS solver without any low-frequency preconditioner.

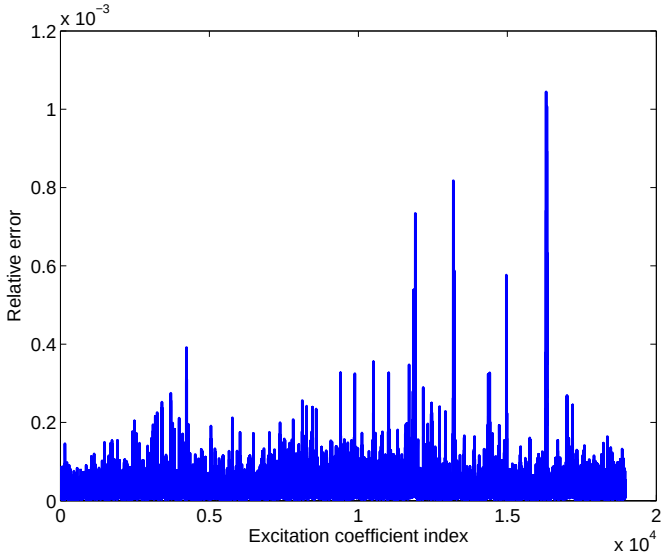


Figure 4.18: Relative excitation error between the brute force solution and the FNFMMS solver without any low-frequency preconditioner.

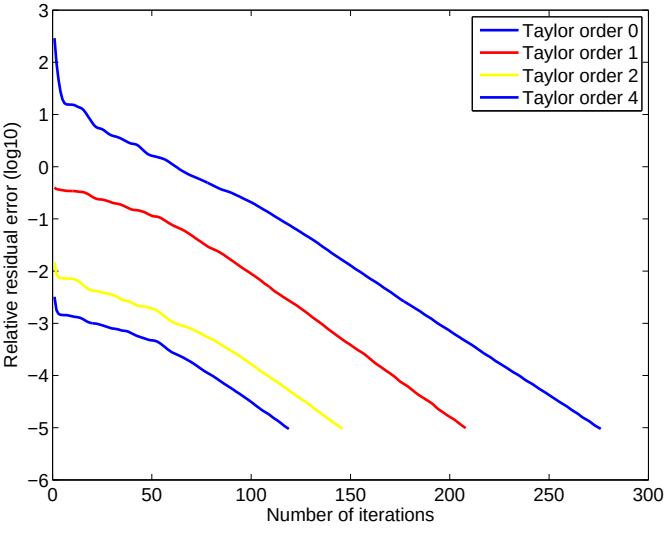


Figure 4.19: Convergence of GMRES with the low-frequency preconditioner - The accuracy for each order is set to the same value of 10^{-5}.

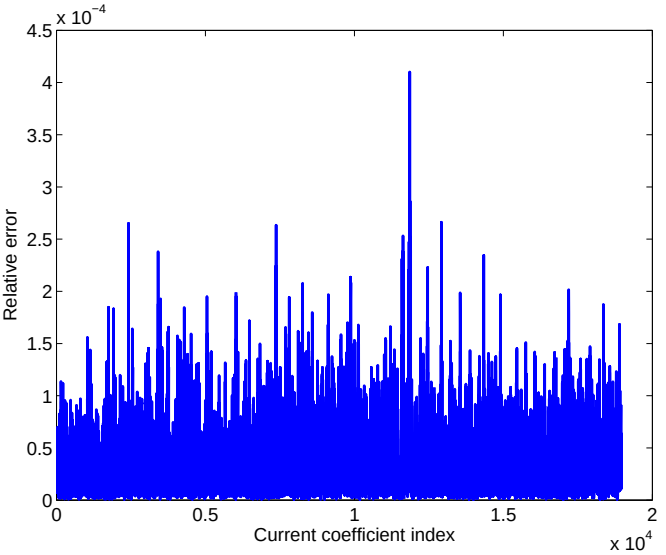


Figure 4.20: Relative current error between the brute force solution and the FNFMMS solver solution with the low-frequency preconditioner.

tive excitation error. The accuracy is slightly better than the one obtained without the low-frequency preconditioner. The computer time required for the GMRES solver using all the Taylor approximation orders is not better than using right away the Taylor approximation of the highest order. Instead of using the same accuracy for each order, the accuracy has been decreased for lower-order approximations as follows

$$error_n = \frac{error_{n-1}}{10} \tag{4.147}$$

$$error_3 = 10^{-5} \tag{4.148}$$

Fig. 4.22 shows the GMRES convergences for this relative residual error criterion.

Tab. 4.1 shows the GMRES computation time for each order of Taylor approximation. Interestingly, the computational time required for the third order is lower for the FNFMMS preconditioned v2 than for the classic FNFMMS preconditioned. The brute force solution with the low-frequency solver has been calculated as well to show the improvement of GMRES convergence on the accurate impedance matrix.

Tab. 4.2 shows the total computational time for each method.

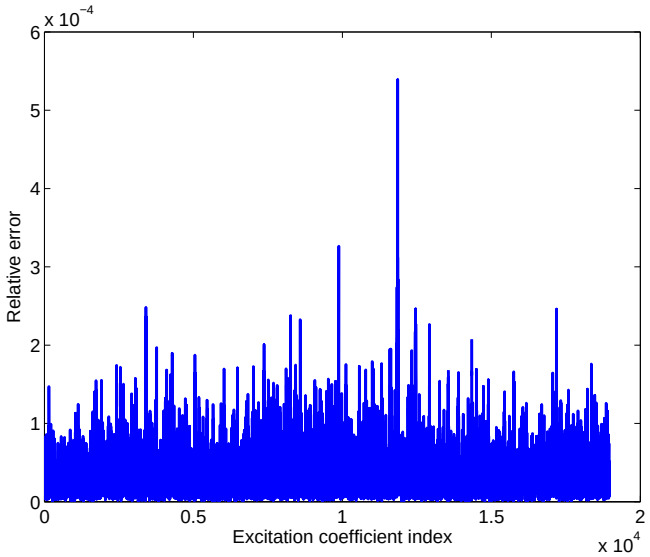


Figure 4.21: Relative excitation error between the brute force solution and the FNFMMS solver solution with the low-frequency preconditioner

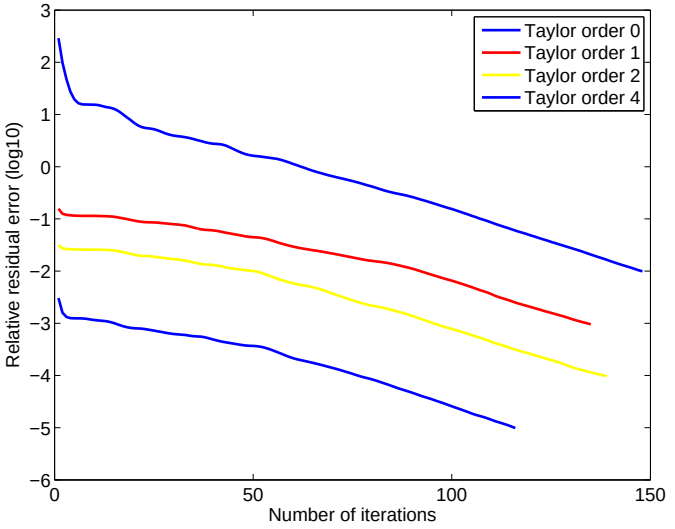


Figure 4.22: Convergence of GMRES with the low-frequency preconditioner - The accuracy for each Taylor order approximation is set according to the criterion defined by Eq. 4.147 and Eq. 4.148.

type \ Taylor order	0	1	2	3
FNFMMS Non preconditioned	250s	305s	602s	1211s
FNFMMS Preconditioned	109s	102s	213s	439s
FNFMMS Preconditioned v2	49s	68s	154s	315s

Table 4.1: GMRES Convergence times

	GMRES	DeadZone	Total
FNFMMS Non preconditioned	2368s	324s	2692s
FNFMMS Preconditioned	863	324s	1187s
FNFMMS Preconditioned v2	586	324s	910s
Brute Force	229s	2784s	3013s
Brute Force Preconditioned	122s	2784s	2907s

Table 4.2: Comparison of total simulation times

Chapter 5

GPU Programming

5.1 GPGPU programming and challenges

A Graphical Processor Unit (GPU) is a device used in a personal computer (PC) that allows fast calculation of 3D rendering scenes. In order to understand why a specific hardware has been built for 3D rendering, one must first understand the principle of rendering. This section is subdivided in several subsections. The first subsection concerns the 3D rendering with openGL [59]. OpenGL is a library used to perform 3D rendering of complex scenes by making use of the dedicated specific hardware for this task namely the GPU. OpenGL has been used in this thesis to display current distributions on the accelerator, electromagnetic fields and the beam dynamics motion. The second subsection explained the GPU hardware and how it can be so efficient for 3D rendering and for scientific computations. Finally, the last section discusses the openCL library [60] and its functioning.

In this thesis, GPU are used for the computation of the beam dynamics. In the next chapter, it will become clear why the hardware of a GPU is particularly well suited for such a solver.

5.1.1 3D rendering with openGL

First of all, a 3D scene is always discretized into a surface mesh before being rendered by a graphic engine. In 3D gaming, the scenes are in general drawn with special software that allows parametric representation of the objects (such as 3ds Max). When the drawing is completed, it is meshed and the result is saved under a 3D model format. The rendering process in computer gaming is never performed on parametric representation (B-

Rep) such as a NURBS [61] because a GPU is designed to render points, lines and polygons. For the movements and deformations of meshed objects, some techniques such as the well-known skeletal animation [62] can be used. At the border between two bones, the positions of the vertices are calculated based on several transformation matrices summed up with different weights.

3D rendering based on rasterisation is subdivided into three main tasks:

- Vertex processing
- Rasterisation
- Fragment processing

Fig. 5.1 illustrates these three steps.

5.1.1.1 Vertex processing

The first step consists in projecting the 3D points also called points in the eye space onto the projected plane called the viewport. The projected coordinates are called the window coordinates. The projection matrix is relatively simple to calculate. Fig. 5.2 illustrates the projection of a 3D point for the $\hat{x}_1$ in the viewport plane. Opengl works with homogeneous coordinates. Let us calculate the projection matrix that associates uniquely eye points to projected points. The frame used by opengl is a right-handed coordinates system. The viewport plane is located at a distance $-n$ from the origin. The dimension of the viewport is defined by the ordered pair $(2l, 2h)$. The screen aspect ratio is defined by $r = l/h$ and in full screen is in general equal to 16/9. This ratio is a constraint forced by the view frame of our 3D application. The viewport plane is defined by

$$\Pi \equiv \{\overline{x} \in \mathbb{R}^3 | \overline{x} = -n\hat{x}_3 + \alpha_1\hat{x}_1 + \alpha_2\hat{x}_2, \alpha_1 \in [-l, l], \alpha_2 \in [-h, h]\} \quad (5.1)$$

The projection is given by the intersection between this plane and the line passing through the origin and the observation point. Therefore, the intersection is given by the parametric resolution of the following equation

$$\beta\overline{x}_o = -n\hat{x}_3 + \alpha_1\hat{x}_1 + \alpha_2\hat{x}_2 \quad (5.2)$$

With the $\hat{x}_3$ component one can directly find the value of $\beta = \frac{-n}{x_3}$. From there, one can get α_1, α_2

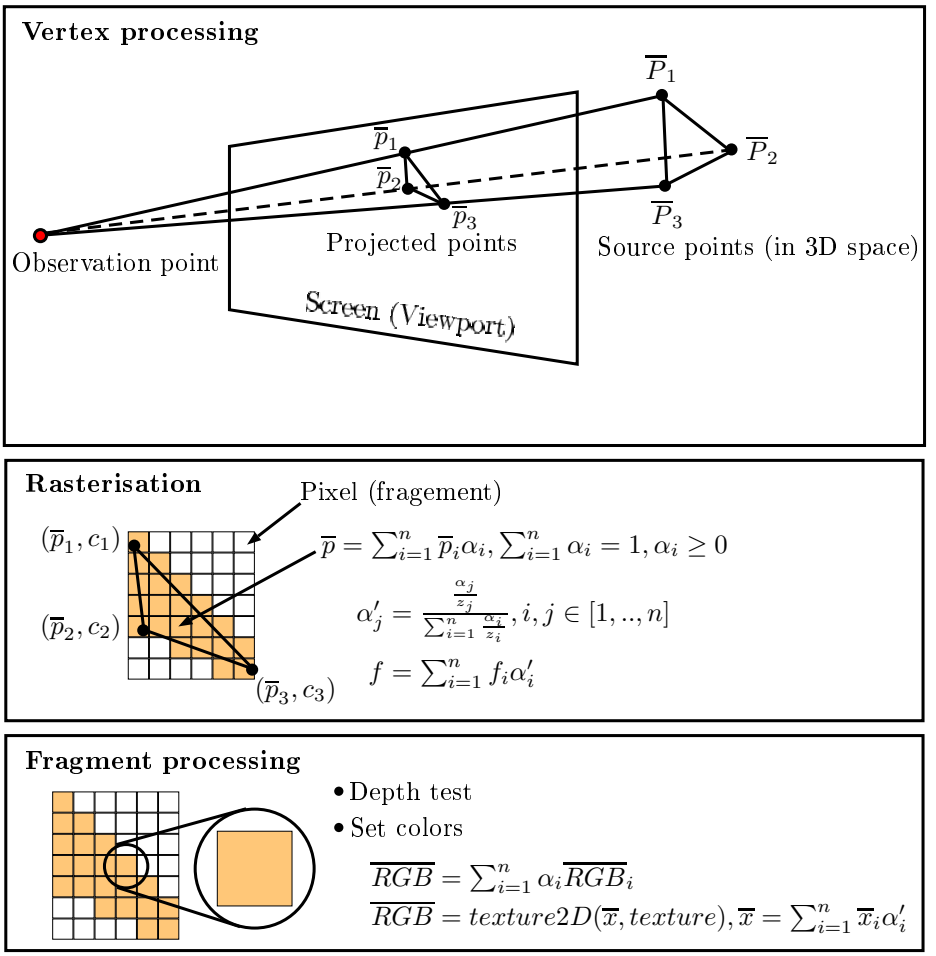
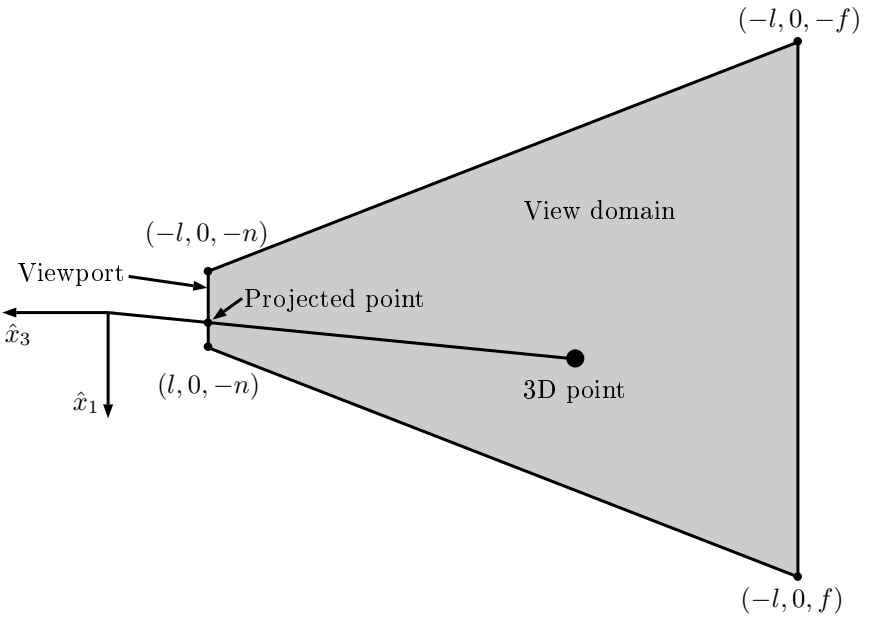


Figure 5.1: Rasterisation principle.



[H]

Figure 5.2: Projection principle.

$$\alpha_1 = -\frac{n}{x_3}x_1 \tag{5.3}$$

$$\alpha_2 = -\frac{n}{x_3}x_2 \tag{5.4}$$

$$\tag{5.5}$$

The frame in the window is defined naturally by $(O_f, \hat{x}_1, \hat{x}_2)$ with $O_f = O - n\hat{x}_3$. Therefore, (α_1, α_2) are the coordinates in the window frame. OpenGL works with normalized coordinates in the interval $[-1, 1]$. The change of variable associated with this rescaling is given by

$$x_{1,w} = \frac{\alpha_1 + l}{2l} = -\frac{nx_1}{2lx_3} + \frac{lx_3}{2lx_3} \tag{5.6}$$

$$x_{2,w} = \frac{\alpha_2 + h}{2h} = -\frac{nx_2}{2hx_3} + \frac{hx_3}{2hx_3} \tag{5.7}$$

The $x_{3,w}$ is used for the depth buffer. Let us determine p_{33}, p_{34} of the projection matrix such that any point on the viewport would lead to a value of $x_{3,w} = -1$ and any point on the far plane would lead to a value of $x_{3,w} = 1$.

$$x_{3,w} = \frac{p_{33}x_3 + p_{43}}{x_3} \tag{5.8}$$

$$\Rightarrow \quad -1 = \frac{-p_{33}n + p_{43}}{-n} \tag{5.9}$$

$$\Rightarrow \quad 1 = \frac{-p_{33}f + p_{43}}{-f} \tag{5.10}$$

This leads to

$$p_{33} = \frac{n+f}{f-n} \tag{5.11}$$

$$p_{43} = \frac{2fn}{f-n} \tag{5.12}$$

The projection matrix is therefore given by

$$P \equiv \begin{pmatrix} \frac{-n}{2l} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{-n}{2h} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{n+f}{f-n} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.13}$$

The projection matrix as it is involves only negative values for the homogeneous coordinate. However, openGL only accepts by design positive number for the clipping coordinate. The projection matrix used in practice with opengl is therefore

$$P \equiv \begin{pmatrix} \frac{n}{2l} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{n}{2h} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{-(n+f)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{5.14}$$

The clip coordinates are defined by the image vector of the eye coordinates transformed by the projection matrix $\overline{x}_c \equiv P\overline{x}$. The window coordinates in this context are given by

$$\begin{pmatrix} x_{1,w} \\ x_{2,w} \\ x_{3,w} \end{pmatrix} = \begin{pmatrix} x_{1,c}/x_{4,c} \\ x_{2,c}/x_{4,c} \\ x_{3,c}/x_{4,c} \end{pmatrix} \tag{5.15}$$

A common way to designate any point in a convex polygon is the use the barycentric coordinates of the points defining it as claimed by the proposition 10.3.5. It is, in particular, useful to obtain the value a a linear function for which the values are known at the polygon points. Indeed, let be $f : E \to F$ with F a normed vector space and f a linear function, then

$$f(\sum_{i=1} \alpha_i x_i) = \sum_{i=1}^{n} \alpha_i f(x_i) \tag{5.16}$$

Let us inject the expression of a point on the primitive in terms of barycentric coordinates in order to investigate the image space of the projection. Let us define $P_r \equiv P(1:3, 1:4)$ and let be $\overline{x} = \sum_{i=1}^{n} \alpha_i \overline{x}_i$ a point in a given primitive

$$\overline{x}_w = \frac{\sum_{i=1}^{n} \alpha_i P_r(\overline{x}_i)}{\sum_{i=1}^{n} \alpha_i x_{3,i}} = \frac{\sum_{i=1}^{n} \alpha_i \overline{x}_{i,w} x_{3,i}}{\sum_{i=1}^{n} \alpha_i x_{3,i}} \tag{5.17}$$

Let us define α_i' by

$$\alpha_i' \equiv \frac{\alpha_i x_{3,i}}{\sum_{i=1}^{n} \alpha_i x_{3,i}} \tag{5.18}$$

Clearly, $\sum_{i=1}^{n} \alpha'_i = 1$ and $\alpha'_i > 0$. Therefore, the images are always expressed as a barycentric combination of the window coordinates. Let us prove the surjectivity of the application. For a set of $\alpha'_i > 0$ such that $\sum_{i=1}^{n} \alpha'_i$, let us find a set of $\alpha_i > 0$ such that $\sum_{i=1}^{n} \alpha_i = 1$ and such that the application send $\alpha \to \alpha'$. It is relatively straightforward to see that choosing $\alpha_i = -\frac{\alpha'_i}{x_{3,i}}$ would give the expected result through the application. However, the condition $\sum_{i=1}^{n} \alpha_i = 1$ is not guaranteed. In order to guarantee this condition, let us simply rescale the α_i as follows

$$\alpha_i = \frac{\frac{\alpha'_i}{x_{3,i}}}{\sum_{i=1}^{n} \frac{\alpha'_i}{x_{3,i}}} \tag{5.19}$$

Therefore, the application is surjective and the image space correspond to the whole convex skull generated by the vertex window points. In order to prove the surjectivity, one created a application that gives the barycentric coordinates in the eye space in function of the barycentric coordinate in the window space. The composition of the two applications gives the identity application and therefore they are both a bijection. It is important to notice that even if these relations give a way to convert the barycentric coordinates uniquely, the primitive convex skull in the window space is not necessary in bijection with the primitive convex skull in the eye space.

Before closing this subsection, let us notice an important consequence about these observations. The third window space coordinate is used for the depth analysis and therefore contains a direct information about the x_3 value in 3D space. This information about x_3 is calculated in the window space by the barycentric combination i.e by a linear combination. It is remarkable that the depth information is calculated simply by a linear combination in the window space. However, it is important to notice that the depth information provided by $x_{3,w}$ is not the depth itself as the following equations show

$$x_{3,w} = \sum_{i=1}^{n} \alpha'_i x_{3,i,w} = p_{33} + \frac{p_{34}}{x_3} \tag{5.20}$$

$$\Rightarrow \sum_{i=1}^{n} \alpha'_i (p_{33} + \frac{p_{34}}{x_{3,i}}) = p_{33} + \frac{p_{34}}{x_3} \tag{5.21}$$

$$\Leftrightarrow \sum_{i=1}^{n} \alpha'_i \frac{1}{x_{3,i}} = \frac{1}{x_3} \tag{5.22}$$

5.1.1.2 Rasterisation

Once the window space coordinate of each vertex of a given primitive has been calculated and the $x_{3,i}$ is kept somewhere in memory, the rasterisation of this primitive takes place. This operation consists in finding all the pixels touching the convex skull of the primitive and calculate for each of these pixels respectively the value of the different functions at the position of the pixel. In order to calculate these values, the barycentric coordinates are calculated. Then the formula developed in the previous subsection allow one to find the barycentric coordinates in the eye space and therefore the value returned by the different functions at the pixel corresponding position in the eye space. For the depth buffer, the $\frac{1}{x_{3,w}}$ is calculated by a simple barycentric combination in the window space. For each primitive process by the rasterisation stage, several fragments associated respectively uniquely to a pixel are generated. A fragment is defined by a pixel and a set of informations associated with this pixel.

5.1.1.3 Fragment processing

The values of the different functions calculated at the rasterisation stage are sent to the fragment processing stage. At this stage, these values are used to define the color of the pixel and to filter the hidden pixels with the help of the depth buffer.

5.2 GPU inside

The 3D rendering of a scene based on rasterisation involves a lot of independent computations on a set of small data. First, each vertex is projected into the clip space by calculating the product of the projection matrix with each eye space vertex coordinate. Then, in the same stage, the window coordinate are calculated. This stage computation is independent for each vertex. Then, each primitive is rasterized. Again, for each primitive, the computation is independent. Finally, each fragment is processed independently in the last stage of the openGL pipeline. The GPU hardware needs to perform a lot a calculation on a set of small data. Furthermore, a give set of data undergo a couple of instructions before being copied back to memory. In other words, this means that several instructions are executed on locally loaded data before copying it back to the main memory. Three characteristics of rasterisation have been identified: compute intensity, data parallelism and data locality. Compute intensity means a good ratio between the number of operations performed per I/O on the main memory. Data parallelism means a good independence between the many sub-tasks of

a given algorithm. Data locality means that each sub-task performs computation on data located in the same memory region. These three conditions allowed engineers to design a specific hardware for the rasterisation.

One may wonder if the instructions set used for the three rendering tasks are the same. Before 2006, the GPU hardware designs were built based on different instructions sets for the different task of the rasterisation process. In particular, the vertex processing had a lot more instructions than the fragment processing. Programmers were therefore using different instructions for these two stages. At the present time, each steps of the rendering pipeline can use the same set of instructions. This is called the Unified Shader Model (USM). A shader is piece of program from the rendering pipeline of an API. If a shader can use any kind of instructions, specific hardware might still be used to perform specific action from the graphics pipeline. For instance, many GPU include a raster engine and a tesselation engine because these operations are almost always intensively used by the game engine, they are much faster on dedicated hardware than running on the main architecture of the GPU and a dedicated hardware represent a reasonable amount of silicon i.e a good performance to price ratio. ATI released the first GPU following this model in 2006 with its HD2000 board based on the TeraScale architecture. Quickly after the release of this board, Nvidia released the Geforce 8 series in 2007 based on the Tesla architecture.

This allowed gaming company to develop more realistic graphics and to improve the ease of programming on GPU. The different API have followed the evolutions of the architectures. There was a major step in the opengl evolution that happened at the transition between the version 2.1 and 3.0; the programmer using opengl has to program at least the vertex shader and the fragment shader. The vertex shader allows to user to perform the most important computation of the vertex shader. For instance, the product of the projection matrix[1] with the homogeneous coordinates happens there. The user can define several variables that are going to be interpolated in the rasterisation stage and therefore accessible in the fragment stage and in particular in the fragment shader. In the fragment shader, the color for the pixel associated to this fragment is calculated. If the color is simply defined by the linear interpolation of the RGB colors defined at each vertex, the value calculated by the rasterisation stage can be used directly and the vertex shader is one line of code. For texture, the 2D texture coordinates interpolated by the rasterisation stage are also directly used by calling a

[1] In general, the matrix used in the vertex shader is not only the projection matrix. Indeed, in general, complex geometry are describe with the help of several frames. The general transformation matrix is therefore first computed (the products of the projection with several transformations matrices) and then used in the vertex shader.

texture function that match the texture coordinate to the right pixel of the picture. For these two cases, the fragment shader is very simple. You may wonder in what kind of situation the fragment shader might be tricky. A example of a more complex shader is for instance a picking fragment shader. The picking process consists in selecting element in the 3D scene by clicking on it. In oder to perform a fast picking, actual game engines use the GPU. A false picture using a specific color for each primitive is rendered in a off-screen buffer. The color seen by the mouse is then matched to the primitive selected.

The architecture this is describe in the next section is the Pascal architecture from Nvidia. This is the latest Nvidia architecture at the time of writing of this thesis. The reader is invited to read interesting aspects about the former architectures by himself. In particular, the Tesla, Fermi and Maxwell architectures might help the reader to understand the former strategies of GPU. All informations related to the different Nvidia architectures can be found on their website [63].

5.2.1 Nvidia Pascal architecture

The GP104 chip is considered through all this section to illustrate the functioning and the features of a Pascal architecture. The GP104 is made of several Graphics Processor Clusters (GPC). Each GPC is made of several Streaming Multiprocessor (SM) and a Raster Engine (RE), as shown in Fig 5.3. The RE is a dedicated hardware used to perform the rasterisation stage of the rendering pipeline. Typically, this allows the GPU to rasterise several triangles per clock cycle. An L2 coherent cache is used to accelerate the transfer between the main memory and the SM. There are height memory controllers for the GP104, each controller has a 256 KB of L2 cache i.e a total of 2MB of coherent L2 cache is available. Next to the L2 cache they are height Render Operations Pipelines (ROP) that perform transactions between buffers in local memory and also perform antialiasing [64] operations.

The Streaming Multiprocessor is made of several piece of hardware namely:

- CUDA core
- Shared memory
- Texture/Cache memory
- Registers

Figure 5.3: The GP104 chip based on Pascal architecture [63].

Figure 5.4: Streaming Multiprocessor of the Pascal architecture [63].

- Load and Store blocks
- Special Function Unit (SFU)
- Texture specific hardware
- Instruction Buffer
- Wrap Scheduler

Fig. 5.4 illustrates all the components.

5.2.1.1 CUDA core, SFU and DPU

A CUDA core is a simple piece of hardware capable of performing basic integer and floating-point operations. It does not decode instructions which is the task of the Wrap Scheduler and Dispatch Units. They are made of an Arithmetic Logic Unit (ALU), and Floating-Point Unit (FPU), potentially source operand registers and destination operand registers as shown in Fig 5.5. The latter two are not always used in each implementation of the architecture. They are link local buffer that can be used to read or write data from the registers and use this data later on. The SFU allows fast

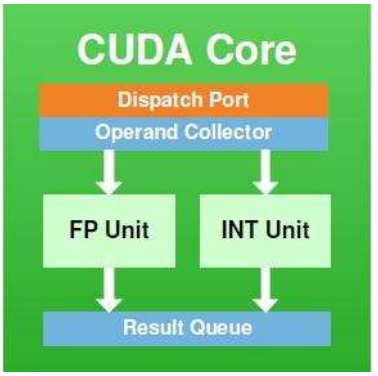


Figure 5.5: Cuda CORE hardware [63].

computation of specific operations such as the evaluation of trigonometric functions and the exponential. When performing 3D rendering, the computation are always done with floating numbers. However, scientists often need to use double precision numbers. Performing precision operations on GPU without specific hardware to do it costs more than twice the computation time of the same operations with floating numbers. In order to avoid such performance loss, Nvidia placed dedicated hardware to handle double precision calculations, the DPU.

5.2.1.2 Instruction decoder and scheduler

The instruction decoder and scheduler is the part of the SM that decoded the instructions present in the instruction buffer and schedule the execution of these instructions on the cores for several threads. The GP104 SM is separated in two block of identical hardware made each of one scheduler, two dispatchers and one instructions buffer. An instruction is executed for a group of 32 threads and if their are less threads to be executed some idle threads are run instead. This is called a wrap execution and it consists in running a single instruction for 32 different set of data. The scheduler fetches an instruction and try to find a warp available to run it. Depending on the instruction, it will look for 8 or 16 available cores, 8 available LD/ST units, 8 or 16 available DP Units or 8 SFU units. For instance, if the data are already loaded in registers (or in the source operands) and the instruction consists in an arithmetic operation, the scheduler looks for 8 or 16 available cores to run the instruction. In this case, the instruction is run for 4 or 2 clock cycles respectively in order to have the 32 executions of the wrap. Once the wrap is identified, the scheduler uses its dispatchers to run the same instruction on these cores. In case of load or store instruction on the

global memory the LD/ST unit calculates the source or destination address for 8 threads. Once these addresses are known, the L1 cache is used to fetch theses data. If the data are contiguous in memory and aligned, the fetch operation might be grouped. The memory section explains in details the functioning of the different memories.

5.2.1.3 Memory principle

In this section, one focuses on the physical memory in the GP104. Each API for GPU computing might have a memory model that slightly diverges from the physical pattern. However, it is relatively easy to guess how the memory model of a given API is linked to the physical memory.

Their are 3 types of memory in a SM: Registers, L1 Cache (unified with the texture cache), shared memory.

The registers are the private memory of the threads. Their are registers also used to hold the status of the thread itself such as the index of the thread being run on a given core. A given SM can run a limited number of threads that is fixed by the manufacturer. Typically, around 1024 threads can run in a given SM. For each of these threads, several registers will be used. There is no switch context when the scheduler decide to run a new wrap in the SM. For each threads the number of registers allocated to it hold the values of these threads till completion of the task (more specifically till the completion of a kernel defined a bit further in the text). It is possible that the number of registers is not sufficient to hold the private variables of the n threads that run on a given SM. In this case, the compiler is going to use global memory to store the private data. This very important point to understand because global memory is more than 100 times slower than registers although local accesses are cached by the L1 cache. When the programmer is not certain that the compiler is going to use only registers, he will likely used to shared memory which is of a speed comparable to registers and is fully controlled by the user. Arrays dynamically indexed are always store on global memory because the compiler cannot figure out the register to use at compile time. Therefore, it uses global memory a pointer arithmetic.

The L1 cache memory is used to cache the access to global memory. It is the second level of the cache system since there is a L2 cache used at the global level. The L1 cache is however a non coherent cache while the L2 cache is coherent. No coherent cache means that the value held by the cache might not be the same as the one held by other caches of the same level and that any write operation from a core to global memory does not

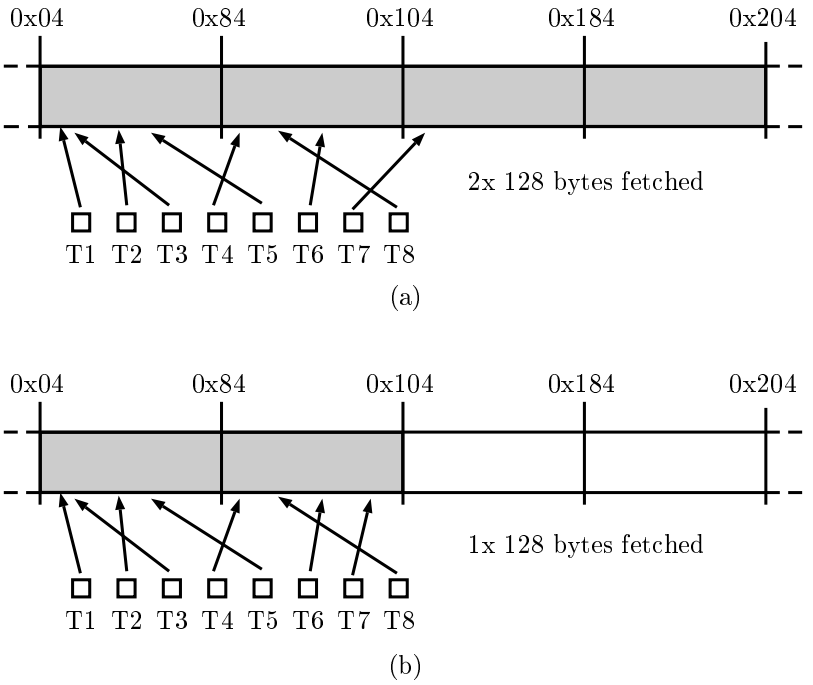


Figure 5.6: Coalescing for global memory access.

warn any other L1 cache pointing to the same memory area been written to. This is a major difference with cache system on a x86 CPU because each cache on a x86 is coherent. The speed of L1 cache is comparable to the speed of the shared memory be does not work exactly the same way. When a thread requests memory from the global memory it first uses a LD/SD unit to calculate the source address. This is done in parallel for a couple a threads (a fraction of a wrap). Once the source or destination addresses are known, the memory controller checks whether or not this addresses are contiguous in memory. If they are, the fetch or write operation can be grouped. This is known as coalescing memory access. Fig /refCoalescing illustrates this concept for 8 threads (instead of 32 threads) for the sake of clearness. For the clearness of the picture, only the end of the address is represented. At the address 0x04 starts a new cache word of 128 bytes. The addresses are given in hexadecimal basis. The subfigure (a) shows a situation where two L1 cache fetches of 128 bytes are required. Indeed, the thread 7 requests a word in the region of memory just after 0x104 which means a full fetch of 128 bytes is performed for the data between 0x104 and 0x204. In the subfigure (b), all threads request memory in the region

0x04 and 0x104. A single fetch is performed. Most likely, the reason of having a 128 cache line is that a LD/ST unit performs I/O operation on 4 words. Since a word is 32 bits or 4 bytes and their are 8 LD/ST units connected on a cache line, the bus width chosen by Nvidia is simply the width of an optimal memory access where the 8 LD/ST units would read contiguous data in memory. L1 cache is not always enable by default. There are two reasons for disabling the L1 cache. First, if the kernel access the global memory at places that do not allow grouping read or write, it is better to bypass the L1 cache and use the 32 bytes line of the L2 cache in order to avoid the waist of bandwidth. The second reason is that former nvidia GPU were designed with a common L1 cache and shared memory. The user could choose to allocate more memory for the cache or for the shared memory. For these devices, in CUDA programming a special flag had to be used at compilation for using more memory for the L1 cache (see Programming guide on nvidia website). With OpenCL, the user does not have to fine control of the device since openCL is a device independent API.

The shared memory is a fast memory entirely controlled by the programmer. It can be used to perform very quick random data access if used correctly. The shared memories in each SM are not connected therefore it can only be used by a group of threads running on the same SM. The shared memory is divided into several banks that can be accessed simultaneously by any thread of the SM. Several threads trying to access the same bank at the same time and for different addresses undergo a bank conflict and several clock cycles will be used to read/write the data. However, if the accesses are performed on the same address and for the same operation only one clock cycle is required to perform the action. On the GP104, 64 KB of shared memory is available per SM. Fig. 5.7 shows 3 different situations for bank access. In the first subfigure (a) the first and second thread access two different data in the first bank resulting in a bank conflict. The subfigure (b) shows almost the same situation but this time the first and second thread access the same data. If the operation is identical (read or write) then there is no bank conflict created by the first and the second thread. The last subfigure (c) shows a situation where the first and the second threads address two different banks so there is no conflict created by these two threads.

5.3 OpenCL

OpenCL is a language that aims at running massively parallelisable program. Nvidia developed their own language called CUDA. However, CUDA is a proprietary language and can only be used for Nvidia devices. Let us

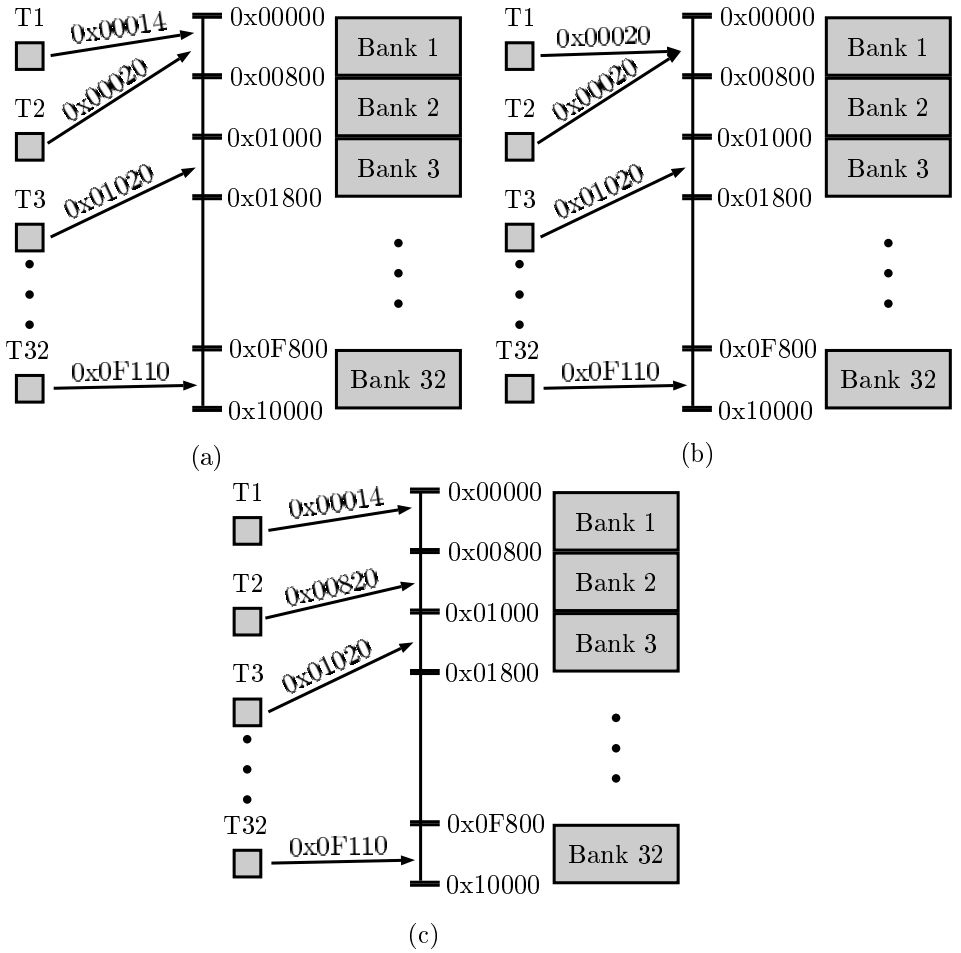


Figure 5.7: Shared memory and bank conflicts.

compare the two languages

Type	CUDA	OpenCL
Portability	Only for Nvidia devices	For all devices supporting openCL and in particular: ATI, Nvidia, Intel, Altera (FPGA), AMD
License	Proprietary language	Open source. Any company can implement openCL for their devices.
Performances	High performance due partially to the fine control of the hardware	10% less performance on Nvidia.
Documentation	Very well documented	Very well documented

The biggest advantage of openCL over CUDA is the portability. Indeed, once the code is written it can be run on many different devices. In particular, the Intel Xeon Phi is a very interesting chip for massive parallelisable algorithm. It is also very interesting to compare the performances on different hardware for the same implementation. Of course, the implementation of openCL plays a role on the performances.

In the following section, the openCL model is presented.

5.3.1 Opencl device abstraction model

Fig. 5.8 illustrates the device abstraction of OpenCL. Their are 3 types of memory in the model. A global memory which is in general huge but much slower than the other memories. Then there is the local memory shared by a group of work items (threads) associated to a Compute Unit (CP). This memory is much faster than the global memory but is also much smaller. Finally, each work item has its own private memory space. The private memory is supposed to be the fastest memory as long as the user does not use more private memory than available. A work item represent a thread and there are several groups of work items each group being associated uniquely to a CU. The local memory is shared by all the WI in the WG.

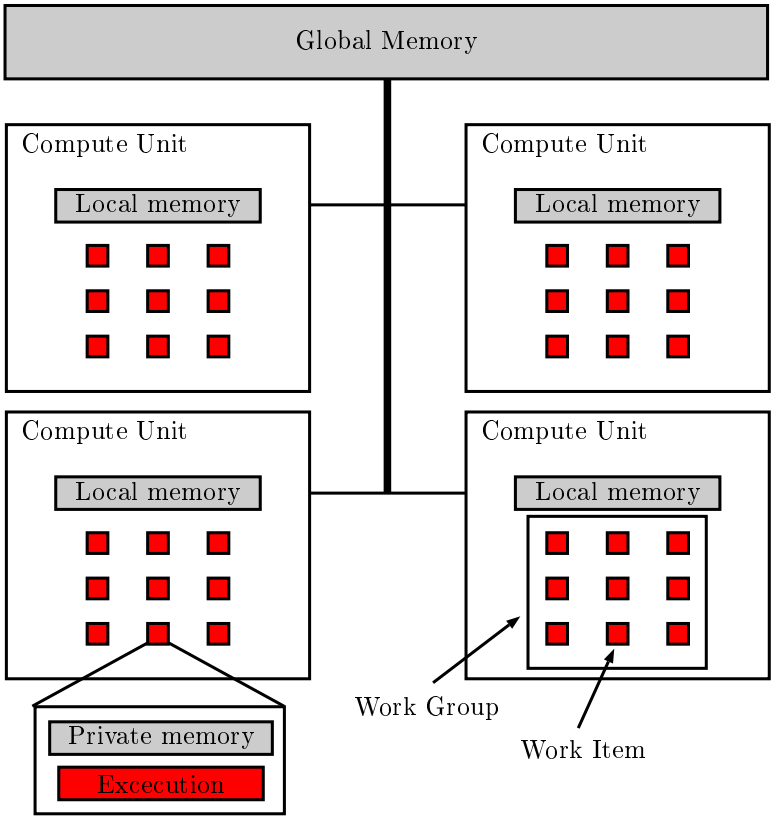


Figure 5.8: Device abstraction OpenCL.

5.3.2 Opencl API

A function that runs on an openCL device is called a kernel. When the programmer request a kernel to be run on a device, opencl first compile the kernel (and save the compilation to gain time for the next use). Then the programmer place the kernel in a command queue associated to the device. The device scheduler then launch the kernel when resources are available for this kernel. It is possible to synchronize the kernels inside a given command queue. For instance, it is possible to forbid a kernel to be run until one or several other kernels are done. The command queue can be set in two different modes: A random mode and a First In First Out (FIFO) mode. In the first mode, the scheduler runs the kernels in the order that suits it.

On a given computer, their maybe several devices that supports openCL. Devices from the same manufacture might be able to share data directly between one another. In openCL, devices are own by a context. A context gather all devices that can share their data without the need to pass by the compute main memory. Each context is associated to a platform which is an openCL specific implementation provided by the manufacture of one or several devices embedded in the computer. Fig 5.9, Fig 5.10 summarizes these explanations.

5.3.3 Work partition

As explained previously, each Compute Unit (CU) runs a number of work items that composes the group associated with it. Opencl imposed that each group has the same number of work items. For each device, there is a preferred group size denoted $L_{pref} \in \mathbb{N}$. When running kernels on Nvidia hardware, the preferred group size is 32 which is not a big surprise since the warp size is 32 threads. If the number of elements running in the group size is smaller than the preferred one then usually some idle work items are created on the device which means that the efficiency decreases. For each CU, there is also a maximum number of work items that can be placed in the group. For instance, on recent Nvidia GPU this number is 1024. Below, the maximum number of work items that can be placed in a group is denoted β.

OpenCL provided n dimensions to partition the work. For each dimension there is a maximum number of work items that can be placed in a group. Let us denote these limits by the ordered pair $L_{max} = (L_{max,1}, ..., L_{max,n}) \in \mathbb{N}^n$ where $L_i \leq \beta$. There is also a limits for each dimension on the global number of work items given by the size of a site_t variable for a specific device. This number can be determined on the device address bit width (for instance 32 bits, then the max number for a given dimension is given

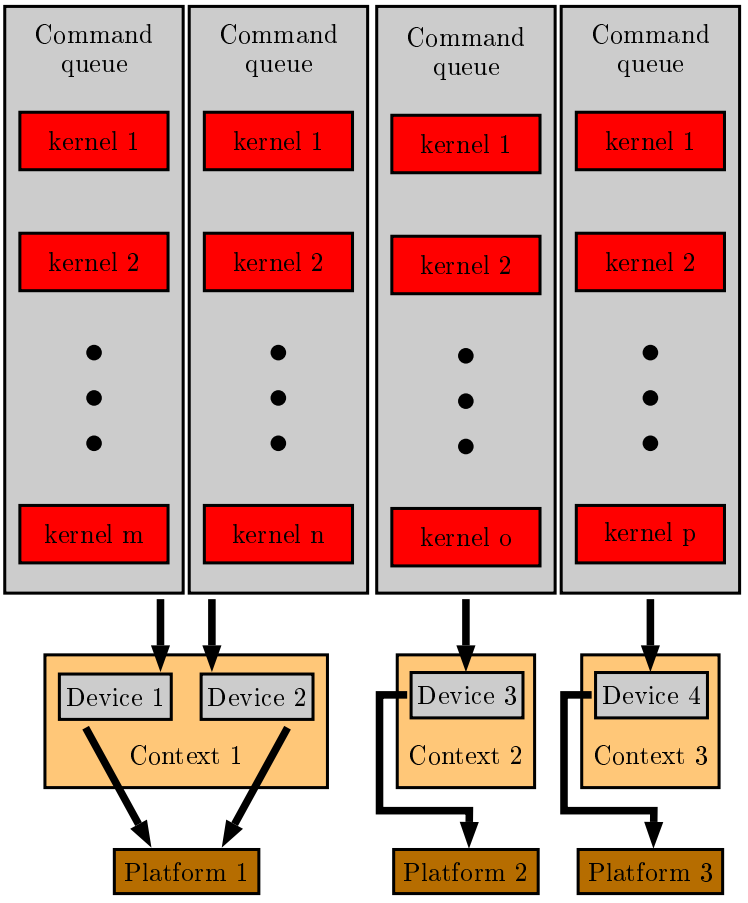


Figure 5.9: This figure shows the different links between the openCL entities.

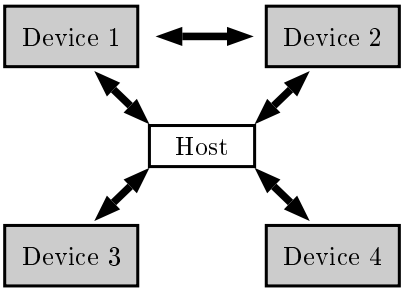


Figure 5.10: This figure illustrates the possible data flow between devices.

by $2^{32} - 1$).

Let us assume that a kernel has a global work load given for each dimension by $(G_1, ..., G_n)$ where $G_i \leq sizeof(size_t)$. One must partition this global work load into groups. Let us denote by $L = (L_1, ..., L_n) \in \mathbb{N}^n$ the group size or the so-called local size. Each group should contain if possible a multiple of the preferred group size for the targeted device i.e $\Pi_{i=1}^{n} L_i = kP_{pref}$ with $k \in \mathbb{N}\backslash\{0\}$. Of course, for each dimension, the group size must be a multiple of the global work size i.e $G_i = k_i L_i$ with $k_i \in \mathbb{N}\backslash\{0\}$. Injecting this last relation into the preferred size condition one gets $\Pi_{i=1}^{n} G_i/k_i = kP_{pref}$. There might be many ordered pairs of $(k_1, ..., k_n)$ that satisfies these constrains. There is still however one criterion left to account for, the amount of local memory required by the group. Since the local memory is relatively small, it might take a dominant part in the partition. Therefore, optimizing an openCL partitioning might be very tricky and there is no recipe for it, the optimal partitions of a kernel is specific to it. Fig 5.11 shows the different relations between these concepts.

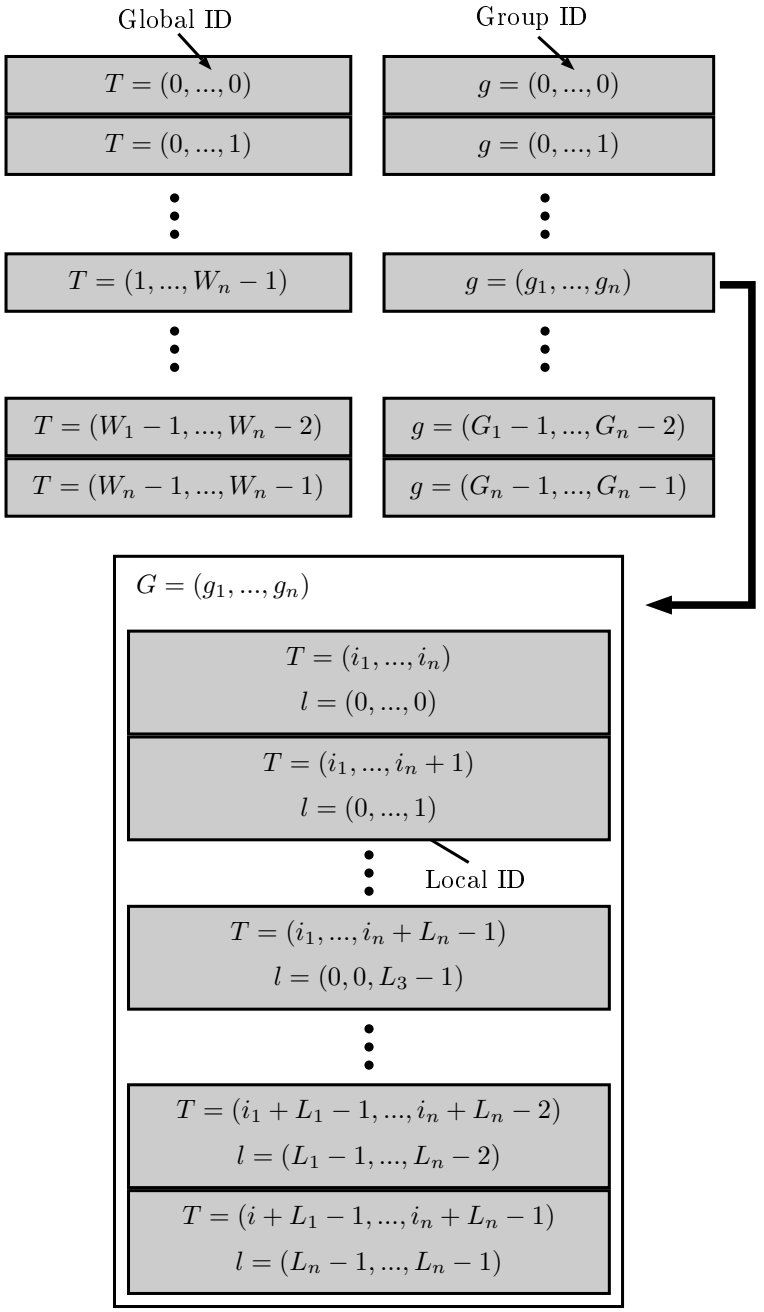


Figure 5.11: The picture illustrates the relation between global ID, group ID and local ID in an opencl partition.

Chapter 6

Beam dynamics

6.1 Introduction

The goal of this chapter is to investigate the feasibility of the implementation of a beam dynamics solver on GPU. The equations of motion of independent particles in an electromagnetic field can be solved in parallel. In particular, a GPU is a perfect piece of hardware for this problem. Including the interactions between particles changes breaks this independence. However, using some approximations, it is possible to maintain a GPU implementation which is still very favourable in terms of the ration performance cost to a CPU.

6.2 Motion solver physical hypotheses

6.2.1 Geometric context

The beam dynamics computation is performed in a region of space delimited by a cylindrical that we call the accelerating tunnel. It is placed right in the middle between the rods where the particles are present. A default accelerating tunnel has been defined at suitable fixed position in order, for instance, to calculate easily the particles unique cells. The cells are a subdivision of the accelerating tunnel as it will be explained in subsection 6.2.4.

The default global frame is defined by $F_1 \equiv (O, \hat{x}_1, ..., \hat{x}_3)$ where the basis is an orthonormal basis for the geometric scalar product. Homogeneous coordinates are used in order to express any affine transformation (see proposition 10.3.3) as a linear application in a fourth dimensional vector space. The default accelerating tunnel is transported at the right place

in the accelerator depending on the localisation of the main geometry. Let $F_2 \equiv (O', \hat{x}'_1, ..., \hat{x}'_3)$ be an other frame placed on the accelerating tunnel at the right position in the accelerator. The default coordinates are therefore expressed in F_2. The transformation matrix between the two system of coordinates is given by

$$T_{2\to 1} \equiv \begin{bmatrix} R^t & T \\ 0 & 1 \end{bmatrix} \tag{6.1}$$

where R is the rotation matrix defined by $\hat{x}'_i = \sum_{j=1}^{3} R_{ij}\hat{x}_j$ and the translation vector by $T \equiv [\overline{OO'}]$. The inverse transformation is given by

$$T_{1\to 2} \equiv \begin{bmatrix} R & -RT \\ 0 & 1 \end{bmatrix} \tag{6.2}$$

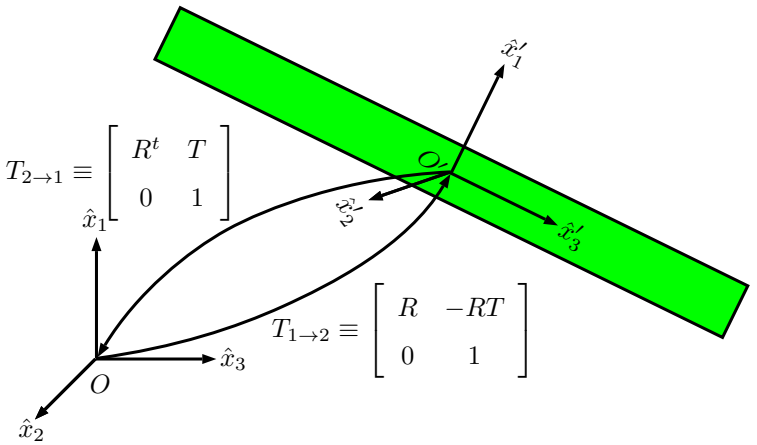
Let us define some sample points by

$$G \equiv \{\overline{x}|\overline{x}_0 + \sum_{i=1}^{3} \alpha_i s_i \hat{x}'_i, \alpha_i \in [1, ..., N_i]\} \tag{6.3}$$

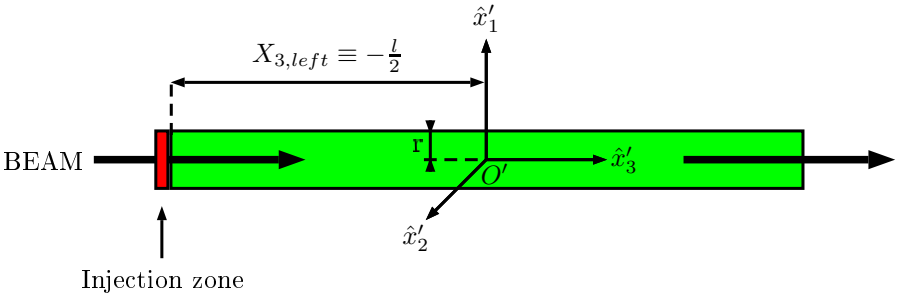
where $N_i \in \mathbb{N}$ and $s_i \in \mathbb{R}$ is the grid step in the direction $\hat{x}'_i$. They represent the points on which the source electric field must be extracted from the MoM simulation. These sample points are defined in the default accelerating tunnel and therefore must be transported to the real accelerating tunnel position. The true grid positions are given by $T_{2\to 1}G \equiv \{T_{2\to 1}(x)|x \in G\}$. The field extracted with the MoM at these positions must be transported back to the default accelerating tunnel. Let be $\overline{f}_{\alpha_1,\alpha_2,\alpha_3}$ the field extracted at the grid position $(\alpha_1, \alpha_2, \alpha_3)$ then the field in the default tunnel is given by $[\overline{f}'_{\alpha_1,\alpha_2,\alpha_3}] = R[\overline{f}_{\alpha_1,\alpha_2,\alpha_3}]$.

6.2.2 Electric field separation hypothesis

The electric field inside the accelerator is generated by two different sources: The RF source feeding the accelerator and the space charge. The space charge is the electromagnetic field generated by the motion of particles inside the accelerator. Since the velocities of the particles are small, the total magnetic field produced by them can be neglected. The electric field scattered by the particles interact with the accelerator (i.e it is a wideband



(a)



(b)

Figure 6.1: (a) Transformations between the main frame F_1 and the frame of the default accelerating tunnel F_2. (b) Definitions of some elements of the accelerating tunnel.

external source for the MoM). Therefore there is a coupling between the RF source feeding the accelerator and the beam. This coupling can be understood also with the help of the energy conservation principle. Since some energy is transferred from the RF source to the beam there must be an electromagnetic coupling between the two sources by conservation of energy principle. The coupling between these two sources is neglected in this thesis. Therefore, the source field produced by the RF source and simulated with the Method of Moments (MoM) is independent of the space charge field calculated separately. The electric field at any point in the accelerator reads

$$\overline{E}(\overline{x},t) = \overline{E}_s(\overline{x},t) + \overline{E}_b(\overline{x},t) \tag{6.4}$$

where s stands for source and b stands for beam (space charge electric field). The space charge electric field is computed with the help of the Coulomb law. The accelerator surrounding the particles generate a static electric field in the presence of particles. Indeed, since the electric potential on and in a perfect electric conductor must be constant, charges will appear on the surface of the accelerator to ensure this condition. Therefore, to obtain an accurate electric field, one should solve the full electrostatic problem taking into account the accelerator. In this thesis, the accelerator is neglected in the computation of the space charge field. Only the field scattered by the particles themselves is accounted for.

6.2.3 Macro-particle Model

The beam is injected in the RFQ at an energy of 30keV and with a current of 4mA. It means that the number of injected particles is given by $I/Q = \frac{4*10^{-3}}{1.6*10^{-19}} = 2.5 * 10^16 p/s$. Let us take a critical time transport in the accelerator corresponding to a particle not gaining any energy [1]. The non-relativist velocity is given by

$$v = \sqrt{\frac{2E}{m_0}} = \sqrt{\frac{2 * 30 * 10^3 * 1.6 * 10^{-19}}{1.6 * 10^{-27}}} = 2.45 * 10^6 m/s \tag{6.5}$$

where m_0 is the mass of a proton and E is the total kinetic energy. The accelerator is 4m long, therefore the transfer time is given by $\Delta t = \frac{4}{2.45*10^6} = 1.6 * 10^{-6} s$. The number of particles for this critical case is given by $\Delta t * \frac{I}{Q} = 4 * 10^{10}$. If the computer were to solve the motion equation

[1] It may happen if the input beam is unmatched with the accelerator.

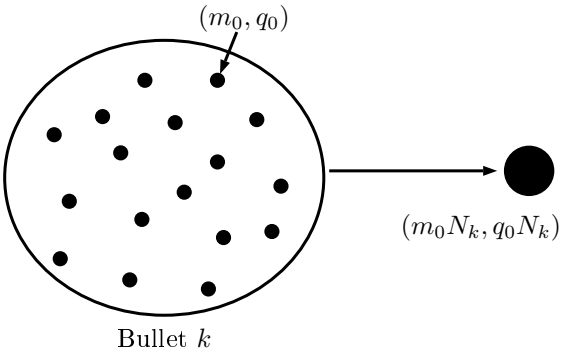


Figure 6.2: Bullet approximation

for each particles, the memory required to hold the positions and the velocities of each particle would be around 1TB which is far too big for present computers. Moreover, if each interaction should be computed, the naïve brute force method scales in $O(n^2)$ with n the number of particles. This is obviously impossible to handle within a reasonable computation time.

The model we decided to follow for the beam dynamics simulation is the so-called macro-particle approximation. It is also called bullet approximation in this thesis. It consists in gathering a certain amount of particles into a bullet and treat this bullet as a single particle. This virtual particle has a mass and a charge corresponding respectively to the mass addition and the charge addition of all the particles concentrated in it. Fig. 6.2 shows this approximation. The force between two particles is given by the following well-known Coulomb formula

$$\overline{F} = \frac{q_1 q_2}{4\pi\epsilon_0} \frac{\overline{r}}{r^3} \tag{6.6}$$

with $\overline{r}$ the distance vector between the two particles, $r = ||\overline{r}||_2$, ϵ_0 the permittivity of free-space, q_1, q_2 the charges respectively of the two macro-particles.

Let us strike the attention of the reader to the fact that the motion of a bullet in a given electric field is identical to the motion of a single proton in the same electric field. Indeed, this is a simple observation in the third Netwon's law

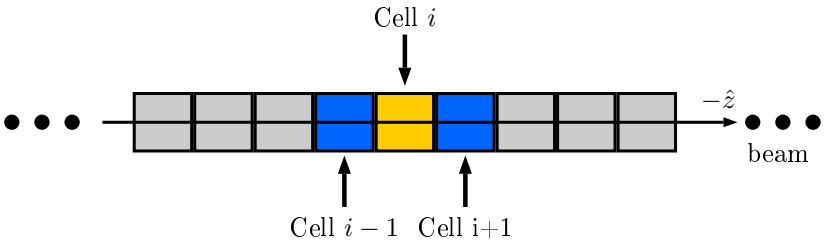


Figure 6.3: Interaction of cell i with its two neighbours namely the cell $i-1$ and the cell $i+1$.

$$\overline{F}_k = m_0 N_k \frac{d^2\overline{O'X}_k}{dt^2} \tag{6.7}$$

$$\overline{F}_k = N_k q \overline{E} \tag{6.8}$$

$$\Rightarrow \quad m_0 \frac{d^2\overline{O'X}^k}{dt^2} = q_0 \overline{E} \tag{6.9}$$

where N_k is the number of particles gather in the bullet k and q_0 is the charge of a proton. Therefore, the beam dynamics errors due to the bullet approximation come from the approximation of the electric field map generated by the approximative particles distribution. Intuitively, if the number of macro-particles is very large, the electric field distribution should not be too far from the real one in the sense of a local average electric field.

6.2.4 Short distance interaction approximation

Without accounting for the space charge effects the motion solver consists simply in solving independent ordinary differential equations. When the space charge effects are taken into account, one should compute the Coulomb interaction between each bullet which is therefore an algorithm in $O(M^2)$ where M is the number of bullets. However, since the Coulomb interaction decreases in $\frac{1}{||\overline{R}||^2}$ where $\overline{R}$ is the distance between two bullets, it makes sense to limit the region of interaction of each particle to a small fraction of space. One potential implementation of this idea consists in dividing the accelerator into small longitudinal cells and the particles in a given cell interact only with their two direct neighbour cells. Fig 6.3 illustrates this concept where the cell i interacts with its two direct neighbours namely the cell $i-1$ and the cell $i+1$.

6.2.5 Source Field Computation

For each particle position, the source electric field has to be calculated. Since it would be cumbersome to perform an electric field extraction for each new particle position during the integration process another strategy had to be used. The electric field is calculated for sampling points placed in a regular grid as defined in the section 6.2.1. The value of the electric field for any position in this grid is given thanks to a simple Lagrange 3-linear interpolation in 3 dimensions. The 3-linear Lagrange interpolation consists simply in interpolating linearly in each dimension recurrently. Additional informations can be found in Appendix in section 10.4.1.

6.2.6 Summary and differential equations

The goal of this section is to formalize a mathematical model of the problem taking into account the approximations of the former sections and investigate the good behaviour of these equations. By good behaviour, one means the existence of a unique solution.

Let $f : U \subset \mathbb{R} \times \mathbb{R}^{6n} \to \mathbb{R}^{6n}$ with n the total amount of particles allow in the simulation be the function that gives the derivatives of the phase space variables for each particles such that the differential equation to solve is given by

$$x' = f(t, x) \tag{6.10}$$

In order to model the beam dynamics through a global differential equation, the function f takes a zero value for the particles that are not yet injected in the accelerator at a given time. Since particles are injected continuously with time and since the interactions for a given particle is calculated only for its two direct neighbour cells, the function f is not a continuous function. However, assuming that one performs the injection at discrete time, and that particles are sorted into cells at discrete time, one ends up with a piecewise continuous function f. Let $t_0, ..., t_n \in \mathbb{R}$ be a series of time points for which an injection and/or a sorting is performed (t_0 being the first injection). For each interval $[t_i, t_{i+1}[$, the function $f : [t_i, t_{i+1}] \times U_2 \to \mathbb{R}^n$, where

$$U_2 \equiv \mathbb{R}^{6M_i} \setminus \cup_{j=1}^{M_i} \cup_{k=1}^{M_i} \{((\overline{x}, \overline{v})_1, ..., (\overline{x}, \overline{v})_{M_i}) \in \mathbb{R}^{6M_i} | \overline{x}_j = \overline{x}_k\} \tag{6.11}$$

with M_i the number of bullets for the time interval $[t_i, t_{i+1}[$ and $(\overline{x}, \overline{v})_i$ the phase space vector (position-velocity) of bullet i, is defined by

$$f(t,x) \equiv((\overline{v},0)_1,...,(\overline{v},0)_{M_i}) + \frac{q_0}{m_0}((0,\overline{E}_s(t,\overline{x}_1) + \sum_{i=2}^{M_i} \frac{Q}{4\pi\epsilon_0}||\overline{x}_1 - \overline{x}_i||),..., (0,\overline{E}_s(t,\overline{x}_{M_i}) + \sum_{i=1}^{M_i-1} \frac{Q}{4\pi\epsilon_0}||\overline{x}_{M_i} - \overline{x}_i||)) \quad (6.12)$$

where q_0, m_0 are respectively the charge and the mass of a proton and Q the charge of bullet.

Now let us try to show the uniqueness and the existence of a solution for a time interval $[t_i, t_{i+1}[$ using well-known theorems of differential equations theory recalled in Appendix section 10.8. First, let us prove that U_2 is an open subset. Let us take $x \in U_2$ and let us show that there exists $\delta > 0$ such that $B(x,\delta) \in U_2$. Since $x \in U_2$ it means that $||\overline{x}_i - \overline{x}_j||_2 \neq 0, \forall i \neq j$. Let us define $b \equiv \min\{||\overline{x}_i - \overline{x}_j||_2 | i \neq j\}$. Let us assume that the norm chosen for the space $\mathbb{R}^{6M}$ is defined by $x \in \mathbb{R}^{6M}, ||x|| \equiv \sqrt{||\overline{x}_1||_2^2 + ||\overline{v}_1||_2^2 + ... + ||\overline{v}_M||_2^2} = ||x||_2$. Now if $||x-y|| < \frac{b}{2}$ it implies that $||\overline{x}_i - \overline{y}_i||_2 < \frac{b}{2}\ \forall i$. Thanks to the triangular inequality, one has $||\overline{x}_i - \overline{x}_j||_2 \leq ||\overline{x}_i - \overline{y}_i||_2 + ||\overline{y}_i - \overline{y}_j||_2 + ||\overline{y}_j - \overline{x}_i||_2$. This leads to $0 < ||\overline{y}_i - \overline{y}_j||_2, \forall i \neq j$ i.e $B(x, \frac{b}{2}) \subset U_2$.

Unfortunately, the interval $[t_i, t_{i+1}[$ is not an open subset of $\mathbb{R}$. However, since the application f is defined and C^1 on $[t_i, t_{i+1}[\times U_2$, one can extend the domain and the function as follows $U \equiv]t_i - (t_{i+1} - t_i), t_{i+1}[\times U_2$

$$f_2(t,x) \equiv \left\{ \begin{array}{ll} f(t,x) & (t,x) \in [t_i, t_{i+1}[\times U \\ f(t_i + (t_i - t), x) & (t,x) \in]t_i - (t_{i+1} - t_i), t_i[\times U \end{array} \right\} \quad (6.13)$$

This application is still C^1 and therefore a local continuous Lipschitz function in x. One knows that for a given (t_i, x_i) there exists an unique maximal solution (see Appendix section 10.8). Let us prove that this solution is global i.e it exists on the whole time domain $]t_i - (t_{i+1} - t_i), t_{i+1}[$. Let us assume that $u :]a, b[\rightarrow \mathbb{R}^n$ is the maximal solution with $a > t_i - (t_{i+1} - t_i)$ and $b < t_{i+1}$. One may notice that if the solution is maximal then the interval must be an open subset. Indeed, if it was not the case, then the solution could be extended locally (there exists a safe cylinder around the limit point) and therefore the solution would not be maximal. Now, let us prove that $b = t_{i+1}$ and by symmetry the result will be the same for a. Let

us take a monotone series $t_i \to b$ and let us prove that $u(t_i) \in K$ where K is a compact of $\mathbb{R}^n$. This comes from the fact that the forces can be bounded for any time in this interval. Using a energy conservation principal, one can obtain the minimum distance between two particles i.e the maximum field due to the Coulomb forces on a given particle. Since the source field is a periodic bounded function, the overall force is bounded. Additional informations on this statement can be found in Appendix section 10.6. The forced being bounded, the function f is bounded too. Therefore, there exists $l \in mathdsN$ such that $u(t_i) \in K, \forall i > l$ where K is a compact. Indeed, let $L = \sup_{t \in]t_i-(t_{i+1}-t_i),t_{i+1}[, x \in U_2} ||f(t,x)|| < \infty$ be the upper bound one has

$$\begin{aligned} u(t_i) &= u(t_l) + \int_{t_l}^{t_i} f(t',x)dt' && (6.14)\\ &\leq u(t_l) + \int_{t_l}^{t_i} |f(t',x)|dt' && (6.15)\\ &\leq u(t_l) + (t_l - t_i)L && (6.16)\\ &\leq u(t_l) + (t_{i+1} - (t_i - (t_{i+1} - t_i))L && (6.17) \end{aligned}$$

Hence, there exists a safe cylinder around $u(b)$. Since the solution passing through $u(b)$ is unique, the proof is complete.

6.3 Selection of sorting method

Sorting particles into cells requires to calculate for each particle the cell containing it and then place the particle in the cell container. There is no need to compare the positions of the particles between them i.e there is no need to sort the particles in a given cell. When using cells of same size, it is straightforward to calculate the position of a given particle in the appropriate cell

$$index = \frac{x^c_{3,i} - X_{3,left}}{CS} \tag{6.18}$$

where $X_{3,left}$ is the extreme left corner of the accelerator (the smallest x_3 coordinate of the accelerating tunnel as shown Fig. 6.3), CS is the cell size and $x^3_{3,i}$ is the $\hat{x}'_3$ coordinate of the bullet i. This computation is of linear complexity with the number of bullets. The tricky part consists in copying the bullets into their cell container in a multithreaded way. The following subsections explained how to achieve such a sorting algorithm called the radix-sort algorithm.

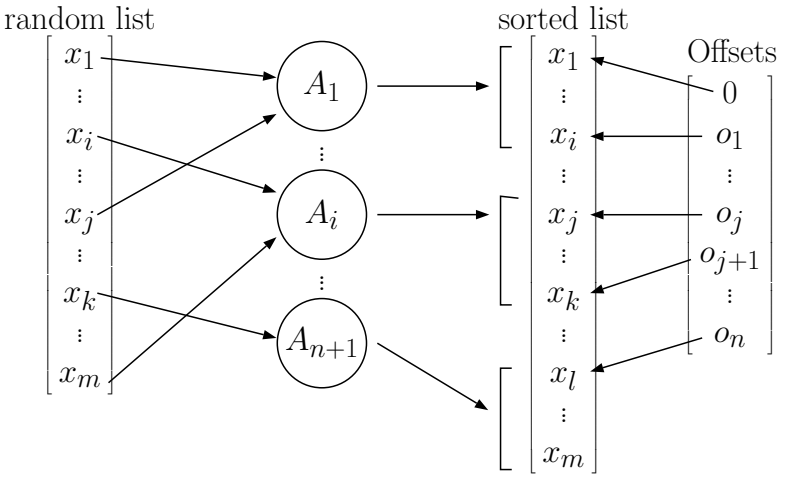


Figure 6.4: The sequential radix sort algorithm.

Let $A_1, ..., A_n \subset \mathbb{R}^3$, be some regions of the 3-dimensional Euclidean space with $A_i \cap A_j = \emptyset$ if $i \neq j$. Let us define $A \equiv \cup_{i=1}^{n} A_i$ and let $A_{n+1} \equiv \mathbb{R}^3 \backslash A$ be the rest. Let be $x_1, ..., x_m \in \mathbb{R}^3$ some points in the 3-dimensional space. Let us define a pre-order relation on $\mathbb{R}^3$ as follows,

$$x \leq y \Leftrightarrow x \in A_i, y \in A_j, i \leq j \tag{6.19}$$

The radix sort algorithm consists in sorting a given list of coordinates according to the pre-order relation defined here-above.

The sequential radix sort algorithm consists in

- counting the number of elements respectively inside each space domain A_i
- creating an offset vector that holds the current free position for an element in the sorted list respectively for each domain
- copying the elements in the new list with the help of the offset vector computed at the previous step

Fig. 6.4 illustrates the functioning of the sequential radix sort algorithm.

6.4 A parallel implementation of the motion solver for GPU

6.4.1 Introduction

The algorithm consists in 4 different parts:

- Sort particles into cells
- Compute forces
- Integrate with an explicit integral method
- Render the results and/or save some results

In the following subsections each steps of the solver implementation are explained in details. Most of the kernel codes are shown and explained. The host code is not shown since it is does not add any important implementation subtlety. GPU hardware constrains are considered in order to have an efficient implementation. The first subsection introduces the data structures used on the GPU. The radix sort algorithm implementation is explained in the second subsection. The GPU implementation of Runge-Kutta explicit integration method of order n is provided in the third subsection. Finally, the space charge computation is addressed in the fourth and last subsection.

6.4.2 Data structure

There are several data structures used by the kernels. First two data structures that contains the position, velocity and cell position of each particles are created. This structure is called the Cinematic Structure of the Particles (CSP) and is presented in Fig. 6.7. A single CSP is active at a given time step of the integration process. The maximum number of particles that can be injected is set by the user. This determines with the preferred group size the total size of these two twin memory blocks (the closest upper value that is a multiple of the preferred group size). The sort algorithm takes the former active CSP, sorts the data and copies them into the new active CSP. Then the Runge-Kutta integration is performed for a single time step and the CSP data are updated. The Runge-Kutta integration uses the kernel responsible for the electric field computation.

The radix-sort algorithm uses two data structures to store the counts and offsets of particles in each cell. There is one data structure said to be global and one data structure said to be private to a work item. A data structure gathering all informations concerning the counts and offsets of

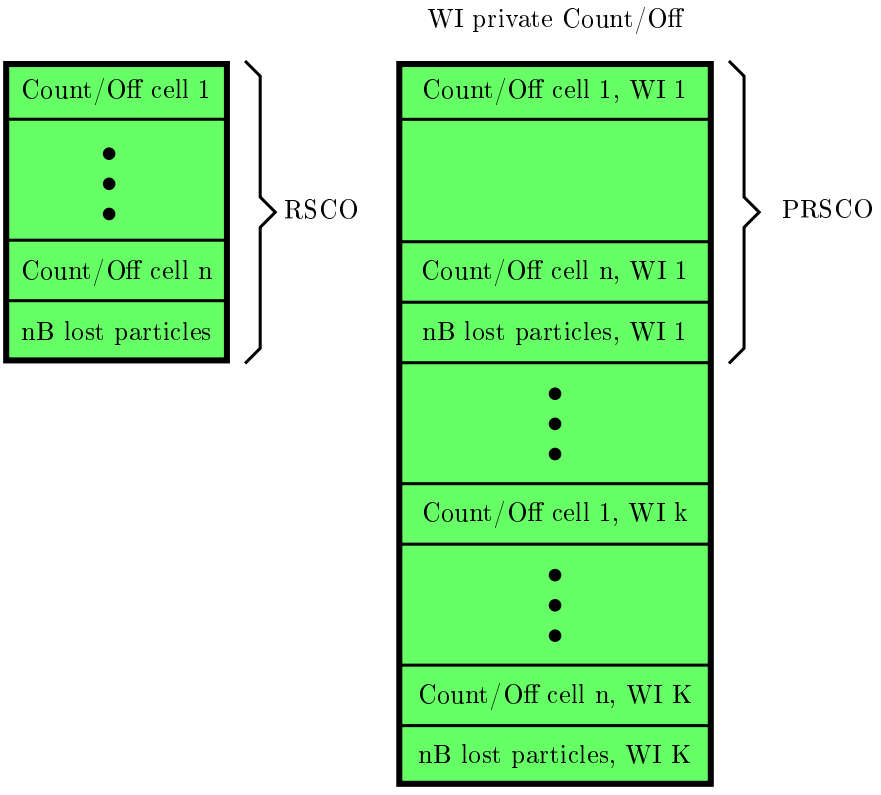


Figure 6.5: Radix Sort internal data structures.

particles for the total set of cells (including the complementary cell for the lost particles) is called Radix-Sort Counts and Offset (RSCO). When this structure is private to a thread (or equivalently to a work item) the structure is called Private Radix-Sort Counting and Offset (PRSCO). These data structures are presented in Fig. 6.5. Futher explanations are provided in the subsection related to the radix-sort algorithm 6.4.3.

The Runge-Kutta algorithm uses a data structure to calculate the new positions and velocities of all the particles. Depending on the order of the method, the data structure size is larger or smaller. An explicit Runge-Kutta method (see [65]) of a given order needs to evaluate at each integration step one or several evaluations of the function f defined by 6.12. Actually, the explicit RungeKutta methods of order higher than one real-

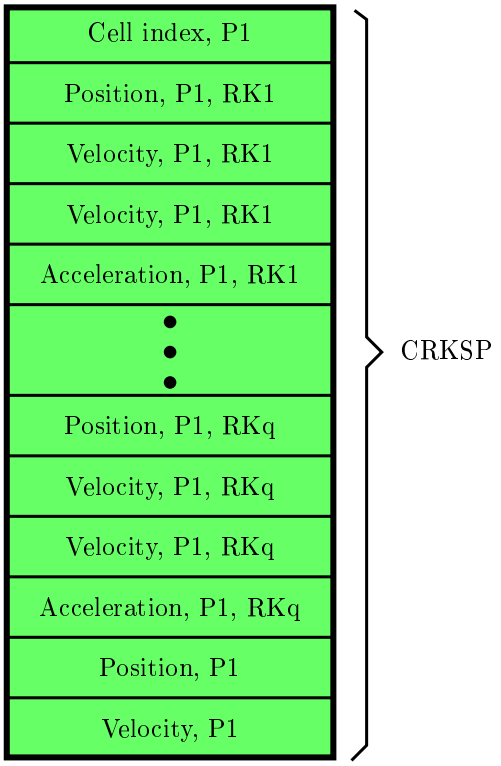


Figure 6.6: Runge Kutta internal data structure.

ize several intermediates (or local) integration steps. These intermediate integration steps are used to evaluate the current global integration step (see section 6.4.5). Let $(f(x_1, t_1), ..., f(x_k, t_k))$ be k evaluations of f during a Runge-Kutta integration step. The function variable is defined by $x_i = (\overline{x}_{1,i}, \overline{v}_{1,i}, ..., \overline{x}_{M,i}, \overline{v}_{M,i})$ where the index $i \in [1, ..., q]$ refers to the current intermediate Runge-Kutta integration step in the current global integration step. Its derivative is defined by $\dot{x}_i = (\overline{v}_{1,i}, \overline{a}_{1,i}, ..., \overline{v}_{M,i}, \overline{a}_{M,i})$. All the data relative to a bullet are gathered together in a contiguous region of memory for optimal cache use in the final integration step where all the intermediate results are used. For each particle, the cell index is copied too. This structure is called the Cinematic Rung-Kutta Structure of a Particle (CRKSP) and is presented in Fig. 6.6. The velocity appears twice for consistency and uniformity reasons and in order to be used with the general Runge-Kutta implemenation.

Fig. 6.7 shows the data flow between the different tasks. Each data takes 4 bytes (a word for present GPU) i.e a float is 4 bytes and an unsigned integer is 4 bytes too.

6.4.3 Radix sort strategy and implementation

6.4.3.1 GPU radix sort

Let us partition the set of points into subsets. Let $S_1, ..., S_k \subset \mathbb{N}$ be k subsets of indexes of particles such that $S_i \cap S_j = \emptyset$ if $i \neq j$ and $[1, ..., M] = \cup_{j=1}^{k} S_k$, where M is the total number of particles. Each subset is associated to a thread. The first operation consist in calculating the total number of elements in each domain so that one can foreseen the necessary amount of space for each domain. Using separate threads to calculate the domain associated to given point is an independent task and therefore fully parallelisable problem. However, incrementing a given count variable associated to a domain from different threads cannot be performed without racing protection. Indeed, even if each cache level were coherent, incrementing a memory block from multiple threads would require a lock that allows only one thread to run at a time the operations protected by it. Cache coherency such as, for instance, the MESIF protocol (see [66], [67]) on Intel CPU only ensures that at any given time the data is seen coherently by each thread in memory. Fig 6.8 shows a piece of code where at some point a variable at address $\&i$ is incremented by a value of one. The pseudo-assembly code shows what are the instructions a classic CPU would have to perform: first fetch the value of $\&i$ in main memory and place it in register R_k, then increment the value of the register R_k by one and finally store the value of register R_k back into memory. Imagine now that this piece of code is running on many threads simultaneously. Imagine the OS scheduler stops a given thread k just after the read operation. The other threads keep running and updating the value at the address $\&i$. When the thread k resumes and fetch its context on the stack, the value hold by the register R_k is an old value but it will be used to perform the incrementation of the variable. This shows the need to use of a exclusive lock system such as mutex to ensure the correctness of the operation.

Now, imagine another situation where all the threads run on a separate physical core and that the scheduler is programmed to let them run till completion (no context switching then). Let us assume that the system is cache coherent for instance a MESIF cache system. Each memory operation on the address $\&i$ is coherent at any time. Let us imagine now that two threads are perfectly synchronized i.e the load operations are performed at the same time. If all caches are in shared mode (a mode that indicates a

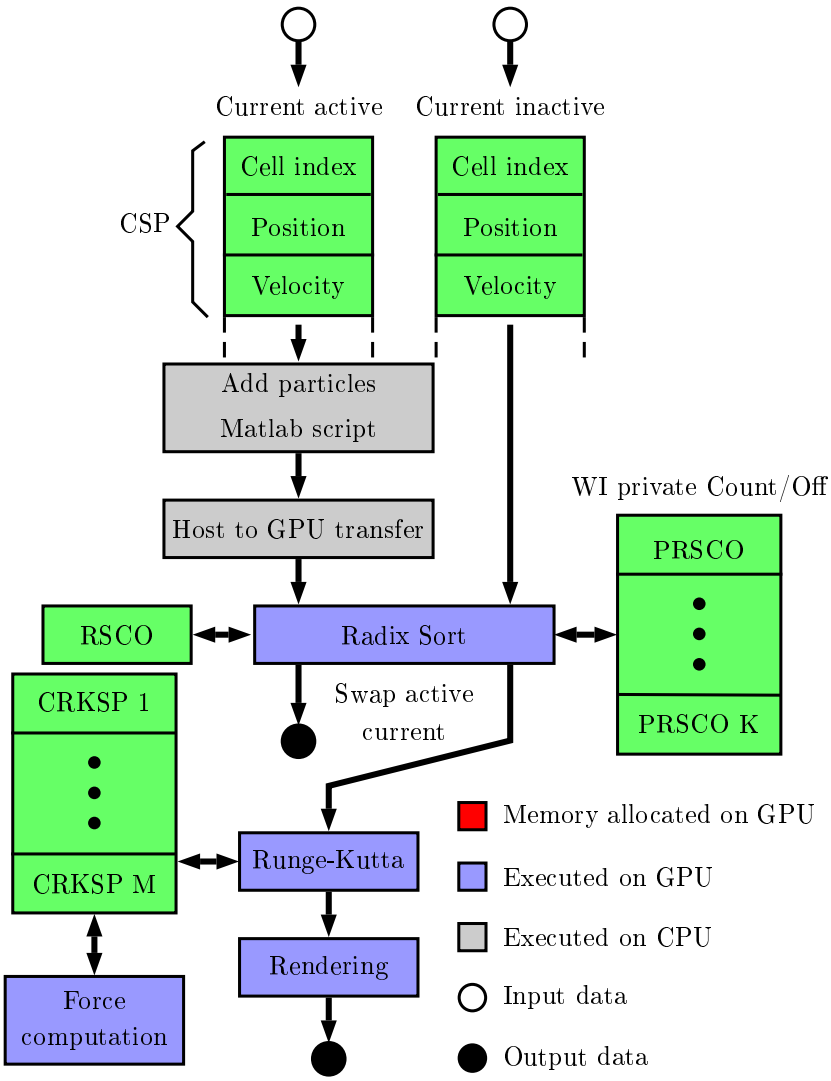


Figure 6.7: Global data definitions and data flow between tasks.

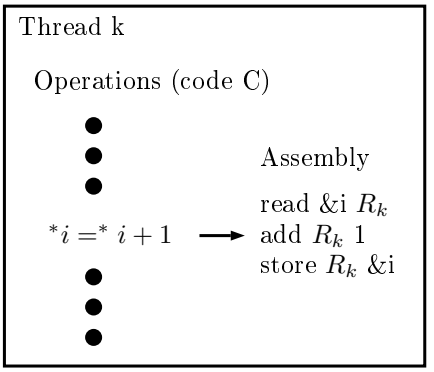


Figure 6.8: An example of code with an increment operation performed in several threads.

coherent share value), then the read operation sees the same value in both synchronized threads. Therefore, the updated value if $\&i$ after the write operations of the two threads ends up to be the value of $\&i$ incremented of one instead of two. Fig. 6.9 illustrates this phenomenon on a 32 bits x86 system for a direct cache system (see for instance [68]). Two CPU fetches at the same time the value pointing by $\&i$. The L1 caches return both the value 5 since they are in a shared status for this cache line. Both CPU then increment this value by one before writing it back to cache. The final value in memory will be therefore 6 instead of 7 if the increment operations were not executed at the same time. This shows again the need in this case of an exclusive lock system.

The need of a mutex for the increment operation slows down a lot the parallelization. In fact, if the number of operations before the increment operation is relatively small, the multi-threaded program does not gain any speed as the number of threads used for this task increases. Indeed, most of the threads will be stuck waiting for the lock region of memory to be available to complete the operation. Therefore a naïve implementation of the counting is highly inefficient.

A solution to this problem is to use a private counter for each thread. Each value stored in these counters are then summed up sequentially. On a GPU, each a thread has a limited amount of private memory. If this amount of private memory is higher than the available number of registers then the global memory (through the L2 cache and optionally through the L1 cache) is used. A solution to this problem would be to use the local

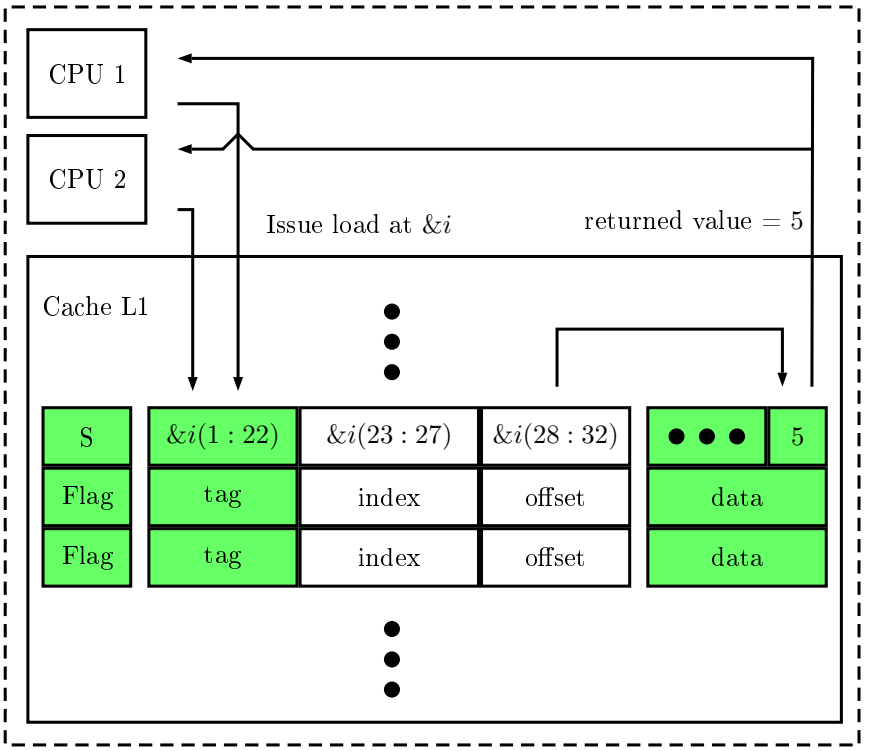


Figure 6.9: An example of a synchronized load operation performed by two different threads in an incrementation process on a 32 bits x86 CPU with a MOSIF cache system.

memory but the local memory is not a large memory either. Let us assume the accelerator is divided in 1000 cells and in each half-SM 64 threads use the registers. The counter are 4 bytes memory block. Therefore, the amount of memory required for the counter is 250KB. It's more than the available amount of registers and shared memory on a half-SM in the pascal architecture. But, the list is almost sorted at any time (indeed, a particle enter and exit its cell by its two direct neighbours). Hence, if the subsets of indexes of particles are chosen such that the indexes in each of them correspond to particles close (in space) from one another, one can assume that close threads are going to use close counters. Therefore, using global cached memory for these counters is acceptable since it will unlikely cause many cache misses. To summarize, a first kernel using its own global memory counters is used to calculate the number of particles in each cell domain A_i. Then an other kernel is used to sum-up each local thread counters associated to the same region of space A_i. This kernel might be inefficient in terms of memory access. However, the number of domains being much smaller than the number of particles, this thread has to do a negligible job. For the same reason, the creation of an offset vector for each space domain A_i is also perform sequentially by an other kernel. Copying the elements using a massive amount of threads (more than the number of domains) require to have for each thread its own offsets table. Indeed, let us consider the offset counter of a given region A_i. This offset has to be incremented each time a data is copied in its allocated memory. Therefore, an exclusive lock would have to be used to protect the incrementations of this offset variable but also to protect the copy operations performed by each thread. In order to avoid this problem, a local offsets table is created for each thread. The local counters used in the first kernel contain for each domain the number of elements for their local subsets of indexes of particles. For each domain, one associates a thread that sums up these local value taking into account the global value calculated with the second kernel.

Now each kernel are going to be explained step by step. The memory occupation for the counters and offset counters and their relations with the kernels is shown Fig 6.10.

The first kernel is shown in the code section 6.1 and it takes 7 arguments namely: the pointer to the position of the particles "pv", the pointer to local counters (counters and offsets data are pointing toward the same memory region) "offset", the number of domains (or cells) "nBDomains" which does not account for the outside domain $A_{n+1} \equiv \mathbb{R}^3 \backslash \cup_{i=1}^{n} A_i$, the number of particles (elements) handle by a single work item (by a single thread) "nBEWI", the extreme left Z coordinate of the accelerator "Z_left" as defined Fig. 6.2.1, the size of a cell "dZ" and the radius of a cell "ra-

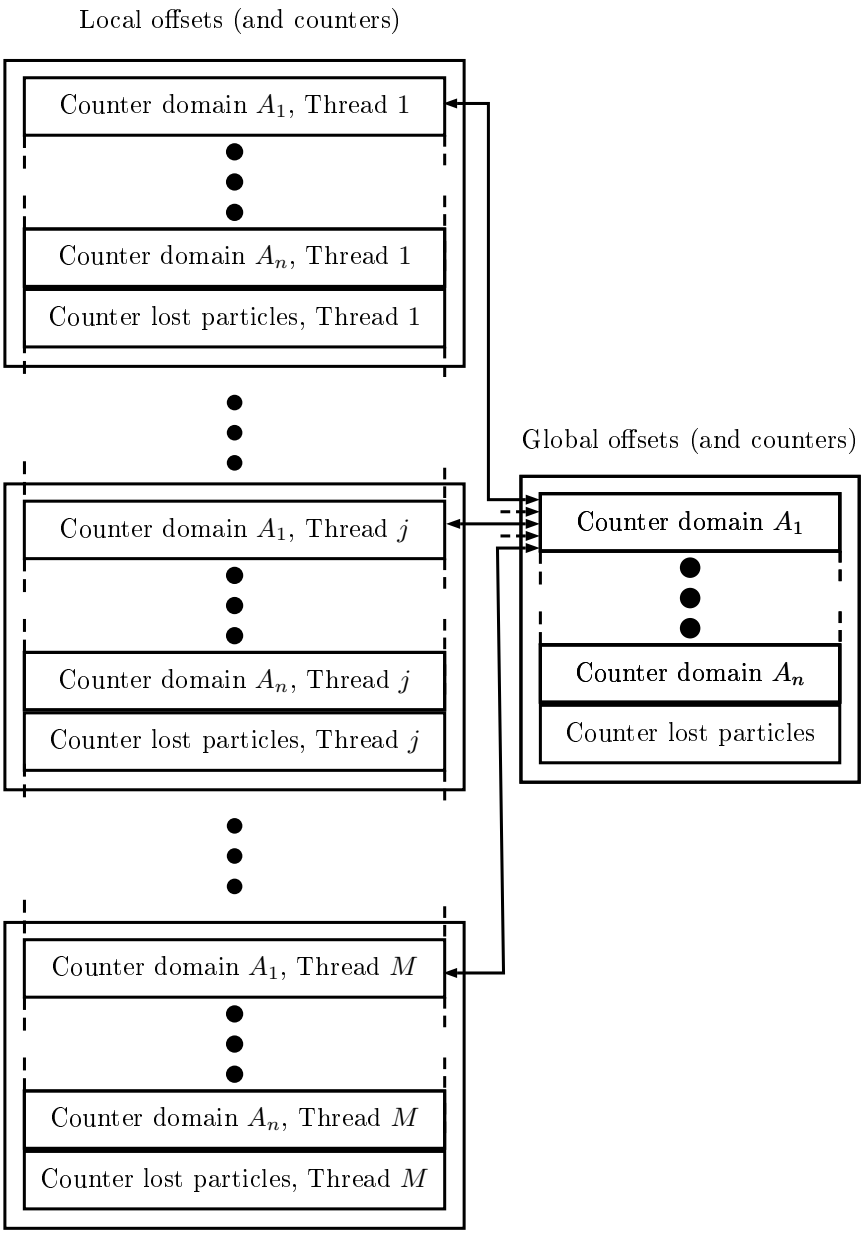


Figure 6.10: Data flow between local and global variables in the GPU radix sort algorithm.

dius". The first line of code fetches from the private registers of the work item its index for the dimension 0 which is the only dimension used here. Then the pointers for the particle positions and for the offsets are placed to point into the memory region foreseen for the current thread. Since there are "nBEWI" particles processed by a given thread and since the memory space used to store the position, velocity and current cell index of a given particle represents 7 words of 4 bytes the number of 4 bytes words the position pointer has to be incremented is given by $nBEWI * 7 * index$. For the counter pointer, the work item index is multiply by the number of domains added by a value of 2. The number of domains including the rest domain is given by $n = nBDomains + 1$ and an additional integer is used to store the number of particles for which the radial component exceeds the cell radius (in order to track particles lost inside the accelerator). The first operation on the data consists in initializing the counters to zero. Then for each particle handled by the work item "index", the cell position of each particle is calculated. Once the cell is found, the local counter relative to this cell is incremented.

Code 6.1: Kernel 1 of radix sort algorithm

```
__kernel void ParticleRadixSortCounting(__global float *pv,
                                        __global unsigned int *offset,
                                        unsigned int nBDomains,
                                        unsigned int nBEWI,
                                        float Z_left,
                                        float dZ,
                                        float radius){

    int index = get_global_id(0);

    pv += 7*index*nBEWI;
    offset += index*(nBDomains+2);

    for(unsigned int i=0;i<=nBDomains+1;i++){
        offset[i] = 0;
    }

    for(unsigned int i=0;i<nBEWI;i++){
        int z_pos = convert_int_sat((pv[3] - Z_left)/dZ);
        float R = sqrt(pv[1]*pv[1]+pv[2]*pv[2]);
        if(z_pos < nBDomains && z_pos >=0){
            if(R < radius){
                offset[z_pos] += 1;
            }
            else{
                offset[nBDomains+1] += 1;
                offset[nBDomains] += 1;
            }
        }
        else{
            offset[nBDomains] += 1;
        }
        pv += 7;
```

```
    }
}
```

The second kernel is relatively simple and is shown in 6.2. It takes 4 arguments namely: the counters pointer "offset", the global counters pointer "domain_offset", the number of cells which does not account for the outside cell and the number of work items (threads) launched for the first kernel. A work item is created for each domain (hence for $nBDomains + 1$). Each work item sums up the local counters corresponding to the cell associated to it and also for the lost particles counter which is an independent counter.

Code 6.2: Kernel 2 of radix sort algorithm

```
__kernel void ParticleRadixSortOffset1(__global unsigned int *offset,
                                       __global unsigned int *
                                           domain_offset,
                                       unsigned int nBDomains,
                                       unsigned int nBWI){

    int index = get_global_id(0);

    offset += index;

    unsigned int total = 0;
    for(unsigned int i=0;i<nBWI;i++){
        total += offset[0];
        offset += (nBDomains+2);
    }

    domain_offset[index] = total;
}
```

The third kernel shown in the code section 6.3 is also very straightforward. It takes 2 arguments namely the global domain offset "domain_offset" and the number of domains "nBDomains". The kernel is run only on a single core and it performs a very little job in comparison the the overall job. One may think this job could be performed on the CPU. However, performing the work on the CPU requires a memory transfer which finally would cost more than just running this sequential work on the GPU. It creates the offsets for each cell. The offset of a given cell is equal to the offset of the previous cell plus the amount of particles in the previous cell.

Code 6.3: Kernel 3 of radix sort algorithm

```
__kernel void ParticleRadixSortOffset2(__global unsigned int *
    domain_offset,
                                       unsigned int nBDomains){
    unsigned int next = 0;
    for(unsigned int i=0;i<=nBDomains;i++){
        unsigned int count = domain_offset[i];
        domain_offset[i] = next;
```

```
        next += count;
    }
}
```

The fourth kernel of the radix sort algorithm is shown in the code section 6.4 and it takes the same arguments as the second kernel. Its goal is to calculate the private offsets of each work item. Each domain is associated to a work item. For a given domain, the kernel simply calculates the local offset value of each subset of indexes of particles. The offset of a given cell A_i in a work item j (corresponding the the subset of indexes of particles j) is simple given by the local offset position of the cell A_i in the work item $j-1$ plus the number of particles in the domain A_i of the work item $j-1$. For the first work-item and for a given domain A_i, the local offset is given by the global offset.

Code 6.4: Kernel 4 of radix sort algorithm

```
__kernel void ParticleRadixSortOffset3(__global unsigned int *offset,
                                       __global unsigned int *
                                           domain_offset,
                                       unsigned int nBDomains,
                                       unsigned int nBWI){

    int index = get_global_id(0);
    offset += index;

    unsigned int count = offset[0];
    if(index <= nBDomains){
        offset[0] = domain_offset[index];
    }
    else{
        offset[0] = 0;
    }
    unsigned int next = offset[0] + count;

    for(unsigned int i=1;i<nBWI;i++){
        offset += nBDomains+2;
        count = offset[0];
        offset[0] = next;
        next += count;
    }
}
```

Finally, the fifth kernel is shown in the code section 6.5 and takes 10 arguments: the source pointer "pv_src" and destination pointer "pv_dst" pointing to respectively the active and inactive CSP data, the destination pointer for storing the positions of the lost particles "lp_dst", the local offsets pointer "offset", the address of the first available memory block for storing the lost particles in the memory block allocated for them, the number of domains "nBDomains", the number of particles (elements) handled by each work item "nBEWI", the extreme left position of the accelerator "Z_left", the cell size "dZ" and the cell radius. The first operations on the

pointers are the same as the one performed for the first kernel. Then, as in the first kernel, the cell position is found for each particles handled by a work item. Once the cell is found, the private offset value for this cell is used to determine the new position of the particle CSP data in memory. The lost particles are also copied in their specific memory region using the entry associated to the rest domain in the offsets table. It is not important to know the cinematic informations of the lost particles. Therefore, instead of copying a full CSP data for the rest cell, a default identical value is written avoiding some read operations for the velocities and the cell positions. It is important to write a CSP data even for lost particles since the total number of threads launched on the GPU does not match in general the exact number of particles in the accelerator. Indeed, as explained in the former chapter section 5.3.3, each group must have the same size and each group size should be a multiple of the preferred group size. Since there is a single dimension used here, the total number of work items must be a multiple of the preferred group size.

Code 6.5: Kernel 5 of radix sort algorithm

```
__kernel void ParticleRadixSortMoving(__global float *pv_src,
                                      __global float *pv_dst,
                                      __global float *lp_dst,
                                      __global unsigned int *offset,
                                      unsigned int start_index_l,
                                      unsigned int nBDomains,
                                      unsigned int nBEWI,
                                      float Z_left,
                                      float dZ,
                                      float radius){

    int index = get_global_id(0);

    __global float *src = pv_src + 7*index*nBEWI;
    offset += index*(nBDomains+2);

    for(unsigned int i=0;i<nBEWI;i++){
        int z_pos = convert_int_sat_rtz((src[3] - Z_left)/dZ);
        float R = sqrt(src[1]*src[1]+src[2]*src[2]);
        if(z_pos < nBDomains && z_pos >=0){
            if(R < radius){
                __global float *dst = pv_dst + 7*offset[z_pos];
                #pragma unroll
                for(unsigned int j=1;j<7;j++){
                    dst[j] = src[j];
                }
                __global unsigned int *pd = (__global unsigned int*)(dst)
                    ;
                pd[0] = z_pos;
                offset[z_pos] += 1;
            }
            else{
                __global float *dst = pv_dst + 7*offset[nBDomains];
                __global float *l_dst = lp_dst + 3*(offset[nBDomains+1]+
                    start_index_l);
```

```
                l_dst[0] = src[1];
                l_dst[1] = src[2];
                l_dst[2] = src[3];

                #pragma unroll
                for(unsigned int j=1;j<7;j++){
                    dst[j] = 0.0f;
                }
                dst[3] = 2.0f * Z_left;

                __global unsigned int *pd = (__global unsigned int*)(dst)
                    ;
                pd[0] = nBDomains;
                offset[nBDomains] += 1;
                offset[nBDomains+1] += 1;
            }
        }
        else{
            __global float *dst = pv_dst + 7*offset[nBDomains];
            #pragma unroll
            for(unsigned int j=1;j<7;j++){
                dst[j] = 0.0f;
            }
            dst[3] = 2.0f * Z_left;
            __global unsigned int *pd = (__global unsigned int*)(dst);
            pd[0] = nBDomains;
            offset[nBDomains] += 1;
        }

        // Set old data to a consistent non-used vector
        #pragma unroll
        for(unsigned int j=1;j<7;j++){
            src[j] = 0.0f;
        }
        src[3] = 2.0*Z_left;
        __global unsigned int *pd = (__global unsigned int*)(src);
        pd[0] = nBDomains;

        src += 7;
    }
}
```

A similar implementation of the radix sort algorithm can be found in [69]. But, since the SM are limited in amount of local memory, for large number of cells the implementation proposed in this paper fails. As explained above, in our case we can let the L1 cache handling the accesses for the counts and offsets tables in global memory without any significant additional computation time since the lists are almost sorted at any time. Since our approach uses the global memory, the amount of memory being much larger, one does not suffer from large number of cells.

6.4.4 Electric field computation

The electric field felt by each particle is divided into two independent components: The source field and the space charge field. The first sub-subsection explains the strategy used to calculate the source electric field and the sec-

ond sub-subsection explains the implementation of the Coulomb electric field computation.

6.4.4.1 The source field computation

In order to avoid the heavy computation of the source field for each new positions of the particles, the source field is calculated on a predefined grid and the data are then linearly interpolated with a 3-linear Lagrangian technique explained in section 6.2.5. Since the particles might escape the internal cells of the accelerator during the Runge-Kutta process, the grid must exceed the dimension of these cells in order to avoid multiway branches in the kernel.

The code section 6.6 shows the kernel responsible for computing the source field. It takes 9 arguments namely: A pointer to the CRKSP blocks "position", a pointer to the electric field values "in_filed", an offset to point to the right position in the CRKSP block "position_offset" depending on the intermediate Runge-Kutta step, the size of a CRKSP block "position_block_size", the phase of the field at the current integration time "phase", the extreme lower left point coordinates of the field samples grid given by the pointer "F_P_low", the memory location of the vector containing the grid steps for $\hat{x}_1, \hat{x}_2, \hat{x}_3$ given by the pointer "F_step" and finally the number of samples for $\hat{x}_1, \hat{x}_2, \hat{x}_3$ given by the pointer "F_SS". One may notice that all the loops have a constant number of iterations. Therefore, many loop unrolling has been performed to improve the performances of the code. The implementation is relatively straightforward and consists basically in the translation of the formula provided by proposition 10.4.1.

Code 6.6: Electric field interpolation kernel

```
__kernel void EFieldInterpolation(__global float *position,
                                  __global double2 *in_field,
                                  unsigned int position_offset,
                                  unsigned int position_block_size,
                                  float phase,
                                  __global float *F_P_low, // Field lower
                                       coordinates
                                  __global float *F_step, // Field step
                                  __global int *F_SS, // Field sample
                                      size
                                  float Q){

    int index = get_global_id(0);

    position += position_block_size*index+position_offset;

    float P[3];
    int f_pos[3];
    int bound[3];
    #pragma unroll
```

```
    for(int i=0;i<3;i++){
        P[i] = position[i];
        f_pos[i] = convert_int_sat((P[i] - F_P_low[i])/F_step[i]);
        bound[i] = F_SS[i]-2;
    }

    float Field[3] = {0.0f,0.0f,0.0f};

    if(f_pos[0] < bound[0] && f_pos[0] > 0 && f_pos[1] < bound[1] &&
         f_pos[1] > 0
       && f_pos[2] < bound[2] && f_pos[2] > 0){

        float FPA[6]; // Field position index around the sampling point
        #pragma unroll
        for(int i=0;i<3;i++){
            FPA[i] = F_P_low[i] + f_pos[i] * F_step[i];
            FPA[3+i] = F_P_low[i] + (f_pos[i]+1) * F_step[i];
        }

        for(int i_x=0;i_x<2;i_x++){
            for(int i_y=0;i_y<2;i_y++){
                for(int i_z=0;i_z<2;i_z++){

                    float E[3];

                    int index = 3*(i_z + f_pos[2]
                                + (f_pos[1] + i_y + (f_pos[0]+i_x) * F_SS
                                    [1]) * F_SS[2]);

                    E[0]=in_field[index].s0 * cos(phase) - in_field[index
                        ].s1 * sin(phase);
                    E[1]=in_field[index+1].s0 * cos(phase) - in_field[
                        index+1].s1 * sin(phase);
                    E[2]=in_field[index+2].s0 * cos(phase) - in_field[
                        index+2].s1 * sin(phase);

                    #pragma unroll
                    for(int j=0;j<3;j++){
                        if(i_x == 0)
                            E[j] *= (P[0] - FPA[3])/(FPA[0] - FPA[3]);
                        else
                            E[j] *= (P[0] - FPA[0])/(FPA[3] - FPA[0]);

                        if(i_y == 0)
                            E[j] *= (P[1] - FPA[4])/(FPA[1] - FPA[4]);
                        else
                            E[j] *= (P[1] - FPA[1])/(FPA[4] - FPA[1]);

                        if(i_z == 0)
                            E[j] *= (P[2] - FPA[5])/(FPA[2] - FPA[5]);
                        else
                            E[j] *= (P[2] - FPA[2])/(FPA[5] - FPA[2]);

                        Field[j] += E[j];
                    }
                }
            }
        }
    }

    position += 9;

    position[0] = Q * Field[0];
```

```
    position[1] = Q * Field[1];
    position[2] = Q * Field[2];
}
```

6.4.4.2 The space charge computation

The implementation of the space charge computation were done with two different strategies. The main difference between them lies in the use of local memory for speeding up the computation. As it will be explained, ussing local memory in our situation didn't improve the computation time and on the contrary harmed the performances mainly because of the extra computation time required to enqueue more kernels. Let us first start with the approach avoiding the use of local memory.

The code in section 6.7 shows the implementation of the kernel. The kernel takes 6 arguments namely: The CRKSP blocks pointer "data", the global offsets "offset", the size of a whole CRKSP block, the number of domains minus the rest domain "nBDomains", the constant K that is equal to $\frac{Q^2}{4\pi\epsilon_0}$ where Q is the total charge of a bullet, ϵ_0 is the vacuum permittivity.

Each particle is assigned to a work item (some fake particles might be used to use a number of work-items multiple of the preferred size). The index of the particle is fetched by the first operation and stored in the index private variable. The "test" pointer is set to point to the particle CRKSP structure. The domain index is then retrieved at the top of the structure and stored in the "index_t" variable. Once the cell is retrieved, the test pointer is placed at the particle position thanks to the dataOffset variable that depends on the order to the Runge-Kutta method and the index of the intermediate integration step. The map variable will be used in a loop to visit the neighbour cells and its own cell. The private variable R is used to store the distance between two particles, F variable is used to store the total space charge field and the T variable is used to store the position of the current particle. The first loop is used to visit the 3 cells closest cells surrounding the particle. The index value "k" contains the current cell index being visited. A test is performed because of the two extreme cells which respectively do not contain a left and right neighbour. Then, the position of each particle inside the current cell is fetched and used to calculate the Coulomb field. The offset RSCO structure obtained with the radix-sort algorithm is used to get the lower and upper indexes of particles inside the cell "k". In order to limit the round-off inaccuracy when summing up the electric field contributions, the two neighbour cells are first visited since they will likely provide smaller values than the self cell of the particle.

Code 6.7: Space charge kernel implementation

```
__kernel void ParticleInteraction2(__global float *data,
                                   __global unsigned int *offset,
                                   unsigned int blockSize,
                                   unsigned int nBDomains,
                                   unsigned int dataOffset,
                                   float K){

    int index = get_global_id(0);

    __global float *test = data + blockSize*index;

    __global unsigned int *domain_pt = (__global unsigned int *)(test);
    int index_t = (int)(domain_pt[0]);

    test += dataOffset; // Placing test to the right phase space data

    int map[3]={-1,1,0};

    float R[3];
    float F[3] = {0.0f,0.0f,0.0f};

    float T[3];
    T[0] = test[0];
    T[1] = test[1];
    T[2] = test[2];
    for(int i=0;i<3;i++){
        int k = map[i] + index_t;
        if(k>=0 && k < (int)(nBDomains)){
            unsigned int low = *(offset+k);
            unsigned int up = *(offset+k+1);
            __global float *source = data + low*blockSize + dataOffset;

            for(int j=low;j<up;j++){
                R[0] = T[0] - source[0];
                R[1] = T[1] - source[1];
                R[2] = T[2] - source[2];
                float r = sqrt(R[0]*R[0]+R[1]*R[1]+R[2]*R[2]);
                if(r != 0.0f){
                    float r3 = r*r*r;
                    F[0] += R[0]/r3;
                    F[1] += R[1]/r3;
                    F[2] += R[2]/r3;
                }
                source += blockSize;
            }
        }
    }

    F[0] *= K;
    F[1] *= K;
    F[2] *= K;

    test += 9;
    test[0] += F[0];
    test[1] += F[1];
    test[2] += F[2];
}
```

One may wonder if local memory could improve the computation time. Each work item is going to access many times the global memory to fetch the positions of the particles in the 2 or 3 cells surrounding the particle associated to it. If the work items are executed in random way at different stages of their executions, the global accesses might end up in very far location from each other i.e a lot of cache misses might occur. Since the opencl implementation is a closed box, it is impossible to know how the compiler foresees the execution of the kernel. If one makes the assumption that work items launched are running till completion and are synchronous in terms of execution of their instructions, the data fetched from the global memory are the same. More precisely, assuming than the threads of a group (or even several wraps) are run virtually synchronously (as explained in the chapter concerning GPU programming, for Nvidia GPU, in general only a fraction of a given wrap is run simultaneously in a SM, but the registers on the SM remain entirely allocated to the work items of this group until another group is executed on the same SM) the fetched values in global memory at a given time are the same. Therefore there is no bandwidth loss in this specific execution scheme. In order to check whether or not this execution scheme might be employed by Nvidia another kernel was written making use of the local memory. A maximum amount of positions of particles were copied in local memory and then for these positions each work item were computing the interactions. Since the number of positions might be higher than the total amount of local memory available an automatic subdivision of the particles were performed. This leads to more potential idle threads than in the former kernel because of the number of the branching conditions. For the sake of completeness the code of this kernel is given on the code section 6.8 but no further explanations are provided. One may notice the local memory barrier required to synchronize the copies.

Code 6.8: Space charge kernel making use of local memory

```
__kernel void ParticleInteraction3(__global float *data,
                                   unsigned int blockSize,
                                   unsigned int dataOffset,
                                   unsigned int start_test_index,
                                   unsigned int start_source_index,
                                   unsigned int stop_source_index,
                                   unsigned int max_local_size,
                                   __local float *cp,
                                   float K){

    int index = get_global_id(0);

    int index_l = get_local_id(0);
    int WGSize = get_local_size(0);

    __global float *test = data + blockSize*(index+start_test_index) +
        dataOffset;
```

```
float R[3];
float F[3] = {0.0f,0.0f,0.0f};

float T[3];
T[0] = test[0];
T[1] = test[1];
T[2] = test[2];

unsigned int total = stop_source_index - start_source_index;
unsigned int nBSteps = total/max_local_size;
unsigned int rest = total - nBSteps * max_local_size;
if(rest != 0){
    nBSteps++;
}

__global float *source = data + start_source_index*blockSize +
     dataOffset;

for(int i=0;i<nBSteps;i++){

    unsigned int test = total/max_local_size;
    unsigned int local_size = 0;
    if(test != 0){
        local_size = max_local_size;
        total -= max_local_size;
    }
    else{
        local_size = total;
    }

    unsigned int size_cp = local_size/WGSize;
    unsigned int rest_cp = local_size - size_cp * WGSize;
    unsigned int start_cp = 0;
    unsigned int stop_cp = 0;

    if(index_l+1 == WGSize){
        start_cp = index_l * size_cp + rest_cp;
        stop_cp = local_size;
    }
    else{
        start_cp = index_l * size_cp;
        stop_cp = (index_l+1) * size_cp;
    }

    __global float *source_pt = source + blockSize * start_cp;
    __local float *cp_pt = cp + 3 * start_cp;

    for(int j=start_cp;j<stop_cp;j++){
        for(int k=0;k<3;k++){
            cp_pt[k] = source_pt[k];
        }
        source_pt += blockSize;
        cp_pt += 3;
    }

    barrier(CLK_LOCAL_MEM_FENCE);

    cp_pt = cp;

    for(int j=0;j<local_size;j++){
        R[0] = T[0] - cp_pt[0];
        R[1] = T[1] - cp_pt[1];
```

```
            R[2] = T[2] - cp_pt[2];
            float r = sqrt(R[0]*R[0]+R[1]*R[1]+R[2]*R[2]);
            if(r != 0.0f){
                float r3 = r*r*r;
                F[0] += R[0]/r3;
                F[1] += R[1]/r3;
                F[2] += R[2]/r3;
            }
            cp_pt+=3;
        }
    }

    F[0] *= K;
    F[1] *= K;
    F[2] *= K;

    test += 9;
    test[0] += F[0];
    test[1] += F[1];
    test[2] += F[2];
}
```

6.4.5 Integration method and implementation

Let us first recall the Runge-Kutta method (see [65]). Let $f : A \subset \mathbb{R}^{6M} \times \mathbb{R} \to \mathbb{R}$ be a function where A is an open subset. The Cauchy problem is given by

$$\dot{x} = f(x,t) \tag{6.20}$$
$$x(t_0) = x_0 \tag{6.21}$$

where $x : I \subset \mathbb{R} \to \mathbb{M}$ with I the maximal open interval of the ordinary differential equation. The Runge-Kutta method consists in solving this problem numerically following the integration scheme hereunder

$$\begin{cases} \left[\begin{array}{lcl} t_{n,i} & = & t_n + c_i h_n \\ y_{n,i} & = & y_n + h_n \sum_{0\le j<i} A_{ij} p_{n,j} \\ p_{n,i} & = & f(t_{n,i}, y_{n,i}) \end{array}\right] 0 \le i < q \\ t_{n+1} = t_n + h_n \\ y_{n+1} = y_n + h_n \sum_{0\le j<q} A_{qj} p_{n,j} \end{cases} \tag{6.22}$$

where h_n is the time step, c_i, A_{ij} are predefined coefficients. The first 3 equations are the intermediate integration steps of the Runge-Kutta method. The last equation corresponds to the integration of the function over a full time step. The forth equation is simply an update of the current integration time. Since the method considered here is an explicit method, one has

$a_{ij} = 0$ if $j \geq i$.

The Runge-Kutta kernel performs the computation of the second equation and the fifth equation. Both of them perform exactly the same calculation although they are presented separately to distinguish the intermediate local integration steps from the global integration step. Between each integration step, the kernels responsible for the computation of the f function are run (the kernel for the Coulomb interaction, the kernel for the source field evaluation and finally a kernel summing up the two contributions and computing the forces). The intermediate times $t_{n,i}, t_n$ are calculated on the host (CPU).

The code section 6.9 shows the Runge-Kutta kernel. It takes 5 arguments: The pointer to the CRKSP blocks, the variable "N" corresponding to $q+1$, the level of integration "level" corresponding to the index i in the equations 6.22, the Butcher tabular "A_ij" and the time step "time_step" corresponding to h_n. The kernel is a straightforward implementation of the second and fifth equations. The loop corresponding to the sum is unrolled in order to avoid useless test logic and branching. The arithmetic operations performed on the pointers can be easily understood from the CRKSP data structure definition as shown Fig. 6.6.

Code 6.9: Runge-Kutta kernel

```
__kernel void RungeKutta(__global float *qf,
                         unsigned int N,
                         unsigned int level,
                         __global float *A_ij,
                         float time_step){

    int index = get_global_id(0);

    __global float *q = qf + index * (12*N-5) + 1;

    float Q[6];
    #pragma unroll
    for(unsigned int i=0;i<6;i++){
        Q[i] = q[i];
    }

    A_ij += (N-1) * level;

    for(unsigned int i=0;i<level;i++){
        float coef = A_ij[i];
        if(coef != 0.0f){
            __global float *f = q + i * 12 + 6;
            #pragma unroll
            for(unsigned int j=0;j<6;j++){
                Q[j] += (time_step * coef) * f[j];
            }
        }
```

```
    }

    q += 12*level;
    #pragma unroll
    for(unsigned int i=0;i<6;i++){
        q[i] = Q[i];
    }
}
```

6.5 Complexity

Let $N_c \in \mathbb{N}$ be the number of cells, N be the number of macro-particles. Let us assume an uniform distribution of particles inside the accelerator such that the number of particles in one cell is given by $n = \frac{N}{N_c}$. The computational complexity of a single cell is given by $C_c = 3n^2$ and for the two extreme cells by $C_c = 2n^2$. The overall complexity is therefore bounded by

$$C < 3N_c n^2 = 3\frac{N^2}{N_c} \tag{6.23}$$

In order to determine the complexity of the method, one must link N_c to the number of particles in the accelerator. Let us define a criterion of error that will allow us to determine such a link. Let us consider a particle placed at the center of a reference frame and on which a force appears due to some neighbour particles placed in a regular grid as shown Fig. 6.11. The particles in this regular grid represent an ideal bunch. The grid steps for each direction $(\hat{x}, \hat{y}, \hat{z})$ are equal. It is assumed that the distribution is going till the infinite negative z, while in a cross section plane one has a finite number of particles given by $(2N'+1)^2$. The total electric field at the position of the particle is given by

$$\overline{F} = -\frac{1}{4\pi\epsilon_0\sqrt{\Delta x}} \sum_{k=-1}^{-\infty} \sum_{i=-N'}^{N'} \sum_{j=-N'}^{N'} \frac{(i,j,k)}{||(i,j,k)||_2^{3/2}} \tag{6.24}$$

Let us first show that the eq. 6.24 is well defined i.e the limit for each dimension converges. It is clear that it is equivalent to show that the limit $\lim_{L\to\infty} g(i,j,L)$ exists with $g(i,j,L) \equiv \sum_{k=1}^{L} \frac{k}{||(i,j,k)||_2^{3/2}}$. Let us prove that $g(i,j,L)$ is a Cauchy series in L.

$$|g(i,j,L) - g(i,j,L')| \leq \sum_{k=L}^{L'} \frac{k}{(i^2+j^2+k^2)^{3/2}} \leq \sum_{k=L}^{L'} \frac{1}{k^2} \tag{6.25}$$

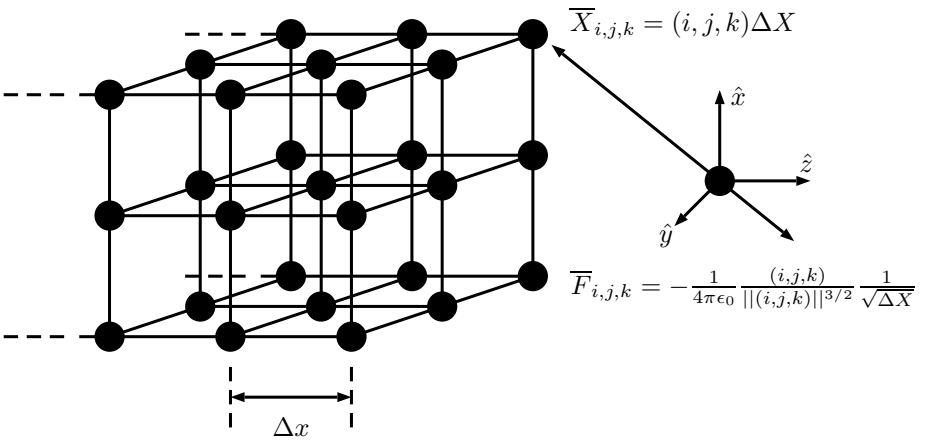


Figure 6.11: Ideal representation of a bunch

It is clear that $\sum_{i=1}^{L} \frac{1}{i^2} \leq 1 + \int_1^{\infty} \frac{1}{x^2} dx = 2$. The series is bounded and monotonically increasing and therefore is a Cauchy series. Because of the completeness of $\mathbb{R}$ the series converges. In passing, with the help of the Residue theorem (see for instance [70]) the value of the series can be analytically calculated and reads $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{pi^2}{6}$.

By symmetry, only the component of the field in the direction $\hat{z}$ remains. Therefore one has

$$\overline{F} = \frac{1}{4\pi\epsilon_0\sqrt{\Delta x}} f \hat{z} \tag{6.26}$$

$$f \equiv \sum_{k=1}^{\infty} \sum_{i=-N'}^{N'} \sum_{j=-N'}^{N'} \frac{k}{||(i,j,k)||^{3/2}} \tag{6.27}$$

The approximate force is defined by

$$\overline{F}_a = \frac{1}{4\pi\epsilon_0\sqrt{\Delta x}} f_a \hat{z} \tag{6.28}$$

$$f_a \equiv \sum_{k=1}^{L} \sum_{i=-N'}^{N'} \sum_{j=-N'}^{N'} \frac{k}{||(i,j,k)||^{3/2}} \tag{6.29}$$

The error criterion is defined by

$$\frac{||\overline{F} - \overline{F}_a||}{||\overline{F}||} < \epsilon \tag{6.30}$$

It is straightforward to observe that $||\overline{F} - \overline{F}_a|| = \frac{1}{4\pi\epsilon_0\sqrt{\Delta x}}|f - f_a|$ and $||\overline{F}|| = \frac{1}{4\pi\epsilon_0\sqrt{\Delta x}}|f|$. It is also straightforward to see that $|f - f_a| = f - f_a$ and $|f| = f$. Therefore, the condition becomes

$$1 - \frac{f_a}{f} = \frac{|f - f_a|}{|f|} < \epsilon \tag{6.31}$$

$$\Leftrightarrow (1 - \epsilon)|f| < |f_a| \tag{6.32}$$

Since f contains an infinite sum it can't be calculated numerically. However, it is possible to obtain a criterion of error to calculate this limit. By definition of $g(i, j, L)$ one has $f = \sum_{i,j=-N'}^{N'} \lim_{k\to\infty} g(i, j, L)$. Let $\tilde{f}_L = \sum_{i,j=-N'}^{N'} g(i, j, L)$ be an approximation of f. Let us define $g(i, j) \equiv \lim_{L\to\infty} g(i, j, L)$. Let us try to bound the error |g(i,j) - g(i,j,L)|

$$\delta\epsilon_{i,j} \equiv |g(i, j) - g(i, j, L)| \tag{6.33}$$

$$\leq \sum_{k=L+1}^{\infty} \frac{k}{(i^2 + j^2 + k^2)^{3/2}} \tag{6.34}$$

$$\leq \sum_{k=L+1}^{\infty} \frac{1}{k^2} = \frac{pi^2}{6} - \sum_{k=1}^{L} \frac{1}{k^2} \tag{6.35}$$

Now the error on f can be easily bounded as follows

$$\delta\epsilon = f - \tilde{f}_L = |f - \tilde{f}_L| < \sum_{i,j=-N'}^{N'} |g(i, j) - g(i, j, L)| \tag{6.36}$$

$$< (2N' + 1)^2 (\frac{pi^2}{6} - \sum_{k=1}^{L} \frac{1}{k^2}) \tag{6.37}$$

$$\Leftrightarrow f < \tilde{f} + (2N' + 1)^2 (\frac{pi^2}{6} - \sum_{k=1}^{L} \frac{1}{k^2}) \tag{6.38}$$

A sufficient to satisfy the condition given by Eq. 6.32 is given by

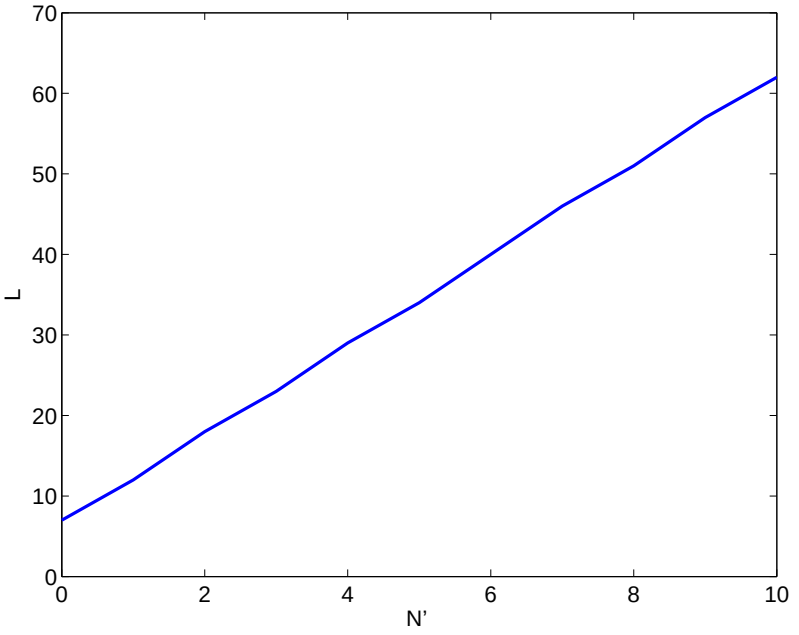


Figure 6.12: L in function of N' for $\epsilon = 0.1$

$$(1-\epsilon)|f| < (1-\epsilon)(\tilde{f} + (2N'+1)^2(\frac{pi^2}{6} - \sum_{k=1}^{L}\frac{1}{k^2})) < |f_a| \tag{6.39}$$

It is important to notice that if f is not calculated enough accurately, the condition given by Eq. 6.39 might not have a solution. In particular, if $\epsilon = 0$, it does not have a solution. Moreover, in order to find the less constraining bound, the value of f should be as accurate as possible. A numerical simulation gives the bound L in function of N'. As observed on Fig. 6.12, L depends linearly on $2N'+1$. Let us overestimate this linear relation by $L = 8(2N'+1)$.

Now, let us assume that the particles populating the accelerator are distributed on a regular uniform grid similar to the one used to represent an ideal bunch. If $L(2N'+1)^2 << N$, The number of particles in a cell is given by $n = L(2N'+1)^2 = 8(2N'+1)^3$. Hence, the number of cells is given by

$$N_c = \frac{N}{L(2N'+1)^2} = \frac{N}{8(2N'+1)^3} \tag{6.40}$$

Injecting this result in Eq. 6.23 one gets

$$C < 3N_c n^2 = 24(2N'+1)^3 N \tag{6.41}$$

Now, we can establish two different complexities based on two different situations. The first situation corresponds to the case of a fully populated accelerator with a given density of particles. In this case, we want to know what is the complexity of the method if the density of particles is increased. In this situation N' depends also on N. In particular one has the following constrains

$$(2N'+1)\Delta x = r \tag{6.42}$$
$$N_z \Delta x = l \tag{6.43}$$
$$(2N'+1)^2 N_z = N \tag{6.44}$$

where r is the size of the beam in the cross-section (length of the square), l is the length of the accelerator and N_z is the number of particle grid steps plus one along the $\hat{z}$ axis. This implies that

$$(2N'+1) = (\frac{Nr}{l})^{\frac{1}{3}} \tag{6.45}$$

The final complexity is given by

$$C < 3N_c n^2 = 3k\frac{r}{l}N^2 \tag{6.46}$$

The second situation corresponds to the case of an accelerator not fully populated and with a fixed level of density of particles. In this case, we want to know the complexity of the method in function of the number of particles injected. In this case N' is constant and the complexity is therefore linear as shown by Eq. 6.41.

6.6 Numerical Results

6.6.1 Validation of the beam dynamics solver with a Paul trap simulation

The well known Paul trap or ion trap [12] consists in a quadrupole fed by an RF source. It aims at keeping an ion beam between the rods. The ex-

tension of the Paul trap is the RFQ. The differences between a Paul trap and a RFQ is that the RFQ accelerates the beam and bunches it.

The field produced by an ion trap is given by the following equations

$$\overline{E}(\overline{x},t) = (kx_1\hat{x}_1 - kx_2\hat{x}_2)\cos(\omega t + \phi) \tag{6.47}$$

where $k \in \mathbb{R}$ is a constant, $\hat{x}_1, \hat{x}_2$ are two orthonormal vectors, x_1, x_2 are the components respectively of $\hat{x}_1$ and $\hat{x}_2$, ω is the angular frequency of the RF source and ϕ the initial phase. The electrostatic equation of the ion trap has been developed in the RFQ principle section 2.1.

Applying the Newton third law, one gets

$$\frac{d^2\overline{x}}{dt^2} - \frac{q_0}{m_0}(kx_1\hat{x}_1 - kx_2\hat{x}_2)\cos(\omega t + \phi) = 0 \tag{6.48}$$

where q_0 is the charge of the particle, m_0 is the mass of a the particle. This can be rewritten component by component

$$\frac{d^2x_1}{dt^2} - \frac{kq}{m}\cos(\omega t + \phi)x_1 = 0 \tag{6.49}$$

$$\frac{d^2x_2}{dt^2} + \frac{kq}{m}\cos(\omega t + \phi)x_2 = 0 \tag{6.50}$$

$$\frac{d^2x_3}{dt^2} = 0 \tag{6.51}$$

Let us rewrite these two first equations, such that it has the form of the Mathieu's equation given by 10.66. Let us define $2t' = \omega t$, the equations becomes

$$\frac{d^2x_1}{dt'^2} - \frac{4kq}{m\omega^2}\cos(2t')x_1 = 0 \tag{6.52}$$

$$\frac{d^2x_2}{dt'^2} + \frac{4kq}{m\omega^2}\cos(2t')x_2 = 0 \tag{6.53}$$

$$\frac{d^2x_3}{dt'^2} = 0 \tag{6.54}$$

Let us define $\epsilon \equiv \frac{4kq}{m\omega^2}$ and $\delta = 0$ so that the equations have the form of Mathieu's differential equation. Now, let us the Ince-Strutt diagram presented in Appendix, section 10.8.6, Fig. 10.5 to obtain a stable solution. Let

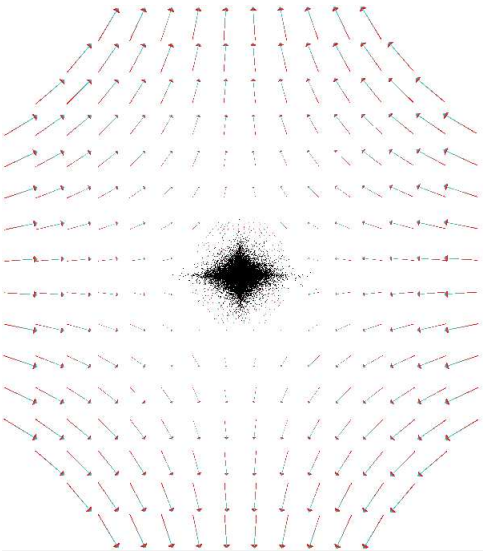


Figure 6.13: Electric field in a cross section of the Paul trap.

us chose of value of ϵ right in the middle of a stable region. For instance, let us chose $\epsilon = 0.5 = \frac{4kq}{m\omega^2}$ for $\delta = 0$. The simulation is performed for protons, therefore $m_0 = 1.672622\ 10^{-27}$ and $q_0 = 1.602176565\ 10^{-19}$. Let us chose the frequency of the RF source to $f = 176Mhz$. The equation of ϵ becomes $k = 1.6\ 10^9$.

The simulations were performed with a Runge-Kutta method of order four. Fig. 6.13 shows the electric field in a cross-section of the ion trap. Fig. 6.14 shows the beam inside the ion trap. The input beam is generated by a Gaussian distribution in the X-Y plane and by an uniform distribution in the Z direction. The is no initial velocity in the X-Y plane by there is an initial velocity in the Z direction given by the initial energy of 30 keV. One can observe on Fig. 6.14 the stability of the beam inside the accelerator as expected. Fig. 6.15, 6.16, 6.17 show the phase space respectively for the directions $\hat{x} = \hat{x}_1, \hat{y} = \hat{x}_2, -\hat{z} = -\hat{x}_3$ at a given time for a thickness of 1cm of cross-section and at a distance of 20cm from the injection. One can see on this picture the ellipse form of the beam in the transversal phase spaces. Since there is no field in the direction $\hat{z}$ the longitudinal phase space will be always flat. Therefore, the longitudinal phase space won't be shown in the next figures. Since the injected beam does not have any emittance (the volume of the beam in the phase space is zero), by the Liouville's theorem, the beam should not have an emittance anywhere else in the accelerator.

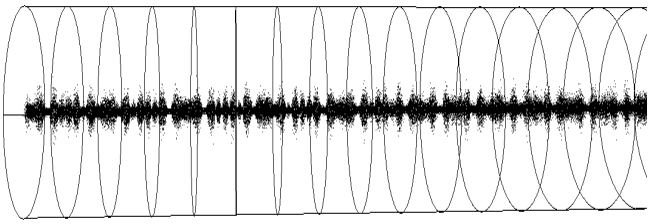


Figure 6.14: Paul trap transversal beam dynamics.

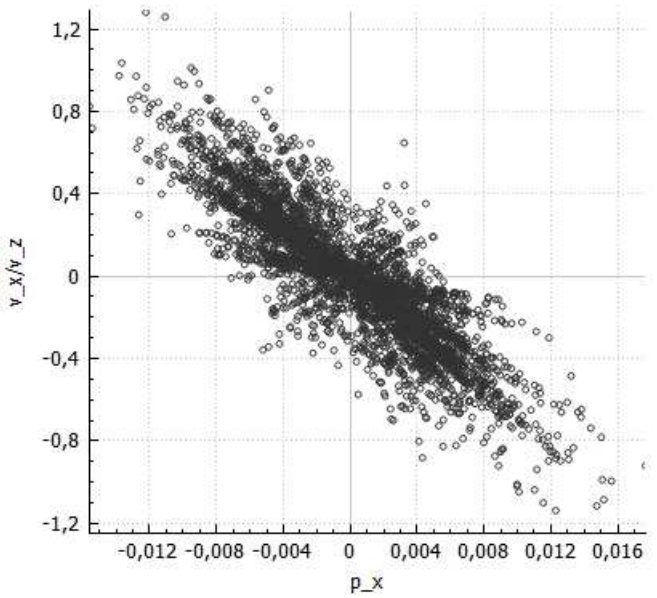


Figure 6.15: X Phase space of a cross-section of 1 cm in the Paul trap.

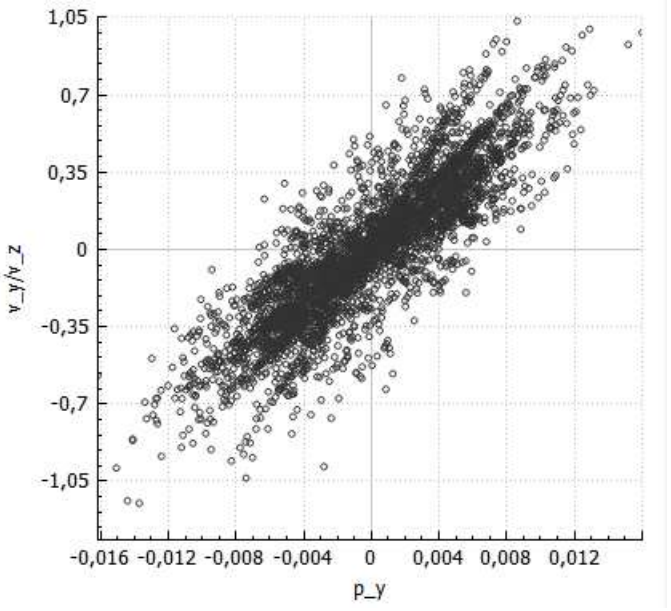


Figure 6.16: Y Phase space of a cross-section of 1 cm in the Paul trap.

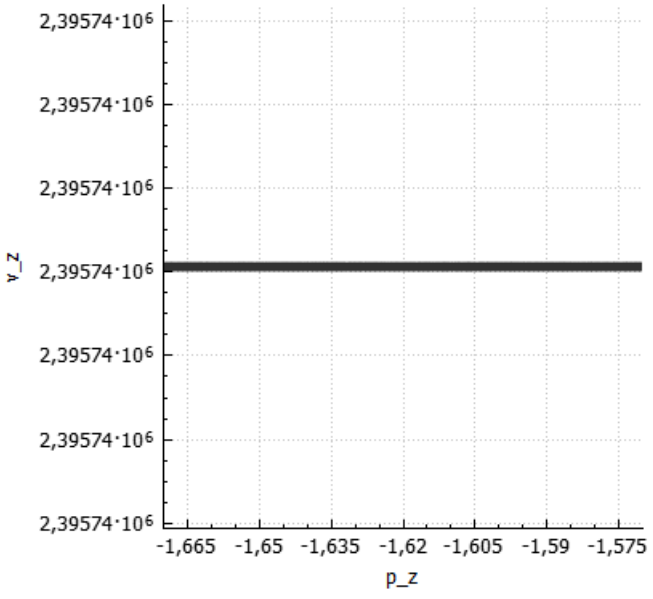


Figure 6.17: Z Phase space of a cross-section of 1 cm in the Paul trap.

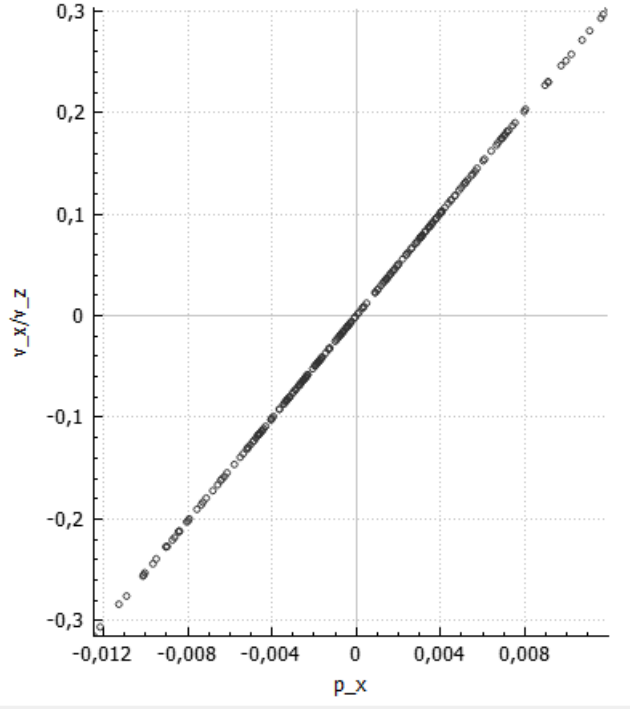


Figure 6.18: X Phase space of a cross-section of 2 mm in the Paul trap

This theorem is true as long as there is no space charge effects. The reason why there is a non zero emittance in the figures 6.15, 6.16 is because one takes a thickness of cross-section of 1cm which is large and therefore the phase spaces correspond to the superposition of zero emittance pieces of beam injected at different time. Let us take a cross-section smaller than the distance travels by the beam for an integration step and let us observe the phase space in this case. Fig. 6.18 and Fig. 6.19 show the transversal phase spaces of the beam in a cross-section of 2mm. One can see that the emittance is equal to zero which is consistent with the Liouville's theorem. This confirmation validates the numerical integration. Let say that the time observation of the particles presented in former figures is t_0. Let us observe the phase space of the same cross-sections but at a time $t = t_0 + \frac{T}{4}$ where $T \equiv \frac{1}{f}$. Fig. 6.20 and Fig. 6.21 show the transversal phase space respectively for each direction $\hat{x}, \hat{y}$. The ellipses have been rotated by a rotation of 90 degrees. If one takes a time $t_0 + T$, the particles distribution in each transversal space phase comes back to the same distribution as the

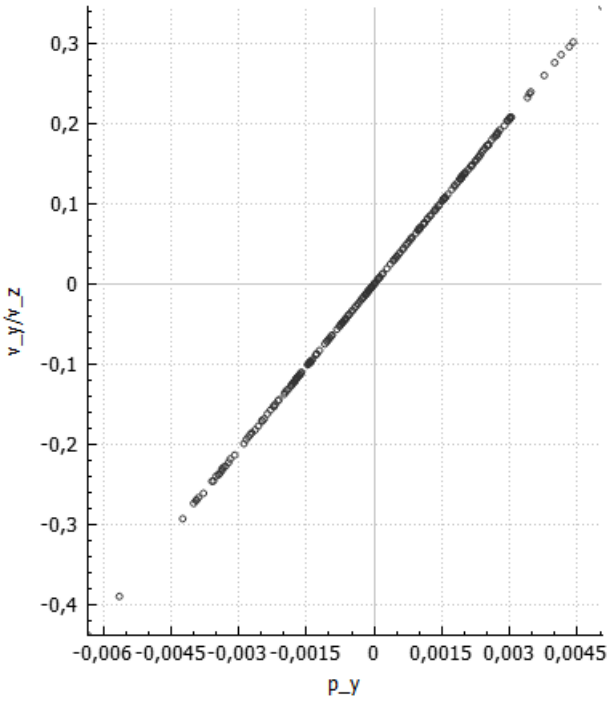


Figure 6.19: Y Phase space of a cross-section of 2 mm in the Paul trap

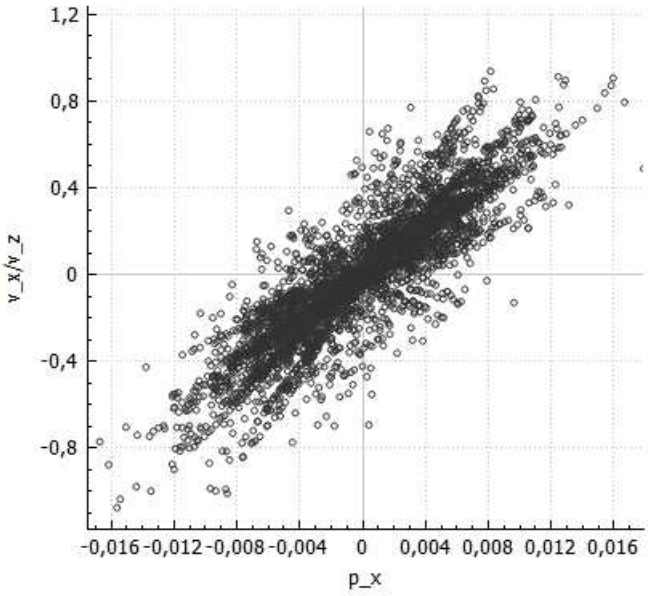


Figure 6.20: X Phase space of a cross-section of 2 cm in the Paul trap at a time $t_0 + \frac{T}{4}$.

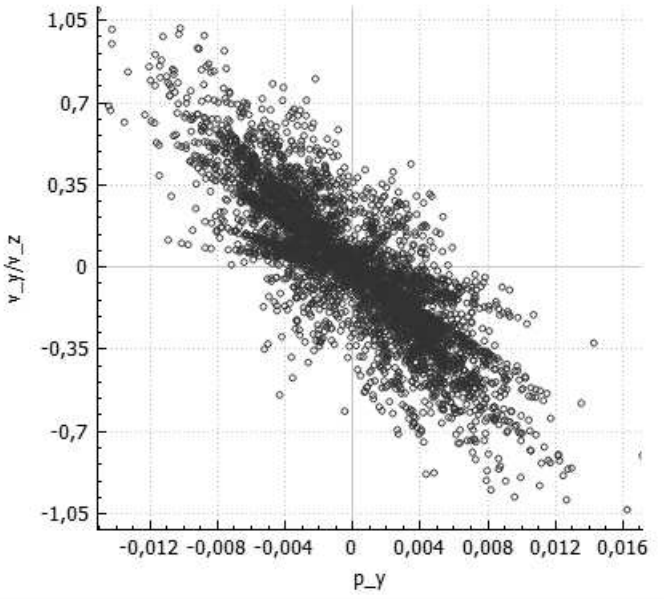


Figure 6.21: Y Phase space of a cross-section of 2 cm in the Paul trap at a time $t_0 + \frac{T}{4}$.

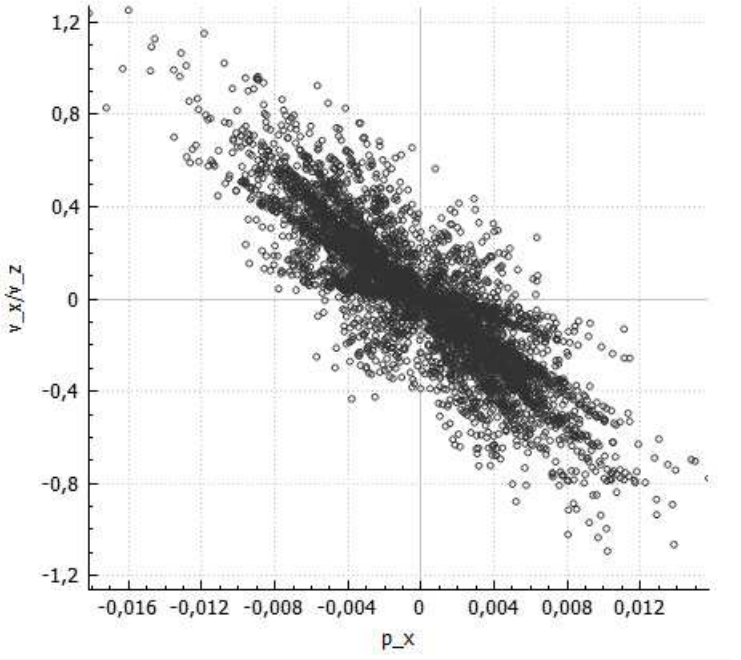


Figure 6.22: X Phase space of a cross-section of 2 cm in the Paul trap with space charge.

one at time t_0. This is again consistent with the Twiss parameters analysis presented in Appendix subsection 10.8.5.

Now let us add some space charge effects into the simulation. There is non-linear term in the differential equation. Therefore, the Floquet's theory presented in Appendix section 10.8.3 cannot be applied anymore or at least not directly [2]. Let us observe the phase spaces in this case. Fig. 6.22, 6.23, 6.24 show the phase spaces respectively for each direction $(\hat{x}, \hat{y}, \hat{z})$. The first thing to observe and without surprise is that the emittance in the longitudinal phase space is not zero anymore. In the transversal phase spaces, the emittance is slightly bigger than for the case without space charge. The beam remains stable with space charge effects and its maximal radius in the cross section remains practically identical. Therefore, one can conclude that at 4mA and for such a Paul trap design, the space charge does not influence significantly the stability and the size of the beam.

[2] If the non-linear function can be linearized, one may in some situations use the Floquet's theory

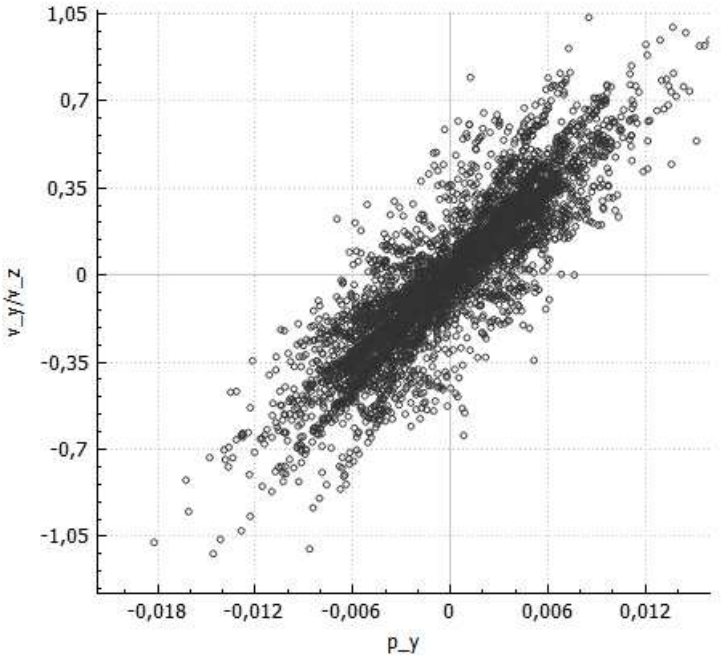


Figure 6.23: Y Phase space of a cross-section of 2 cm in the Paul trap with space charge.

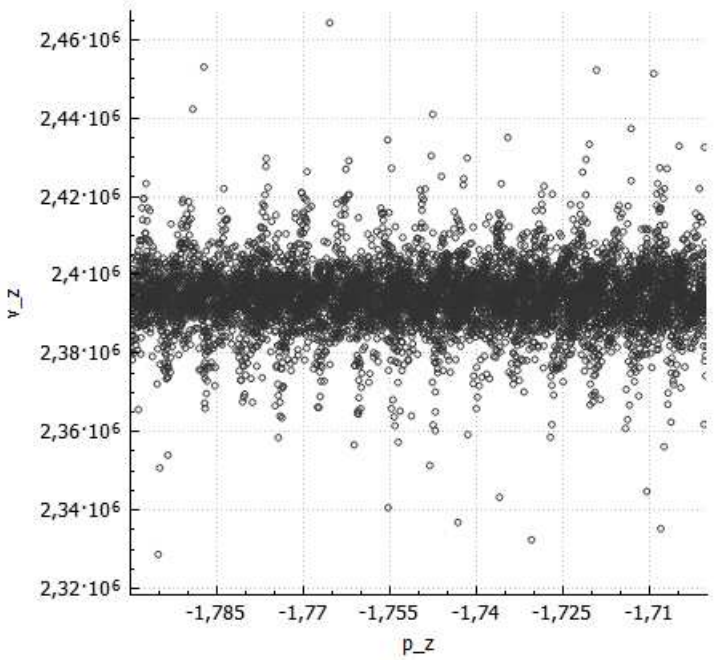


Figure 6.24: Z Phase space of a cross-section of 2 cm in the Paul trap with space charge.

6.6.2 Gain factor between the CPU and the GPU implementation

OpenCL is a portable langage. Therefore, the comparison of performances between CPU and GPU can be done using the same openCL implementation. In order to perform calculations on the GPU, a GPU device data structure is passed to the openCL functions while for calculations on the CPU a CPU device data structure is passed. The openCL API provides functions to measure the computation time of a given kernel. Therefore, it is relatively easy to compute the FLOPs once the number of operations executed by a kernel is known. The simulations where performed on a DELL workstation equiped with a Xeon E5 1660 v3 CPU and a Nvidia Quadro k620. To the knowledge of the writer, there is no benchmark or specifications providing the theoretical maximum FLOPs of the CPU. However, the Xeon E5-2640 v3 is very close to the specifications of the E5 1660 v3 and the article [71] claims that its theoretical peak GFLOPS performance is 300 GFLOPs. The Nvidia quadro k620 can provide a theoretical peak of 812 GFLOPs as the article [72] claims.

Fig. 6.25 shows the GFLOPs obtained with the GPU and the CPU in function of the number of particles inside the Paul trap. The red curve corresponds to the CPU while the blue curve corresponds to the GPU. The average FLOPs obtained on the GPU is 130 GFLOPs while the average FLOPs for the CPU is 15 GFLOPs. Fig. 6.26 shows the ratio between the GFLOPS obtained on the GPU divided by the GFLOPS obtained on the CPU. The GPU outperforms the CPU by a average factor of 9.35 with a standard deviation of 3.46. The ratio between the average FLOPs is equal to 8.61. Now, let us take into account the price of the CPU versus the price of the GPU. The advice price of the Xeon E5 1660 v3 provided by intel is 1080$ altought the best price found in Europe for this ship is 1310€. The Nvidia quadro K620 can be found for 160$ on the newegg website and for 188€ on Amazon France. The ration FLOPS/$ for the CPU is therefore 13.9 MFLOPs/$ while for the GPU it is 800 MFLOPs/$. It means the GPU is 57 times more efficient in terms of FLOPs/Price.

6.6.3 Beam dynamics

In this subsection, the beam dynamics results are presented. The source electric field has been calculated from the MoM currents obtained with a brute force method (computation of the full impedance matrix + direct inversion of the impedance matrix). Fig. 6.27 shows the bunches created at 50 cm from the injection. The space charge has been disabled in this simulation. The bunches starts fading away around the longitudinal middle

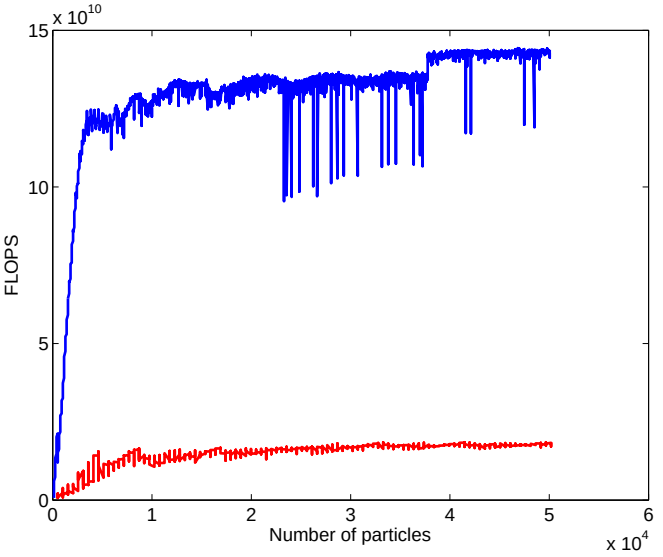


Figure 6.25: GFLOPs GPU vs CPU.

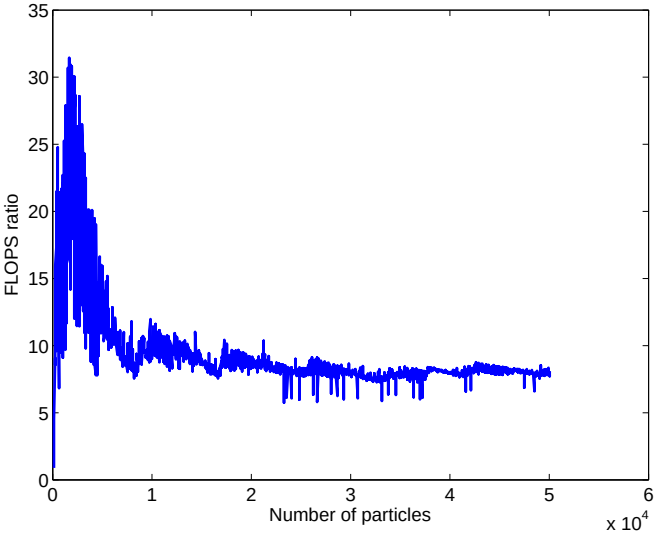


Figure 6.26: GFLOPs GPU divided by GFLOPs CPU.

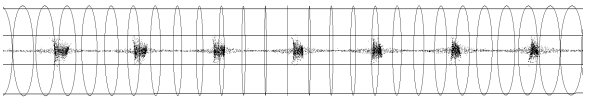


Figure 6.27: Transversal beam dynamics view.

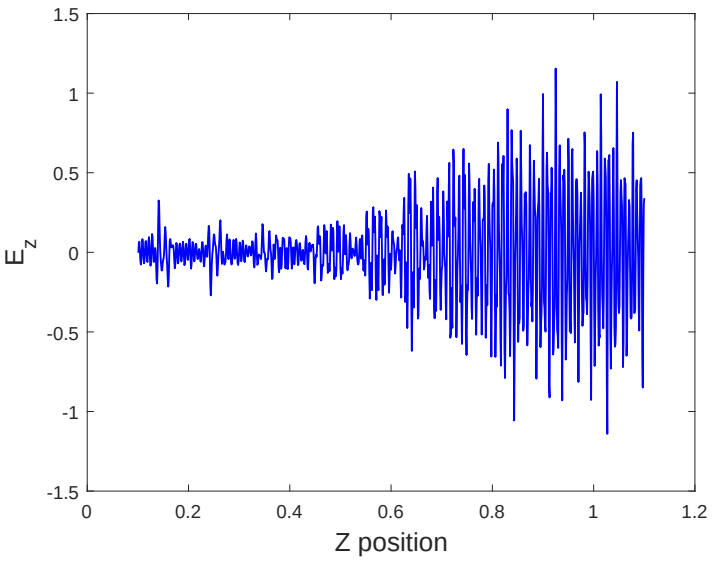


Figure 6.28: Longitudinal electric field in function of the longitudinal coordinate at a given time.

of the accelerator. The reason of the failure of the simulation comes probably from the strong variations of the electric field in function of the longitudinal coordinate as shown Fig. 6.28. We believe the strong variation of the electric field is due partially to the coarse regularity of the RWG basis functions. Another source of error comes from the singular behaviour of the electric field operator. Hence, using smoother basis functions potentially in conjunction with an analytical extraction of singularities might solve the problem.

Chapter 7

Conclusion

The Myrrha project aims at building an Accelerator Driven System (ADS) for nuclear waste transmutation. The accelerator feeding the sub-critical nuclear core is a LINear ACcelerator (LINAC) accelerating a proton beam of 4mA to an energy of 600MeV. At the very beginning of the accelerator a cavity known as the Radio-Frequency Quadrupole (RFQ) plays a very important role in the overall stability of the beam. Its goal is to focus, bunch a give a first acceleration to the beam. Because the RFQ is a very critical component of a LINAC, accurate simulations are performed in order to validate the designs but also to have a better understanding of the physical phenomenons implied in such a technology.

The electromagnetic fields simulation of a RFQ is challenging in terms of computational cost. The Method of Moments (MoM) has been used with the final goal of improving the computation time of present solvers while maintaining the accuracy of a full-wave solver. One of the final goals is the numerical optimization of such an accelerator. The electromagnetic fields simulation of the Myrrha RFQ with the MoM faces several challenges. First, the impedance matrix of the EFIE is ill-conditioned, mainly because of the small size of the accelerator in comparison to the wavelength. The bad condition number of the impedance matrix implies a poor convergence of iterative solvers such as GMRES and therefore a long computation time. Second, the direct computation of the impedance matrix is of complexity $O(N^2)$ where N is the number of basis functions. The mesh of the RFQ must be fine in order to capture the rods curvature accurately. This implies a relatively large number of basis functions. Iterative solvers entail matrix-vector multiplication at each iteration. Classic matrix-vector accelerators for electromagnetic solvers cannot be used here since these methods fail when the overall structure is smaller than two wavelengths. Finally, the

electric field accuracy suffers from the singularities of the basis functions and their discontinuities when it is evaluated in the very near field (i.e very close to the boundaries). The first issue has been solved with the help of an efficient and fast preconditioner proposed in [19] and described in section 4.3. The second issue has been solved with the help of a newly proposed method called the Fast Near Field MoM Solver (FNFMMS) presented in section 4.4. The low-frequency preconditioner is necessary to guarantee good performance of the FNFMMS as shown by the computation time tables 4.2 and 4.1. The third issue has not been solved in this thesis but an idea that consists in using smoother basis functions have been proposed to solve it.

There exist two principal ways to simulate the beam dynamics. The first one consists in considering the particles as a continuum of charges. In this case, the well-known Vlasov's equation is often used to calculate the motion of this continuum of charges. Another method consists in considering macro-particles and applying the laws of motion directly on them. The latter approach has been selected in this thesis. It is mathematically simpler to handle than the first approach and therefore it allowed us to focus on another aspect which is the efficient implementation of the solver on Graphical Processor Unit (GPU). Neglecting the space charge, any particle trajectory can be calculated independently of the others. In this case, a GPU is a perfect piece of hardware to solve massively all these independent differential equations. Accounting for the space charge implies a coupling between particles which breaks the independence between the differential equations. The differential equations become a single huge differential equation to solve. Because the computation of the force experienced by each particle implies accesses to all the positions of all the other particles, this problem seems at first glance not any-more adapted for GPU. Indeed, all these large memory accesses are not handled very well by a GPU. Furthermore, the positions of all particles must be updated at the same time which implies a lower level of parallelism. However, using some approximations presented in section 6.2, the memory accesses could be improved such that 16% of the theoretical GPU performance was achieved. The GPU outperforms the CPU by a factor of 60 in terms of ratio between performance and financial cost. The GPU solver has been validated for a Paul trap or Ion trap presented in section 6.6.1. The beam dynamics simulation of Myrrha's RFQ was performed with the electric field calculated with the MoM. However, because of several issues regarding the electric field extraction, the simulations did not succeed to produce an overall valid solution. The creation of bunches has been observed till the middle of the accelerator to fade away slowly as the particles were getting closer to the end of the accelerator. No significant acceleration

was observed, probably due to the destruction of bunches but also because of the very strong variations of the electric field due to the extraction issues. We believe the problem comes from the weak regularity of the Rao-Wilton-Glisson (RWG) basis functions. In particular, by definition of the RWG, the density of charges is approximated by a constant function over a mesh cell. Hence, a prospective solution would be to use smoother basis functions.

To conclude, the MoM has been applied for the simulation of a four-rod RFQ for the first time. Some of the challenges have been solved and some are still present. Two new methods came out of the research perfomed in this thesis. The new FNFMMS method can be applied for a wide range of problems. In particular, it can be used for simulations concerning the computation of the electromagnetic fields scattered by small objects in comparison to wavelength and in preference with a dense mesh.

Chapter 8

Acknowledgments

Chapter 9

Scientific output

9.1 Publications

- C. Raucy, E. de Lera Acedo, C. Craeye, D. Gonzalez-Ovejero, and N.R. Ghods. Experimental validation of fast simulation methods in the framework of the ska telescope project. In *Antennas and Propagation (EUCAP), 2012 6th European Conference on*, pages 2225–2229, 2012
- E. de Lera Acedo, N.R. Ghods, P. Scott, P. Doherty, K. Grainge, A. Faulkner, P. Alexander, D. Gonzalez-Ovejero, C. Raucy, C. Craeye, N. Drought, N. Troop, P. Van der Merwe, and H.C. Reader. Ska aa-low front-end developments (at cambridge university). In *Antennas and Propagation (EUCAP), 2012 6th European Conference on*, pages 616–620, March 2012
- C. Raucy, E. de Lera Acedo, N. Razavi-Ghods, and C. Craeye. Characterization of ska-aalow antenna elements in the array environment. In *Electromagnetics in Advanced Applications (ICEAA), 2012 International Conference on*, pages 514–517, Sept 2012
- C. Raucy, Craeye C., and Vandeplassche D. Simulation of a radio frequency quadrupole with the method of moments. In *Antennas and Propagation (EUCAP), 2014 8th European Conference on*, April 2014
- C. Raucy, Craeye C., and Vandeplassche D. Rfq solver based on the method of moments. In *5th International Particle Accelerator Conference (IPAC)*, June 2014
- C. Raucy, F. P. Andriulli, and C. Craeye. Stabilization of the modelling of a radio-frequency quadrupole based on quasi-helmholtz projectors. In *Electromagnetics in Advanced Applications (ICEAA), 2015 International Conference on*, pages 1441–1444, Sept 2015

Chapter 10

Appendix

10.1 Limits

Proposition 10.1.1:

Let be $u_n, v_n : \mathbb{N} \to E$ with E a normed space. Let us assume that $u_n \to u$ and $v_n \to v$ then

- $u_n + v_n \to u + v$
- $(u_n|v_n) \to (u|v)$ where $(.|.)$ is a scalar product on E, if $\exists N \in \mathbb{N}$ such that $||u_n|| < K < +\infty$, $\forall n > N$

Demonstration 10.1.1:

The first relation is a direct consequence of the norm definition

$$||u_n + v_n - (u + v)|| = ||(u_n - u) + (v_n - v)|| \leq ||u_n - u|| + ||v_n - v||$$

Now, one has

$$\forall \epsilon > 0, \exists N \in \mathbb{N} \text{ such that } \forall n > N \Rightarrow ||u_n - u|| < \epsilon$$
$$\forall \epsilon > 0, \exists M \in \mathbb{N} \text{ such that } \forall n > M \Rightarrow ||v_n - u|| < \epsilon$$

Choosing $N_{max} = \max N, M$ one has

$$\forall n > N_{max} \Rightarrow ||(u_n + v_n) - (u + v)|| \leq 2\epsilon$$

Choosing $\epsilon = \frac{\epsilon'}{2}$ one has

$$\forall \epsilon' > 0, \exists N \in \mathbb{N} \text{ such that } \forall n > N \Rightarrow ||(u_n + v_n) - (u + v)|| \leq \epsilon'$$

The second relation is proven by developing the norm expression of the limit

$$\begin{aligned} |(u_n|v_n) - (u|v)| &= |(u_n|v_n) - (u_n|v) + (u_n|v) - (u|v)| \\ &= |(u_n|(v_n - v)) + (u_n - u|v)| \\ leq &|(u_n|(v_n - v))| + |(u_n - u|v)| \end{aligned}$$

Using the Cauchy-Schwarz inequality one has

$$|(u_n|v_n) - (u|v)| \leq ||u_n||||v_n - v|| + ||u_n - u||||v||$$

Using the condition $||u_n|| < K$ for $n > N$ one has

$$|(u_n|v_n) - (u|v)| \leq K||v_n - v|| + ||u_n - u||||v||$$

Then by convergence of u_n and v_n, for $\epsilon > 0$ one can find N, M such that

$$\begin{aligned} ||v_n - v|| &< \frac{\epsilon}{2K} \\ ||u_n - u|| &< \frac{\epsilon}{2||v||} \end{aligned}$$

Let us define $N_{max} = \max(N, M)$ then one has

$$\forall n > N_{max} \Rightarrow |(u_n|v_n) - (u|v)| \leq \epsilon$$

This ends the proof of the proposition.

The same proposition can be formulated for functions.

Proposition 10.1.2:

Let be $u, v : F \setminus \{x_0\} \to E$ with E, F two normed space. Let us assume that $\lim_{x \to x_0} u(x) = U$ and $\lim_{x \to x_0} v(x) = V$ then

- $\lim_{x \to x_0} u + v = U + V$
- $\lim_{x \to x_0} (u|v) = (U|V)$ where $(.|.)$ is a scalar product on E, if $\exists \delta > 0$ such that $||u|| < K < +\infty$, $\forall x \in B(x_0, \delta)$

Demonstration 10.1.2:

The first relation is a direct consequence of the norm definition

$$||u(x) + v(x) - (U + V)|| = ||(u(x) - U) + (v(x) - V)|| \leq ||u(x) - U|| + ||v(x) - V||$$

Now, one has

$$\forall \epsilon > 0, \exists delta > 0 \text{ such that } \forall x \in B(x_0, \delta) \Rightarrow ||u(x) - U|| < \epsilon$$
$$\forall \epsilon > 0, \exists delta' > 0 \text{ such that } \forall x \in B(x_0, \delta') \Rightarrow ||v(x) - V|| < \epsilon$$

Choosing $\delta_{min} = \min(\delta, \delta')$ one has

$$\forall x \in B(x_0, \delta_{min}) \Rightarrow ||(u(x) + v(x)) - (U + V)|| \leq 2\epsilon$$

Choosing $\epsilon = \frac{\epsilon'}{2}$ one has

$$\forall \epsilon' > 0, \exists \delta > 0 \text{ such that } \forall x \in B(x_0, \delta) \Rightarrow ||(u(x) + v(x)) - (U + V)|| \leq \epsilon'$$

The second relation is proven by developing the norm expression of the limit

$$\begin{aligned} |(u(x)|v(x)) - (U|V)| &= |(u(x)|v(x)) - (u(x)|V) + (u(x)|V) - (U|V)| \\ &= |(u(x)|(v(x) - V)) + (u(x) - U|V)| \\ leq \quad &|(u(x)|(v(x) - V))| + |(u(x) - U|V)| \end{aligned}$$

Using the Cauchy-Schwarz inequality one has

$$|(u(x)|v(x)) - (U|V)| \leq ||u(x)||||v(x) - V|| + ||u(x) - U||||V||$$

Using the condition $||u(x)|| < K$ for $x \in B(x_0, \delta)$ one has

$$|(u(x)|v(x)) - (U|V)| \leq K||v(x) - V|| + ||u(x) - U||||V||$$

Then by convergence of $u(x)$ and $v(x)$, for $\epsilon > 0$ one can find δ', δ'' such that

$$\begin{aligned} ||v(x) - V|| &< \frac{\epsilon}{2K} \\ ||u(x) - U|| &< \frac{\epsilon}{2||V||} \end{aligned}$$

Let us define $\delta_{min} \equiv \min(\delta, \delta', \delta'')$ one has

$$\forall x \in B(x_0, \delta_{min}) \Rightarrow |(u(x)|v(x)) - (U|V)| \leq \epsilon$$

10.2 Standard analysis

10.2.1 The exponential of a linear application

10.3 Affine geometry

Definition 10.3.1:

Let be E and F two vector space on the same field K (in general $\mathbb{R}$ or $\mathbb{C}$). An isomorphism between E and F is a bijective linear application.

Proposition 10.3.1:

Let be E and F two vector space of finite dimension. Let be $A : E \to F$ a linear application, then if f is injective and $dim(E) = dim(F)$, f is an isomorphism.

Demonstration 10.3.1:

Let show that the application is surjective. Let be $x_1, ..., x_n \in E$ a basis, if one shows that $y_1 = A(x_1), ..., y_n = A(x_n)$ is a basis then the proof is complete. Let us show that $y_1, ..., y_n$ is a linear independent family of vectors. Let us assume that exists $\alpha_1, ..., \alpha_n \in K$ such that $\alpha_i y_i = 0$ it implies that $\alpha_1 A(x_1) + ... + \alpha_n A(x_n) = A(\alpha_1 x_1 + ... + \alpha_n x_n) = 0$. Since the application is injective, one has $\alpha_1 x_1 + ... + \alpha_n x_n = 0$. Since $x_1, ..., x_n$ is a basis of E it implies that $\alpha_1 = ... = \alpha_n = 0$. Since the dimension of the spaced are equal, the linear independent family $y_1, ..., y_n$ is a basis.

Proposition 10.3.2:

Let be E a vector space of dimension n and $x_1, ..., x_n$ a basis. Let us define the linear application $i : E \to \mathbb{R}^n, \sum_{i=1}^n \alpha_i x_i \to (\alpha_1, ..., \alpha_n)$ then the application is an isomorphism.

Demonstration 10.3.2:

The linearity is easily shown

$$\begin{aligned} i(\alpha x + \beta y) &= i(\alpha \sum_{i=1}^{n} \alpha_i x_i + \beta \sum_{i=1}^{n} \beta_i y) = i(\sum_{i=1}^{n} (\alpha\alpha_i + \beta\beta_i) x_i) = (\alpha\alpha_1 + \beta\beta_1, ..., \alpha\alpha_n + \\ i(\alpha x + \beta y) &= \alpha(\alpha_1, ..., \alpha_n) + \beta(\beta_1, ..., \beta_n) = \alpha i(x) + \beta i(y) \end{aligned}$$

The injectivity is also straightforward

$$i(x) = (\alpha_1, ..., \alpha_n) = 0 \Rightarrow \alpha_1 = ... = \alpha_n = 0 \Rightarrow x = 0$$

Since the vector spaces have the same dimension by proposition 10.3.2 the demonstration is complete.

On a finite vector space, once a basis is defined the isomorphism given in proposition 10.3.2 is denoted $[x] = i(x)$. The isomorphism depends on a basis (it's not a canonic isomorphism). Hence, when a confusion can occur the notation is a bit different namely $[x]_{x_1,...,x_n}$.

Definition 10.3.2:

Let be E, E' two affine space associated respectively to the vector space V and V'. A function $f : E \to E'$ is an affine application if exists $\overline{f} : V \to V'$ such that

$$\forall X \in E, f(X) = O' + \overline{f}(\overline{OX})$$

Proposition 10.3.3:

Let be E an space associated to the vector space V. Let be $(O, x_1, ..., x_n)$ a frame. Let be $f : E \to E$ an affine application then, the affine application expressed in these two frames has the following form

$$[\overline{Of(X)}] = T + A[\overline{OX}]$$

where $A \in \mathbb{R}^{n \times n}$, $A_{ij} = f_i(x_j)$ with f_i the i component of $\overline{f}$ in the base $x_1, ..., x_n$ and $T \equiv [\overline{OO'}]_i$.

Demonstration 10.3.3:

By definition of an affine application one has

$$\forall X \in E, \overline{Of(X)} = \overline{OO'} + \overline{f}(\overline{OX})$$

This vectorial expression can be expressed in $\mathbb{R}^n$ thanks to the isomorphism as follows

$$[\overline{Of(X)}] = [\overline{Of(X)}] + i(\overline{f}(i^{-1}([\overline{OX}])$$

The matrix is identified by the linear application $A : \mathbb{R}^n \to \mathbb{R}^n, x \to A(x) = i(\overline{f}(i^{-1}(x))$ in its canonic basis and the translation vector by $T \equiv i(\overline{Of(X)})$.

Definition 10.3.3:

Let be E a normed vector space. It is said that $V \subset E$ is convex if

$$\forall x, y \in E, \{z | z = x + \alpha(x - y), \alpha \in [0, 1]\} \subset V$$

Definition 10.3.4:

Let be E a normed vector space and $x_1, ..., x_n \in E$ some points in E. The convex hull of $V \subset E$ is given by the intersection of all convex subset containing V.

Proposition 10.3.4:

Let be E a normed vector space and $x_1, ..., x_n \in E$ some points in E. The convex hull of these points is a convex subset of E.

Demonstration 10.3.4:

Let be $A_i \in E$ a family of convex subset of E. Let us define $A \equiv \cap_i A_i$ and let us proof that A is convex i.e $\forall x, y \in A, \{z | z = x + \alpha(x-y), \alpha \in [0, 1]\} \subset A$. Since $\{z | z = x+\alpha(x-y), \alpha \in [0, 1]\} \subset A_i$ the proof is complete.

Proposition 10.3.5:

Let be E a normed vector space and $x_1, ..., x_n \in E$ some points in E. Let be $V \subset E$ the convex skull of these points then

$$V = \{x | x = \sum_{i=1}^{n} \alpha_i x_i, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i > 0\}$$

Demonstration 10.3.5:

Let us prove it by recurrence. For $n = 1$ it is obvious. Let us assume it is true for $n - 1$ and let us proof this true for n.

First, let us prove that $x = \sum_{i=1}^{n} \alpha_i x_i$ with $\sum_{i=1}^{n} \alpha_i = 1, \alpha_i > 0$ implies $x \in V$. This a simple rewriting of the expression

$$x = \sum_{i=1}^{n-1} \alpha_i x_i + \alpha_n x_n$$

If $\alpha_n = 1$ by definition one has $x \in V$. If $\alpha_n < 1$ this implies $\sum_{i=1}^{n-1} \alpha_i = 1 - \alpha_n \neq 0$. Therefore one may write

$$x = (1 - \alpha_n) \sum_{i=1}^{n-1} \frac{\alpha_i}{1 - \alpha_n} x_i + \alpha_n x_n$$

with $\sum_{i=1}^{n-1} \frac{\alpha_i}{1-\alpha_n}$. By hypothesis, $x' = \sum_{i=1}^{n-1} \frac{\alpha_i}{1-\alpha_n} x_i \in V$. There one has

$$x = (1 - \alpha_n)x' + \alpha x_n$$

By definition of the convex set $x \in V$.

Now, let us prove that $V = \{x | x = \sum_{i=1}^{n} \alpha_i x_i, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i > 0\}$ is a convex set. Let us take two points $x, y \in V$ and let us prove that $x\alpha + (1 - \alpha)y \in V$ for any $\alpha \in [0, 1]$

$$\begin{aligned} x &= \sum_{i=1}^{n} \alpha_i x_i \\ y &= \sum_{i=1}^{n} \beta_i x_i \\ &\Rightarrow \quad \alpha x + (1 - \alpha)y = \sum_{i=1}^{n} \alpha\alpha_i x_i + (1 - \alpha)\beta_i x_i \\ &\Leftrightarrow \quad \alpha x + (1 - \alpha)y = \sum_{i=1}^{n} (\alpha\alpha_i + (1 - \alpha)\beta_i) x_i \end{aligned}$$

Since, $\sum_{i=1}^{n} \alpha\alpha_i + (1 - \alpha)\beta_i = 1$ and $\alpha\alpha_i + (1 - \alpha)\beta_i > 0$ the proof is complete.

10.4 Interpolation

10.4.1 3-linear Lagrange interpolation

Definition 10.4.1:

Let E be a vector space of dimension n and $(b_1, ..., b_n)$ a basis of this space. Let $G \equiv \{x | x = X_0 + \sum_{i=1}^{n} \alpha_i s_i b_i, \alpha_i \in [0, ..., N_i]\}$ be a grid with $N_1, ..., N_n \in \mathbb{N}$. Let $f : E \to \mathbb{R}$ be a function known at the grid points. The n-linear Lagrange interpolation is defined for each $x \in env(G)$ by the following recurrence formula

$$f_j(x_1, ..., x_j, X_{j+1,k_1}, ..., X_{n,k_{n-j}}) \equiv$$
$$f_{j-1}(x_1, ..., x_{j-1}, X_{j,l}, X_{j+1,k_1}, ..., X_{n,k_{n-j}}) \frac{x_j - X_{j,l+1}}{X_{j,l} - X_{j,l+1}} +$$
$$f_{j-1}(x_1, ..., x_{j-1}, X_{j,l+1}, X_{j+1,k_1}, ..., X_{n,k_{n-j}}) \frac{x_j - X_{j,l}}{X_{j,l+1} - X_{j,l}} \quad (10.1)$$
$$f_0 \equiv f \quad (10.2)$$

where $X_{j,l} \equiv X_{0,j} + l s_j$, $k_i \in [1, ..., N_{j+i}]$ and $x_j \in [X_{j,l}, X_{j,l+1}]$.

Fig 10.1 illustrates this recurrence in the case of an Euclidean vector space of dimension 3 and for an orthonormal basis $b_1, ..., b_3$.

Proposition 10.4.1:

The recurrence formula can be expressed as

$$f_n(x_1, ..., x_n) = \sum_{i_1=0}^{1} ... \sum_{i_n=0}^{1} f(X_{1,l_1+i_1}, ..., X_{n,l_n+i_n}) \prod_{i=1}^{n} \frac{x_i - X_{i,l_i+1-i_i}}{X_{i,l_i+i_i} - X_{i,l_i+1-i_i}} \quad (10.3)$$

where $l_1, ..., l_n \in \mathbb{N}$ such that $x_i \in [X_{i,0} + l_i s_i, X_{i,0} + (l_i + 1) s_i]$.

Demonstration 10.4.1:

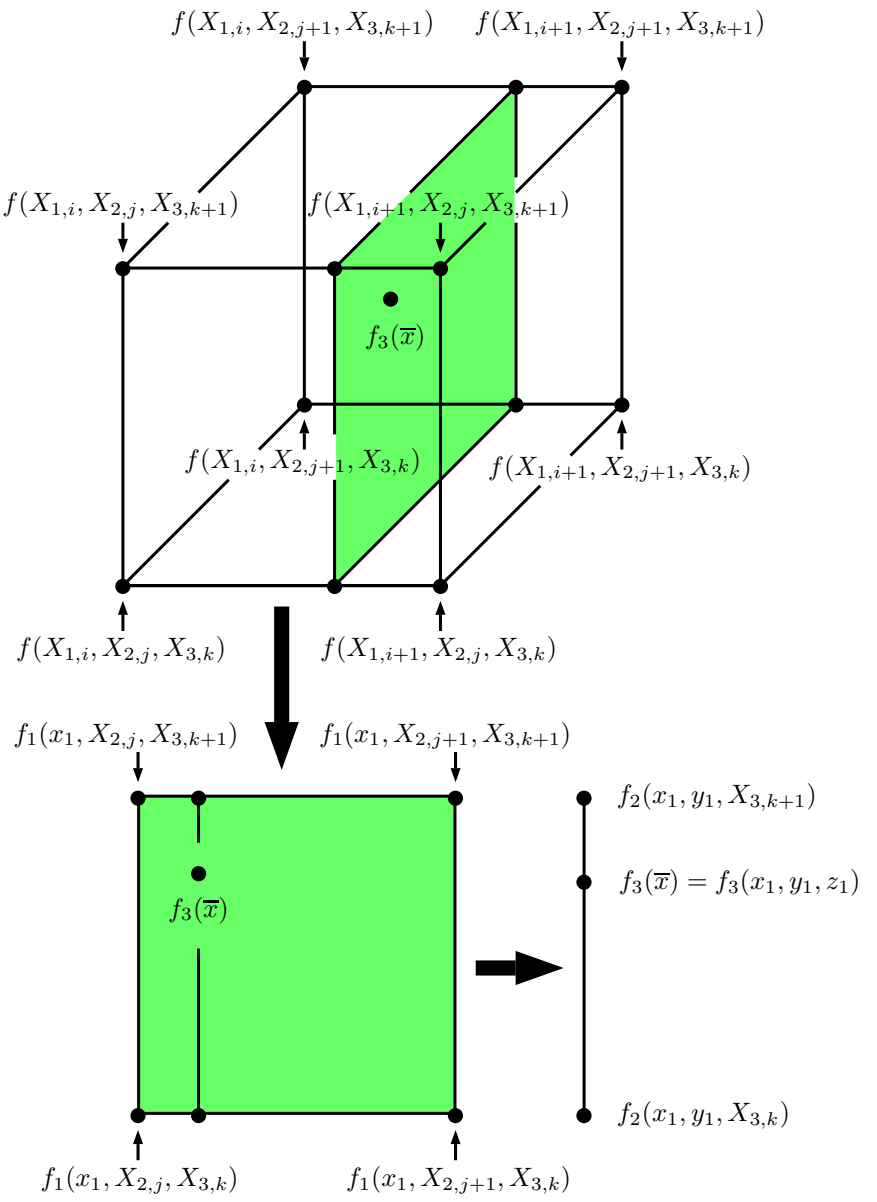


Figure 10.1: 3-linear Lagrangian interpolation for a 3 dimensional vector space.

This can be shown easily by recurrence on the vector space dimension. For a one-dimensional vector space it's direct. Let us assume that it's true for $n-1$ and let us show it is true for n. By definition one has

$$f_n(x_1,...,x_n) = f_{n-1}(x_1,...,x_{n-1},X_{n,l})\quad\frac{x_n - X_{n,l+1}}{X_{n,l} - X_{n,l+1}} + f_{j-1}(x_1,...,x_{n-1},X_{n,l+1})\frac{x_n - X_{n,l}}{X_{n,l+1} - X_{n,l}} \qquad (10.4)$$

The functions $f_{n-1}(x_1,...,x_{n-1},X_{n,l})$ and $f_{j-1}(x_1,...,x_{n-1},X_{n,l+1})$ can be seen as a function defined one an n-1-dimensional vector space with respectively one parameter $X_{n,l}$ and $X_{n,l+1}$. Injecting the formula for these two functions for the sub-grid defined by $G' = proj(G)$ where $proj$ is the parallel projection defined by $proj(\sum_{i=1}^{n} \alpha_i b_i) = \sum_{i=1}^{n-1} \alpha_i b_i$ one gets

$$
\begin{aligned}
f_n(x_1,...,x_n) = & \sum_{i_1=0}^{1} ... \sum_{i_{n-1}=0}^{1} f(X_{1,l_1+i_1},...,X_{1,l_{n-1}+i_{n-1}},X_{n,l}) \\
& \prod_{i=1}^{n-1} \frac{x_i - X_{i,l_i+1-i_i}}{X_{i,l_i+i_i} - X_{i,l_i+1-i_i}} \frac{x_n - X_{n,l+1}}{X_{n,l} - X_{n,l+1}} \\
& + \sum_{i_1=0}^{1} ... \sum_{i_{n-1}=0}^{1} f(X_{1,l_1+i_1},...,X_{1,l_{n-1}+i_{n-1}},X_{n,l+1}) \\
& \prod_{i=1}^{n-1} \frac{x_i - X_{i,l_i+1-i_i}}{X_{i,l_i+i_i} - X_{i,l_i+1-i_i}} \frac{x_n - X_{n,l}}{X_{n,l+1} - X_{n,l}} \qquad (10.5) \\
f_n(x_1,...,x_n) = & \sum_{i_1=0}^{1} ... \sum_{i_{n-1}=0}^{1} f(X_{1,l_1+i_1},...,X_{1,l_{n-1}+i_{n-1}},X_{n,l}) \\
& \prod_{i=1}^{n} \frac{x_i - X_{i,l_i+1-i_i}}{X_{i,l_i+i_i} - X_{i,l_i+1-i_i}} \\
& + \sum_{i_1=0}^{1} ... \sum_{i_{n-1}=0}^{1} f(X_{1,l_1+i_1},...,X_{1,l_{n-1}+i_{n-1}},X_{n,l+1}) \\
& \prod_{i=1}^{n} \frac{x_i - X_{i,l_i+1-i_i}}{X_{i,l_i+i_i} - X_{i,l_i+1-i_i}} \qquad (10.6) \\
f_n(x_1,...,x_n) = & \sum_{i_1=0}^{1} ... \sum_{i_n=0}^{1} f(X_{1,l_1+i_1},...,X_{1,l_n+i_n}) \\
& \prod_{i=1}^{n} \frac{x_i - X_{i,l_i+1-i_i}}{X_{i,l_i+i_i} - X_{i,l_i+1-i_i}} \qquad (10.7)
\end{aligned}
$$

where $i_n = l$.

This proposition is used in the subsection concerning the implementation of the interpolation kernel.

corollary 1:

The n-linear interpolation is invariant under any permutation of coordinates.

Demonstration 10.4.2:

This is direct consequence of the proposition 10.4.1 that leads to a formula independent of the order of coordinates.

10.5 Unicity theorem of Poisson equation - Classic analysis

Theorem 1:

Let $D \in \mathbb{R}^3$ be a compact domain with regular piecewise oriented boundary surfaces ∂S (i.e a differential oriented manifold) then, the Poisson equation has a unique solution if the Dirichlet conditions are imposed on the boundaries.

Demonstration 10.5.1:

This is relatively straightforward, using the first Green's identity. Let us first recall the Green's identity. Let $\alpha : D \to \mathbb{R}^3$ be a C^1 function and $\beta : D \to \mathbb{R}^3$ be a C^2 function. Let us define $\overline{\gamma} \equiv \alpha\overline{\beta}$ and let us apply the Ostrogradski's theorem

$$\int_D \nabla \cdot \overline{\gamma} dV = \int_{\partial S} \overline{\gamma} d\overline{S} \tag{10.8}$$

The divergence of γ can be rewritten as follows $\nabla \cdot \alpha\overline{\beta} = \alpha\nabla^2\beta + \nabla\alpha \cdot \nabla\beta$. The first Green's identity consists simply in injecting this expression in the Ostrogradski's theorem

$$\int_D \alpha\nabla^2\beta + \nabla\alpha \cdot \nabla\beta dV = \int_{\partial S} \alpha\overline{\beta} d\overline{S} \tag{10.9}$$

Let $\phi_1, \phi_2 : D \to \mathbb{R}^3$ be two C^2 function satisfying $\nabla^2\phi_1 = \nabla^2\phi_2 = 0$ and $\phi_1 = \phi_2$ on ∂S. Let us define $\xi \equiv \phi_1 - \phi_2$ let us set $\alpha = \beta = \xi$ in the first Green's identity. One ends up with

$$\int_D (\nabla\xi)^2 dV = \int_{\partial S} \xi\overline{\xi} dS = 0 \tag{10.10}$$

Since ξ is a continuous function, this implies that $\nabla\xi = 0$ on D or in other words that ξ is constant on D. Therefore, since $\xi = 0$ on ∂S surrounding D, this implies that $\xi = 0$ everywhere and the uniqueness is demonstrated.

10.6 Coulomb potential energy and energy conservation

By defintion of the kinetic energy of a bullet i one has

$$\begin{aligned}
\Delta E_{c,i} &\equiv \int_{t_1}^{t_2} (\overline{F}(\overline{x}_i(t'),t)|\overline{v}_i(t')dt' \\
&= m\int_{t_1}^{t_2} (\overline{a}_i(t')|\overline{v}_i(t')dt' \\
&= m\int_{t_1}^{t_2} \frac{1}{2}\frac{d||\overline{v}_i(t')||^2}{dt'}dt' \\
&= \frac{mv^2(t_2)}{2} - \frac{mv^2(t_1)}{2}
\end{aligned}$$

Now, the forces are split into two component: the source field and the Coulomb field. The latter leads to a potential while the first cannot in general be assumed as such.

$$\Delta E_{c,i} = q\int_{t_1}^{t_2} (\overline{E}_s(\overline{x}_i(t'),t)|\overline{v}_i(t')dt' + q\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2} \frac{q}{4\pi\epsilon_0}\frac{(\overline{x}_i - \overline{x}_j|\overline{v}_i(t'))}{||\overline{x}_i - \overline{x}_j||^3}$$

Let us define $\Delta E_{s,i} \equiv q\int_{t_1}^{t_2}(\overline{E}_s(\overline{x}_i(t'),t)|\overline{v}_i(t')dt'$. Since, $\nabla_{\overline{x}_i}\frac{-1}{||\overline{x}_i-\overline{x}_j||} = \frac{\overline{x}_i-\overline{x}_j}{||\overline{x}_i-\overline{x}_j||^3}$ and $\nabla_{\overline{x}_j}\frac{-1}{||\overline{x}_i-\overline{x}_j||} = -\frac{\overline{x}_i-\overline{x}_j}{||\overline{x}_i-\overline{x}_j||^3}$ one has

$$\begin{aligned}
E_{c,i} &\equiv \Delta E_{s,i} + \frac{q^2}{4\pi\epsilon_0}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}(\nabla\frac{-1}{||\overline{x}_i - \overline{x}_j||}|\overline{v}_i(t'))dt' \\
&= \Delta E_{s,i} + \frac{q^2}{4\pi\epsilon_0}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}\frac{d}{dt'}\frac{-1}{||\overline{x}_i - \overline{x}_j||} + \frac{(\overline{x}_i - \overline{x}_j|\overline{v}_j(t'))}{||\overline{x}_i - \overline{x}_j||^3}dt' \\
&= \Delta E_{s,i} + \frac{q^2}{4\pi\epsilon_0}\sum_{j=1,j\neq i}^{M}(\frac{1}{||\overline{x}_i(t_1) - \overline{x}_j(t_1)||} - \frac{1}{||\overline{x}_i(t_2) - \overline{x}_j(t_2)||}) + \frac{q^2}{4\pi\epsilon_0}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}(\overline{x}
\end{aligned}$$

Now, the total variation of kinetric energy is given by

$$\begin{aligned}\Delta E_c &\equiv \sum_{i=1}^{M}\Delta E_{s,i} + \frac{q^2}{4\pi\epsilon_0}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\left(\frac{1}{||\overline{x}_i(t_1)-\overline{x}_j(t_1)||}-\frac{1}{||\overline{x}_i(t_2)-\overline{x}_j(t_2)||}\right)\\ &- \frac{q^2}{4\pi\epsilon_0}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}\frac{(\overline{x}_i+\overline{x}_j|\overline{v}_j(t'))}{||\overline{x}_i-\overline{x}_j||^3}dt'\end{aligned}$$

One can observe that

$$\begin{aligned}\Delta E_c &= \sum_{i=1}^{M}\Delta E_{s,i} + \frac{q^2}{4\pi\epsilon_0}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\left(\frac{1}{||\overline{x}_i(t_1)-\overline{x}_j(t_1)||}-\frac{1}{||\overline{x}_i(t_2)-\overline{x}_j(t_2)||}\right)\\ &- \frac{q^2}{4\pi\epsilon_0}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}\frac{(\overline{x}_i+\overline{x}_j|\overline{v}_j(t'))}{||\overline{x}_i-\overline{x}_j||^3}dt'\\ &= \sum_{i=1}^{M}\Delta E_{s,i} + q\frac{q^2}{4\pi\epsilon_0}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}\frac{(\overline{x}_i-\overline{x}_j|\overline{v}_i(t'))}{||\overline{x}_i-\overline{x}_j||^3}dt'\end{aligned}$$

This leads to the identity

$$\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\left(\frac{1}{||\overline{x}_i(t_1)-\overline{x}_j(t_1)||}-\frac{1}{||\overline{x}_i(t_2)-\overline{x}_j(t_2)||}\right)=\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\int_{t_1}^{t_2}\frac{(\overline{x}_i-\overline{x}_j|\overline{v}_i(t'))}{||\overline{x}_i-\overline{x}_j||^3}dt'$$

The total kinetic energy is therefore given by

$$\Delta E_c \equiv \sum_{i=1}^{M}\Delta E_{s,i} + \frac{q^2}{4\pi\epsilon_0}\sum_{i=1}^{M}\sum_{j=i+1}^{M}\left(\frac{1}{||\overline{x}_i(t_1)-\overline{x}_j(t_1)||}-\frac{1}{||\overline{x}_i(t_2)-\overline{x}_j(t_2)||}\right)$$

10.7 Special relativity

One may wonder if the special relativity theory should be used for the beam dynamics solver. This short chapter is devoted to answer this question. First, a brief recall about the origin of special relativity is presented in section 10.7.1. Then the theory is applied to our specific application in section 10.7.2. The theory of special relativity can be found in many books and publications. We suggest the reader to refer to [15] for any complementary reading.

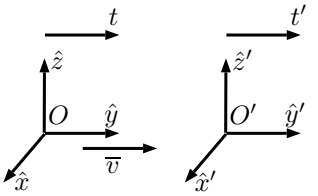


Figure 10.2: Example of two intertial frames

10.7.1 Special relativity origin and main results

The Galilean invariance claims that all laws of physics should be invariant between inertial frames. An inertial frame is a system of coordinates for which any free body (i.e no external force applied on it) has a constant uniform rectilinear motion relative to it. Before the theory of special relativity, the space-time (four-vector) relation between two inertial frames was given by the Galilean transformation namely

$$\begin{pmatrix} \overline{x}' \\ t' \end{pmatrix} = \begin{pmatrix} R & \overline{v}' \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \overline{x} \\ t \end{pmatrix} + \begin{pmatrix} \delta\overline{x}' \\ \delta t' \end{pmatrix} \tag{10.11}$$

where R is a rotation matrix, $\delta t'$ and $\delta\overline{x}'$ respectively the time shift between the clocks in each frame and the initial displacement between the two inertial frames. One considers an orthonormal basis for the geometric coordinates.

Maxwell's equations under this transformation are not invariant. Additionally, around the beginning of the twentieth century several physicists discovered that the speed of light is invariant in an inertial frame. These discoveries pushed Albert Einstein to formulate the two famous principles of special relativity

- The laws of physics must be invariant in an inertial frame
- The speed of light is invariant in an inertial frame

The second postulate allows one to obtain the well-known Lorentz transformations that link the space-time coordinates of two inertial frames. Let us calculate this transformation for the frames shown Fig. 10.2 In case the space-time orientations is preserved, it reads

$$\begin{pmatrix} c\Delta t' \\ x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \gamma & -\beta\gamma & 0 & 0 \\ -\beta\gamma & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\Delta t \\ x \\ y \\ z \end{pmatrix} \tag{10.12}$$

where $\gamma \equiv \frac{1}{\sqrt{1-\beta^2}}$ and $\beta \equiv \frac{v}{c}$. It can be shown that under this transformation, Maxwell's equations are invariant. Now, the Newton's laws of dynamics must be still invariant. A simple way to try to obtain this generalization is to consider the collision of two rigid bodies observed in two distinct inertial frames. Since an elastic collision between two rigid bodies involves conservation of momentum and energy which themselves involve velocities, let us first try the find the relation between the velocities of a given body as observed in different inertial frames.

One may wonder if it is possible to obtain such a relation with the help of the Lorentz transform itself. In order to do that, let us try to find a scalar space-time invariant. Indeed, if one can obtain a scalar invariant that would be related directly to the proper time (the time of the clock placed in a intertial frame that would lie on the body in motion) and therefore to the time itself in each frame one might be able to calculate the derivative of the positions relative to this invariant. The proper time is defined as the time in the proper frame and the proper frame is defined by the frame lying on the body so that the velocity and the position expressed relatively to it and at any time is equal to zero. Notice that such a frame might undergo some accelerations and therefore is not a inertial frame. But, at a given time, an inertial frame can be placed on the body in motion. Therefore, a proper frame is defined at any time by such an intertial frame.

The invariant bilinear form under Lorentz's transformation is given by

$$M \equiv \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10.13}$$

This bilinear form is not a scalar product since it is not positive definite. However, it defines a pseudo-metric called the Minkowski metric. It can

be shown that the Lorentz transformation is in fact a particular element of the Poincarré's group that is defined by the group of isometries of the Minkowski space. In passing, it is a Lie group. One may define a new invariant quantity given by

$$\Delta s^2 = c^2\Delta t^2 - \Delta x^2 - \Delta y^2 - \Delta z^2 = M_{\mu\eta}\Delta x^\mu \Delta y^\eta \tag{10.14}$$

using the Einstein notation for the indices. The proper time can be calculated with

$$\tau(t) = \tau(t_0) + \int_{t_0}^{t} \frac{1}{c}\sqrt{M_{\mu\eta}dx^\mu dx^\eta} \tag{10.15}$$

Since the velocity in the proper frame is equal to zero, and $\frac{dx^0}{dt} = \frac{1}{\gamma}$ one gets

$$\tau = \tau_0 + \int \frac{1}{\gamma}dt$$

Using the invariant with the four-vector position, one may obtain a new four-vector (a four-vector is a vector that can be transformed using the Lorentz transformations). For instance, one may obtain the four-vector velocity as follows

$$\overline{V} \equiv \lim_{\Delta t\to 0}\begin{pmatrix} c^2\frac{\Delta t}{\Delta s} \\ c\frac{\Delta \overline{x}}{\Delta s}\end{pmatrix} \tag{10.16}$$

$$\overline{V} = \lim_{\Delta t\to 0}\begin{pmatrix} c^2\frac{\Delta t}{\sqrt{c^2\Delta t^2-\Delta x^2-\Delta y^2-\Delta z^2}} \\ c\frac{\Delta x}{\sqrt{c^2\Delta t^2-\Delta x^2-\Delta y^2-\Delta z^2}}\end{pmatrix} \tag{10.17}$$

$$\overline{V} = \lim_{\Delta t\to 0}\begin{pmatrix} c^2\frac{1}{\sqrt{c^2-(\frac{\Delta x}{\Delta t})^2-(\frac{\Delta y}{\Delta t})^2-(\frac{\Delta z}{\Delta t})^2}} \\ c\frac{\frac{\Delta x}{\Delta t}}{\sqrt{c^2-(\frac{\Delta x}{\Delta t})^2-(\frac{\Delta y}{\Delta t})^2-(\frac{\Delta z}{\Delta t})^2}}\end{pmatrix} \tag{10.18}$$

By assumption, the function $x : T \subset \mathbb{R} \to D \subset \mathbb{R}^3$ with T, D two open subsets is differentiable. Therefore, one has

$$\lim_{\Delta t\to 0}\frac{\Delta \overline{x}}{\delta t} = \overline{v} \tag{10.19}$$

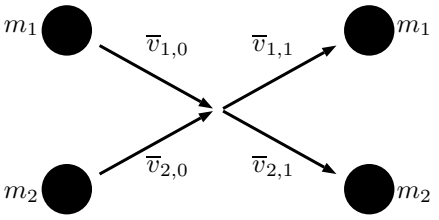


Figure 10.3: Elastic collision between two bodies

By continuity of the square power, the addition property of the limits given by proposition 10.1.2, the continuity of the square root function and the continuity of the inverse function everywhere except at zero one has

$$\lim_{\Delta t \to 0} \frac{1}{\sqrt{c^2 - (\frac{\Delta x}{\Delta t})^2 - (\frac{\Delta y}{\Delta t})^2 - (\frac{\Delta z}{\Delta t})^2}} = \frac{1}{\sqrt{c^2 - v_x^2 - v_y^2 - v_z^2}} \tag{10.20}$$

Now, by the property of the product of two limits again given by proposition 10.1.2 one has

$$\overline{V} = \begin{pmatrix} c^2 \frac{1}{\sqrt{c^2 - v_x^2 - v_y^2 - v_z^2}} \\ c \frac{\overline{v}}{\sqrt{c^2 - v_x^2 - v_y^2 - v_z^2}} \end{pmatrix} \tag{10.21}$$

$$\overline{V} = \begin{pmatrix} c\gamma \\ \gamma \overline{v} \end{pmatrix} \tag{10.22}$$

$$\tag{10.23}$$

In order to find the new laws of dynamics, let us write the equations of conservation of the momentum and the energy of two rigid bodies colliding (elastic collision) as show Fig. 10.3. We know that the momentum and the energy must be preserved and we have an approximation of what should be the momentum and the energy at a speed relatively low in comparison to the light speed. Thanks to the four-vector velocity, we have a clue of what should be the new definition of momentum and energy. Indeed, we have

$$\overline{p}_{1,0} + \overline{p}_{2,0} = \overline{p}_{1,1} + \overline{p}_{2,1} \tag{10.24}$$

$$E_{1,0} + E_{2,0} = E_{1,1} + E_{2,1} \tag{10.25}$$

Because of the linearity of the Lorentz's transformation, it is natural to consider the four-vector velocity as the vector containing partially the

information on the momentum and the energy. Indeed, let us assume that there exists a quantity called the mass m_0 invariant in each inertial frame, let us define the four-vector momentum by

$$\overline{P} \equiv m_0 \overline{V} \tag{10.26}$$

Let us postulate that the four-vector momentum has to be conserved in case of an elastic collision of two rigid bodies. Then, because the four-vector momentum is transformed from one inertial frame to another with Lorentz's transformation the postulate is invariant in each inertial frame. This sentence can be formulated with equations as follows

$$\begin{aligned} m_{1,0}\overline{V}_{1,0} + m_{2,0}\overline{V}_{2,0} &= m_{1,0}\overline{V}_{1,1} + m_{2,0}\overline{V}_{2,1} && (10.27)\\ &\Downarrow\ L \\ m_{1,0}\overline{V}'_{1,0} + m_{2,0}\overline{V}'_{2,0} &= m_{1,0}\overline{V}'_{1,1} + m_{2,0}\overline{V}'_{2,1} && (10.28) \end{aligned}$$

Furthermore, when the velocities of the bodies are much lower than the speed of light, the approximation of γ at the first order reads

$$\gamma(v) \approx 1 + \frac{v^2}{2c^2} \tag{10.29}$$

Injecting this first order approximation in the four-vector momentum one gets

$$m_0\overline{V} \approx \begin{pmatrix} (m_0c^2 + \frac{m_0v^2}{2})\frac{1}{c} \\ m_0\overline{v} \end{pmatrix} \tag{10.30}$$

Since m_0 is invariant, m_0c^2 is also invariant. Therefore, for low velocities, the classic conservation of energy and momentum is respected. All these observations suggest to postulate the following new definitions for the momentum and energy

$$\begin{aligned} \overline{p} &\equiv \gamma m_0 \overline{v} && (10.31)\\ E &\equiv \gamma m_0 c^2 && (10.32) \end{aligned}$$

where m_0 is the mass, $\overline{v}$ the velocity. The famous formula $E_0 = m_0c^2$ is the so-called energy at rest. This formula links the mass to the energy at

rest. Hence, there is an equivalence between mass and energy. This is one of the greatest discovery of the twentieth century.

The total energy $E = \gamma m_0 c^2$ is equal to the energy at rest plus the kinetic energy. Therefore, the kinetic energy is given by

$$T \equiv E - E_0 = (\gamma - 1) m_0 c^2 \tag{10.33}$$

It is interesting to calculate the expression of the force

$$\overline{f} = \frac{d\overline{p}}{dt} = m_0 \frac{d\gamma \overline{v}}{dt} = m_0 \frac{d\gamma}{dt} \overline{v} + m_0 \gamma \overline{a} \tag{10.34}$$

The derivation of γ can be easily performed and reads

$$\frac{d\gamma}{dt} = \gamma^3 \frac{< \overline{v} | \overline{a} >}{c^2} \tag{10.35}$$

The expression of the force is therefore

$$\overline{f} = m_0 \gamma (\frac{\gamma^2}{c^2} < \overline{v} | \overline{a} > \overline{v} + \overline{a}) \tag{10.36}$$

It would be convenient to obtain the expression of the acceleration in function of the force. The expression of the force is equal to the addition of two components, one parallel to the speed vector and one parallel to the acceleration. This equation can be split into the some of two orthogonal components. Let $\overline{a}_{\|} \equiv < \overline{a} | \hat{v} > \hat{v}$ be the parallel component of the acceleration relatively to the speed vector and let $\overline{a}_{\perp} \equiv \overline{a} - < \overline{a} | \hat{v} > \hat{x}$. It is clear that

$$\begin{aligned} < \overline{v} | \overline{a} > \hat{v} &= < \overline{v} | \overline{a}_{\|} > \hat{v} && (10.37) \\ &= v ||\overline{a}_{\|}|| \hat{v} && (10.38) \\ &= v \overline{a}_{\|} && (10.39) \end{aligned}$$

Injecting this result in the expression of the force one gets

$$\overline{f} = m_0 \gamma (\frac{\gamma^2}{c^2} v^2 \overline{a}_{\|} + \overline{a}) \tag{10.40}$$

Now, using the orthogonal decomposition of the acceleration one obtains

$$\begin{aligned}
\overline{f} &= m_0\gamma(\frac{\gamma^2}{c^2}v^2\overline{a}_{\parallel} + \overline{a}_{\parallel} + \overline{a}_{\perp}) && (10.41)\\
&= m_0\gamma((1+\frac{\gamma^2}{c^2}v^2)\overline{a}_{\parallel} + \overline{a}_{\perp}) && (10.42)\\
&= m_0\gamma(\frac{c^2}{c^2-v^2}\overline{a}_{\parallel} + \overline{a}_{\perp}) && (10.43)\\
&= m_0\gamma(\gamma^2\overline{a}_{\parallel} + \overline{a}_{\perp}) && (10.44)\\
&= m_0\gamma\overline{a}_{\perp} + m_0\gamma^3\overline{a}_{\parallel} && (10.45)
\end{aligned}$$

This expression allows one to easily calculate the expression of the acceleration in function of the force. Indeed, using the orthogonal property of the two components and the fact that $\overline{a}_{\parallel}$ is collinear by definition to the speed vector one gets

$$\begin{aligned}
\overline{a}_{\parallel} &= \frac{1}{m_0\gamma^3v^2} < \overline{f}|\overline{v} > \overline{v} && (10.46)\\
\overline{a}_{\perp} &= \frac{\overline{f} - m_0\gamma^3\overline{a}_{\parallel}}{m_0\gamma} && (10.47)
\end{aligned}$$

After some trivial algebraic manipulations the acceleration is given by

$$\overline{a} = \frac{1}{m_0\gamma}(\overline{f} - \frac{< \overline{f}|\overline{v} > \overline{v}}{c^2}) \qquad (10.48)$$

10.7.2 RFQ and special relativity

This section aims at answering the following question: does the special relativity should be considered for our RFQ simulations. Considering the fact that the output kinetic energy of the particles should be 1.5MeV, the special relativity parameters read for this worse case

$$\gamma = \frac{E}{m_0c^2} = \frac{1.5*10^6*1.6*10^{-19} + m_0c^2}{1.7*10^-27*(3*10^8)^2} = 1.0015 \qquad (10.49)$$

The β parameter is given by

$$\beta = \sqrt{1-\frac{1}{\gamma^2}} = \sqrt{1-\frac{1}{1.0015^2}} = 0.055 \qquad (10.50)$$

In order to evaluate the impact of the relativist effects on the acceleration let us try to bound the relative acceleration factor defined by

$$\frac{|a_i - a_{c,i}|}{|a_{c,i}|} \tag{10.51}$$

where $\overline{a} = \sum_{i=1}^{3} a_i \hat{x}_i$ and $\overline{a}_c = \sum_{i=1}^{3} a_{c,i} \hat{x}_i$ the acceleration obtained by the third Newton's law.

By definition of the Newton's law, one gets

$$\overline{a} - \overline{a}_c = \frac{1}{m_0}(\overline{f}(\frac{1}{\gamma} - 1) - \frac{< \overline{f} | \overline{v} >}{c^2} \frac{\overline{v}}{\gamma}) \tag{10.52}$$

By definition of the norm, an upper bound can be calculated as follows

$$\begin{aligned} |a_i - a_{c,i}| &= \frac{1}{m_0} |f_i(\frac{1}{\gamma} - 1) - \frac{< \overline{f} | \overline{v} >}{c^2} \frac{v_i}{\gamma}| && (10.53) \\ &\leq \frac{1}{m_0}(|f_i||\frac{1}{\gamma} - 1| + \frac{| < \overline{f} | \overline{v} > |}{c^2} \frac{v_i}{\gamma}) && (10.54) \\ &\leq \frac{1}{m_0}(|f_i||\frac{1}{\gamma} - 1| + \frac{||\overline{f}||v}{c^2} \frac{v_i}{\gamma}) && (10.55) \\ &\leq \frac{1}{m_0}(|f_i||\frac{1}{\gamma} - 1| + \frac{\beta^2}{\gamma} ||\overline{f}||) && (10.56) \end{aligned}$$

Dividing this upper bound by the corresponding absolute value component of the acceleration provided by the Newton third law, one obtains

$$\frac{|a_i - a_{c,i}|}{|a_{c,i}|} \leq |\frac{1}{\gamma} - 1| + \frac{\beta^2}{\gamma} \tag{10.57}$$

Using the values obtained at the beginning of this section, one gets

$$\frac{|a_i - a_{c,i}|}{|a_{c,i}|} \leq 0,0045 \tag{10.58}$$

From the relative error on the acceleration one can obtain an upper bound error for the beam deviation without knowing the integration time Δt required for a particle to cross the accelerator

$$\epsilon_i = |\int_{t_0}^{t_0+\Delta t}\int_{t_0}^{t_0+\Delta t}(a_i - a_{c,i})dt'dt''| \quad (10.59)$$

$$\leq \int_{t_0}^{t_0+\Delta t}\int_{t_0}^{t_0+\Delta t}|a_i - a_{c,i}|dt'dt'' \quad (10.60)$$

$$\leq 0,0045\int_{t_0}^{t_0+\Delta t}\int_{t_0}^{t_0+\Delta t}|a_{c_i}|dt'dt'' \quad (10.61)$$

$$\leq 0,0045L \quad (10.62)$$

where L is the length of the accelerator. Therefore, the value $0,0045$ is too large to conclude that the relativist effects are negligible with this global scale upper bound. The problem has to be investigated more locally. As it will be shown in the chapter devoted to differential equations (Ch. 10.8) the equations of motion obtained for the RFQ (neglecting the space charge) are the so-called Hill's equations. The design parameters of the accelerators have been chosen such that the solutions to these equations have a good stability. A good stability means in particular that a small perturbation in the initial conditions of a particle won't impact significantly its trajectory. Hence, if local perturbations due to the relativist effects are not too large, they should not impact significantly the particles trajectories. There is no easy way to quantify these considerations. Simulations have shown that relativist effects can be neglected for the RFQ simulations. But, for downstream cavities following the RFQ relativist effects must be considered.

10.8 Theory of ordinary differential equations

10.8.1 Introduction

All the theory of ordinary differential equations focuses on solving the so-called Cauchy problem. Most of the results shown in the following sections can be found in [65].

Definition 10.8.1:

Let be $U \subset \mathbb{R} \times \mathbb{R}^n$ an open subset and $f : U \to \mathbb{R}^n$ a continuous function. Let be $u : I \to \mathbb{R}^n$ with $I \subset \mathbb{R}$ an open subset and $(t, u(t)) \in U, \forall t \in I$, the u is called a solution of the differential equation if $u'(t) = f(t, u(t)), \forall t \in I$.

The Cauchy problem consists in finding a solution passing by a initial condition $(t_0, u(t_0) \in U$.

Ordinary means that the equation depends only on one variable. If the equation were depending one several variables that one would call it a partial differential equation. The theory for this family of equations is relatively different from the theory of ordinary differential equations.

10.8.1.1 An intuitive explanation of a differential equation in a single dimensional vector space

Before getting in the theory itself, a short intuitive explanation about how a ordinary differential equation can lead to a unique existing solution. Let us first recall the definition of a derivative of a function of a single variable

Definition 10.8.2:

Let be $f : I \to R$, with I an open subset and let be $t_0 \in I$ a given point, then the derivative a the function at this point t_0 exists if there exists a value $'f(x_0)$ such that

$$\forall \epsilon > 0, \exists \delta > 0 \text{ such that } |x - x_0| < \delta \Rightarrow |f(x) - f(x_0) - f'(x_0)(x - x_0)| < \epsilon |x - x_0|$$

Now let us investigate the meaning of this definition. Fig 10.4 illustrates the definition of the derivative

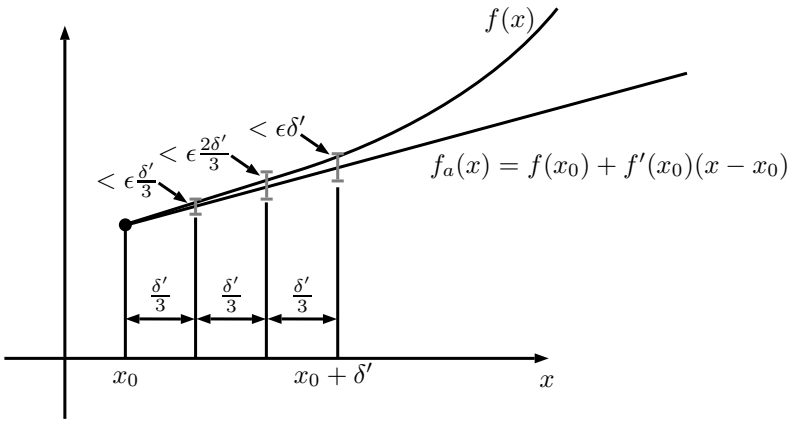


Figure 10.4: Analysis of the derivative definition

Let us take and ϵ smaller than the derivative $f'(x_0)$. Let us take a $\delta' < \delta$ obtained by the definition of the derivative. The definition of the derivative means that for all the values $x \in]x_0 - \delta, x_0 + \delta[$ the gain of information given by $f'(x_0)(x - x_0)$ can be infinitely better than $\epsilon|x - x_0|$. Thanks to this analysis, one can understand from what a derivative give a local information of the function. The fundamental theorem of analysis[1] confirms this intuition by linking the derivative of a function to its primitive.

Theorem 2:

Let be $I \subset \mathbb{R}$ an open interval and $f : I \to \mathbb{R}$ a continuous function, then the primitive of f defined by $F(x) = F(a) + \int_a^x f(x')dx'$ where $x, a \in I$ and $\int$ is the Riemann integral, is derivable and its derivative is equal to f.

Demonstration 10.8.1:

Let us calculate the limit of $\frac{F(x)-F(x_0)}{x-x_0} = \frac{\int_{x_0}^{x} f(x')dx'}{x-x_0}$. Since f is continuous function, by definition one has $\forall\epsilon > 0, \exists\delta > 0$, such that $|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon$. Hence, one has $f(x_0) - \epsilon \leq f(x) \leq f(x_0) + \epsilon$ for any $x \in [x_0 - \delta, x_0 + \delta]$. Injective this expression in the ratio above one gets

[1] Here one focuses only on the first version of the fundamental theorem. A stronger version exists for the Lebesgue integral

$$|x - x_0| < \delta \Rightarrow f(x_0) - \epsilon \leq \frac{\int_{x_0}^{x} f(x')dx'}{x - x_0} \leq f(x_0) + \epsilon$$
$$\Leftrightarrow |x - x_0| < \delta \Rightarrow |\frac{\int_{x_0}^{x} f(x')dx'}{x - x_0} - f(x_0)| \leq \epsilon$$

This ends the demonstration.

Now, let us take a function $f : I \to \mathbb{R}$ where $I \subset \mathbb{R}$ an open interval and let us assume that the function is C^1. Let us take an closed interval $[a, b] \subset I$, the derivative $f'(x)$ is therefore uniformly continuous i.e $\forall \epsilon > 0, \exists \delta > 0$, such that $\forall x_1, x_2 \in [a, b]$ such that $|x_1 - x_2| < \delta \Rightarrow |f'(x_1) - f'(x_2)| < \epsilon$. Choosing a $\epsilon > 0$, one can cover the interval $[a, b]$ with interval $I_i = [a_i, b_i[$ of length strictly smaller than δ and such that $I = \sqcup_{i=1}^{N} I_i$. Let us define the middle point of each of these interval by $c_i = \frac{b_i + a_i}{2}$ then one has

$$\sum_{i=1}^{N} f'(c_i)(b_i - a_i) - \epsilon(b - a) \leq \int_a^b f'(x')dx' = f(b) - f(x) \leq \sum_{i=1}^{N} f'(c_i)(b_i - a_i) + \epsilon(b - a)$$

This means that knowing the derivative of a function and an initial condition $f(x_0)$ one may recover with a precision as accurate as desired the function at any point $f(x)$ knowing the value of the derivative on several points. The function $f(t, x)$ gives the value of the derivative at any point $(t, x) \in U$. Therefore, using this function to evaluate the derivative of x might be used to recover the approximate the Cauchy solution. However, since value of x is not exact for all integration steps, the value provided by the function is not exactly the value that one would obtain with an exact solution. This could have been shown directly from the definition of the derivative.

One could have chosen any other point in the interval $[a_i, b_i[$ since the error if bounded for all $|x - y| < \delta$. The famous Euler method consists in choosing the first point of the interval namely a_i. The equation becomes in this case

$$\sum_{i=1}^{N} f'(a_i)(b_i - a_i) - \epsilon(b - a) \leq \int_a^b f'(x')dx' = f(b) - f(x) \leq \sum_{i=1}^{N} f'(a_i)(b_i - a_i) + \epsilon(b - a)$$

The approximation is given by

$$\int_a^b f'(x')dx' \approx \sum_{i=1}^{N} f'(a_i)(b_i - a_i)$$

This form is particularly well suited for solving the differential equation defined in 10.8.1. Indeed, at each step of the integration the value of u is known and therefore the derivative can be evaluated at any step of the integration by the approximate value of u.

10.8.2 Uniqueness and existence of a solution

10.8.2.1 Solution maximal

Definition 10.8.3:

Let be I_1, I_2 two open intervals of $\mathbb{R}$ and let be $u_1 : I_1 \to \mathbb{R}^n$ and $u_2 : I_2 \to \mathbb{R}^n$, then u_2 is called an extension of u_1 if $I_1 \subset I_2$ and $u_1(x) = u_2(x), \forall x \in I_1$.

The extension is said to be maximal if any extension $u_2 : I_2 \to \mathbb{R}^2$ of u_1 implies that $I_2 = I_1$.

Proposition 10.8.1:

Let be $U \subset \mathbb{R} \times \mathbb{R}^n$ an open subset and $f : U \to \mathbb{R}^n$ a continuous function. Let be $I \subset \mathbb{R}$ and open subset and $u : I \to \mathbb{R}^n$ a solution of the differential equation $u' = f(t, u)$. Then, there exists a maximal extension of u (no necessary unique).

Demonstration 10.8.2:

Let us bring the bounds of I out, $I = |a, b|$ where the notation $|$ means either] or [. It is sufficient to show that there exists a maximal extension for the upper bound since the lower bound proof consists in the same operations.

Since one does not have the uniqueness of the extension, let us build a series of extension compatible with each other by recurrence. Let be $u_k : |a, b_k| \to \mathbb{R}$ an extension and $S_k \equiv \{x | \exists e : |a, x| \to \mathbb{R}$ an extension of $u_k\} \subset \mathbb{R}$. Let be $c_k \equiv \sup S_k$. By definition of thesupremum, there exists $e : |a, x| \to \mathbb{R}$ and extension such that $c_k - x < \frac{1}{k}$. Let us define $u_{k+1} \equiv e$. The idea behind this definition is to force the series to take larger b if possible while in the same time making this bound converging to a value. Let us proof this last part. Since $S_{k+1} \subset S_k$ one has $c_k \leq c_{k+1}$. One the other hand,

one has $b_k \leq c_k$ and $b_k \leq b_{k-1}$. Let us summarize this observation

$$b = b_0 \leq b_1 \leq b_2 \leq ... \leq b_n \leq ... \leq ... \leq c_m \leq c_{m-1} \leq ... \leq c_1$$

Now let us assume that $\lim_{k\to\infty} c_k = C < \infty$. Then, by definition of b_k one has $\lim_{k\to\infty} b_k = C$. Let us define the extension $e : |a, C| \to \mathbb{R}, x \to e(x) = u_k(x)$ such that $x \in I_k$. Since the extension u_k are consistent, the extension e is well defined. , Furthermore, there is no extension compatible with e that has a larger upper bound than C. Indeed, if it was the case, by definition this extension would be in each S_k and therefore any c_k would be larger than C which is a paradox.

Let us assume that $\lim_{k\to\infty} c_k = \infty$. Then, by definition of b_k one has $\lim_{k\to\infty} b_k = \infty$. One can define an extension $e : |a, \infty| \to \mathbb{R}, x \to e(x) = u_k(x)$ such that $x \in I_k$. Again, the extension u_k are consistent and therefore the extension e is well defined. The upper is maximal and therefore there is no way to extend it any further.

10.8.2.2 Construction of a local solution

Let us try to build a local solution of the differential equation defined in 10.8.1. The idea is to build a solution with the Euler explicit method and take smaller integration in order to converge toward a solution. First one must find a domain of integration which is compatible with the domain U of the continuous function f. Since the integration must be performed in a finite number of step, the interval of integration must be bounded. Let us take a time domain with the following form $D_t \equiv [t_0 - T, t_0 + T]$. Let us assume a solution of the differential equation $u : [t_0 - T, t_0 + T] \to \mathbb{R}^n$ such that $(t, u(t)) \in U, \forall t \in [t_0 - T, t_0 + T]$. The fundamental theorem of analysis tell us that

$$u(t) - x_0 = \int_{t_0}^{t} u'(t')dt' = \int_{t_0}^{t} f(t', u(t'))dt'$$

Let us define a compact $K \equiv [t_0 - T_0, t_0 + T_0] \times B[x_0, r] \subset U$. Since f is a continuous function and K is a compact there is a maximal value of the norm of this function on the compact, $M \equiv \max_{(t,x)\in K} ||f(t, x)||$. Let us take the following norm on $|||| : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}, (t, x) \to |t| + ||x||_2$. For this norm, each derivative in absolute value is smaller than M on K, $|f_i(t, x)| < M$. Therefore, one has

$$||u(t) - x_0|| = || \int_{t_0}^{t} f(t', u(t'))dt'||_2 \leq || \int_{t_0}^{t} |f(t', u(t'))|dt'||_2 \leq || \int_{t_0}^{t} (M, ..., M)dt'||_2 = M(t -$$

In order to make sure the solution remains in the compact domain so that the continuous function f is bounded and the equation above remains valid, one must ensure that $u(t) \in B[x_0, r]$ and $t \in [t_0 - T_0, t_0 + T_0]$. The first condition is ensured if $M(t - t_0) \leq r$. Therefore, taking a time domain $[t_0 - T, t_0 + T]$ with $T \equiv \min(T_0, \frac{r}{M})$ ensures that the solution remains in the compact domain. Let us formalize this result.

Definition 10.8.4:

Let be $C \equiv [t_0 - T, t_0 + T] \times B[x_0, r] \subset U$ a compact domain, then C is said to be a safe cylinder if $T \leq \frac{r}{M}$ where $M \equiv \max_{(t,x) \in C} ||f(t, x)||$.

Proposition 10.8.2:

Let be $K = [t_0 - T_0, t_0 + T_0] \times B[x_0, r]$ a compact such that $K \subset U$, then there exists a safe cylinder $C \subset K$.

Demonstration 10.8.3:

It sufficient to take $T \leq \min(T_0, \frac{r}{M})$ as explained here-above.

Let us try to build a solution thanks to the Euler Explicit Method (EEM) approximation. Let be $I \subset \mathbb{R}$ an interval and $u_k : I \to \mathbb{R}^n$ a series of approximate EEM solution such that $\lim_{k \to \infty} u_k = u$ and $u : I \to \mathbb{R}^n$. Let us try to find the conditions such that u is a solution of the differential equation passing by (t_0, x_0). The error between the potential exact solution and the EEM approximation is defined by

$$E \equiv ||u_k(t) - u(t)||$$

Now let us bring the function f in this error equation. If u is a solution of the differential equation then one would have by definition $u(t) = u(t_0) + \int_{t_0}^{t} f(t', u(t'))dt'$. Now, $u_k'(t)$ is the approximate integration of the function $f(t, u_k(t))$. Let us define a new error that matches E is u is a solution.

$$
\begin{aligned}
E_2 &\equiv || \int_{t_0}^{t} f(t', u(t')) - u_k'(t') dt' || \\
&= || \int_{t_0}^{t} (f(t', u(t')) - f(t', u_k(t'))) + (f(t', u_k(t')) - u_k'(t')) dt' || \\
&\leq || \int_{t_0}^{t} f(t', u(t')) - f(t', u_k(t'))|| + || \int_{t_0}^{t} f(t', u_k(t')) - u_k'(t')) dt' ||
\end{aligned}
$$

If the solution $(t, u_k(t)$ are entirely contain in a compact domain then f is uniformly continuous on this domain. If the solution u_k where converging uniformly toward u then for any error one could find an index from which the first part of the error is bounded. Concerning the second part of the error, again if the EEM solutions are contained in a compact domain, since the function is uniformly continuous on this domain one could define a minimum step such that this norm is smaller than a given epsilon. Let us formalize this idea.

Proposition 10.8.3:

Let be $u : [t_0 - T, t_0 + T] \to \mathbb{R}^n$ an EEM solutions and $[t_0 - T, t_0 + T] \times B[r, x_0]$ is a safe cylindrical domain. Then, the EEM solution image is entirely contained in $[B[r, x_0]$.

Demonstration 10.8.4:

Let be u an EEM solution that is a piecewise C^1 and continuous function with $t_0 < t_1, ..., t_N$ a subdivision of $[t_0, t_0 + T]$ and u is C^1 on each $[t_i, t_{i+1}[$ interval. The interval steps are defined by $h_i \equiv (t_{i+1} - t_i)$. The proposition can be proven by recurrence as follows: for each interval of time $[t_0, t_n]$ let us show that $|u(t) - x_0| < \sum_{i=0}^{n-1} h_i M < r$. The last inequality comes from the fact that $\sum_{i=0}^{N} h_i < T$. For the first integration step of the EEM one has $||f(t_0, x_0)|| \leq M$ and therefore one has $||u(t) - x_0|| = ||f(t_0, x_0)(t - t_0)|| \leq M|t - t_0| \leq Mh_0 \leq r$. Now, let us assume that this property is true till t_{n-1} and let us prove this is true till t_n. By hypothesis, one has $|u(t) - x_0| < \sum_{i=0}^{n-1} h_i M < r, \forall t \in [t_0, t_{n-1}]$. Let us take $t \in [t_{n-1}, t_n]$, one has $|u(t) - x_0| \leq |u(t) - u(t_{n-1}| + |u(t_0) - u_{t_{n-1}}|$. But since $||f(t_{n-1}, x_0)|| \leq M$ once again one has $|u(t) - u(t_{n-1})| \leq M|t - t_{n-1}|$ and hence $|u(t) - x_0| \leq \sum_{i=1}^{n-1} h_n M < r$.

Proposition 10.8.4:

Let be $u : [t_0 - T, t_0 + T] \to \mathbb{R}^n$ an EEM solutions and $[t_0 - T, t_0 + T] \times B[r, x_0]$ is a safe cylindrical domain then

$$\forall \epsilon > 0, \exists \delta > 0, \text{ such that } h_i < \delta, \forall i \Rightarrow \forall t \in [t_0, t_0 + T], ||u'(t) - f(t, u(t))|| < \epsilon$$

Demonstration 10.8.5:

First, let us substitute the value of $u'(t) = f(t_n, u(t_n)), forallt \in [t_n, t_{n+1}]$ in the inequality, one has to prove that

$$\forall \epsilon > 0, \exists \delta > 0, \text{ such that } h_i < \delta, \forall i \Rightarrow \forall t \in [t_0, t_0 + T], ||f(t_n, u(t_n)) - f(t, u(t))|| < \epsilon$$

Since any EEM solution defined on $[t_0, t_0 + T]$ is contained in the compact $K \equiv [t_0 - T, t_0 + T] \times B[x_0, r]$, and since the function f is continuous by the uniform continuity of f on this compact one has

$$\forall \epsilon > 0, \exists \delta > 0, \text{ such that } \forall ||(t_1, x_1) - (t_2, x_2)|| < \delta \Rightarrow ||f(t_1, x_1) - f(t_2, x_2)|| < \epsilon$$

This is true for any norm since each norm is equivalent on a finite dimensional vector space. However, let us the norm defined by $||(t, x)|| \equiv |t| + ||x||_2$. Since the function f is bounded on the compact K, one has

$$||u(t) - u(t_n)|| \leq M|t - t_n| \leq Mh_n, \forall t \in [t_n, t_{n+1}]$$

This implies

$$||(t, u(t)) - (t_n, u(t_n))|| \leq M|t - t_n| + |t - t_n| \leq h_n(M + 1)$$

Hence, if one chooses h_i such that $h_i < \frac{\delta}{M+1}$, by definition of the norm one has

$$||(t, u(t)) - (t, u(t_n))|| < \delta$$

This ends the demonstration.

Now let us demonstrate the local existence of a solution.

Proposition 10.8.5:

Let be $u_k : [t_0, t_0 + T] \to \mathbb{R}^n$ a series of EEM solutions contained in the safe cylindrical compact domain $K = [t_0 - T, t_0 + T] \times B[x_0, r]$ and such that $\lim_{k\to\infty} \max_{i\in[1,\ldots,N_k]} h_i = 0$ where N_k is the number of steps of the k EEM solution. If the series u_k converges uniformly toward a function $s : [t_0, t_0 + T] \to \mathbb{R}^n$ then this function a solution of the differential equation and is contained in the compact domain K.

Demonstration 10.8.6:

Let us prove that

$$|| \int_{t_0}^{t} f(t', u(t'))dt - u(t)|| = 0$$

Proposition 10.8.4 implies that for a given $\epsilon > 0$, exists $K \in \mathbb{N}$ such that

$$\forall k > K, \forall t \in [t_0, t_0 + T], ||u_k'(t) - f(t, u_k(t))|| < \epsilon \qquad (10.63)$$

Now, since f is uniformly continuous on K and u_k converges uniformly, it implies that for a given $\epsilon > 0, \exists K_2 \in \mathbb{N}$ such that

$$||f(t, u(t)) - f(t, u_k(t))|| < \epsilon \qquad (10.64)$$

Indeed, by definition of an uniform continuous function, for a given $\epsilon > 0$ exists a $\delta > 0$ such that

$$||(t_1, x_1) - (t_2, x_2)|| < \delta \Rightarrow ||f(t_1, x_1) - f(t_2, x_2)|| < \epsilon$$

Now, since the series is uniformly convergent, one has

$$\exists K_2, \forall k \geq K_2 \Rightarrow \forall t \in [t_0, t_0 + T], ||u_k(t) - u(t)||_2 < \delta$$

Combining the two former results one gets $\exists K_2, \forall k \geq K_2 \Rightarrow \forall t \in [t_0, t_0 + T], ||f(t, u(t)) - f(t, u_k(t))|| < \epsilon$.

Combining eq. 10.63 and eq. 10.64 one gets

$$\forall k \geq \max(K_2, K) \Rightarrow \forall t \in [t_0, t_0 + T], ||u_k'(t) - f(t, u(t))|| < 2\epsilon$$

After integration, one gets

$$\begin{aligned} \forall k \geq \max(K_2, K) \Rightarrow ||u_k(t) - x_0 - \int_{t_0}^{t} f(t', u(t'))dt'|| &\leq || \int_{t_0}^{t} u_k'(t') - f(t', u(t'))dt'|| \\ &\leq \int_{t_0}^{t} ||u_k'(t') - f(t, u(t'))||dt' \\ &\leq 2\epsilon(t - t_0) \leq 2\epsilon T \end{aligned}$$

Finally, again by the uniform convergence of u_k one gets

$$\exists K_3 \in \mathbb{N}, \forall k > K_3 \Rightarrow \forall t \in [t_0, t_0 + T], ||u_k(t) - u(t)|| < \epsilon$$

Combining the two later results one gets

$$\forall k \geq \max(K_1, K_2, K_3), ||u(t) - x_0 - \int_{t_0}^{t} f(t', u(t'))dt'|| < \epsilon(2T + 1)$$

Since $u(t)$ is a continuous function by property of the uniform convergence, it implies that $u(t) \in C^1[t_0, t_0 + T]$.

Now let us prove that a series of EEM solutions for which the maximum time step length tends toward zero converges uniformly toward a solution.

Proposition 10.8.6:

Let be $K = [t_0 - T_0, t_0 + T_0] \times B[x_0, r] \subset U$ a compact, and $f : U \to \mathbb{R}^n$ a continuous and a local Lipschitz continuous function. In particular, the function is k-Lipschitz continuous on the compact K. Let be $I \subset \mathbb{R}$ and interval and $u_1 : I \to \mathbb{R}^n$ and $u_2 : I \to \mathbb{R}^n$ such that $||f(t, u_1(t)) - u_1'(t)|| < \epsilon_1$, $||f(t, u_2(t)) - u_2'(t)|| < \epsilon_2$ and $t_0 \in I$, then one has

$$||u_1(t) - u_2(t)|| \leq (\epsilon_1 + \epsilon_2)\frac{e^{k(t-t_0)} - 1}{k} \forall t \in I \cap [t_0 - T_1, t_0 + T_2]$$

where $[t_0 - T_1, t_0 + T_2]\ times B[x_0, r]$ is a safe cylinder for which $u_1([t_0 - T_1, t_0 + T_2]) \in B[x_0, r]$ and $u_2([t_0 - T_1, t_0 + T_2]) \in B[x_0, r]$.

Demonstration 10.8.7:

First, let us show that a safe cylinder exists such that $u_1([t_0 - T_1, t_0 + T_2]) \in B[x_0, r]$ and $u_2([t_0 - T_1, t_0 + T_2]) \in B[x_0, r]$. Let us define $T \equiv \min(\frac{r}{M+\epsilon_1+\epsilon_2}$, and $K' = [t_0 - T, t_0 + T] \times B[x_0, r]$ a safe cylinder. Let us take $T_1, T_2 \in \mathbb{R}$ such that $T_1, T_2 \leq T$ and $[t_0 - T_1, t_0 + T_2] \subset I$. If u_1 and u_2 do not escape the compact domain $B[x_0, r]$ for any $t \in I$ then K' satisfies the conditions. On the contrary, let us assume that u_1 escapes the domain $B[x_0, r]$ and let us define $\tau \equiv \inf\{t \in \mathbb{R}^+ | u_1(t_0 + t) \notin B[x_0, r]\}$. Since $\forall t \in [t_0, t_0 + \tau[, ||u_1(t) - x_0|| \leq r$ by continuity of u_1 one has $||u_1(t_0 + \tau) - x_0|| \leq r$ and $\tau \notin \{t \in \mathbb{R}^+ | u_1(t_0 + t) \notin B[x_0, r]\}$. Indeed, let us assume a series t_n converging toward t, then with the help of the norm property and the definition of continuity one has

$$\forall \epsilon > 0, \exists N \in \mathbb{N}, \forall n > N \Rightarrow ||u_1(t_n) - u_1(t_0 + \tau)|| < \epsilon$$
$$|||u_1(t_n) - x_0|| - ||u_1(t_0 + \tau) - x_0||| \leq ||u_1(t_n) - u_1(t_0 + \tau)||$$
$$\Rightarrow \forall \epsilon > 0, \exists N \in \mathbb{N}, \forall n > N \Rightarrow |||u_1(t_n) - x_0|| - ||u_1(t_0 + \tau) - x_0||| < \epsilon$$

This implies that $||u_1(t_0+\tau) - x_0|| < \epsilon + ||u_1(t_n) - x_0|| \leq \epsilon + r, \forall \epsilon$. Now, let use the fact that f is bounded on K to show the safe cylinder property

$$
\begin{aligned}
||u_1(t) - u_1(t_0)|| &= ||\int_{t_0}^{t} u_1'(t')dt'|| \leq \int_{t_0}^{t} ||u_1'(t')||dt', t \in I \\
&\leq \int_{t_0}^{t} ||f(t, u_1(t'))|| + ||u_1'(t') - f(t, u_1(t'))||dt', t \in I \\
&< \int_{t_0}^{t} (M + \epsilon_1)dt' = (t - t_0)(M + \epsilon_1), \forall t \in [t_0, t_0 + \tau]
\end{aligned}
$$

This means that if $t - t_0 \leq T$ where $T \equiv \frac{r}{M+\epsilon_1}$ it implies that $u_1(t) \in B[x_0, r]$. It also implies that $\tau \geq T$. The same reasoning can be done the lower bound of the interval and for u_2. If u_2 is also escaping $B[x_0, r]$ then one ends up with the definition for the upper bound of the interval $T \equiv \frac{r}{M+\max(\epsilon_1,\epsilon_2)}$.

10.8.2.3 Uniqueness and existence of a local solution: The Cauchy-Lipschitz theorem

Now, let us try to bound the distance between the two functions u_1 and u_2.

$$
\begin{aligned}
||u_1(t) - u_2(t)|| &= ||\int_{t_0}^{t} u_1'(t') - u_2'(t')dt'|| \\
&\leq \int_{t_0}^{t} ||u_1'(t') - u_2'(t')||dt' \\
&\leq \int_{t_0}^{t} ||u_1'(t) - f(t', u_1(t'))|| + ||f(t, u_1(t')) - f(t, u_2(t'))|| \\
&+ ||f(t', u_2(t')) - u_2'(t')||dt' \\
&\leq \int_{t_0}^{t} ||f(t', u_1(t')) - f(t', u_2(t'))||dt' + (\epsilon_1 + \epsilon_2)(t - t_0), \forall t \in I \\
&\leq \int_{t_0}^{t} k||u_1(t') - u_2(t')||dt' + (\epsilon_1 + \epsilon_2)(t - t_0), \forall t \in I \cap [t_0 - T, t_0 + T
\end{aligned}
$$

Let us define $x(t) \equiv \int_{t_0}^{t} ||u_1(t') - u_2(t')||dt'$. The inequality reads

$$x'(t) \leq kx(t) + (\epsilon_1 + \epsilon_2)(t - t_0)$$

Now, let us try to obtain an easy integrable inequality

$$
\begin{aligned}
(x'(t) - kx(t))e^{-k(t-t_0)} &\leq (\epsilon_1+\epsilon_2)(t-t_0)e^{-k(t-t_0)} \\
(x(t)e^{-k(t-t_0)})' &\leq (\epsilon_1+\epsilon_2)(t-t_0)e^{-k(t-t_0)} \\
x(t)e^{-k(t-t_0)} - x(t_0) &\leq (\epsilon_1+\epsilon_2)\int_{t_0}^{t}(t'-t_0)e^{-k(t'-t_0)}dt' \\
x(t)e^{-k(t-t_0)} - x(t_0) &\leq (\epsilon_1+\epsilon_2)(-\frac{1}{k}(t-t_0)e^{-k(t-t_0)} + \frac{1}{k}\int_{t_0}^{t} e^{-k(t'-t_0)}dt') \\
x(t)e^{-k(t-t_0)} - x(t_0) &\leq (\epsilon_1+\epsilon_2)(-\frac{1}{k}(t-t_0)e^{-k(t-t_0)} - \frac{e^{-k(t-t_0)}-1}{k^2}) \\
x(t)e^{-k(t-t_0)} - x(t_0) &\leq (\epsilon_1+\epsilon_2)\frac{1-(k(t-t_0)+1)e^{-k(t-t_0)}}{k^2}
\end{aligned}
$$

Since $x(t_0) = 0$ one ends up with

$$
\begin{aligned}
x(t)e^{-k(t_0-t)} &\leq (\epsilon_1+\epsilon_2)\frac{1-(k(t-t_0)+1)e^{-k(t-t_0)}}{k^2} \\
x(t) &\leq (\epsilon_1+\epsilon_2)\frac{e^{k(t-t_0)}-(k(t-t_0)+1)}{k^2}
\end{aligned}
$$

Injecting this result in the first inequality one gets

$$
\begin{aligned}
||u_1(t) - u_2(t)|| &\leq k(\epsilon_1+\epsilon_2)\frac{e^{k(t-t_0)}-(k(t-t_0)+1)}{k^2} + (\epsilon_1+\epsilon_2)(t-t_0) \\
&\leq (\epsilon_1+\epsilon_2)\frac{e^{k(t-t_0)}-1}{k}
\end{aligned}
$$

We have now all the tools to state the famous Cauchy-Lipschitz theorem.

Theorem 3:

Let be $U \subset \mathbb{R}\times\mathbb{R}^n$ and open subset, and $f : U \to \mathbb{R}^n$ a local Lipschitz continuous function. Let be $K \equiv [t_0-T, t_0+T]\times B[x_0, r] \subset U$ a safe compact cylinder on which the function f is k-Lipschitz. Then, there exists a unique C^1 function $u : [t_0-T, t_0+T] \to \mathbb{R}^n$ such that

$$\begin{aligned} u'(t) &= f(t,u(t)), \forall t \in [t_0 - T, t_0 + T] \\ u(t_0) &= x_0, (t_0, x_0) \in K \end{aligned}$$

u is called the local solution of the differential equation.

Demonstration 10.8.8:

First let us prove the existence. Let us take a series of EEM solution $u_k : [t_0 - T, t_0 + T] \to \mathbb{R}^n$ such that $||u'_k(t) - f(t, u_k(t))|| \leq \epsilon_p, \forall t \in [t_0 - T, t_0 + T]$, $\lim_{p\to\infty} \epsilon_p = 0$. Thanks to the proposition 10.8.6, the series is uniform Cauchy series on $[t_0 - T, t_0 + T]$. Since, $u_k([t_0 - T, t_0 + T) \in B[x_0, r]$ a compact domain and therefore a complete metric space, the series converge uniformly toward a function $u = \lim_{k\to\infty} u_k(t)$. Thanks to the proposition 10.8.5 the function $u(t) = \lim_{p\to\infty} u_p(t)$ is a solution of the differential equation. This proves the existence.

The uniqueness of the solution is a straightforward application of proposition 10.8.6. Let us assume two solutions $u_1, u_2 : [t_0 - T, t_0 + T] \to \mathbb{R}^n$ such that $u_1(t_0) = u_(t_2)$ then one has $||u_1(t) - u_2(t)|| \leq (\epsilon_1 + \epsilon_2)\frac{e^{k(t-t_0)}}{k} = 0$.

10.8.2.4 Uniqueness and existence of a global solution

Theorem 4:

Let be $U \subset \mathbb{R} \times \mathbb{R}^n$ an open subset and $f : U \to \mathbb{R}^n, (t, x) \to f(t, x)$ a local Lipschitz continuous function in x. Let be $u_1, u_2 : I \to \mathbb{R}^n$ two solutions of the differential equation $u'(t) = f(t, u(t))$ then if there exists $t \in I$ such that $u_1(t) = u_2(t)$ the solutions are equal everywhere on I.

Demonstration 10.8.9:

Let us assume that $u_1(t_0) = u_2(t_0)$ and that the solutions are not equal everywhere. Let us define $\tau = \inf\{t | u_1(t) \neq u_2(t), t \in [t_0, \sup I|\}$. Since $u_1(t) = u_2(t), \forall t \in [t_0, \tau[$ by continuity of the solutions one has $u_1(\tau) = u_2(\tau)$. Now, since U is an open subset, there exists a safe cylinder $K \equiv [\tau - T, \tau + T] \times B[u_1(\tau), r]$ such that the function is k-Lipschitz on it. The Cauchy-Lipschitz theorem tell

us that the solution u_1 and u_2 must be equal the safe cylinder which is therefore a contradiction.

corollary 2:

Let be $U \subset \mathbb{R} \times \mathbb{R}^n$ an open subset and $f : U \to \mathbb{R}^n, (t, x) \to f(t, x)$ a local Lipschitz continuous function in x. Then, there exits a unique maximal solution $u : I \to \mathbb{R}^n$ such that $u'(t) = f(t, u(t))$ and $u(t_0) = x_0$.

10.8.3 The Floquet Theorem

In this section, we focus on a special family of ODE: A spacial linear and time periodic EDO. Let be $A : \mathbb{R} \to \mathbb{R}^{n \times n}$ a continuous and periodic application, and let be $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$f(t, x) = A(t)x$$

Let us consider the following EDO $u'(t) = f(t, u(t)) = A(t)u(t)$. Let be $u_1, u_2 : I \to \mathbb{R}$ two solutions of the EDO. Thanks to the spacial linearity of the equation is is straightforward to observe that a linear combination of these two solutions $u = \alpha_1 u_1 + \alpha_2 u_2, \alpha_1, \alpha_2 \in \mathbb{R}$ is still a solution of the equation

$$u'(t) = \alpha_1 u_1'(t) + \alpha_2 u_2'(t) = \alpha_1 f(t, u_1(t)) + \alpha_2 f(t, u_2(t)) = f(t, \alpha_1 u_1(t) + \alpha_2 u_2(t))$$

Now, since the function A is continuous and of period T, it is bounded. Indeed, the maximum is defined by

$$M \equiv \max t' \in [t, t + T] ||A(t')||$$

where the norm is a matrix norm. M is independent of t by periodicity of A i.e $||A(t)|| \leq M, \forall t \in \mathbb{R}$. If the norm is the operator norm, one has $||A(t)x|| \leq M||x||, \forall x \in \mathbb{R}^n$. If means that the function f is a Lipschitz function in the spacial coordinate. This implies that for any $T \in \mathbb{R}^+$ there exists a safe cylinder $K = [t_0 - T, t_0 + T] \times B[x_0, r]$. Indeed, a sufficient condition to have a safe cylinder is $T \leq \frac{r}{M}$. Therefore, for a chosen T, an appropriate radius r is chosen to satisfied this condition. This implies directly that there exists a global a unique solution by the Cauchy-Lipschitz theorem.

Theorem 5:

Let be $A : \mathbb{R} \to \mathbb{R}^{n\times n}$ a continuous and periodic function of period T, and let be $f : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^n$ a function defined by $f(t,x) = A(t)x$. Let be $(t_0, x_0) \in \mathbb{R} \times \mathbb{R}^n$ an initial condition then the ODE defined by

$$u'(t) = f(t, u(t))$$

has a unique solution $u : \mathbb{R} \to \mathbb{R}^n$ passing by the initial condition $u(t_0) = x_0$.

Let be $\{x_1, ..., x_n\}$ a basis of $\mathbb{R}^n$. For any basis vector, there exists a unique solution $u_i : \mathbb{R} \to \mathbb{R}^n$ passing by (t_0, x_i). Let be a given initial condition (t_0, x_0). By definition of the basis, there is a unique decomposition of x_0 in the basis $\{x_1, ..., x_n\}$, $x_0 = \sum_{i=1}^n \alpha_i x_i, \alpha_i \in \mathbb{R}$. As stated at the beginning of this section, any linear combination of some solutions is still a solution. Therefore, the solution for any given initial condition is given by $u = \sum_{i=1}^n \alpha_i u_i$. Moreover, the solution u_i never intersect each other by uniqueness. In other words, the vectors $u_i(t)$ are linearly independent at any time.

Definition 10.8.5:

Let be $\{u_1, ..., u_n\}$ such that each function is a solution of the ODE and at a given time t, the vectors $\{u_1(t), ..., u_n(t)\}$ are linearly independent. Such a set of functions is called a fundamental system of solutions.

Proposition 10.8.7:

Let be $\{u_1, ..., u_n\}$ a fundamental system of equations, then the vectors $\{u_1(t), ..., u_n(t)\}$ are linearly independent at any time $t \in \mathbb{R}$.

Demonstration 10.8.10:

This a direct consequence of the uniqueness of the solution. By definition, at a given time t_0, the vector $\{u_1(t_0), ..., u_n(t_0)\}$, i.e each

solution has a different initial condition. By uniqueness of the solution the proposition is demonstrated.

Proposition 10.8.8:

Let be $\{u_1, ..., u_n\}$ a system of solution of the basis $\{x_1, ..., x_n\}$ and for the initial time t_0. The set of functions given by $\{\sum_{j=1}^{n} T_{1j}u_j, ..., \sum_{j=1}^{n} T_{nj}u_j\}$ with $T \in \mathbb{R}^{n\times n}$ a non-singular matrix is still a fundamental system of equation.

Demonstration 10.8.11:

This is straightforward by linearity of the ODE. Since T is non-singular, the functions remain linearly independent and correspond respectively to a new basis vector.

Proposition 10.8.9:

Let be $\{u_1, ..., u_n\}$ the solutions of the basis vector $\{x_1, ..., x_n\}$ at the initial time t_0, then the solutions $\{u_1(t+T), ..., u_n(t+T)\}$ are also a set of fundamtental solutions of the ODE.

Demonstration 10.8.12:

This is again a straightforward demonstration. Let us inject the function $u_i(t+T)$ in the ODE and verify it is a solution

$$u_i'(t+T) = A(t+T)u_i(t+T) = A(t)u_i(t+T)$$

The initial conditions for this new sets of solutions are given by $\{u_1(t_0+T), ..., u_n(t_0+T)\}$.

Proposition 10.8.10:

Let be $\{u_1, ..., u_n\}$ and $\{v_1, ..., v_n\}$ two sets of fundamental solutions, then there is a non-singular matrix $T \in \mathbb{R}^{n\times n}$ such that $u_i = \sum_{j=1}^{n} A_{ij}u_j$.

Demonstration 10.8.13:

Let us chose a time $t_0 \in \mathbb{R}$, and let us observe that the two fundamental systems provide respectively two basis of $\mathbb{R}^n$ at t_0 namely $\{u_1(t_0), ..., u_n(t_0)\}$ and $\{v_1(t_0), ..., v_n(t_0)\}$. Therefore, there exists a non-singular matrix $T_0 \in \mathbb{R}^{n\times n}$ such that $u_1(t_0) = \sum_{i=1}^{n} T_{ij} v_j(t_0)$. By prop. 10.8.8, one can define a fundamental system by $\{w_1 = \sum_{i=1}^{n} T_{0,1j} v_j, ..., \sum_{i=1}^{n} T_{0,nj} v_j\}$. This fundamental system has the same initial conditions as $\{u_1, ..., u_n\}$ and therefore by uniqueness they must be equal.

Theorem 6:

Let be $\{u_1, ..., u_n\}$ a fundamental system, there exists a non-singular matrix $X \in \mathbb{R}^{n\ timesn}$ such that $\{v_1 = \sum_{i=1}^{n} u_j X_{j1}, ..., \sum_{i=1}^{n} u_j X_{jn}\}$ with $\{v_1 = u_1(t+T), ..., v_n = u_n(t+T)\}$.

Demonstration 10.8.14:

This a direct consequence of prop. 10.8.9 and prop. 10.8.10.

This important to strike that this theorem is the heart of the Floquet theory. Indeed, it allows to highlight a periodic term in the fundamental solution as shown in the following theorem. Before going further let us recall a result about the exponential of a Matrix that can be found in [78] and in [79].

Theorem 7:

Let C be a real square matrix. Then there exists a real solution X to the equation $C = e^X$ if and only if C is non-singular.

The theory relative to the exponential of a linear application is recalled in appendix in subsection 10.2.1.

Theorem 8:

Let be $A : \mathbb{R} \to \mathbb{R}^{n\times n}$ a continuous and periodic function, and let us consider the differential equation $u'(t) = A(t)u(t)$, then any fundamental solution $\{u_1, ..., u_n\}$ can be expressed as follows

$$U(t) = P(t)e^{-Rt}$$

where $U(t)$ is the matrix where a column i correspond to a vector u_i, $R \in \mathbb{R}^{n\times n}$ and $P : \mathbb{R} \to \mathbb{R}^{n\times n}$ a periodic function.

Demonstration 10.8.15:

From theorem 6, one has

$$U(t+T) = U(t)X$$

where $X \in \mathbb{R}^{n\times n}$ is a non-singular matrix. From theorem 7 there exists a matrix R such that $X = e^{RT}$. Let us define $P(t) \equiv U(t)e^{Rt}$ and let us show that P is periodic.

$$\begin{aligned} P(t+T) &= U(t+T)e^{R(t+T)} \\ &= U(t+T)e^{RT}e^{Rt} \\ &= U(t+T)Xe^{Rt} \\ &= U(t)e^{Rt} \\ &= P(t) \end{aligned}$$

Therefore, one ends up with $X(t) = P(t)e^{-Rt}$ where P is periodic.

There is a similar theorem considering the eigenvalues of the matrix X.

Theorem 9:

Let be $A : \mathbb{R} \to \mathbb{R}^{n\times n}$ a continuous and periodic function, and let us consider the differential equation $u'(t) = A(t)u(t)$. Let be a fundamental system of solution $\{u_1, ..., u_n\}$ and let be the non-singular matrix $X \in \mathbb{R}^n$ such that $\{v_1 = \sum_{i=1}^n u_j X_{j1}, ..., \sum_{i=1}^n u_j X_{jn}\}$ with $\{v_1 = u_1(t+T), ..., v_n = u_n(t+T)\}$ as given by theorem 6. Let be $e \in \mathbb{R}^n$ an eigenvector of X with a corresponding eigenvalue $\lambda \in \mathbb{R}$

then there exists a solution of the ODE given by

$$x(t) = p(t)e^{-\mu t}$$

where $p : \mathbb{R} \to \mathbb{R}^n$ is a periodic function of period T and $e^{\mu} = \lambda$ and $x(t) = U(t)e$.

Demonstration 10.8.16:

Let us define the solution $x(t) \equiv U(t)e$. Because of the relation $U(t+T) = U(t)X$, one has $x(t+T) = U(t+T)e = U(t)Xe = \lambda U(t)e = \lambda x(t)$. Let us define $p(t) \equiv e^{\mu t}x(t)$ where $e^{\mu T} = \lambda$ and let show that p is a periodic function of period T.

$$\begin{aligned} p(t+T) &= e^{\mu(t+T)}x(t+T) \\ &= e^{\mu t}e^{\mu T}x(t+T) \\ &= e^{\mu t}\lambda x(t+T) \\ &= e^{\mu t}x(t) \\ &= p(t) \end{aligned}$$

corollary 3:

Under the hypotheses of theorem 9, let $e_1, ..., e_k$ be a series of eigenvectors of X with $k \leq n$ and let $\lambda_1, ..., \lambda_k$ be their respective eigenvalues such that $\lambda_1 \neq ... \neq \lambda_k$ then there exists a set of linearly independent solutions given by

$$x_i(t) = p_i(t)e^{-\mu_i t}$$

where $p_i : \mathbb{R} \to \mathbb{R}^n$ are a periodic functions of period T and $e^{\mu_i} = \lambda_i$ and $x_i(t) \equiv U(t)e_1$.

Demonstration 10.8.17:

The only thing to show here is the independence of the solutions. This a direct application of the uniqueness. Since each eigenvalue

is linearly independent of the others (because they have different eigenvalues), each eigenvalue constitutes a linearly independent initial condition from the other.

Let us try to characterize the eigenvalues. Let us start by recalling a well known theorem.

Theorem 10:

Let be $X \in \mathbb{C}^{n\times n}$ a complex matrix, and let be $\lambda_1, ..., \lambda_k \in \mathbb{C}$ the eigenvalues with their respective multiplicity $r_1, ..., r_k \in \mathbb{N}$ then

$$\begin{aligned} det(X) &= \Pi_{i=1}^{k} \lambda_i^{r_i} \\ trace(X) &= \sum_{i=1}^{k} r_i \lambda_i \end{aligned}$$

Demonstration 10.8.18:

Let be the polynomial $p(\lambda) \equiv det(X - \lambda I)$. Since the field is the complex number, the polynomial can be expressed as follow $p(\lambda) = \Pi_{i=1}^{k}(\lambda_i - \lambda)^{r_i}$. Therefore, one has

$$p(0) = det(X) = \pi_{i=1}^{k} \lambda_i^{r_i}$$

Now, let us investigate the coefficient of λ^{n-1} in the polynomial. From the polynomial splitting one has $(-1)^{n-1}(r_1\lambda_1 + ... + r_k\lambda_k)$. Now, regarding the determinant, only the terms on the diagonal produce a power of λ^{n-1}. Indeed, the determinant can be expressed as follows $det(X - \lambda I) = \sum_{\sigma\in P_n} \epsilon(\sigma) \sum_{i=1}^{n} (X - \lambda I)_{\sigma(i),i}$ where P_n is the set of permutation of dimension n and ϵ is the signature of the permutation. Clearly, to make appear a coefficient of λ^{n-1} at least $n-1$ terms of the permutation must satisfy $\sigma(i) = i$. Therefore, the only permutation that can satisfy this is the identity permutation $\sigma(i) = i, \forall i$. The coefficient is therefore $(-1)^{n-1}(X_{11} + ... + X_{nn})$. Identifying the coefficients one has

$$trace(X) = r_1\lambda_1 + ... + r_k\lambda_k$$

Proposition 10.8.11:

Let be $A : \mathbb{R} \to \mathbb{R}^{n\times n}$ a continuous and periodic function, and let us consider the differential equation $u'(t) = A(t)u(t)$. Let be a fundamental system of solution $\{u_1, ..., u_n\}$ and its matrix form $U : \mathbb{R} \to \mathbb{R}^{n\times n}$ where the column i of U is given by a vector u_i. The Wronskian is defined by $W(t) \equiv det(U(t))$. Then, one has

$$W(t) = W(t_0)e^{\int_{t_0}^{t} trace(A(t'))dt'}, \forall t_0 \in \mathbb{R}$$

Demonstration 10.8.19:

The demonstration can be found in [65].

corollary 4:

Let be $A : \mathbb{R} \to \mathbb{R}^{n\times n}$ a continuous and periodic function, and let us consider the differential equation $u'(t) = A(t)u(t)$. Let be a fundamental system of solution $\{u_1, ..., u_n\}$ and let be the non-singular matrix $X \in \mathbb{R}^n$ such that $\{v_1 = \sum_{i=1}^{n} u_j X_{j1}, ..., \sum_{i=1}^{n} u_j X_{jn}\}$ with $\{v_1 = u_1(t + T), ..., v_n = u_n(t + T)\}$ as given by theorem 6. Then, one has

$$det(X) = e^{\int_{t}^{t+T} trace(A(t'))dt'}$$

Demonstration 10.8.20:

Since $U(t + T) = U(t)X$ one has $det(X) = det(U(t + T))det(U^{-1}(t))$. Now, by prop. 10.8.11, one has $det(U(t + T)) = det(U(t))e^{\int_{t}^{t+T} trace(A(t'))dt'}$. Since $det(X(t)) = \frac{1}{det(X^{-1}(t))}$ the demonstration is complete.

The last result shows that the characteristic multipliers and exponents are an intrinsic property of the differential equation and therefore does not depend on the fundamental system of equation chosen.

Proposition 10.8.12:

The characteristic multipliers and exponents does not depend on the choice of the fundamental system

Demonstration 10.8.21:

Let us consider two fundamental systems on their matrix form U_1 and U_2. One has by theo. 6 that there exists $X_1, X_2 \in \mathbb{R}^{n\times n}$ such that

$$\begin{aligned} U_1(t+T) &= U_1(t)X_1 \\ U_2(t+T) &= U_2(t)X_2 \end{aligned}$$

Now, since U_1 and U_2 are two fundamental systems one has from prop. 10.8.10 there exists $C \in \mathbb{R}^{n\times n}$ non-singular such that $U_2(t) = U_1(t)C$. Therefore, one has

$$\begin{aligned} X_1 &= U_1^{-1}(t+T)U_1(t) \\ &= CU_2^{-1}(t+T)U_2(t)C^{-1} \\ &= CX_2C^{-1} \end{aligned}$$

Since C is non-singular the eigenvalues of X_1 and X_2 are equals.

10.8.4 Hill Equation

The Hill equation is probably one of the most important equation in the accelerator community.

Definition 10.8.6:

Let be $f : \mathbb{R} \to \mathbb{R}$ a continuous and periodic function, then the Hill equation is defined by

$$u''(t) + f(t)u(t) = 0$$

with $u : \mathbb{R} \to \mathbb{R}$ a C^2 function.

Let us first rewrite the equation as a first order differential equation. Let be $v : \mathbb{R} \to \mathbb{R}$ a C^1 function defined by $v(t) \equiv u'(t)$, then the second order differential equation can be written as follows

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = A \begin{bmatrix} u \\ v \end{bmatrix}, \text{ with } A(t) \equiv \begin{bmatrix} 0 & 1 \\ -f(t) & 0 \end{bmatrix}$$

Let us apply the Floquet theory to investigate the condition of stability of such an equation. Let us take the two following initial conditions

$$u_{0,1} \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$u_{0,2} \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

at time $t_0 = 0$. The two unique solutions respectively of $u_{0,1}$ and $u_{0,2}$ are denoted respectively s_1 and s_2 and they constitute a fundamental system. The matrix form of the fundamental system is written $U(t) \equiv [s_1(t)s_2(t)]$. The matrix $X \in \mathbb{R}^{2\times 2}$ is defined by

$$X \equiv U(t+T)U^{-1}(t) = U(t+T)$$

The eigenvalues of X are denoted by λ_1 and λ_2. It is important to keep in mind that they might be equal. From the former subsection one knows that

$$\begin{aligned} \lambda_1 + \lambda_2 &= trace(U(T)) = u_1(T) + v_2(T) \\ \lambda_1\lambda_2 &= det(X) = e^{\int_0^T 0dt'} = 1 \end{aligned}$$

Let us define $\phi \equiv \frac{u_1(T)+v_2(T)}{2}$ then

$$\begin{aligned}
&\lambda_1 = 2\phi - \lambda_2 \\
&\Rightarrow \lambda_2(2\phi - \lambda_2) = 1 \\
&\Leftrightarrow \lambda_2^2 - 2\lambda\phi + 1 = 0
\end{aligned}$$

Therefore one has

$$\lambda = \frac{2\phi \pm \sqrt{4\phi^2 - 4}}{2} = \phi \pm \sqrt{\phi^2 - 1}$$

10.8.4.1 Case $\phi \in]-1, 1[$

This case correspond to $trace(U(T)) < 2$ and the absolute values of the eigenvalues are equal to one

$$|\lambda| = \sqrt{\phi^2 + 1 - \phi^2} = 1$$

The trigonometric complex decomposition gives

$$\lambda = |\lambda|(\cos(\alpha) + i\sin(\alpha)) = \cos(\alpha) + i\sin(\alpha)$$

for a given $\alpha \in \mathbb{R}$. The decomposition is obviously not unique. This implies that the two characteristic exponents are purely complex. From the Floquet Theory, the general solution is given by

$$x(t) = c_1 e^{i\mu t} p_1(t) + c_2 e^{-i\mu t} p_2(t)$$

with $p_1(t)$ and $p_2(t)$ two periodic functions. Since $A(t)$ is real, one has

$$(x'(t))^* = (A(t)x(t))^* = A^*(t)x^*(t) = A(t)x^*(t)$$

for any solution $x(t)$. Therefore, if $x_1(t) \equiv p_1(t)e^{i\mu t}$ is the solution corresponding to the first eigenvalue, the solution corresponding to the second eigenvalue must be $x_2(t) \equiv p_1^*(t)e^{-i\mu t}$. The general solution is given by

$$x(t) = c_1 e^{i\mu t} p(t) + c_2 e^{-i\mu t} p^*(t)$$

with $p(t)$ a periodic solution. Therefore the solution is **stable**! Using the trigonometric complex decomposition, one can rewrite $p(t) = |p(t)|e^{i\phi(t)}$ with $\phi : \mathbb{R} \to \mathbb{R}$ a function as regular as p. The two solutions becomes

$$\begin{aligned} x_1(t) &= |p(t)|e^{i\gamma(t)} \\ x_2(t) &= |p(t)|e^{-i\gamma(t)} \end{aligned}$$

with $\gamma(t) \equiv \mu t + \phi(t)$. Since the solution is real, one has

$$\begin{aligned} x(t) &= |p(t)|Real(c_1 e^{i\gamma(t)} + c_2 e^{-i\gamma(t)}) \\ &= |p(t)|(c_{1,r}cos(\gamma(t)) - c_{1,i}sin(\gamma(t) + c_{2,r}cos(\gamma(t)) + c_{2,i}\sin(\gamma(t))) \\ &= |p(t)|((c_{1,r} + c_{2,r})cos(\gamma(t)) + (c_{2,i} - c_{1,i})\sin(\gamma(t))) \\ &= |p(t)|(Acos(\gamma(t)) + B\sin(\gamma(t))) \end{aligned}$$

where $A \equiv c_{1,r} + c_{2,r}$ and $B \equiv c_{2,i} - c_{1,i}$.
Now, let us define

$$\begin{aligned} \cos(\alpha) &\equiv \frac{A}{\sqrt{|A|^2 + |B|^2}} \\ \sin(\alpha) &\equiv \frac{B}{\sqrt{|A|^2 + |B|^2}} \end{aligned}$$

The solution becomes

$$\begin{aligned} x(t) &= \sqrt{|A|^2 + |B|^2}|p(t)|(cos(\alpha)cos(\gamma(t)) + \sin(\alpha)\sin(\gamma(t))) \\ &= \sqrt{|A|^2 + |B|^2}|p(t)|cos(\gamma(t) - \alpha) \\ &= \sqrt{\epsilon}\sqrt{\beta(t)}cos(\gamma(t) - \alpha) \end{aligned}$$

where $\beta(t) \equiv |p(t)|^2$ and $\epsilon = |A|^2 + |B|^2$. This equation is called the betatron function. It is very well known in the accelerator community as it is a very important equation. Let us investigate the relation between $\beta(t)$ and $\gamma(t)$. For that, let us inject the solution $x_1(t)$ in the differential equation

$$\begin{aligned} x_1'(t) &= |p(t)|'e^{i\gamma(t)} + |p(t)|i\gamma'(t)e^{i\gamma(t)} \\ x_1''(t) &= |p(t)|''e^{i\gamma(t)} + |p(t)|'i\gamma'(t)e^{i\gamma(t)} + |p(t)|'i\gamma'(t)e^{i\gamma(t)} + |p(t)|i\gamma''(t)e^{i\gamma(t)} - |p(t)|(\gamma'(t \\ &= (|p(t)|'' + i(2|p(t)|'\gamma'(t) + |p(t)|\gamma''(t)) - |p(t)|(\gamma'(t))^2)e^{i\gamma(t)} \\ &= f(t)|p(t)|e^{i\gamma(t)} \\ &\Rightarrow (|p(t)|'' - |p(t)|(\gamma'(t))^2 - f(t)|p(t)| + i(2|p(t)|'\gamma'(t) + |p(t)|\gamma''(t)))e^{i\gamma(t)} = 0 \\ &\Leftrightarrow |p(t)|'' - |p(t)|(\gamma'(t))^2 - f(t)|p(t)| + i(2|p(t)|'\gamma'(t) + |p(t)|\gamma''(t)) = 0 \end{aligned}$$

Separating the real part and the imaginary part one gets

$$\left\{ \begin{array}{rcl} |p(t)|'' - |p(t)|(\gamma'(t))^2 - f(t)|p(t)| & = & 0 \\ 2|p(t)|'\gamma'(t) + |p(t)|\gamma''(t) & = & 0 \end{array} \right. \tag{10.65}$$

The last equation can be rewritten as

$$2|p(t)|'\gamma'(t) + |p(t)|\gamma''(t) = (|p(t)|^2\gamma(t)')' = 0$$

This implies

$$|p(t)|^2\gamma'(t) = K$$

where $k \in \mathbb{R}$ a constant. This equation can be rewritten

$$\gamma(t) = \gamma(t_0) + \int_{t_0}^{t} \frac{K}{|p(t')|^2} dt' = \gamma(t_0) + \int_{t_0}^{t} \frac{K}{\beta(t')} dt'$$

One can chose $p(t)$ such that $K = 1$ without lost of generality. This later relation is also fundamental in the accelerator community.

10.8.4.2 Other cases

The other cases lead to an unstable solution at the exception of the case $\phi = 1$ that may lead to a periodic solution. Therefore, the stability condition used in the accelerator community is $trace(U(T)) < 2$.

10.8.5 Twiss parameters

Let us investigate the betatron solution in the phase space.

$$\begin{array}{rcl} x(t) & = & \sqrt{\epsilon}\sqrt{\beta(t)}cos(\gamma(t) - \alpha) \\ x'(t) & = & \sqrt{\epsilon}(\frac{1}{2}\frac{\beta'(t)}{\sqrt{\beta(t)}}cos(\gamma(t) - \alpha) - \sqrt{\beta(t)}\gamma'(t)\sin(\gamma(t) - \alpha)) \\ & = & \sqrt{\epsilon}(\frac{1}{2}\frac{\beta'(t)}{\sqrt{\beta(t)}}cos(\gamma(t) - \alpha) - \frac{1}{\sqrt{\beta(t)}}\sin(\gamma(t) - \alpha)) \\ & = & \sqrt{\epsilon}\frac{1}{\sqrt{\beta(t)}}(\frac{1}{2}\beta'(t)cos(\gamma(t) - \alpha) - \sin(\gamma(t) - \alpha)) \end{array}$$

Let us get rid of the cosine and sine

$$\begin{aligned}
\cos(\gamma(t)-\alpha) &= \frac{x(t)}{\sqrt{\epsilon}\sqrt{\beta(t)}} \\
\sin(\gamma(t)-\alpha) &= \sqrt{1-\cos^2(\gamma(t)-\alpha)} = \sqrt{1-\frac{x^2(t)}{\epsilon\beta(t)}} = \frac{\sqrt{\epsilon\beta(t)-x^2(t)}}{\sqrt{\epsilon\beta(t)}}
\end{aligned}$$

$$\begin{aligned}
x'(t) &= \frac{1}{\beta(t)}(\frac{1}{2}\beta'(t)x(t) - \sqrt{\epsilon\beta(t)-x^2(t)}) \\
\epsilon\beta(t)-x^2(t) &= (x'(t)\beta(t) - \frac{1}{2}\beta'(t)x(t))^2 \\
\epsilon\beta(t)-x^2(t) &= (x'(t))^2\beta^2(t) - \beta(t)\beta'(t)x'(t)x(t) + \frac{1}{4}(\beta'(t))^2x^2(t) \\
\epsilon &= (x'(t))^2\beta(t) - \beta'(t)x'(t)x(t) + \frac{(\frac{1}{4}(\beta'(t))^2+1)}{\beta(t)}x^2(t)
\end{aligned}$$

Let us define

$$\begin{aligned}
a(t) &\equiv -\frac{1}{2}\beta'(t) \\
b(t) &\equiv \frac{1+a^2(t)}{\beta(t)}
\end{aligned}$$

The equation can be rewritten as follows

$$\epsilon = (x'(t))^2\beta(t) + 2a(t)x'(t)x(t) + b(t)x^2(t)$$

This is the equation of an elipse. Since $a(t), b(t), \beta(t)$ are periodic, this equation means that after a period T, the particles end up in this elipse. The elipse orientation changes with times but is periodic of period T. The three parameters above are called the **Twiss parameters**.

10.8.6 Mathieu's differential equation

Let us focus now on a particular case of Hill equation. Let us consider the following differential equation

$$u''(t) + (\delta + 2\epsilon\sin(2t))u(t) = 0 \tag{10.66}$$

We will use this equation to validate the beam dynamics solver. From the Floquet Theory, one knows that a stable solution is of the form

$$\begin{aligned} x_1(t) &= e^{i\mu t}p(t) \\ x_2(t) &= e^{-i\mu t}p^*(t) \end{aligned}$$

The goal is to determine the stability region of the differential equation. The stability condition only depends on the two parameters ϵ and δ. It has been shown by O.Haupt [28] that the points satisfying $trace(U(T)) = 2$ fill out curves in a δ, ϵ-plane separating it into regions in which (δ, ϵ) give stable or unstable solutions of the Mathieu's differential equation. There exists several methods to obtain the stability region namely, the Fourier method, the perturbation method, the direct method (Meissner Equation) or numerical approaches. The Fourier method is explained in [80]. The idea is to inject the periodic function corresponding to $trace(U(T)) = 2$ and find the condition on ϵ and δ such that this solution can exist. Since the solution is periodic a Fourier transform can be used to express it. Using a finite number of terms as an approximation, the values of δ and ϵ can be calculated.

The stability regions for the Mathieu's equation are presented in Fig. 10.5

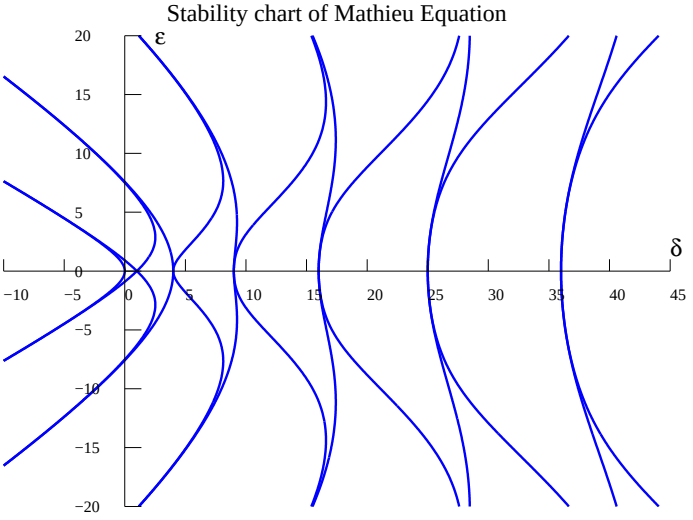


Figure 10.5: Stability chart of Mathieu's equation.

Bibliography

[1] The official web site of the myrrha project. Available on the following web site: http://myrrha.sckcen.be/.

[2] The official web site of the sck-cen. Available on the following web site: http://www.sckcen.be/fr/SCK-CEN.

[3] A. Fokau. Accelerator-driven system: Source efficiency and reactivity determination. Royal Institute of Technology, Stockholm, 2010.

[4] J.C. Evans, E.L. Lepel, R.W. Sanders, C.L. Wilkerson, W. Silker, C.W. Thomas, K.H. Abel, and D.R. Robertson. Long-lived activation products in reactor materials. Technical Report NUREG/CR - 3474 PNL - 4824, Division of Engineering Technology Office of Nuclear Regulatory Research, U.S Nuclear Regulatory Commission Washington D.C, August 1984.

[5] Anttila M., A. Crégut, M.T. Cross, Z. Dlouhy, J. Elkert, R. Fedorowicz, M. Genova, G. Imbard, M. Klein, M. Laraia, C. Le Goaller, F Madrid, D.W Reisenweaver, C. Sancho Llerandi, V.V. Shidlovskii, Y. Sivintsev, R.I Smith, and L. Valencia. Radiological characterization of shut down nuclear reactors for decommissioning purposes. Technical Report No. 389, International Atomic Energy Agency, Vienna, 1998.

[6] Stanley Humphries. *Principles of Charged Particle Acceleration.* Wiley-Interscience, 1986.

[7] D. Vandeplassche, J.-L. Biarrotte, H. Klein, and H. Podlech. The myrrha linear accelerator. In *2rd International Particle Accelerator Conference (IPAC)*, September 2011.

[8] John J. Barnard and Steven M. Lund. Uspas 2011: Beam physics with intense space charge. New York, Spring Session 2011. Lecture notes.

[9] D. A. Edwards and M. J. Syphers. *An Introduction to the Physics of High Energy Accelerators.* Wiley-VCH, Boschstrasse 12, 69469 Weinheim, Germany, 1992.

[10] Martin Reiser. *Theroy and Design of Charged Particle Beams.* Wiley, June 2008.

[11] T. P. Wangler. *RF Linear accelerators.* Wiley, June 2007.

[12] W. Paul and H. Steinwedel. Notizen: Ein neues Massenspektrometer ohne Magnetfeld. *Zeitschrift Naturforschung Teil A*, 8:448–450, July 1953.

[13] W. C. Gibson. *The Method of Moments.* Taylor & Francis Group, 2008.

[14] A. Stratton Julius. *Electromagnetic theory.* Mcgraw Hill Book Company, 1941.

[15] J. D. Jackson. *Électrodynamique classique.* John Wiley & Sons, 1999.

[16] J-C. Nédélec. *Acoustic and Electromagnetic Equations*, volume 144. Spinger, Applied Mathematical Sciences, 2000.

[17] C. Craeye and D. González-Ovejero. A review on array mutual coupling analysis. *Radio Science*, 46:RS2012, April 2010.

[18] G. Vecchi. Loop-star decomposition of basis functions in the discretization of the efie. *Antennas and Propagation, IEEE Transactions on*, 47(2):339–346, Feb 1999.

[19] F.P. Andriulli. Loop-star and loop-tree decompositions: Analysis and efficient algorithms. *Antennas and Propagation, IEEE Transactions on*, 60(5):2347–2356, 2012.

[20] F.P. Andriulli, K. Cools, I. Bogaert, and E. Michielssen. On a well-conditioned electric field integral operator for multiply connected geometries. *Antennas and Propagation, IEEE Transactions on*, 61(4):2077–2087, April 2013.

[21] D.R. Wilton and A. W. Glisson. On improving the electric and field integral equations at low frequencies. In *URSI Radio Sci. Meet. Dig.*, Los Angeles, CA, June 1981.

[22] J.R. Mautz and R. Harrington. An e-field solution for a conducting surface small or comparable to the wavelength. *Antennas and Propagation, IEEE Transactions on*, 32(4):330–339, 1984.

[23] J.-M. Song, C. C. Lu, and W. C. Chew. Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *IEEE Transaction Antennas and Propagation*, 45(10):1488–1493, 1997.

[24] A. A. Vlasov. The vibrational properties of an electron gas. *Soviet Physics Uspekhi*, 10(6), 1968.

[25] S. Humphries. *Principles of charged particle acceleration.* Wiley-interscience publications. J. Wiley, 1986.

[26] J. Lafontaine. *Introduction aux varietes differentielles.* EDP Sciences, 2010.

[27] L. Haine. Elément de géometrie différentielle lmat2110. Louvain-la-Neuve, CRC, 2013. Lecture notes.

[28] O Haupt. Über lineare homogene differentialgleichungen zweiter ordnung mit periodischen koeffizienten. *Math Ann*, Bd. 79, 1919.

[29] J. J. Barnard and Steven M. Lund. Beam physics with intense space charge. New York, 2011. Lecture notes.

[30] M. Berger and B. Gostiaux. *Géométrie différentielle: variétés, courbes et surfaces.* Mathématiques (PUF). Presses universitaires de France, 1987.

[31] R. Coifman, V. Rokhlin, and S. Wandzura. The fast multipole method. *IEEE Antennas and Propagation Magazine*, 35(3):7–12, 1993.

[32] J. Song and W. C. Chew. Interpolation of translation matrix in mlfma. *Microwave and Optical Technology Letters*, 30:109–114, 2001.

[33] B. K. Alpert and V. Rokhlin. A fast algorithm for the evaluation of legendre expansions. *SIAM Journal on Scientific and Statistical Computing*, 12(1), January 1991.

[34] Eric Darve and Pascal Havé. Efficient fast multipole method for low-frequency scattering. *J. Comput. Phys.*, 197(1):341–363, June 2004.

[35] V. Rokhlin. Diagonal forms of translation operators for the helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, 1:82–93, 1993.

[36] Ozgur Ergul and Bariscan Karaosmanoglu. Approximate stable diagonalization of the green's function for low frequencies. *IEEE Antennas and Wireless Propagation Letters*, 13:1054 – 1056, 2014.

[37] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[38] P. Van Dooren and P. A. Absil. Analyse numérique. Université Catholique de Louvain, Ecole Polytechnique de Louvain, 2010. Lecture notes.

[39] Y. Saad and M. H. Schultz. Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7, July 1986.

[40] Gérard Meurant and Jurjen Duintjer Tebbens. The role eigenvalues play in forming gmres residual norms with non-normal matrices. *Numerical Algorithms*, 68(1):143–165, 2015.

[41] Yousef Saad. *Iterative methods for sparse linear systems (second edition)*. PWS, 2004.

[42] Mark Embree. How descriptive are gmres convergence bounds? 1999.

[43] Jun-Sheng Zhao and Weng Cho Chew. Integral equation solution of maxwell's equations from zero frequency to microwave frequencies. *Antennas and Propagation, IEEE Transactions on*, 48(10):1635–1645, 2000.

[44] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37:123–146, 2010.

[45] A. Napov and Y. Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM J. Sci. Comput*, 34:A1079–A1109, 2012.

[46] Y. Notay. Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM J. Sci. Comput*, 34:A2288–A2316, 2012.

[47] Hongwei Cheng, William Y. Crutchfield, Zydrunas Gimbutas, Leslie F. Greengard, J. Frank Ethridge, Jingfang Huang, Vladimir Rokhlin, Norman Yarvin, and Junsheng Zhao. A wideband fast multipole method for the helmholtz equation in three dimensions. *J. Comput. Phys.*, 216(1):300–325, July 2006.

[48] Min Hyung Cho and Wei Cai. A wideband fast multipole method for the two-dimensional complex helmholtz equation. *Computer Physics Communications*, 181(12):2086 – 2090, 2010.

[49] S. Kapur and J. Zhao. A fast method of moments solver for efficient parameter extraction of mcms. In *In Design Automation Conference*, pages 141–146, 1997.

[50] Mario Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[51] J. M. Tamayo, A. Heldring, and J. M. Rius. Multilevel adaptive cross approximation (mlaca). *IEEE Transactions on Antennas and Propagation*, 59(12):4600–4608, Dec 2011.

[52] Neil Muller, Lourenço Magaia, and B. M. Herbst. Singular value decomposition, eigenfaces, and 3d reconstructions. *SIAM Review*, 46(3):518–545, 2004.

[53] C. Raucy, E. de Lera Acedo, C. Craeye, D. Gonzalez-Ovejero, and N.R. Ghods. Experimental validation of fast simulation methods in the framework of the ska telescope project. In *Antennas and Propagation (EUCAP), 2012 6th European Conference on*, pages 2225–2229, 2012.

[54] D. Gonzalez-Ovejero and C. Craeye. Fast numerical characterization of non-uniform arrays. In *Antennas and Propagation, 2009. EuCAP 2009. 3rd European Conference on*, pages 2107–2110, 2009.

[55] L. Matekovits, V.A. Laza, and G. Vecchi. Analysis of large complex structures with the synthetic-functions approach. *Antennas and Propagation, IEEE Transactions on*, 55(9):2509–2521, 2007.

[56] E. Lucente, A. Monorchio, and R. Mittra. An iteration-free mom approach based on excitation independent characteristic basis functions for solving large multiscale electromagnetic scattering problems. *Antennas and Propagation, IEEE Transactions on*, 56(4):999–1007, 2008.

[57] D. Gonzalez-Ovejero and C. Craeye. Interpolatory macro basis functions analysis of non-periodic arrays. *Antennas and Propagation, IEEE Transactions on*, 59(8):3117–3122, 2011.

[58] R. Maaskant, R. Mittra, and A. Tijhuis. Fast analysis of large antenna arrays using the characteristic basis function method and the adaptive cross approximation algorithm. *Antennas and Propagation, IEEE Transactions on*, 56(11):3440–3451, 2008.

[59] Dave Shreiner, Graham Sellers, John M. Kessenich, and Bill M. Licea-Kane. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley Professional, 8th edition, 2013.

[60] M. Scarpino. *OpenCL in Action: How to Accelerate Graphics and Computation*. Manning, 2012.

[61] David F. Rogers. *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[62] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88*, pages 26–33, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[63] NVIDIA Corporation. Nvidia official website. Available on the following link: www.nvidia.com.

[64] Jason Gregory. *Game Engine Architecture, Second Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2014.

[65] J-P. Demailly. *Analyse numérique et équations différentielles*. edp sciences, 17, avenue du Hoggar Parc d'Activité de Courtaboeuf - BP 112 91944 Les Ulis Cedex A - France, Avril 2016.

[66] M. E. THOMADAKIS. *The Architecture of the Nehalem Processor and Nehalem-EP SMP Platforms*. PhD thesis, TEXAS A&M UNIVERSITY, March 2011.

[67] Mark S. Papamarcos and Janak H. Patel. A low-overhead coherence solution for multiprocessors with private cache memories. *SIGARCH Comput. Archit. News*, 12(3):348–354, January 1984.

[68] Norman P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. *SIGARCH Comput. Archit. News*, 18(2SI):364–373, May 1990.

[69] T. Harada and L. Howes. Introduction to gpu radix sort. *Advanced Micro Devices, Inc*, 2011.

[70] Lars V. Ahlfors. *Complex analysis*. McGraw-Hill Book Co., New York, third edition, 1978. An introduction to the theory of analytic functions of one complex variable, International Series in Pure and Applied Mathematics.

[71] Microway. Detailed specifications of the intel xeon e5-2600v3. Available on the following web site: https://www.microway.com/knowledge-center-articles/detailed-specifications-intel-xeon-e5-2600v3-haswell-ep-processors/.

[72] TechPowerUp. Nvidia kadro k620. Available on the following web site: https://www.techpowerup.com/gpudb/2600/quadro-k620.

[73] E. de Lera Acedo, N.R. Ghods, P. Scott, P. Doherty, K. Grainge, A. Faulkner, P. Alexander, D. Gonzalez-Ovejero, C. Raucy, C. Craeye, N. Drought, N. Troop, P. Van der Merwe, and H.C. Reader. Ska aa-low front-end developments (at cambridge university). In *Antennas and Propagation (EUCAP), 2012 6th European Conference on*, pages 616–620, March 2012.

[74] C. Raucy, E. de Lera Acedo, N. Razavi-Ghods, and C. Craeye. Characterization of ska-aalow antenna elements in the array environment. In *Electromagnetics in Advanced Applications (ICEAA), 2012 International Conference on*, pages 514–517, Sept 2012.

[75] C. Raucy, Craeye C., and Vandeplassche D. Simulation of a radio frequency quadrupole with the method of moments. In *Antennas and Propagation (EUCAP), 2014 8th European Conference on*, April 2014.

[76] C. Raucy, Craeye C., and Vandeplassche D. Rfq solver based on the method of moments. In *5th International Particle Accelerator Conference (IPAC)*, June 2014.

[77] C. Raucy, F. P. Andriulli, and C. Craeye. Stabilization of the modelling of a radio-frequency quadrupole based on quasi-helmholtz projectors. In *Electromagnetics in Advanced Applications (ICEAA), 2015 International Conference on*, pages 1441–1444, Sept 2015.

[78] Earl A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, Asaf Ali Road, New Delhi 110 002, 1955.

[79] Earl A. Coddington and R. Carlson. *Linear Ordinary Differential Equations*. SIAM - Society for Industrial and Applied Mathematics, Philadelphia, 1951.

[80] G.W. Hill. On the part of the motion of the lunar perigee which is a function of the mean motions of the sun and moon. *Acta Math.*, 1-36, 1886.