UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia
"Galileo Galilei"

Corso di Laurea Magistrale in Fisica

Exploring the application of the parareal algorithm to edge simulations using the SOLPS-code

Supervisors:
Dr. Debasmita Samaddar
Dr. Emilio Martines

Laureando:
Michele Marin

Anno Accademico: 2015/2016

Abstract

One of the most challenging obstacles to economically viable magnetically confined fusion power is how to deal with hot plasma when it inevitably interacts with solid surfaces. The physics in this regime is particularly complex due to the presence of neutrals. Numerical simulations require large wallclock times and are often incapable of incorporating all the desired physics due to computational restrictions.

The aim of this placement project is to explore the applicability of a novel technique – the Parareal Algorithm – to the SOLPS code in edge physics simulations. The SOLPS code is widely used to study plasma wall interactions and has also been used to design the ITER wall. The Parareal Algorithm parallelizes the time domain – which is counter-intuitive as it apparently violates causality.

However, it is a predictor-corrector approach, and if applied correctly, can greatly reduce computation time. This makes complex simulations of the plasma edge in ITER like machines feasible.

Sommario

Uno degli ostacoli più impegnativi da superare per generare elettricità in modo economicamente sostenibiletramite tramite la fusione nucleare a confinamento magnetico è il capire come trattare il plasma caldo quando inevitabilmente interagisce con le superfici solide. La fisica in tale regime è particolarmente complessa a causa della presenza degli atomi neutri. Le simulazioni numeriche hanno bisogno di tempi lunghi e non sono spesso in grado di incorporare tutta la fisica desiderata a causa di restrizioni computazionali.

L'obiettivo di questo progetto è quello di esplorare l'applicabilità di una nuova tecnica, - Il Parareal - al codice SOLPS per la simulazione di plasmi al bordo. Il codice SOLPS è largamente usato per studiare le interazioni tra il plasma e le pareti ed è stato anche utilizzato per progettare le pareti di ITER. Il Parareal Algorithm parallellizza il dominio temporale, metodo contro-intuitivo e che sembra violare la casualità.

Esso utilizza tuttavia un approccio predittore-correttore e, se applicato correttamente, può ridurre sensibilmente il tempo di calcolo. Questo rende fattibili le complesse simulazioni al bordo del plasma di macchine come ITER.

Contents

Chapter 1

Introduction

In this thesis the parareal, an algorithm that parallelize time in order to speed up codes, will be implemented and tested on SOLPS, a code designed to simulate the plasma on the edge of a nuclear fusion reactor.

In recent years the fight to climate change has gained increasing importance. The development of renewables proceeds steadily, but the current rate of growth is not fast enough. A new energy source is needed and the combined effort of 35 nations is focusing on a promising alternative, fusion energy.

An experimental fusion reactor, ITER, is currently being built in the South of France and will start working in 2025. It will contain a plasma that will be confined and heated to a temperature high enough for thermonuclear reactions to take place. The confinement cannot be perfect, so the heat fluxes going out from the reactor must be dealt with. This is done by shaping the magnetic fields to make the plasma interact with the walls in a prescribed area. The divertor configuration, as it is called, it's crucial to make fusion economically feasible.

ITER will be a unique machine, so the only way to predict the behaviour of its plasma, in order to design the divertor, is thorough numerical simulations. It does not exist a single code capable to model the whole plasma, because the range of parameters is too wide, but the interest in this thesis is only in the Scrape Off Layers, where the interaction with the physical structure take place. The most complete code capable to model the plasma in that region is SOLPS, whose last version is SOLPS-ITER.

The code is composed by a fluid model, B2.5, and a kinetic model, Eirene. The fluid model is fast and reliable in simulating charged particles, but it is not fit to follow neutrals trajectories. This is the reason why it was coupled with a more computationally expensive Montecarlo kinetic code, which can capture the complex processes that neutrals are subjected to in the divertor area. SOLPS is very slow, about one step per minute, and ITER pulses will require a longer time compared to previous experimental reactors, so a tool to speed it up would be useful.

The algorithm studied in this thesis is the parareal, which uses an anti-intuitive

parallelization in time. The idea is to guess the evolution of the system with a faster, rougher model and then iteratively correct it until convergence is reached. The "coarse" solution has to capture some of the physics, but not all of it. Find a model that is as fast and similar as possible to the original "fine" one is the challenge that has to be overcome. The parareal was previously applied to SOLPS-5, a former version of the code, for DIII-D and MAST cases. The optimization of the algorithm and the scaling with the number of cores were studied and gains of the order of ten were observed.

After a first part of training on both SOLPS and the parareal, a case based on the ASDEX tokamak, placed in Garching, was studied for SOLPS-5. The same case with a reduced grid and bigger time steps was taken as a coarse solution. Both fine and coarse used the stand - alone version of the code. The gain and the scaling with the number of processors were observed again, proving the adaptability of the parareal. Then the algorithm was successfully implemented on SOLPS-ITER. The same ASDEX case was studied, showing gains similar to the ones obtained with SOLPS-5. It was tried to change the correction script, to check if there was some improvement in the performance.

The last case considered was based on ITER and it was by far the most complex one. Different combinations of fine and coarse were explored and a calibration protocol was developed in the coupled (fine) - stand-alone (coarse) to enable the user to choose the best fast solver. No gain was obtained, but steps were taken in the research of a suitable coarse solution. To speed up this work and similar ones that will be carried around in the future, a series of tools were developed to automatically run the parareal on new cases.

In chapter one the need for new forms of energy will be presented, together with more details on the ITER experiment. Information about plasma physics and edge physics in particular will be useful to understand the code. SOLPS will be analysed in all its parts in chapter two, with a focus on the fluid and the kinetic difference in the treatment of neutrals. Chapter three will be a description of the parareal algorithm, including a list of the coarse models used in this work. The last chapter will begin with the training done to learn to use the various tools. It will then range over all the cases tested, specifying for each one the strategies adopted.

More work will be needed to run the ITER case with parareal on SOLPS-ITER, which is the most fusion relevant. Suggestions will be presented on how to continue this work towards that objective.

Chapter 2

Background

2.1 Social background

2.1.1 Energy problem

Available energy

Energy has always been crucial to mankind, we need it to do basically everything, from producing food to any recreational activity. Being so important, energy is one of the limiting factors to the growth of our population. In other words, if the amount of energy available were to drop, the population would drop as well. Right now in the world there are around 7460 million people, rising at a rate of 80 million for year. Forecasts (figure 2.1, [1]) show how this growth is not about to stop in the next decades.

Before 2025, the world population is predicted to increase by 6.5 percent and reach 8 billions. The energy demand is supposed to increase accordingly, but this will not be the case. A citizen of the United States consumes about 12 times more than a citizen of a developing country, such as India. The energy consumption pre-capita of those countries will surely rise, changing that 6.5 percent to a much higher number.

According to [2] world energy consumption will change from 298 (2013) to 344.6 (2020) million barrels per day, a growth around 15 percent. Oil, coal and natural gas are predicted to still provide 75 percent of that energy by 2040.

In the "other renewables" are included solar and wind power, that are here predicted to multiply seven times. Even if they were to expand a lot faster, e.g. twenty times, they wouldn't still be able to significantly reduce the fossil fuel demand. There are two main problems in continuing to rely on fossil fuels. The first one is that none of them is limitless, and in 25 years we will have used up roughly half of the world reserves of oil and gas. Coal will last just a little bit more.

According to OPEC, the oil world reserve amount to 1492 billion barrels, which at this rate won't last much more than 50 years. Fossil fuels will reach a peak of

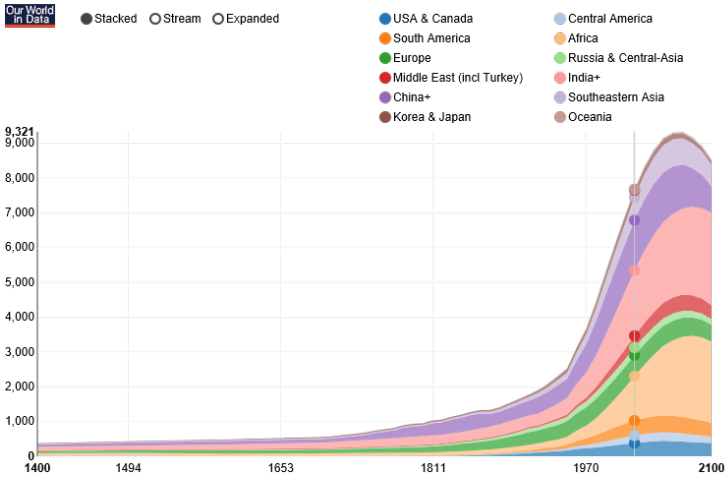

Figure 2.1: Population growth forecasts and history, millions of people

Energy source / year	2013	2020	2030	2040
Oil	84.4	90.1	96.1	100.6
Coal	76.1	84.2	92.4	98.3
Gas	59.2	69.1	87.7	111.5
Nuclear	13.1	13.9	17.5	23.5
Hydro	6.3	7.4	8.9	10.2
Biomass	26.2	29.1	33.6	38.1
Other renewables	2.4	4.3	8.4	17.4
Total	267.6	298.0	344.6	399.4

Table 2.1: World energy consumption and forecasts divided by energy source. All the data are in million barrels per day

production in ten or fifteen years and will then start to slow down. By then it will be necessary that the renewable energy sources can balance the loss due to the end of the carbon fossil era (fig. 2.2).

Those projections show a substantial flattening of the energy growth around 2030-2035, not due to a decline in the world economy or energy consumption but to a smarter management. To give an example, the number of cars is predicted to rise close to 800 million vehicles by 2040, but the global energy demand for transportation will increase about only 340 percent. The reason is that cars are becoming more and more fuel efficient, to the point that it will be possible to travel 45 miles per gallon in comparison with the current 25. Houses will be more efficient too, and urbanization will let more people use electricity sources cheaper than the biomass which rural population are still using in developing countries. Even taking all of this

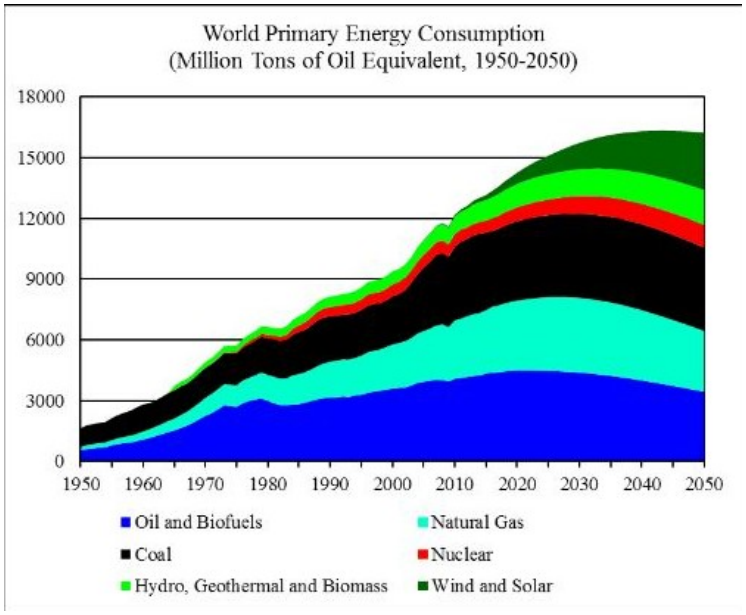

Figure 2.2: Sources: World historical oil, natural gas, and coal consumption from 1950 to 1964 is assumed to be the same as production; world primary energy consumption and its composition from 1965 to 2015 is from BP (2016)

into account, however, it is clear than renewables will have to keep increasing until they can cover the whole demand. In the following the various alternatives will be analysed.

Hydroelectric energy is still a growing industry, but the growth is not very fast and is expected to reach saturation in around 25 years. Moreover, there is an environmental impact that has to be taken to account. Attempts are being made to use the energy of the oceans and the geothermal heat, but those technologies are still quite expensive and can be exploited only in small areas. Recent developments seem to promise that they will become competitive in the future, but they are not sectors in expansion.

Nuclear fission energy is already widely used, but is not cheap and brings considerable risks. Incidents such as the ones in Chernobyl and in Fukushima made the development and the constructions of nuclear fission facilities much slower and some countries decided not to use this energy source. What's more, nuclear fission produces waste that can last thousands or tens of thousands years. A new set of reactors, Generation III, are safer, more efficient and cheaper, but this is not enough to greatly expand this sector. The most promising alternatives to fossil fuels are solar and wind energy. Wind is cheap and readily available, plentiful and widely redistributed. It is predicted to nearly triple its potential in the next ten years and big investments are being done to build new wind farms. The cost of solar panels, on the other hand, became more than 70 times smaller in the past few years, making it competitive. The solar industry too is growing exponentially, enormous solar farms

are being built and solar panels are being installed on the roofs of a growing number of houses.

Still, wind and solar are respectively two and three times more expensive than natural gas, and they provide not much more than 1 percent of the global demand. Even an exponential growth will take more than 50 years to become significant. Even if they were to expand faster, there is another problem with them: high variability. If there is no wind or no sun the implants won't work, but cities need energy full time. Storing electricity is very challenging, and high grid penetration is not an easy task. It is important to note that solar and wind energy tend to be complementary, cause high pressure areas tend to bring clear skies and low wind and vice-versa. Combined implants are therefore quite promising.

Climate change

Climate change is another reason why humanity should reduce its energy dependence on non-renewable sources as soon as possible. "Climate change" is referred to any change in the statistical properties of the climate system. It is important to underline that climate changes always happened in the history of the earth and are expected to happen again. There were times in which the levels of carbon dioxide were much higher than today and it was totally natural. The main reasons to climate change are

1. Modifications in the solar activity
2. Changes in the reflectivity of earth's atmosphere or surface
3. Greenhouse effect, that regulates how much heat is retained by the earth's atmosphere.

The reason why this is considered a problem is that the climate change that is happening now is the fastest in history and is most likely man-made. In the past, in fact, these phenomena were happening in geological timescales, taking tens of thousands of years. The time scale now is completely different. Only in the last ten years there was an increase in the medium levels of carbon dioxide by around 5 %. The threshold of 400ppm was broken in march 2015 and at the current rate in 20 more years the concentration will be 450ppm. [3] The world's emission of CO_2 will reach its peak in 2030 according to most predictions, and by then the concentration will already be around 440ppm. This value is found simply supposing a rough linear behaviour and is inferred from the official data from NASA (fig. 2.3) By that time, optimistically, the temperature difference will be around 1 $°C$. This value is again obtained in the same way, taking into account the data shown in figure 2.4. The world emission will then start to fall, but there is very little chance that

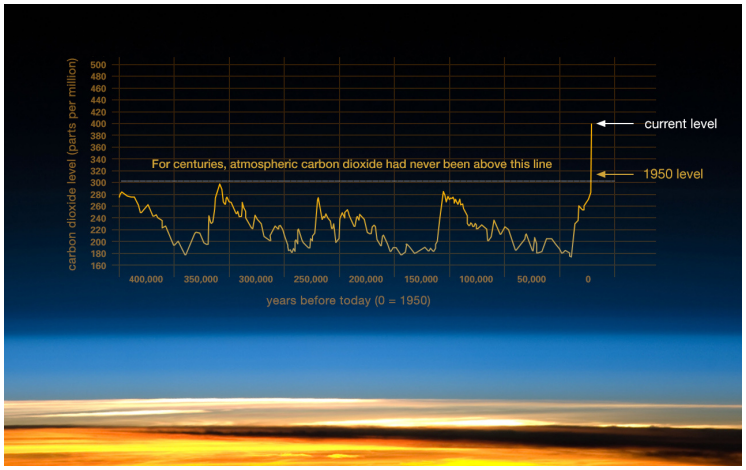

Figure 2.3: This graph, based on the comparison of atmospheric samples contained in ice cores and more recent direct measurements, provides evidence that atmospheric CO2 has increased since the Industrial Revolution. (Credit: Vostok ice core data/J.R. Petit et al.; NOAA Mauna Loa CO2 record.)

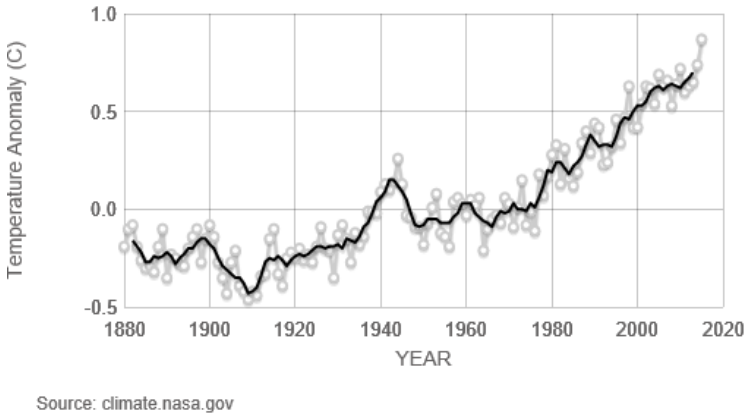

Figure 2.4: Average World's temperature in recent years. Data source: NASA's Goddard Institute for Space Studies (GISS). Credit: NASA/GISS

the concentration growth will slow down for at least ten more years, and by then the concentration will already be around 465ppm.

The first consequence of this fact is that the greenhouse effect is increasing, trapping more heat and so warming the planet. Almost every month in 2016 has been the hottest on record. The average temperature of the earth, in fact, has risen by 0.78 degrees in less than 40 years, and is still rising. An increase of 2 $^{\circ}C$ would be irreversible and catastrophic for the planet. Another big problem are the positive feedbacks that this process implies. The most important are the two that follow.

1. Ice being white, it reflects most of the light it receives, while the sea, being darker, tends to absorb much more heat. The increasing temperature is melting ice all over the world and, in particular, around the poles. The ice is replaced by sea water, reducing the amount of light and heat reflected back. Thus the earth takes more heat, melts more ice and so on.

2. While the concentration of CO2 in the atmosphere is rising, the oceans absorb part of it and become more acid. But there is a limit on how much CO2 they can take, and the more acid they become the less they can absorb. So the concentration in the atmosphere increases and so on.

The effects of climate change are numerous. It raises the rate of extreme weather conditions, like floods, droughts and heat waves, which can be dangerous to the world. The changes in patterns and amount of rainfall can harm the production of hydroelectric energy and decrease the quality and quantity of water supplies. The risk of desertification is growing higher in an increasing number of areas around the world. The rising of sea levels puts coastal communities and ecosystems at serious danger. Above $^{\circ}2C$ every ecosystem is in danger, migration and reproduction habits of many species are changing and will change even more. This sums up to the mass extinction we are already causing, with 20 % of the species likely to become extinct by as soon as 2028. Even without running out of fossil fuel, using all the reserves would be more than enough to break the $2^{\circ}C$ limit.

Humanity has put an enormous system out of equilibrium, and that system is our only home.

2.1.2 Nuclear fusion

The most cost-effective solutions to the emission problem are to improve fuel-efficiency and to substitute coal with natural gas, but those solutions would only mitigate it. The renewables that are now at our disposal are not enough for entirely replacing fossil fuels and, even if they were, are not cheap enough to do it in time. From all said above the need to think about new forms of energy is very clear. A large worldwide project is going on in order to make mankind able to use the very same energy source of our sun, nuclear fusion. Fusion would be a virtually

inexhaustible, relatively safe, continuous and competitive energy source. Nuclear fusion happens every time that two atoms overcome the Coulomb barrier and fuse together, losing a little amount of mass in the process. This mass is given back in the form of kinetic energy, the amount depending on the atoms involved. Fusion is a natural process in the universe and all the stars in the main sequence burn hydrogen to exist. While inside the sun the conditions favourable to fusion are achieved by strong gravity, to obtain such conditions on earth is really challenging. Density and gravity fields on earth are much lower, so the temperature has to be higher. There are two main ways to do it.

The first is inertial confinement, that consists basically in heating up a small pellet of fuel usually containing a little amount of deuterium and tritium to a very high temperature. Usually, a laser beam is split up and then amplified trillions of times. The target is compressed 20 times the density of water and heated up to 100 millions degrees, shock waves travel into the centre of the pellet compressing it even further and starting the fusion process. In a few milligrams pellet there is the same amount of energy that can be obtained burning a barrel of oil. The main problem of inertial confinement regards the implosion, that has to be perfectly symmetric. To achieve that, the sphericity of the target has to be high, with aberrations no more than few micrometers. Moreover, the beams have to be of equal power and of the same shape, or instabilities are likely to develop. Fast progress are being made in this field, but a commercial fusion reactor of this kind is still quite far from being possible.

The second alternative is magnetic confinement and is currently the most promising one. The idea is to control and confine the fuel in a designated area using strong magnetic fields. There is more than just one way to do that, and the most studied ones at present time are the tokamak, the stellarator and the reversed field pinch.

1. A tokamak is the acronym for the Russian "toroidal chamber with magnetic coils". As the name suggests, a tokamak is a toroidal shaped chamber in which the fuel is heated up and confined. To overcome the Coulomb force the temperature has to be millions of degrees, so the fuel inside is in the fourth state of matter, plasma.

   Plasma is basically ionized matter and will be described in more details later. What is important now is that ions and electrons tend to follow the magnetic field lines while being pushed by the Lorentz force. So, a strong magnetic field can keep the plasma inside the chamber long enough to be heated and to start nuclear fusion. The point in which the reaction become self-sustaining is called "ignition point" and is what the international community is trying to achieve.

   There are a lot of problems in doing so. First, perfect confinement is impossible to achieve, so at some point the plasma will inevitably interact with the walls. To enable the control with magnetic fields the plasma density has

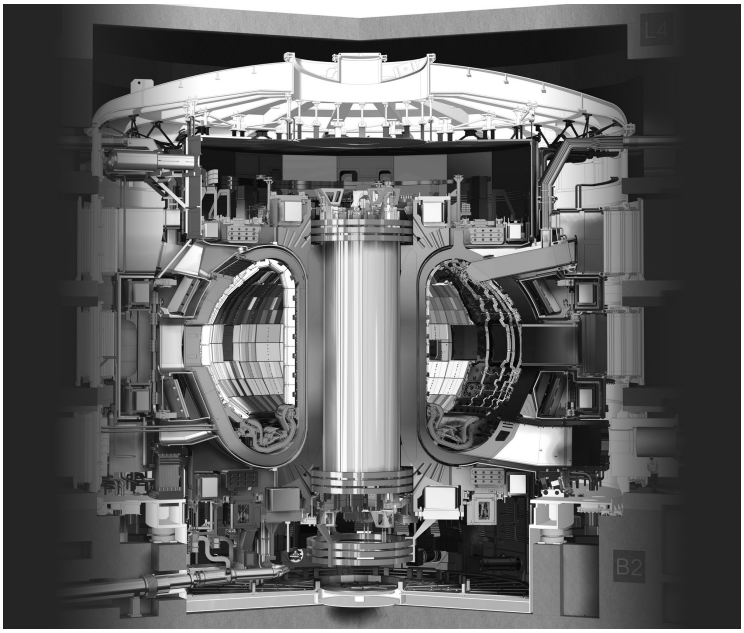

Figure 2.5: ITER Tokamak, official ITER website

to be quite low, on the order of at least a millionth times the density of air, but lower densities means higher temperatures. In order to reach ignition the temperature of the core, the inner part of the plasma, has to be risen to 150 million degrees. On the edge it goes down to thousands of degrees, but this is still challenging for whatever material has to face such a heat flux.

Moreover, Tokamaks suffers instability issues that can lead to disruptions, in which huge amount of heat fluxes reach the walls, thereby damaging it. So measures must be taken to avoid and mitigate such disruptions, that in a real fusion reactor would seriously damage and melt the walls. Heating the plasma to reach ignition inside the tokamak is also technologically challenging. There are three main ways to heat the plasma, summarized below.

(a) **Ohmic heating**. This is a widely used and well known process and it follows the simple principle that an object is heated when a current flows through it. The current is created by electromagnetic induction where the fluctuating the magnetic field generates a flow of current. Those collisions between ions and electrons create a resistance that result in heat. However, the resistance of the plasma goes down when the temperature rises, so this kind of heating process works efficiently only till a certain threshold.

(b) **Neutral beam injection**. The principle, again, is very simple. Charged particles are accelerated to a high energy and injected to the plasma where, through collisions, they transfer energy to the plasma particles.

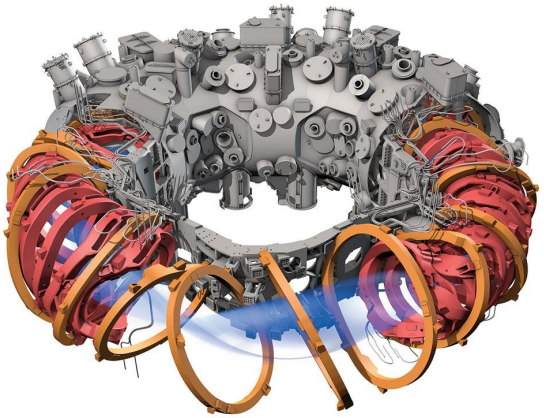

Figure 2.6: concept of a stellarator

However, charged particles cannot be injected directly, because the strong magnetic fields would deviate them to the walls. This is the reason why the beam has to be neutral, and the particles are made to go through a "beam neutralizer".

(c) **Radio-frequency heating**. Electromagnetic waves resonate with ion and electrons, speeding them up and raising the temperature. This is a really powerful method and can provide tens of MW of power.

There are even more challenges to overcome. A bigger machine implies a better confinement, so the tokamak needs to be a megastructure, with all the construction problems and the costs that follow. The powerful magnetic fields are generated with huge coils, but traditional conductors are not able to support the high currents needed, so superconductors have to be used. This means that the coils must be kept at cryogenic temperature, around 4K, at a short distance from a 150 million degrees plasma. Considering the size, this means that a huge cryostat is needed. Moreover, the neutrons coming out of the chamber due to the fusion reaction have to be collected, so another giant structure has to surround the cryostat.

2. In a stellarator the geometry is not anymore a torus, but something more complicated. The basic idea is to balance the drifts making the particles move into different magnetic fields as they travel into the confinement area. Using both geometry and magnetic fields for confinement there is no need for the additional current that is present in the tokamaks, with all the instabilities that come with it. Stellarators are even more complicated than tokamaks, so this yet promising technology is still quite far from being able to become commercial.

3. The third configuration is the reversed field pinch. In this kind of device a

unique magnetic configuration is used to sustain currents and confinement similar to the ones in the tokamaks with a lower magnetic field. The main disadvantage is that this system is more susceptible to turbulence. The biggest RFP at the moment is RFX in Padua.

Between the three, the closest to be economically feasible is the first one. Due to all the technological problems that are described above, nuclear fusion is an extremely challenging objective and research has been conducted for over 60 years. We are still not currently able to produce energy from nuclear fusion, but considerable improvement has been made. The record in fusion energy production is now taken by Jet, in Culham, with 14 MW of fusion power out of 24 MW of power input.

Previous forecasts about nuclear fusion availability revealed themselves to be wrong, so it's said that nuclear fusion is always 50 years in the future. Now, however, we have for the first time a schedule about the construction of commercial power plants. The next step will be ITER.

2.2 Scientific background

2.2.1 ITER

China, the European Union, India, Japan, Korea, Russia and the United States are working together to build what will be the largest tokamak in the world, ITER, "the way" in Latin. ITER will be located in south France and will be the first device to produce net energy, although it will not put any current in the grid. With an input power of 50 MW it should be able to produce 500MW of output. ITER has been designed to reach five main tasks. [4]

1. Produce at least 500 MW of power. This will demonstrate that the future, bigger power plants, the DEMO devices that will actually generate electricity, are possible.

2. Demonstrate that all the various different technologies needed in a power plant can work together. ITER will be the most complex device in the world and will need technologies able to provide vacuum, heating control, plasma diagnostic, cryogenics, remote maintenance and more, all working in harmony at the same time.

3. Reach the ignition point. The plasma in ITER will be heated by the nuclear reactions within it. This should provide higher fusion energy and longer pulses.

4. Test tritium breeding. The tritium available is not enough for the commercial use of fusion power, but luckily there is a way to produce it inside the vacuum vessel. ITER should demonstrate that this method is efficient in producing enough tritium.

5. Demonstrate the safety characteristic of a fusion device. To sustain a reaction is for itself a task hard enough, so when it fails, the plasma just vanishes, melting the walls in the worst case scenario. Therefore, fusion is intrinsically safe, and ITER should be a proof of this. Concerning radioactive material, ITER will produce radioactive waste like every other nuclear power plant, but the difference will be that the decay time will be around 12 years. This means that in about 100 years it will all be back to normal, ready to be recycled. So, radioactive waste in fusion is relatively very low compared to fission.

The construction started in 2010 and the first plasma is now scheduled for December 2025. At the present time, the various parts of ITER are being crafted and transported in France. This 15 billion dollars device will be around 30 x 30 m and will break a number of world records. The machine will weigh 23000 tons, and will have a plasma chamber of $840m^3$. The superconducting strand that will be rolled to form the coils will be 100 thousands kilometers long and will store 51 Gj of magnetic energy. The central solenoid will sustain a plasma current of 15 MA that will generate magnetic fields as strong as 13 Tesla. A system capable to immobilize a rocket will be used to keep the machine on the ground in case of a major disruption. The $16000m^3$ vacuum chamber will be the largest stainless steel vacuum chamber in the world. Around that, a vast array of support systems will provide the right conditions to fusion. From water cooling supply to remote handling and the heating systems, all will have to work at the same time. Given that ITER will be an experimental machine, a large number of diagnostics will collect tens of terabytes of data every day.

The next step after ITER will be DEMO, a new set of machines that will actually be able to collect heat and to put electricity in the grid. A lot of different concepts are being developed for DEMO, like a stellarator DEMO, a liquid-divertor DEMO and so on. Then will begin the mass production, the costs for a central four times smaller than for DEMO.

A machine so big, and with such an high power, necessarily has a number of issues to solve. One of the most important problems is how to deal with the heat flux that will inevitably reach the walls. Perfect confinement is, in fact, impossible to achieve, and a tokamak needs to have a system able to deal with the escaping particles. What the latest tokamaks do, and ITER will do as well, is to use geometric and magnetic topologies to make the plasma interact with a particular region of the walls explicitly designed for this purpose. The name of this section is the "divertor" and on ITER it will be in the lower section of the torus. Before concentrating on the divertors, however, we will discuss a few more details about what a plasma is.

2.2.2 Definition of Plasma

Matter can exist in four different states: solid, liquid, gas and plasma. Most of the ordinary matter, in the universe, is in plasma state, and this is because plasma state covers an extremely wide range of parameters. The density can vary from a few particles for m^3 to the 10^{30} that can be found in the cores of stars, the temperature from a couple of Kelvin in the intergalactic medium to 10^8 in our laboratories on earth.

A plasma is basically an ionized gas, but there are three particular conditions that are needed for a gas to be a plasma.

1. The plasma approximation: Every charged particle has to interact with many other particles, so they can exibit a collective behaviour. Using Λ to indicate the average number of particles in the Debye sphere, in a plasma it must hold $\Lambda > 1$. The Debye sphere is defined as a sphere with a radius equal to the Debye length $\lambda = \sqrt{\frac{\varepsilon_0 KbT}{ne^2}}$.

2. Bulk interactions: The plasma has to be significantly bigger than the Debye spheres. This condition makes the bulk iterations more important than the edge iterations and allow the plasma to be quasi-neutral. If any violation of the quasi-neutrality arise, in fact, strong electromagnetic fields would soon restore it.

3. Plasma frequency: The electron plasma frequency (measuring plasma oscillations of the electrons) is large compared to the electron-neutral collision frequency (measuring frequency of collisions between electrons and neutral particles). When this condition is valid, electrostatic interactions dominate over the processes of ordinary gas kinetics.

In a plasma, as stated above, there are many particles inside the Debye sphere, and this means that a lot of particles can interact simultaneously. A plasma exhibit therefore a collective behaviour and so is a "complex system", which is very challenging to describe from an analytical point of view.

A complex system is one that exhibits some peculiar characteristics, like feedback loops, that can be both damping and amplifying, some degree of spontaneous order, numerosity. Less rigorously, it is a system whose properties cannot be inferred by the properties of its parts. Examples are ants, climate, economies, nervous system and many more. Due to the extreme variety of such systems, science of complex systems needs to be an inherently interdisciplinary subject. The aim is to find some common behaviour or laws that can allow models and predictions to be developed. Sometimes a complex system is also chaotic, if it is sensitive to the initial conditions, topologically mixing and with dense periodic orbits. Normally, the analytical tools to deal with this kind of problem are non linear differential equations. Another

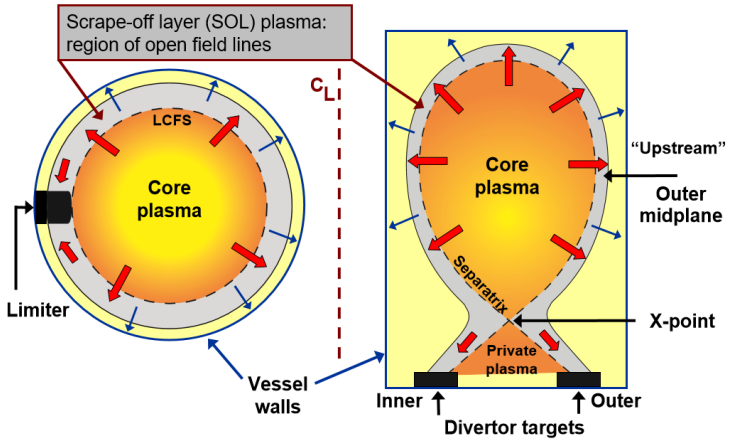

Figure 2.7: Schematic view of a divertor [8]

kind of complex systems are stochastic systems, where other tools are used, like Markovian processes, master equation, detailed balance and so on.

Plasma is a good example of a chaotic system with its strong sensitivity to initial conditions and its non linear properties. The only way to predict the behaviour of such a system is to simulate it with a computer, but those two last characteristics are major problems for a simulation. We will be interested in non-equilibrium dynamics, and in particular in how the plasma inside the tokamak relaxes to a steady state.

2.2.3 Divertor

Inside a tokamak there is a plasma with a temperature of tens of millions degrees that cannot be perfectly confined. This means that the high heat flux coming out from the core has to be dealt with. The first idea for preventing the plasma to reach the walls was to use a limiter. A limiter is a material that is put inside the plasma, near the wall, in such a way that the heat flux loses its power before being able to make contact with the actual wall. A magnetic surface that does not intersect any solid surface is said to be closed. The edge of the plasma defines the last close flux surface (LCFS), otherwise called "Separatrix", after which there is the region of the plasma commonly addressed as "Scrape Off Layers" (SOL) [24]. The SOL, or plasma edge, is usually small compared to the whole plasma, but it is crucial to understand its behaviour.

The limiter configuration, however, has some problems, first of all with the impurities. When the plasma reaches the limiter, in fact, the hot particles make the material evaporate and end up into the core. The main mechanism for the plasma to erode the walls is the sputtering. The particles can be ejected by colliding with energetic particles in the plasma, and this would be physical sputtering, or chemically interact with those particles, under the name of chemical sputtering. The fuel becomes more diluted and there is a risk of increased radiation and instabilities.

Moreover, most of the ionization for a limiter configuration happens far from the limiter, so there is not a local cool down system. The temperature is therefore high and the heat fluxes can easily melt the walls.

A new configuration with divertor was thus developed. This basically consists in changing the magnetic geometry to make the plasma interact with a particular material in a designated area, away from the core. A magnetic quadrupole field generates what is called an X-point, through which passes the "separatrix". The poloidal magnetic field is zero in the X-point. The separatrix is the equivalent of the LCFS for the limiter, the region near the core is called upstream and the region that face the walls is called downstream. Below the X-point there appears an area of cold and high density plasma, which is separated from the core. The separatrix hits the divertor in the strike point, where most of the heat exhausts.

A series of tools are used to make this interaction as favourable as possible. About 15*percent* of the total fusion power reaches the divertor. For ITER, this means around 15 MW, that will concentrate on the vertical target with a heat flux of 10 $\frac{MW}{m^2}$ during the steady state and 20 $\frac{MW}{m^2}$ during the slow transient. To minimize this the divertor is carefully designed, the flux is enlarged with dedicated coils and the angle between the flux and the divertor is made as large as possible.

Moreover, there was an intense research to choose the material for the divertor, and in the end Tungsten was chosen, with the highest melting point and reduced sputtering. The divertor is used both to sustain the heat flux and to purify the plasma from α particles, unburnt fuel and eroded particles. In a burning plasma the helium concentration is to be maintained below 15 %, so it is important to remove it fast enough. [5] A tokamak may have more than one X-point. In that case the geometry is addressed, for example with two X-points, as "double null" instead of "single null".

The ITER divertor will work in a semi-detached mode. When the density becomes high enough the electron conduction in the SOL becomes finite and leads to a temperature drop, that allows recombination and heat loss through radiation. In the end the density of charged particles reaching the divertor becomes negligible due to a "neutrals cushion" and the area of flux exhaustion broadens. The plasma in the divertor region becomes completely separated in what is the detached mode. This phenomenon is crucial because it allows a better pumping of the helium ash and decreases drastically the particle flux at the target plates.

In the normal operation mode the density decreases smoothly to the edge of plasma (L-mode). During the experiments on ASDEX, however, researchers discovered the H-mode, in which the gradient is much higher. The turbulent transport is affected by this barrier and the density and the temperature in the core increases. ITER will work in H-mode. Some basics of edge plasma physics will be introduced to better explain how these phenomena arise and what are the issues in trying to model it.

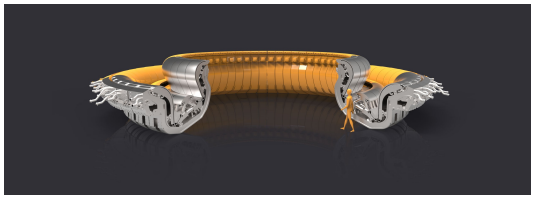

Figure 2.8: ITER divertor

2.2.4 Edge physics

The study of the SOL is very important to understand the physics in the tokamak, one of the reasons being that is the place where plasma-wall interactions happen [22]. As stated above, the hot particles from the plasma hit the solid surfaces and damage them. The point in fusion is to make it economically feasible, so it's imperative that the walls can handle the plasma and do not need to be replaced too often. The choice of the material is therefore extremely important. Beryllium was chosen for the ITER walls, due to its reasonable melting point and the relative harmlessness as an impurity in the core. The divertor, instead, is made in tungsten, because it has to bear a higher heat load and the divertor is sufficiently isolated and the impurities are not significantly introduced in the core.

When impurities reach the core there are a lot of unwanted effects: the plasma cools down, the fuel becomes more diluted, the kinetic pressure rises. Heavy atoms can be ionized a lot of times, so when they reach the core they start to radiate and this is followed by a power loss. Besides that, the divertor area have to be able to remove the fusion ash, helium, to avoid it to dilute the core. For the same purpose, the walls have to be low in tritium retention.

To model such a system several species have to be taken into account, along with their interactions, the presence of neutrals and the possibility of ionization. There are some considerations that can be easily done, but after that more complex models have to be used. A completely general feature in a wall-plasma interaction is the developing of a thin layer between the two, under the name of Debye sheath. Being lighter, the electrons reach the wall first, charging it and creating a potential difference. The ions are then accelerated towards the wall and that potential is screened in a length of the order of the Debye length.

Between the plasma and the Debye sheath there is yet another area, the presheath, separated from the sheath in the point in which the velocity of the ions become bigger than the ion sound speed $C_s = \sqrt{\frac{K_b T_e}{m_i}}$. The walls reach the floating potential, which is the one reached by a surface with no net current flow. It's usually calculated as

$$\Delta V = \frac{K_b T_e}{e} \frac{1}{2} ln(\frac{m_i}{2\pi m_e}) \tag{2.1}$$

The reason why the floating potential is so important is that it controls the heat and energy flux on the walls. To give an understanding of how complicated is the

SOL physics, a simple model will be presented. Some assumptions will be made, and then this model will be compared with a more realistic situation.

1. T_i and T_e uniform, while usually there are temperature gradients and finite conductivity.

2. Weak thermal equipartition, so electrons and ions do not strongly interact between each other. This is not always the case.

3. No ionization or recombination, that are instead present and sometimes important.

4. No friction with neutrals, that are usually quite hard to model.

5. No radiative cooling, while an important parameter that have to be taken in to account in a fusion reactor is the power loss via radiation.

Simply assuming the conservation of particles and a classical model for diffusion it follows that the plasma parameters drop with an exponential behaviour, the fastest being the energy flux and the slowest the temperature. Another important relation about ionization can be found considering that the flux of neutrals should be proportional to

$$n_n v_n = \sigma v_i n_n \lambda_{iz} (n + n_{LCFS})/2 \tag{2.2}$$

Where n_n is the neutral density, v_n the neutrals velocity, σ the collision section, v_i the ions velocity, λ_{iz} the characteristic lenght for ionization and n_{LCFS} the density at the last closed flux surface. This basically means that the flux of neutrals is equal to their ionization. The flux out of the LCFS can be calculated as

$$\Gamma_{out} = D_\perp \frac{n_{LCFS}}{\lambda_n} \tag{2.3}$$

Where $D_\perp$ is the perpendicular diffusion coefficient and λ_n is the characteristic length for the density gradient. With the assumption that the two characteristic length are equal (there is a correlation, just a rough estimation is done here) it can be found

$$n_{LCFS} = \lambda_n \frac{\sigma v_i}{2 v_n} n^2 \tag{2.4}$$

This relation links the density in the SOL with the density in the core and represents yet another proof of the importance of this region.

Changing to a complex SOL is therefore crucial to understand precisely what happens in the whole plasma. Qualitatively, the biggest approximation is the absence of collisions, and taking into account collisions has actually a favourable effect for the purpose of a fusion machines. The friction with the neutrals cools down the plasma, facilitates ionization and radiation and allows volume recombination. The temperature in the complex plasma model is therefore lower than the one found with

the simple one. Moreover, scattering, backscattering and absorption are needed to correctly explain the behaviour of impurities, that can be of main importance in the divertor area.

Analytical approaches cannot describe all these complex phenomena, so a numeric model is needed. There are, in general, two main ways to describe the plasma, depending on its parameters, and those are the kinetic and the fluid models. Both introduce approximations and should be applied carefully. They will now be presented.

Kinetic Model

The most basic model is the kinetic one, that treats the plasma as a collection of particles interacting through Coulomb forces and collisions. The traditional models for neutrals are simply extended, but this results in much more complex problems. Point particles will be considered, since quantum mechanical effects are mostly negligible in the SOL plasma. The distribution function is normalized to the density as

$$n^m(x,t) = \int d^3v f^m(x,v,t) \tag{2.5}$$

Microscopic fields are obtained from the Maxwell equations using charge density (from the density of the distribution function) and electric current (from the velocity distribution). Dealing with a big number of particles is hard, so it's usually better to work with the distribution function. Using the Liouville theorem the Klimontovich equation can be written

$$\frac{df^m}{dt} = \frac{\partial f^m}{\partial t} + \frac{d\mathbf{x}}{dt}\frac{\partial f^m}{\partial \mathbf{x}} + \frac{d\mathbf{v}}{dt}\frac{\partial f^m}{\partial \mathbf{v}} \tag{2.6}$$

where for plasma $\frac{d\mathbf{v}}{dt}$ can be written as

$$\frac{d\mathbf{v}}{dt} = \frac{q}{m}[\mathbf{E}^m(\mathbf{x},t) + \mathbf{v} \times \mathbf{B}^m(\mathbf{x},t)] \tag{2.7}$$

What is usually done is to take the average of those quantities and to linearize them. If the collision time is bigger enough that the time scale of the process considered the right term can be put to zero and the Vlasov equation is obtained. This equation is useful because the solutions are entropy conserving and are incompressible in the six dimensional phase space.

Another useful equation that is strictly related to the Vlasov one is the Fokker-Plank equation, i.e

$$(\frac{\partial}{\partial t} + v_{k_a}\nabla_a + \frac{e_a(\vec{E} + v_a \times \vec{B})}{m_a} \cdot \nabla_v) f_a(s, v_a, t) = \sum_{b\in(e,i)} \vec{C_{ab}} + \vec{S_a} \tag{2.8}$$

where $\vec{C_{ab}}$ summarize the collisions between species while $\vec{S_a}$ represent sources and sinks. This is more general, but makes the problem much more difficult.

The first simplification that will be very useful to the particular problem of modelling the plasma inside a tokamak is to ignore the toroidal direction, thanks to the fact that the problem is axisymmetric and that the transport parallel to the magnetic field is a lot faster than the transport across it. This turns the 3D problem in just 2D, bringing crucial simplification. Still, this will not be enough.

Kinetic models are accurate, but simulations based on them are very cpu expensive. For this reason sometime is useful to consider fluid models.

Fluid Model

The distribution function gives much more information than what is generally requested. Often, the interest is on knowing just a set of plasma parameters, like the temperature, the energy flow, the pressure and so on. All these variables are actually fluid moments. There are equations that can be found to calculate the mean value and the evolution of those moments.

Moments are defined as $M^n = \int \mathbf{v}_1\mathbf{v}_2...\mathbf{v}_n f d^3v$. They will therefore be tensors of rank n.

The zero order moment is just the density, then velocity, then pressure. By contracting this first three, taking the module of one or more velocities, more thermodynamic quantities are found, like

$$p = \int \frac{mv^2}{2} f d^3v \tag{2.9}$$

or

$$T = \frac{1}{n}\int \frac{m}{3}(\mathbf{v}-\mathbf{u})^2 f(\mathbf{x},\mathbf{v},t)d^3v = \frac{m}{3}\langle v'^2\rangle \tag{2.10}$$

From those variables equations can be written to model the plasma. The Braginskii equations [10] are between the most important in plasma edge physics. When they are not used, most of the times is just a variation to them. They read as

$$\frac{d(nv)}{dx} = S_p \tag{2.11}$$

$$\frac{d}{dx}[(m_i v^2 + 2kT)n] = -m_i(v_i - v_n)\nu_{in}n \tag{2.12}$$

$$\frac{d}{dx}((\frac{1}{2}m_i v^2 + 5kT)nv - q_{k_e}) = Q_r + Q_e \tag{2.13}$$

The first equation involves the zero fluid moment and needs the first ones to be solved, the second allows to calculate the first moments but require the thirds and so on. There is the need for closure, i.e. some set of equations that permits to

calculate higher order from lower order. Usually the closure express second order terms using the zero and the first order.

The expressions can be a little different depending on the approximations in use. There are some approximations that are needed to get a simple enough model. First of all, the distribution is considered to be the equilibrium Maxwellian distribution, that can be linearly perturbed obtaining $f = f^0 + f'$. Nonlinear terms in the expansion are therefore not retained, and perturbations from the Maxwellian equilibrium due to electrostatic fluctuations are not considered.

The cross field transport calculated in this model is not in good agreement with the experimental data for the SOL. This is due to the fact that collisions are not the main cause of it. The problem turns into a non linear and numerical simulations are needed, with time dependent "anomalous" transport coefficients. In this case, closure requires non-diagonal viscosity terms in the momentum flux tensor. Following this approach, there is a lot of physics that can be added or ignored, every complication bringing a finer model but with an increasing computational cost.

Here a simple and widely used model is presented as an example. Often, the quasi-neutrality allows to utilize a single mass-average velocity for the plasma, in what is called the single fluid model. From this model, after some algebra, some simplifications and neglecting small terms, MHD can be derived. Magnetohydrodynamic (MHD), is a continuum fluid model that treat the plasma as a fluid electrically conducting, collisional and magnetic.

The equations for MHD are

$$\frac{\partial(nM)}{\partial t} + \nabla \cdot (nM\mathbf{u}) = 0 (countinuity) \tag{2.14}$$

$$nM\frac{\partial(\mathbf{u})}{\partial t} = \mathbf{j} \times B + \nabla(p) (equationofmotion) \tag{2.15}$$

$$\mathbf{E} + \mathbf{u} \times \mathbf{B} = \frac{\mathbf{j}}{\sigma_{el}} (Ohm's law) \tag{2.16}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} (Faraday - Henry) \tag{2.17}$$

$$\nabla \times \mathbf{B} = -\mu_0 \mathbf{j} (Ampere) \tag{2.18}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.19}$$

And the closure is done supposing the fluid incompressible, isothermal, adiabatic and pressure-less. Another big simplification in MHD is the assumption of thermodynamic equilibrium. Thus, the approximation doesn't work well in a low collisional plasma. The collisionality depends strongly on the temperature

$$\tau_{coll} = \frac{T^{\frac{3}{2}}}{n_e ln\Lambda} \tag{2.20}$$

where $ln\Lambda$ is the Coulomb logarithm. This means that the fluid model is usually valid for dense and cold plasma, when the free path length ($\lambda = \tau * v$) is smaller than the other characteristic length of the system.

Given the extreme complexity of the SOL, the MHD is not enough to fully describe the behaviour of the plasma inside it. Not only the limitations of such a simple model do not hold for all the situations that are studied, but the interest is also upon different species, ionization, interaction with the walls and so on. Before exploring the code that was used for this thesis, heat fluxes will be briefly presented a little more in detail.

Heat fluxes

The particles tend to move along the field lines, but collisions and other effects make them drift radially, towards the walls. One of the most important parameters in tokamaks physics, and in this thesis, is the heat flux at the divertor plates. There are three main mechanisms for heat transport, conduction, convection and radiation. The fastest in the case that is considered is the convection, in which the heat is carried along with the moving particles. The diffusive heat flux is calculated by

$$q = n\frac{kT}{m\upsilon}\frac{dkT}{dx} = \chi\frac{dkT}{dx} \tag{2.21}$$

and it can be seen that χ depend on T as $T^{\frac{5}{2}}$. The heat conduction is far stronger for electrons than for ions. The biggest contribution to the heat flux at the divertor is the power transmitted by ions and electrons through the sheath that forms between the plasma and the divertor. A current is created in the Debye sheath due to the ions trying to re-establish quasi neutrality. When a magnetic field is also present the charged particles move along with it, with a speed of $c_s = k\frac{Z_iT_e+\gamma T_i}{m_i}$

The particle flux to the target is $\Gamma_{se} = n_{se}v_{se}$ with a consequent energy flux corresponding to $kT_e\Gamma_{se}$. The power on the divertor coincides then to the kinetic energy of ions plus the acceleration due to the potential $\Gamma = \frac{1}{2}m_ic_s^2 + 3kZ_iT_e$. Another contribution is the chemical exothermic reaction of fuel ion recombination with the neutrals at the surface. The heat deposited at the divertor can so be estimated with $q_{k,dep} = (7\frac{kT}{e}/ + 18.1)e\Gamma_{se}$. As just explained, the intensity of the heat flux is controlled by the parallel transport, but the shape and the width are controlled by the perpendicular transport. The standard way to describe it is simply accounting for diffusive and convective fluxes through

$$\Gamma_\perp = D_\perp\frac{dn}{dx} + nv_\perp \tag{2.22}$$

A purely diffusive transport model seems to describe fairly well the experimental results, however the diffusion coefficient $D_\perp$ observed is too big to be explained by

classical transport. This is therefore a good example of a system in which neo-classical transport and turbulence are important. Usually those coefficients are taken from experiments rather than being derived from first principles. The same thing holds for the heat diffusion coefficients, which are both anomalous and extremely important for edge physics. Assuming quasi-neutrality, it can be written $q_{\perp_{e,i}} = -n(\chi_{\perp_{e,i}} \nabla_{\perp} T_{e,i})$. Different mechanism can dominate in various conditions, for example electron convection and neutral conduction is important in high density, while in low density diffusion is more important. Both parallel and perpendicular transport are stronger near the separatrix, but the temperature is also highest around that area and the parallel energy transport dominates over the radial heat transport ($\approx T^{\frac{5}{2}}$). Most of the energy is therefore deviated towards the divertor and doesn't reach the walls.

A code that is widely used to simulate the plasma in the Scrape Off Layer is the SOLPS. This is the code used to design the divertor for ITER and is the code that was used in this thesis. [6], [7].

Chapter 3

SOLPS

3.1 What is SOLPS

The version of SOLPS, Scrape Off Layer Plasma Simulation, used to model the divertor of ITER was SOLPS4.3, but since then both at IPP-Garching (EU) and in St. Petersburg (RF) the code was developed independently, leading to SOLPS5.1 and SOLPS5.2. The main difference is that SOLPS 4.x (x being a subversion) was using B2 for the fluid model, while more recent versions use B2.5. Then, those version were merged in the new tool that will be used for ITER, SOLPS-ITER. From SOLPS-5 to SOLPS-ITER an important change was the change from Braginskii to Balescu model. The two models are different in the following quantities

quantity	functional dependence	Prefactor (Braginskii)	Prefactor (Balescu)
$\kappa_{e\|\|}$	$\frac{T_e^{\frac{5}{2}}}{m_e ln\Lambda}$	3.16	$\frac{5}{2}\frac{2.16Z_{eff}}{1+0.27Z_{eff}}$
$\kappa_{i\|\|}$	$\frac{T_i^{\frac{5}{2}}}{m_e Z_i^2 ln\Lambda}$	3.9	3.98
$\eta_{i\|\|}$	$\frac{T_i^{\frac{5}{2}}}{m_e Z_i^2 ln\Lambda}$	1.92	1.81

Table 3.1: Differences between Braginskii and Balescu

SOLPS-ITER consist of a lot of different packages, the most important being B2.5 and Eirene [14]. Those are respectively the fluid and kinetic codes, written in FORTRAN and able to work both stand-alone and coupled. The code is always evolving, trying to introduce more physics and to become easier for the user. Amongst other things, it was decided to add a more user-friendly interface and attempts and progress are being made trying to implement monitoring frameworks and graphical postprocessing tools. A manual [9] contains almost all the information about the code, missing just the most recent developments and some of the

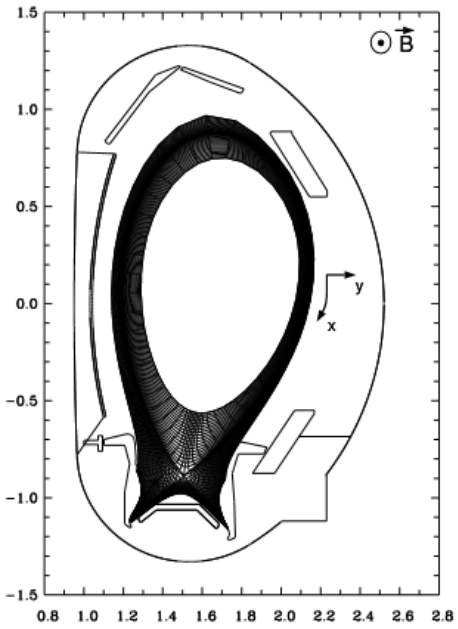

Figure 3.1: Coordinate system: x is the poloidal coordinate, y is the radial coordinate orthogonal to the flux surfaces

commands.

3.2 B2.5

B2.5 is a code based on a fluid model of the plasma. It is meant to describe a multi-species plasma in a 2D geometry. It was first created by Bas Braams in 1983 and then developed continuously in the next years. It is composed of more than 20000 lines and the output is post-processed so it can be read in by other codes (In particular, Eirene).

The equations are the Braginskii equations written on a curvilinear orthogonal coordinate system [11]. The x-coordinate varies along flux surfaces and the y-coordinate varies perpendicular to flux surfaces, while z is the toroidal direction and is ignored.

3.2.1 Continuity equation

Considering all the various contributes to the velocities, the continuity equation is

$$\frac{\partial n}{\partial t} + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x}\left(\frac{\sqrt{g}}{h_x} n (b_x V_{\parallel} + b_z V_{\perp})\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial y}\left(\frac{\sqrt{g}}{h_y} n V_y\right) = S_n \tag{3.1}$$

With the velocities composed by

$$V_\perp = V_\perp^a - \frac{1}{enB}\frac{1}{h_y}\frac{\partial nT_i}{\partial y} + V_\perp^{in} + V_\perp^{vis} + V_\perp^s \tag{3.2}$$

$$V_y = V_y^a - \frac{B_z}{enB^2}\frac{1}{h_x}\frac{\partial nT_i}{\partial x} + V_y^{in} + V_y^{vis} + V_y^s \tag{3.3}$$

where the terms are, respectively, the $E \times B$ drift plus the thermo-diffusive velocity, the diamagnetic flow of ions, the inertial, viscous and ion-neutral friction forces.

In the case of anomalous transport $V_\perp^a$ and V_y^a are modified by the addiction of the anomalous terms to the classic transport

$$V_\perp^a = V_{\perp classical}^a - D_{AN}^n \frac{1}{h_x n}\frac{\partial n}{\partial x} - D_{AN}^p \frac{1}{h_x n}\frac{\partial p}{\partial x} + V_\perp^{AN} \tag{3.4}$$

$$V_y^a = V_{\perp classical}^a - D_{AN}^n \frac{1}{h_y n}\frac{\partial n}{\partial y} - D_{AN}^p \frac{1}{h_y n}\frac{\partial p}{\partial y} + V_y^{AN} \tag{3.5}$$

The divergent part of diamagnetic velocity is kept, while is usually neglected, because the corresponding flux is still relatively large and can be comparable with the diffusive flux even for anomalous diffusion coefficients. The Coriolis force is taken into account too. The parallel viscosity driven by parallel ion heat $q_{i\|}^0 = \kappa_{i\|}^0 b_x \frac{1}{h_x}\frac{\partial T_i}{\partial x}$ is absent in the Braginskii equations, however, in a tokamak, where parallel particle flux multiplied by temperature is of the order of the ion heat flux, this term is important.

3.2.2 Current equations

$j_\perp$ and j_y are obtained from radial and poloidal projections of the total momentum balance equation. The main contributes that result from this calculation are a diamagnetic current j^{dia}, inertia and gyroviscosity j^{in}, viscosity j^{vis} and ion-neutral friction j^s. The diamagnetic current is most of the times the biggest contribute and is expressed as

$$j_{\perp,y}^{dia} = \frac{1}{b_z, 1}\frac{n(T_e + T_i)B_z}{h_{y,x}}\frac{\partial}{\partial y, x}(\frac{1}{B^2}) \tag{3.6}$$

While, for example, the current driven by the parallel viscosity is just a correction to the diamagnetic one, the vertical guiding centre drift of ions caused by the centrifugal force is responsible for a term that is often important.

3.2.3 energy balance equations

There are some problems in choosing the equations form for the energy balance,

1. The expression must be as simple as possible, and the most convenient format is an Ohm-like expression

2. The system of equations must be informed about non-classical contributions

3. The classical core of the equations must not be disturbed

In a multi-fluid case the standard Ohm's law cannot be used in the radial direction, because it is not sure if that current is heating electrons or ions. For this reason, additional heat exchange terms must be added, and those terms have to be produced simultaneously with the radial current. This is done introducing an *ad hoc* parameter $\eta_\perp$ to correct the electron friction density.

The balance equations are therefore

$$\begin{aligned}&\frac{3}{2}\frac{\partial nT_e}{\partial t}+\frac{1}{\sqrt{g}}\frac{\partial}{\partial x}(\frac{\sqrt{g}}{h_x}q_{ex})+\frac{1}{\sqrt{g}}\frac{\partial}{\partial y}(\frac{\sqrt{g}}{h_y}q_{ey})+\frac{nT_e}{\sqrt{g}}(\frac{\sqrt{g}}{h_x}(V_\parallel-\frac{j_\parallel}{en})b_x)=\\&=Q_e+nT_eB\frac{1}{h_xh_y}(\frac{\partial\phi}{\partial y}\frac{\partial}{\partial x}(\frac{1}{B^2})+\frac{\partial\phi}{\partial x}\frac{\partial}{\partial y}(\frac{1}{B^2}))\end{aligned} \tag{3.7}$$

for electrons, and a similar one for ions. The heat fluxes are opportunely closed, taking into account classical and anomalous transport. Q_e is written as $Q_e=-Q_\Delta-Q_u-Q_{AN}$ and the additional heat exchange terms are considered in

$$Q_{AN}=\sum_a(u^a-u^e)_\perp n_a\frac{Z_a^2}{Z_{eff}}e\eta_\perp j_\perp \tag{3.8}$$

A complete analysis of the terms present in the code can be found in [9].

3.2.4 Characteristics

Problems can arise from all the simplifications made in some particular regions or conditions. For example, unphysical flow at the plates must be avoided. The flow should be at least sonic at the sheath entrance [23],

$$Vk\geq C_s-\frac{b_z}{b_x}V\perp \tag{3.9}$$

and the continuity is accounted with the sheath current-voltage characteristics

$$j_x=en(b_xc_s-b_x\frac{q}{\sqrt{2\pi}}\sqrt{\frac{T_e}{m_e}}exp(-\frac{e\Phi}{T_e})(1-\gamma)) \tag{3.10}$$

where γ is the secondary electron emission coefficient. If the density is low enough the conductive heat flux can become important due to the strong temperature gradients. In this case, the simple sheath model does not hold and corrections need to be introduced, usually in the form of heat flux limiters. The flux limiter is implemented as

$$\kappa_k=\frac{\kappa_{SH}}{1+|q_{SH}/q_{fl}|} \tag{3.11}$$

with

$$q_{fl} = \alpha n_e T_e^{\frac{3}{2}} \sqrt{m_e} \tag{3.12}$$

Heat fluxes limiters are important for convergence and for a correct description of the data, so the user have to pay attention on how they are used by the code.
The fluid model holds well when all characteristic lengths (electron and ion mean free path, neutral mean free path, heat conduction length) are smaller than the parallel connection length. These values can variate from sub centimetres to tens of meters. There are three main situations in which the fluid model stops being reliable.

1. At low temperatures, like near the target in the divertor, steeper gradients increase the connections lengths and the processes stop from being local.
2. At high temperature the heat conduction mean free path is larger than the connection length, so flux limiters are necessary.
3. The whole sheath cannot be resolved by a fluid model and is represented by boundary conditions at the sheath entrance deduced from kinetic models.

Quantitative results in those cases are questionable, and the user have to be careful.

The fluid model is usually coupled with Eirene, the kinetic model than can describe the neutrals behaviour with higher accuracy. However, there is the possibility to run the fluid model in stand-alone version. The model for surface sources and sputtering is picked up by the user depending on the situation, the volume sources are given by plasma recombination rates. The neutrals are treated as other particles, with the same temperature of other ions and the same equations, without obviously the terms due to charged particle motions. The transport coefficients are

$$D^n_{AN,0} = 0 \tag{3.13}$$

$$D^p_{AN,0} = C + \sqrt{\frac{T_i}{m_N}} \frac{1}{\sigma_{CX} n_i + \sigma_{in} n_e} \tag{3.14}$$

where m_N, C,σ_{CX}, σ_{in}, are the mass of the neutral species, the hydrogen gas diffusion, and the charge-exchange and ionization cross-sections. For the stand-alone version, a new flux limiter was introduced, in the form

$$X \Rightarrow \frac{X}{(1 + |\frac{X}{\alpha F}|^{\gamma})}^{\frac{1}{\gamma}} \tag{3.15}$$

where F is the maximum and α and γ two values inserted by the user. The case studied is partially detached, so those conditions can be met and the flux limiters had been introduced changing the value α and γ. The heat conductivity is limited in the same way and is calculated by

$$\chi_N = n_N \lambda_{CX} \upsilon_T \tag{3.16}$$

The main problems with B2.5 arise when it tries to model neutrals. When neutrals are introduced in the plasma, depending on the situation, two things can happen. If the temperature is over 10eV the ionization length is on the same order of the charge-exchange length, so the neutral is soon completely ionized and a fluid model is sufficient. Otherwise a lot of processes, mainly charge-exchange, can take place before the particle reaches ionization. To describe those situations a new tool is needed.

3.3 Eirene

Eirene is a code based on a kinetic model and is capable of correctly simulating the behaviour of multi-species neutrals in a 2D or 3D geometry. It is designed to solve a system of linear kinetic transport equations and its most important applications are in connection with plasma-fluid codes. Neutrals are the primary source for fuelling the plasma and they have a strong influence on the behaviour of the plasma near the edge. The kinetic model doesn't consider the gas as a fluid, but is based on the distribution function, $f(\mathbf{r}, \mathbf{v}, t, i)$. The density is then calculated as $n_i(\mathbf{r}, t) = \int d^3v f(\mathbf{r}, \mathbf{v}, t, i)$ The central equation in this approach is the Boltzmann equation

$$\begin{aligned} &[\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} + \frac{F(\mathbf{r}, \mathbf{v}, t)}{m} \cdot \nabla_{\mathbf{v}}] f(\mathbf{r}, \mathbf{v}, t) + \sum_t (\mathbf{r}, \mathbf{v}) \mid \mathbf{v} \mid f(\mathbf{v}) = \\ &= \int d^3v' C(v' \longrightarrow v) \mid \mathbf{v'} \mid f(\mathbf{v'}) + Q(\mathbf{r}, \mathbf{v}, t) \end{aligned} \tag{3.17}$$

Sometime, however, instead of the particle density distribution the scalar transport flux ϕ is chosen as dependent variable. It is defined as

$$\phi(x) :=\mid \mathbf{v} \mid \cdot f(\mathbf{r}, \mathbf{v}, i) \tag{3.18}$$

And, from it, the pre-collision density can be derived, as

$$\psi(x) = \Sigma_t(x)\phi(x) = \nu_t(x) f(x) \quad with \quad \Sigma_t(\mathbf{r}, E, \Omega, i) = \Sigma_t \nu_t(\mathbf{r}, E, \Omega, i) / \mid \mathbf{v} \mid \tag{3.19}$$

The integral equation for this last quantity is

$$\psi(x) = S(x) + \int dx' \psi(x) K(\mathbf{r}', \mathbf{v}', i' \longmapsto \mathbf{r}, \mathbf{v}, i) = S(x) + \int dx' \psi(x) C(\mathbf{r}'; \mathbf{v}', i' \longmapsto \mathbf{v}, i) T(\mathbf{r}' \longmapsto \mathbf{r}, \mathbf{v}', \dots \tag{3.20}$$

Where C is a collision kernel and T is a transport kernel. This equation is particularly suited for a Montecarlo method of solution. It can be shown that the solution, under not too restrictive conditions, is unique. The interest, however, is not on the detailed solution for $\psi(x)$, and the code follows a different approach.

The N neutral particles are launched by a source of strength S, then their trajectories are followed for a length $l = -\lambda ln(r)$. For each step, a uniformly distributed random number ξ on [0,1] is compared with the probability $G_T(l_0) = 1 - exp[-\frac{l_0}{\lambda}]$. If it's smaller, a collision has occurred, and the particle is moved to the sampled point of collision. The complex interactions between the neutrals and the plasma background is simulated here, with all the possible results of a collision. Eriene uses a discrete Markoff chain, whose histories are used to derive, for every cell, average values of a set of responses R, defined as

$$R = \int dx \psi(x) g_t(x) = \frac{1}{N} \sum_{i=1}^{N} X_g(w_i) \tag{3.21}$$

Where the unbiased estimators X_g are, in EIRENE

$$X_g^e(w)_i^n = \sum_{l=0}^{N-1} (\int_{x_l}^{x_{end}} ds g_t(s) exp(- \int_0^s ds' \Sigma_t(s'))) \prod_{l=1}^{l-1} \frac{c(x_j)}{1 - c(p_a(x_j))} \tag{3.22}$$

Here x_{end} is the point in which the trajectory ends. This can happen because the particle was absorbed or because the code reached the maximum number of cycles allowed for that particle. Those are *extensive* quantities, which are calculated for every cell. *Intensive* quantities can be derived simply when the denominator is know exactly, while they require a little more work when that is not the case.

The variances are calculated in the usual standard way. A simple improvement to the code can be achieved by using sources stratification to speed up the generation of the particles. This method, however, is not used in the current work. In order to takes the surfaces into account, reflection models had been implemented. The reflection kernel is composed by three parts: fast particles reflection, thermal particle re-emission and particle absorption. The first one is usually just a mono-energetic, cosine angular distribution and the latter is dealt with changing the state of the particle in a "limbo" state that is outside the phase space. The reflection model for thermal re-emission is the Behrisch matrix reflection model, helped by a database that was produced with the TRIM code from large random samples of test particle trajectories.

Is important to notice that the ions that hit the walls come back as neutrals. Some more options or simplifications can be used too, like ignoring the volume of the sheath or letting a Maxwellian flux drift in it. Eirene geometry is implemented using a combination of boundary representation (BREP) and Voxel geometry. The first defines the majority of the surfaces using algebraic functions up to the second order, while the second creates the triangular mesh. The code also uses some databases, namely H2VIBR, HYDHEL, METHANE, PHOTON, SPUTER, TRIM. For a complete description of the data in those files see the manual [12].

Until this point the problem was linear. The first and biggest cause of non-linearity is the coupling with the plasma background. The non-linearity is not in the collision integral, but in the dependence of the collision parameters on the plasma parameters, that for long enough simulations vary with time.

Another contribute to non-linearity are the neutral-neutral collisions. For example, elastic self-collisions can lead to viscous effects, or elastic cross collisions to a heating of the molecular gas. In some cases, and the gas in the divertor area is a good example, even inelastic collisions can become important. The strategy adopted by EIRENE is to parametrize the single particle distribution and to use an iterative scheme. An additional, artificial specie is added, with the same parameters of the corresponding specie in the last time step. The non-linear part of the collision kernel is calculated considering this new specie.

3.3.1 Coupled B2.5-Eirene

The coupling between Eirene and B2.5 is done in an explicit way, with Eirene called at every time step.

The first thing to remember when running the coupled version of SOLPS is that the code relies on the Eirene part to compute the neutral particles, but the fluid model still follows them. So, it is necessary to hide the neutrals in B2.5, and this is done by rescaling them of a factor 10^{-10}. Both neutrals and neutral sources are rescaled, and then at the end of every run the correct value from Eirene is substituted in the fluid model. During the coupling, the two codes exchange information between each other. B2.5 provides a plasma background that is used by Eirene to compute neutrals trajectories and reactions, while Eirene gives the energy and momentum sources and sinks that are then summed again in B2.5. The fluid model then relaxes the solution given by Eirene through its internal iterations till the residuals are reasonably low.

Eirene works on a different (triangular) grid than B2.5, so it's important to run with two grid that match each other. In particular, if one attempts to change the grid size in B2.5, the grid size has to be changed in Eirene too [16].

3.4 Running SOLPS-ITER

The suite is composed by

1. Div-Geo/Unip, to generate the geometry and the input file
2. CARRE/Tria/Triageom, to generate the discrete grid for Eirene and b2.5
3. Sonnet, again for the geometry files
4. B2.5, the fluid model

5. Eirene, the kinetic model

6. B2.5-Eirene, the coupled code that is usually used for the simulations.

The first step is to run Div-Geo, which creates the magnetic equilibrium and the topology desired. Then the plasma-facing components and the number of cells are specified. Then an output file is passed to CARRE that generates the grid. Then triangulation takes place, where Unip translate the grid in a triangular grid of the vacuum vessel that lie outside the main plasma.

First b2ag creates a b2fgmtry file and a fort.30 files, with information respectively about the geometries for B2.5 and Eirene. Then Tria creates the triangular grid and Triageom merges it with the plasma grid. The input files created in those steps are the fort.33, fort.34 and fort.35, with the vertex positions, connectivity and triangle neighbour relationships. The next step is to choose the physics parameters, using some input files, i.e.

1. b2ah.dat with the default physics parameters.

2. b2ar.dat with the default atomic physics rates file.

3. b2ai.dat, that prepares the initial state of the plasma and save it in b2fstati.

4. b2mn.dat with various switches and numeric parameters, like the length of the time steps

5. b2.boundary.parameters with the boundary conditions, like input power (from the core) and puffs strength.

6. b2.feedback_control.parameters, with some feedback switches and parameters.

7. b2.feedback_save.parameters, with some time-dependent feedback parameters.

8. b2.neutrals.parameters with information about the neutral particles for both the fluid model and Eirene, when coupled. It contains information about neutral sources, or stratas, and recycling.

9. b2.numerics.parameters, with the possibility of changing the time-steps for the different regions.

10. b2.wall_save.parameters with information on the wall status and meant to assure a smooth restart. They don't need to be modified by the user.

11. b2.transport.parameters with the various transport coefficients.

12. b2.user.parameters with additional data for diagnostic. input.dat with the input data for Eirene.

13. b2.sputter_save.parameters with information about the sputtering for a smooth restart. Not intended for human use.

b2fstati is the initial plasma state file acting as input for any SOLPS simulation. b2ai creates a plasma state with a flat profile. The output of each simulation for the plasma state is stored in the file b2fstate, and is copied in as b2fstati during all subsequent restarts. There is a tool, namely b2yt, that is used to adapt the number of species and the geometry of the starting b2fstate to the one needed. A scripts allows the user to submit the run and to specify the maximum simulation time and whether the standalone or the coupled version of B2.5 is used.

Then the run is started, and a series of tools allow the user to check if it is converging. Some log files allow to see the evolution of more than a hundred variables in real time, and the basic variables can be plotted on a 2D model of the tokamak. A complete, real scale run can last for months. Post-processing tools are being developed to allow easier confrontations between the different runs.

Looking to figure 3.2 it can be seen how the grid is considered. The boundary conditions are added on the surfaces, divided into six main regions. It is worth notice that the border of the grid are also border in the geometrical rendering. The regions are named after the four cardinal points, with the obvious correspondence with the borders in the first figure.

The central "South" is the core, the other two lateral "South" the inboard and outboard divertor regions. a lot of spaces is gave to the divertors, which occupy the whole "West" and "East" borders. The "North" is the border between the SOL and the first wall.

The width and number of the time-steps can be varied by the user. For the ITER case that will be studied, the initial time-step was set to $10^{-7}s$. For the time-dependent version of Eirene, in fact, the time-step must be shorter than the time between re-ionization and subsequent neutralization of the particles, because the Monte Carlo procedure does not include intermediate ionized phases. Studies had been made about this parameter and it seems that with this width the code is stable. For the stand-alone version bigger time-steps can be taken without altering the performance.

In a steady state the time-steps can be much bigger. Given that the plasma in a pulse spends most of the time in a steady state, this greatly reduces the runtime needed for a complete run. Still, one of the biggest problems of SOLPS-ITER is that it is slow, usually one time-step takes one minute or more. It is easy to see that even using time-steps of $10^{-4}s$ or $10^{-3}s$ a simulation long as an ITER pulse, (hundreds of seconds) takes months. With $10^{-7}s$, for the transient part, even the simulation of a millisecond becomes computationally expensive. From here the need to find ways to simulate faster becomes obvious. One of the ways to do so is the parareal.

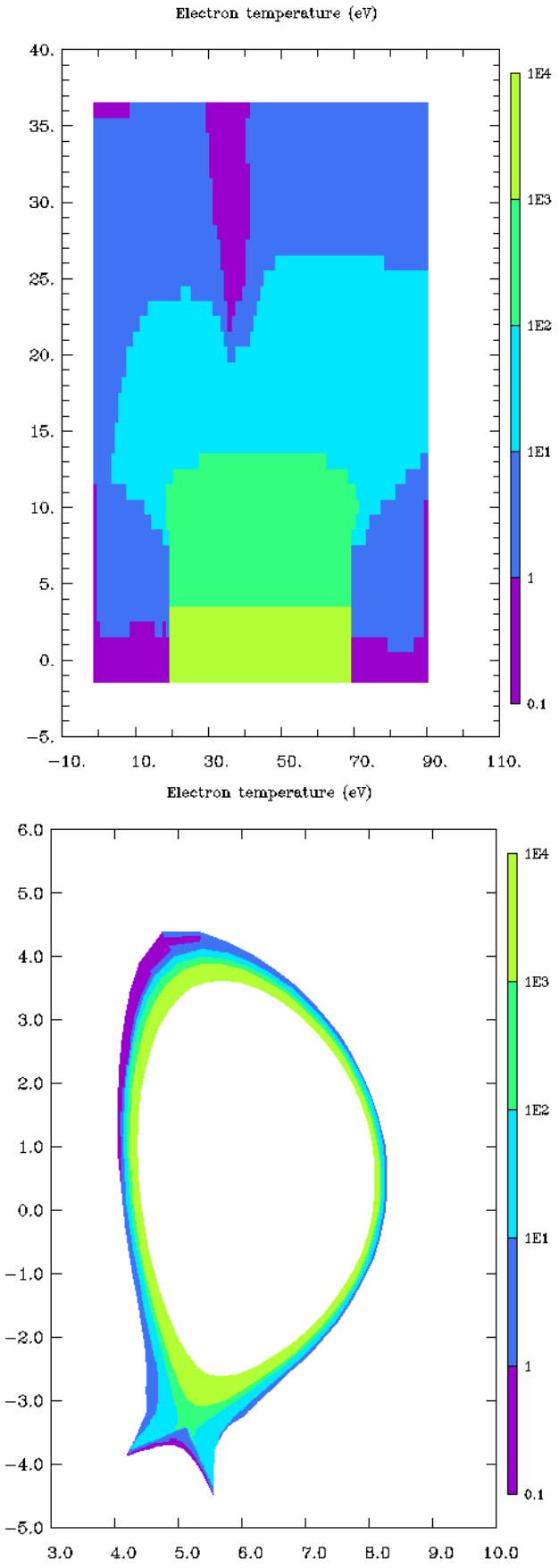

Figure 3.2: Temperature of the electrons, ITER, SOLPS-ITER. The first figure shows the computational grid while the second one show a rendering on the geometric grid

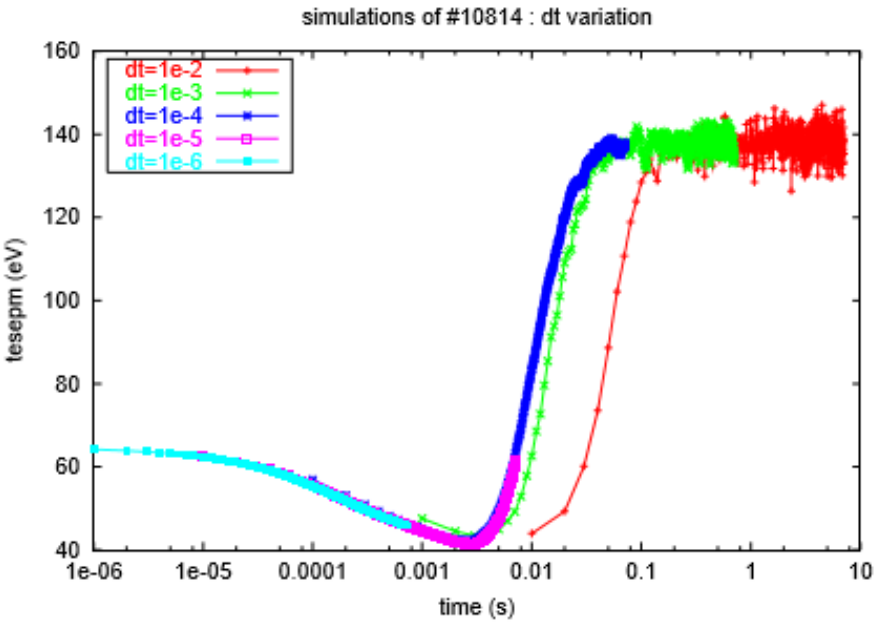

Figure 3.3: Upstream (outer midplane) electron temperature for a range of time-steps

Chapter 4

The Parareal

4.1 A general presentation

The parareal was first introduced in 2001 and is now widely used and studied. It is a method that can be applied to almost every initial values problem and has already been tested with interesting results [27], [26]. The core concept of the parareal algorithm is to parallelize the time domain in time dependent computations, thus making it a novel, counter-intuitive technique. Parareal exploits available computing resources to achieve this purpose and speed up codes. More and more powerful supercomputers are being built every year, breaking a record after another. This means that the availability of cores is constantly increasing, and that scientists need to find ways to use them.

Running a code on multiple cores allows to use more computational power and to considerably speed up simulations. Usually the simplest way is spatial parallelization, with each processor calculating a different region of the mesh. In the end of each iteration the information are exchanged between the cores and a new iteration can start. Problems can arise in case of non-local phenomena, but even when spatial parallelization works sooner or later it will reach saturation. Still, simulations are too long and there are more cores than the ones already used in most modern machines. The next step is then to parallelize time. This is more difficult for a time dependent solver, because for each iteration the results of the previous one are needed.

Apparently, time parallelization violates causality. Still, there is a way to achieve it. The idea is to "guess" the evolution of the simulation and then iteratively correct it. The first tool is a "coarse" solver, that will be called "G" from now on, that should be a lot faster than the "fine" solver, the original solver that will be called "F". A G just 20 times faster can bring good results if the number of iterations is not too high, but coarse solvers even 40 times faster are not that hard to obtain. The "coarse" solver should capture some of the physics, but not all of it. The hardest part is to be able to understand what are the most important processes in order to include them

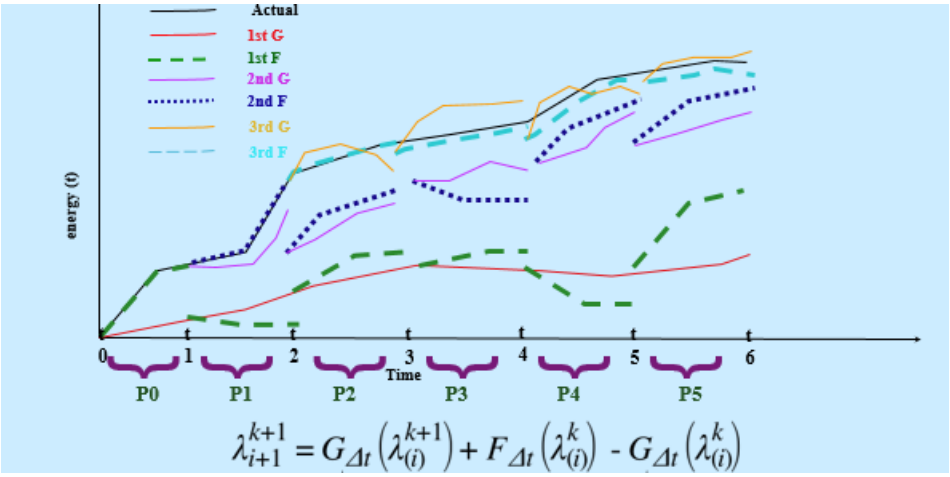

Figure 4.1: Scheme of the operating principle of the parareal [26]

in the coarse solver. This choice should be less case-specific as possible, in order to be able to use the same coarse solver for a big number of different cases.

The first step is to run the coarse solution in serial, operation that, if G is fast enough, should require just a fraction of the total time. Then the parallelization takes place. The coarse solution is divided in slices, one for each core available, and then for each slice a fine run begins with the input data provided by the coarse. At this point, it is better to introduce a little of notation.

1. λ_i^k is the initial data for the slice i in the iteration k.
2. $F(\lambda_i^k)$ is the output of the fine slice i in the iteration k.
3. $G(\lambda_i^k)$ is be the same for a coarse run.

After this first step, the second iteration begins. The first slice had already been computed by the fine solver, so there is nothing more to do. The second slice takes the input values from the output of the first slice, that again were calculated with the fine solver and do not need any further work: $\lambda_2^2 = F(\lambda_1^1)$. Then the coarse is run again in serial, but from now on the initial data are calculated by the parareal correction

$$\lambda_{i+1}^k = F_i^{k-1} + G_i^k - G_i^{k-1} \tag{4.1}$$

The correction in SOLPS is not made over all the variables in the code [25], but only on

1. na, the ion and neutral density
2. ua, the ion and neutral parallel velocity
3. te, the electron temperature

4. ti, the ion temperature

5. po, the electric potential

6. ne, the electron density

while

1. zamin, zamax, zn, specifing charge range and charges of the various species

2. am, specifing atomic mass

3. uadia, specifing the total drifting velocity for a specie

4. fna, the particle flux

5. fhe, the electron energy flux

6. fhi, the ion energy flux

7. fch, the current

8. kinrgy, the kinetic energy

are taken from the previous fine run without apply any correction.

This method is iterated until convergence is reached. The convergence is obtained if

$$\sigma_i^k < \xi \forall i < I \tag{4.2}$$

with ξ a maximum chosen by the user. The error is calculated by

$$\sigma^k = \int t_{i-1}{}^{t_i} \frac{|\lambda^k(s) - \lambda^{k-1}(s)|}{|\lambda^{k-1}(s)|} \tag{4.3}$$

on the maximum power flux on the inboard divertor.

In the end, the aim is to obtain the same result of the serial fine run. Given that the first slice for every iteration have to converge, because is a fine run that takes the input files from another fine run, the maximum number of iteration required to convergence is the number of slice. If that were to be the case the parareal would not provide any gain, but usually the number of iterations required to achieve convergence is much smaller. The performance of the parareal scale with the number of processors, reaching saturation after a certain point. Empirically, the ideal number of processors is of the order of the ratio $\frac{\tau_F}{\tau_G}$ where $\tau_{F,G}$ are the total time to run a serial fine and coarse run.

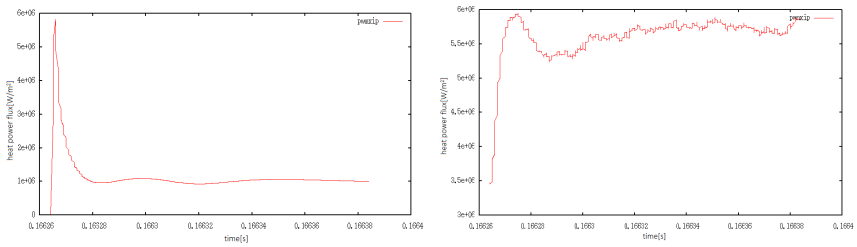

Figure 4.2: Comparison of a fine and a coarse run for SOLPS-ITER. The coarse has the same grid, bigger time steps, and is using the stand-alone version of the code. The variable is the maximum total power flux on the inboard divertor [W]

4.2 Choosing G

Clearly, to choose an appropriate coarse solver is crucial to get good results. The choice of G is not trivial at all, every problem is different and often what works for a situation can stop working for another one. Specific solutions can be found for various problem, but sometime this is not enough.

In edge physics, for example, there are many different regimes and the coarse solution must work in all of them. This means that it has to be quite general and this reduces the number of possibilities. The first and simplest idea is to just increase the time-steps. Studies on the behaviour of the program using bigger time-steps were already made and were showing instability of the program when it was run for too long. When the time step length is not increased too much, however, the method is plausible.

One of the reasons why the time-steps cannot become too long is the CFL condition, that substantially claims that there has to be a limitation on the ratio between the spatial and the temporal scales. A solution to avoid this is to change the grid size, making the cells bigger. This allows to take even bigger time-steps.

Changing the grid itself, without increasing the time-steps, does not change much of the computational time. This is due to the fact that most of the computational time is in this case taken by Eirene. Eirene launch a fixed number of particles and follow them for a certain amount of time, so it is not very affected by a change in the grid. However, it is possible to decrease the number of particles with a reduced grid, because what is important is to have a reasonable statistical error is the amount of particles per cell and not the total number of particles.

Another idea is to use just the fluid model instead of the coupled version of the code. Eirene is by far the slowest part of the code, so just turning it off would speed it up considerably. Moreover, the fluid model works quite well with bigger time-steps, that enhance the gain even more. A factor of around six or seven can be obtained by switching off Eirene, that combined with, for example, ten times bigger time steps gives a code that is sixty times faster.

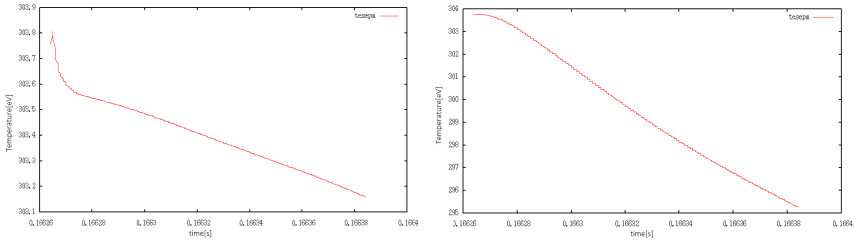

Figure 4.3: Comparison of a fine and a coarse run for SOLPS-ITER. The coarse has the same grid, bigger time steps, and is using the stand-alone version of the code. The variable is the electron temperature at the separatrix $[Wm^{s-1}]$

A third possible idea is to combine the two methods, using B2.5 with a reduced grid as a course solution. Greater computational gain can be achieved in this way, but the price is a solution that is further away from the fine one. If the coarse and the fine solution are too different the code is likely to crash or to take a high number of iterations to converge.

4.3 Running the Parareal

The codes are run on The Max Planck Computing and Data in Garching, a supercomputer where all the packages required by SOLPS-ITER are already installed. The length of the queue depends on the number of cores that the user is asking, up to around 500, and on the maximum time length.

There are two main scripts to interface the user with the parareal.

1. $parareal\_SOLPS2.pbs$ in which input and output folders are defined and the maximum time and number of processors are specified.

2. $SOLPS\_parareal2.conf$ that contains information about input scripts, input files, the number of slices, the maximum time for the run.

The first one is submitted and calls the .conf, which starts a python environment, the IPS framework. This is a powerful tool that helps in speeding up the parareal. It consists in a collection of python scripts. In this framework the fine iteration doesn't have, for example, to wait for all the coarse to finish the entire run. Every slice start as soon as it has the data ready from the two coarse and the fine needed.

The parareal scripts are then called by the python ones. They create and initialize the folders in which the various slices will run and manage the copying of files. Every script creates its own log files to allow the user to check if everything is working.

Chapter 5

The parareal on SOLPS

5.1 The parareal on SOLPS-5

5.1.1 Training and SOLPS runs

In the first part of this thesis some simulations with SOLPS5 were run to understand how the code works and how to change some input parameters useful for the parareal. The length of the time steps and their number are defined in b2mn.dat as respectively "ntim" and "dtim" and were the main parameters changed during this first part.

The first simulations were done using a DIII-D case that was already well tested and studied [28]. DIII-D is a Tokamak built in 1980 in San Diego and still working today. The species considered were Deuterium and Carbon, with all the possible degree of ionization, for a total of 9 species, two neutrals and seven charged. The geometry was a 96x36 grid and it was used the coupled version of the code.

The typical behaviour of a run is shown in figures 5.1 and 5.2, where tesepm and nesepm were chosen as variables. Tesepm is the electron temperature at the midplane on the separatrix, while nesepm is the density of electrons in the same point. The choice to print those particular variables is totally arbitrary, the post-processing codes allow the user to print all the variables in the code (see the manual [13] for the complete list of variables).

The next step was to continue a run and to check if it was correctly restarted. Being able to restart a run is crucial for the parareal, because every slice is a separate run that has to be restarted from a previous run. The files involved in this process, in this case, were "*b2.neutrals_save_parameters*" and "b2fstati"-"b2fstate". "b2fstati" contains information about the initial values for the plasma and "b2fstate" is the same file but with the final parameters. "b2fstate" has to be the "b2fstati" for the new run.

The information about the plots are stored in "b2time.nc", that is a NetCDF (Network Common Data Form) file, a machine independent data format that is

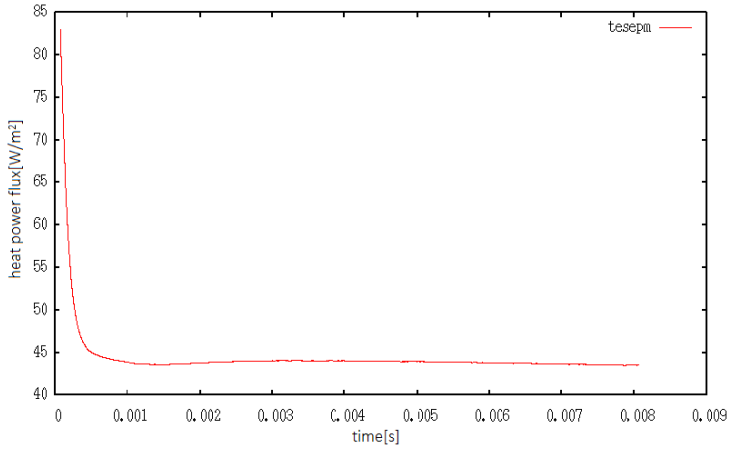

Figure 5.1: Temperature [eV] on the separatrix, SOLPS-5, DIII-D, run with 4000 steps

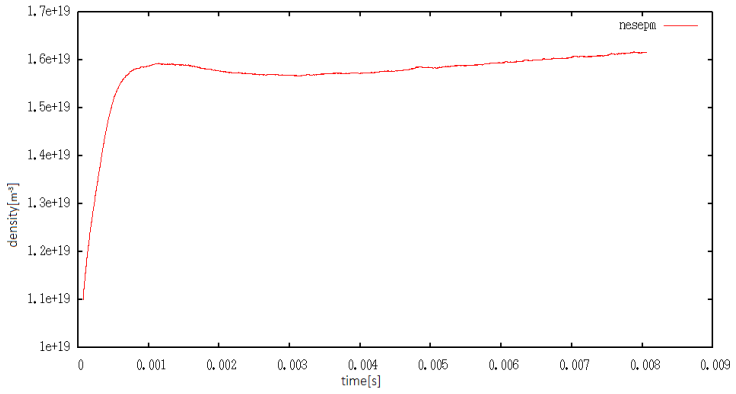

Figure 5.2: Density [particles on cubic meter] on the separatrix, SOLPS-5, DIII-D, run with 4000 steps

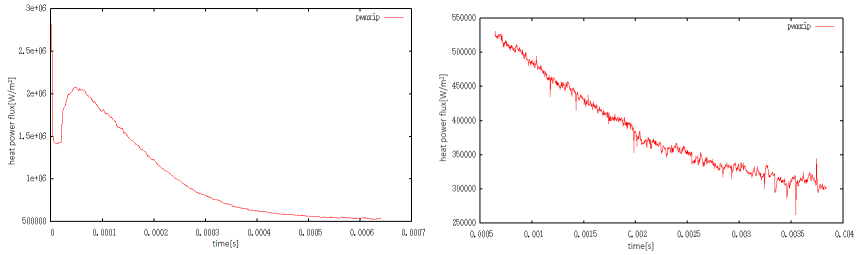

Figure 5.3: pwmxip for two long, consecutive runs, DIII-D, SOLPS-5

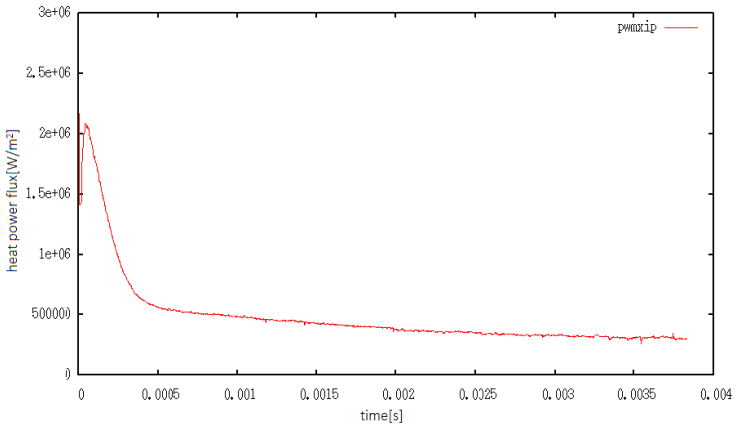

Figure 5.4: Joint run for pwmxip, DIII-D, SOLPS-5

specifically designed to store array-like variables. A special tool was developed to join two ".nc" files.

In figures 5.3 and 5.4 an example of the union of two plots is presented. It is important to check that the variables do not have discontinuities in the point where they are joined and that the transition is smooth. The variables chosen for the example are pwmxip and pwmxap, the maximum total power fluxes at the inboard and outboard divertors respectively, in $W \cdot m^{-2}$.

5.1.2 Training and parareal runs

The setup for a parareal run was then prepared. The same DIII-D case was used as an example, choosing the simplest G, i.e. a run with just bigger time steps. Every simulation has to be run in a different directory, in which the user can change the relevant parameters. A setup for a parareal run mainly requires two directories, a prototype folder for the fine runs (Frun) and one for the coarse runs (Grun). Since the total time length of the slices has to be constant, the product of ntim and dtim has to be constant in the two directories.

$$ntim_F * dtim_F = ntim_G * dtim_G \tag{5.1}$$

Once this simple parareal run was successfully completed a test run with a

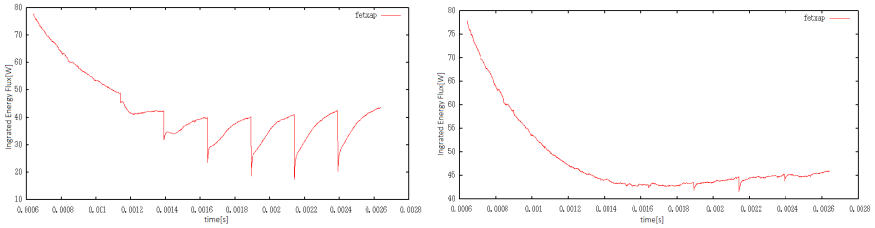

Figure 5.5: first and last iteration of a parareal run, fetxap, DIII-D, SOLPS-5, plot of the fine slices

slightly advanced G was built. The grid was changed from 96-36 to 48-36. To do so there exist a special routine, "b2yt". "b2yt" allows to change the grid and the number of species, but only the first option will be used in this work.

A further step is required in this case. Since the parareal correction has to work with input files of the same kind, b2yt has to be used two times every time a coarse slice is computed. A "fine" set of input files is copied in the directory where the coarse run will take place, then b2yt is used. The run is started with the changed grid, then the output is converted again ready for the correction.

In figure 5.5 it can be seen how the convergence plot is not completely smooth in general. The variable chosen for this example is fetxap, the integrated poloidal total energy flux on the outboard divertor. The convergence check was done just on pwmxip and pwmxap, so the other variables can display those little discontinuities. However, it was seen that usually the restrictions in use are sufficient to keep the "jumps" on the other variables under reasonable limits.

5.1.3 The ASDEX case with SOLPS-5

ASDEX is a tokamak built in 1980 at IPP, Garching. The parareal was applied to an ASDEX case using SOLPS-5. Both the fine and the coarse were run without Eirene, changing the length of the time steps and the grid. The standalone run is faster and more stable than the coupled one. Bigger time-steps can be taken too, but the price is a less reliable simulation. The time steps of the fine run were chosen $1.5 * 10^{-5}[s]$. Ten times smaller time steps were showing an almost identical solution, while with ten times bigger time steps the values were quantitatively different.

The case considered will be a single Deuterium species, with a 96-36 starting grid and the time steps in the core ten times bigger than in the other parts of the SOL. This artifice is sometimes used in SOLPS and is proved to increase convergence. There is a particle flux coming from the core and a leakage on the divertor area.

It was seen in [27] that a careful calibration of $ntim_F$ and $ntim_G$ results in an optimization of the performance of the parareal. In particular, when $ntim_F$ is made too small or too big, the algorithm fails. When it is too big the reason is that the coarse solution evolves far from the fine one and when the parareal correction is applied the plasma state that result is out of the range of parameters in which the

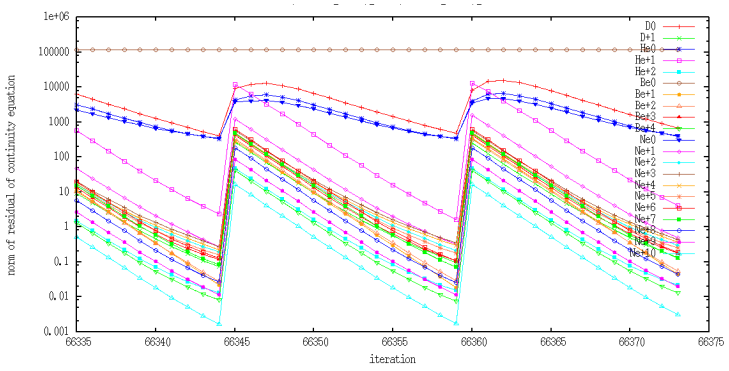

Figure 5.6: Typical residual behaviour for a SOLPS-ITER run. ITER, all 21 species

code can work.

SOLPS is, in fact, quite sensitive to the initial state of the plasma, and a random alteration is likely to make it crash. This is the reason why usually a simulation is not started from an arbitrary initial state but from the end of a previous run. A crash consists in an exponential growth of the residuals of the continuity equation. A typical residual behaviour is shown in figure 5.6. Only two steps are shown in the figure: during each step the internal iterations in B2.5 keep lowering the residuals for each species.

Given that the calculation of residuals involves divisions, when the density is very low some residuals can remain high and never decrease. As long as they are stable, however, this is not a problem for the run. This is the case for neutral Berillium in Figure 5.6.

It is less clear why the code crashes when $ntim_F$ is made too small, intuitively the run does not have time to relax the solution and this makes the residuals grow too much. This issue is currently being studied to be solved in a more mathematically strict point of view. This behaviour was seen in this case too, as is clear from table 5.1.

The computational gain ξ is simply calculated as

$$\xi = \tau_{serial}/\tau_{parareal} \tag{5.2}$$

A problem was found in calculating τ_{serial}. The same serial run, in fact, can take different times to be completed. The reason is related to the proprieties of the supercomputer in Garching. It is not clear if some core is faster depending on the kind of core, the heating or if the problem is in the sharing of information or memory between the cores. The difference can be even of a factor 1.5 and must therefore be taken into account.

To overcome this problem, a long parareal simulation was run and the times were taken for each slice from the log file. The results is shown in figure 5.7. It can be seen that the length of the simulations is basically divided in two populations,

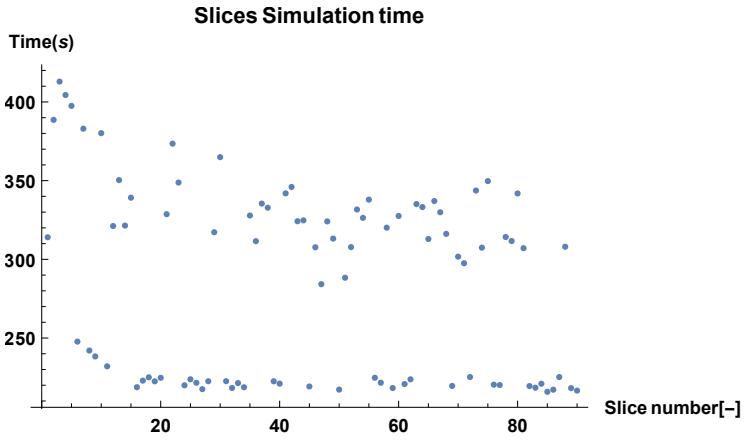

Figure 5.7

one almost stable around 220 s and the other one varying around 320 s.

A simple average was taken over 100 slices and it was found that a good estimate for the time required for a serial run is

$$\tau_{serial} = \tau_{step} \cdot N_{tot} \tag{5.3}$$

where $\tau_{step} = 1.38 \pm 0.03$ and N_{tot} is the total number of time steps of the simulation.

It has to be noticed that this estimate is very rough and case dependent. All the simulations considered here start from the same point, so this approach is reliable enough in this particular case. The interest is not on the precise value of the gain, so the error on τ_{step} was chosen arbitrarily bigger than the error on the average.

It was seen that no gain was obtained with just 16 processors, so a set of results with 64 processor will be presented to show the effect of the variation of $ntim_G$ and $ntim_F$. The reason of the failure of the parareal with a small number of processors is that the parareal scripts take some time to run, especially when they have to change the grid and copy files. The effect is negligible when the runs are very long and the number of iterations is a lot smaller than the number of slices, but otherwise it can be dominant.

A ratio between 10 and 20 proved itself to be most effective. The gain does not grow much more than three even when a higher number of processors are used. Using Eirene for the fine solution would considerably slow the run and, if the coarse solution does not take too much iteration to reach the new solution, this would assure a higher gain. This option was not explored for SOLPS5, most of the attention being on SOLPS-ITER.

It can be seen that with 400 step in the fine run the solution is already too far from the coarse one for the code to be able to run. Every run with more step resulted in a crash, or the ratio $\frac{ntim_F}{ntim_G}$ was so low that no real gain was possible.

ntim fine	ntim coarse	grid	serial time (s)	parareal time (s)	gain
40	4	48-36	1776	aborted	-
50	5	48-18	2208	2167	1.02
100	5	48-18	4416	2343	1.88
200	10	48-18	8832	2940	3.00
200	20	48-18	8832	2709	3.26
300	10	48-18	13248	6677	1.98
300	15	48-18	13248	5206	2.54
400	20	48-18	17760	aborted	-

Table 5.1: Gain. The error on the parareal time is of the order of seconds, while is on the order of minutes on the serial time

There is another thing that can happen when the slices are made too long or too short. The parareal is implicitly based on the fact that the soution of the coarse for just a single slice can match the fine solution with a really good accuracy (usually $\varepsilon < 0.05\%$).

When $ntim_G$ is too small the coarse solution just does not have the time to match the fine one, and it results in an endless oscillation that never reaches convergence. It was not observed, because the most probable outcome is a crash, but it can probably oscillate also when $ntim_G$ is too big.

In table 5.2, 5.3 and 5.4 the scaling with the processors is described.

processors number	serial time (s)	parareal time (s)	Gain
16	2208	1681	0.76
32	4416	2085	2.11
64	8832	2980	2.96
128	17664	5552	3.18
256	35328	10100	3.49

Table 5.2: Gain varying the number of processors, $ntim_F = 200, ntim_G = 10$

The gain grows quite fast at the beginning and then it tends to saturate. There will be a number of processors that will make the code crash at some point, because the first fine and coarse run will eventually evolve far from each other, but this problem was not met in the runs studied. Hopefully that number is bigger than the maximum number of available processors.

Intuitively, a bigger $\frac{ntim_F}{ntim_G}$ ratio improves performance when the number of processors is high. A small ratio, in fact, would leave the coarse solution not so much ahead of the fine one, and there would be unused processors. The scaling was then

tested increasing it from 20 to 30.

processors number	serial time (s)	parareal time (s)	Gain
16	3312	2624	1.26
32	6624	4140	1.6
64	13248	6677	1.98
128	26496	11419	2.32

Table 5.3: Gain varying the number of processors, $ntim_F = 300, ntim_G = 10$

The performance are worse than the first case. This is due to the fact that the parareal took more iterations to converge, slowing the process. The effect may be mitigated increasing the number of processors even further, but a simpler solution was found. As it can be seen below, increasing $ntim_F$ without modifying the ratio was more effective.

processors number	serial time (s)	parareal time (s)	Gain
16	3312	2215	1.49
32	6624	2789	2.37
64	13248	5206	2.5
128	26496	8182	3.23

Table 5.4: Gain varying the number of processors, $ntim_F = 300, ntim_G = 15$

5.2 The parareal on SOLPS-ITER

SOLPS-ITER was then installed and run, both on the coupled and on the stand-alone versions. Some changes had to be made on the python and on the parareal scripts to adapt them to the new code.

There are some special files, recognizable by the presence of "save" in the name of the file, that are changed during the run. There are three of those files:

1. $b2.wall\_save\_parameters$
2. $b2.feedback\_save\_parameters$
3. $b2.sputter\_save\_parameters$

Their role is to assure a smooth restart of the run, a feature needed by the parareal. SOLPS-ITER reads only the ones that are already present in the run folder. They are then changed and the final version is needed to continue the run.

None, some or all of them can be present, and the code has to be able to recognize how many of them are present. It also has to be able to read and copy only the useful ones, managing them properly between the directories.

Three more files, fort.11, fort.13 and fort.15, can be made read in by the code. Those are Eirene input and output files, containing respectively information about tallies, background and test particle population. The problem with those files is that they are not just time-dependent, but they are also grid dependent. This means that, for the first coarse slice, is not possible to use the fort files from the fine run.

The idea is to create the fort files for the coarse from a very short coarse run that does not read in any fort files. This has to be done manually by the user. The input files have then to be copied in a special folder and the code will take care of them from that point onwards. At the present state of art, the files are copied even if they are not needed. This does not slow much the code and provide more versatility.

Another important feature of SOLPS-ITER needs to be discussed in the case that Eirene is used for the fine solution and the stand-alone for the coarse one. As stated before, the fluid model is not very reliable dealing with the neutrals. For this reason, two switches exist in b2mn.dat that rescale the neutrals for the coupled version. They are suppressed by a factor 10^{-10} in B2.5 and dealt with by Eirene.

Using the output files from the coupled version as input for the stand-alone is necessary to rescale the neutrals again, or the fluid model would see a fluid without neutrals and the results won't be reliable. There are four different cases, depending on whether the fine (or coarse) run is going to pass the data to a fine or coarse run. For each case the switches in b2mn.dat have to be changed.

Only after all these issues had been addressed, the physics simulations could be performed.

5.2.1 The ASDEX case with SOLPS-ITER

The same single D species case that was used for SOLPS-5 is considered again for a parareal run with SOLPS-ITER. Boundary conditions were changed a little bit in order to be able to run the code, that can be quite unstable when some options are used. In this case, as for the SOLPS-5 case, the stand-alone version was used both for the coarse and the fine. This choice was due to the fact that the process to change the grid with SOLPS-ITER is a little bit different from earlier versions of the code.

First, the grid used by B2.5 has to be changed using b2yt. This requires a geometry file created by Div_Geo or CARRE, a mesh generator available with the SOLPS package. The number of cells can be divided only by integer numbers. To run the coupled version, however, it is necessary to provide three more files, namely fort.33, fort.34 and fort.35. To obtain those files is not trivial, but they have to be created just once for each grid-size. Once this step is done, they have to be put in

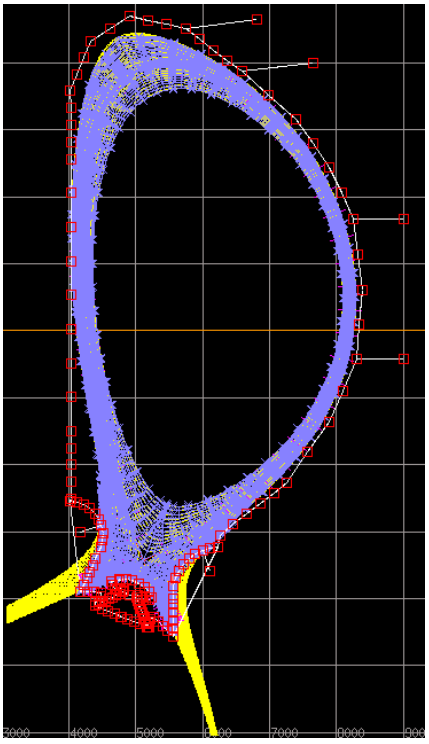

Figure 5.8: A typical SOLPS mesh with the equilibrium, from a DivGeo session, whole profile. The equilibrium file is represented by the yellow lines, the mesh by the blue lines. The white lines represents the physical surfaces

a set of special folders and the code will pick the ones for the correct grid in use.

The documentation is not very complete yet for what concerns the changing of the grid [17], [18], [19], [20], [21], so a short manual was written to enable other SOLPS-ITER user to change Eirene grid when necessary. A workshop in November 2016 should clarify how to use the instruments that are being developed to do so.

The problem for the ASDEX case was that this process, called triangulation, requires three files: a Div-Geo file, an equilibrium file and a mesh file. Those files were not available, so it was impossible to change the grid for the coupled version.

Another kind of work was carried on in this case. As explained when the parareal was presented, the code does not correct all the variables, but just a subset of them. Moreover, it does not correct the neutrals, because it was first meant to work with a coupled - stand-alone case. It was therefore interesting to try to change the correction script and see if the performance were to increase. Four different sub-cases are possible.

1. Correcting just a sub-set of variables and not correcting the neutrals
2. Correcting just a sub-set of variables and correcting the neutrals
3. Correcting all the variables and not correcting the neutrals

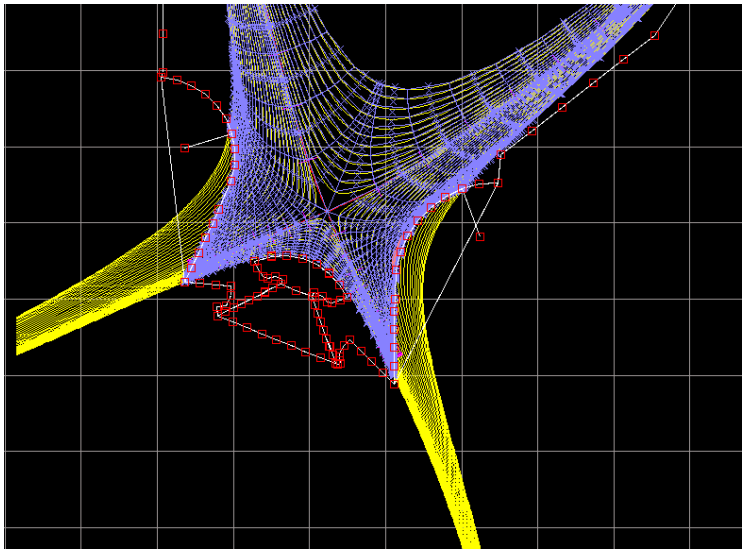

Figure 5.9: Typical computational mesh from DivGeo session, zoomed in the divertor area. The equilibrium file is represented by the yellow lines, the mesh by the blu lines

4. Correcting all the variables and correcting the neutrals

The results are summarized in the table below. The reduced grid was 48 x 36 and the number of processors 16. For all the runs it was $\frac{ntim_F}{ntim_G} = 10$. The same approach as in section 5.1.3 was used to find the error on the serial run simulation time.

Again, runs with $ntim_G$ too small, equal to 5 in those cases, crashed. It is less clear why the run with $ntim_G = 15$ crashed, apparently there was a problem with the convergence. The gain is still around 2, but with just 16 processors this is a decent result, better than with SOLPS5. No relevant differences were noticed changing the corrections, but the data is not sufficient to strongly confirm that.

It was decided not to proceed further in the studying of this case. It was in fact noticed that, with long runs, there was a visible discontinuity for some variables in joining two plots (fig.5.10). The same effect was responsible for a kick in the residuals during the restart of a run. This was a problem for the code in general and in particular for the parareal. At the time of this thesis work, SOLPS-ITER was still a code under development. Although new releases continue to occur, the code is more stable now.

In version 3.0.5, in particular, a large number of new variables were added in the plasma-state files. This is likely to improve the performances of the parareal, but a lot of scripts will have to be changed and the performance will have to be tested again. This kind of work was out of the scope of this thesis.

type of correction	number of fine steps	serial time (s)	parareal time (s)	Gain
1	50	713	aborted	-
1	80	1141	723	1.6
1	100	1426	710	2.0
1	120	2140	1052	1.6
1	150	2215	aborted	-
2	50	713	aborted	-
2	80	1141	703	1.6
2	120	2140	1043	1.6
2	150	2215	1012	2.1
3	50	713	aborted	-
3	120	2140	1234	1.4
3	150	2215	1019	2.1

Table 5.5: Gain the corrections

5.2.2 The ITER case with SOLPS-ITER

This was the most complex and interesting case studied during this thesis. Even if the parareal is still not working on an ITER-case, steps were taken in understanding its physics and how to handle it.

The geometry is shaped with a single null, with a 90 x 36 grid. It is a four species case, Deuterium, Helium, Beryllium and Neon, for a total of 21 species, 4 neutrals and 17 charged. The power output from the core is set to 50MW as a boundary condition. The electron decay length in the SOL is set to $\lambda_e = 30[cm]$. For the neutrals, the leakage in the core is specified. The core works as a source of charged ion species for deuterium and helium, with strength of $\nu_D^+ = 9.1 \cdot 10^{21}$ and $\nu_{He}^{++} = 2.13 \cdot 10^{20}$ particles $\cdot s^{-1}$.

The absorption coefficients for the divertor are specified, in particular neutral Beryllium is absorbed completely. Neutral Neon is pumped in the SOL area with a strength of $\nu_{Ne} = 4 \cdot 10^{19}$ particles $\cdot s^{-1}$. Transport coefficients are defined for the charged species specifying the characteristic length for density, momentum and heat flux.

The anomalous transport coefficients were corrected from the starting ones. In order to do so a new matrix, which proved itself to make the code more stable, were added in the input file b2mn.dat. The values stored in this file, in fact, have priority and rewrite the corresponding values in other files. Some technical information about the sputtering model were changed too using the switches in b2mn.dat.

At first a simple coarse case with Eirene computed on a reduced grid and longer time steps was used. As most of the simulation time is taken by Eirene, however, the

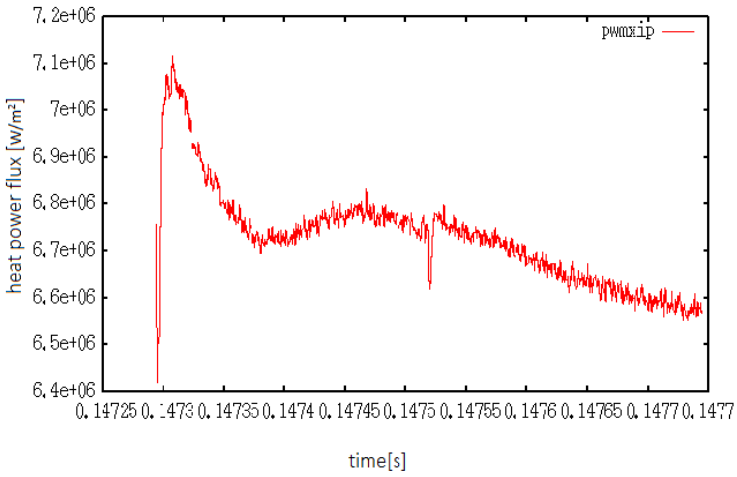

Figure 5.10: Discontinuity in two consecutive long runs. ITER machine on SOLPS-ITER, maximum heat power flux on the inboard divertor

speed up was not decisive enough. The kinetic code, in fact, launches and follows always the same number of particles, regardeless from the grid. The convergence was not very fast and the $\frac{\tau_F}{\tau_G}$ was not high enough to balance the large number of iterations.

More changes were needed in this case to try to adapt the coupled version with the standalone version. At first a lower temperature was observed in the divertor area in the stand-alone case, so it was tried to increase the heat flux from the core in order to bring the temperature up. However, it was seen that the heat flux from the core had a very little influence on the temperature. This is mainly due to the fact that the heat takes a lot of time travelling from the core to the divertors, so it is impossible to reach the values needed without a substantial modification of all the plasma parameters.

Moreover, it was seen that the coarse runs were crashing after some hundred steps. The heat flux limiters presented in section 3.2.4 were introduced both for the stand-alone and for the coupled version, but it was not enough. This was happening because the behaviour of the simulation that was not taking neutrals into account was too different from the coupled one. This fact can give an idea on how much neutrals influence is important in an experiment like ITER.

A possible solution here is to stabilize a stand-alone run and then use that b2fstate to start a new coupled run. Eirene, in fact, usually tends to give robust results and the simulations with the coupled code are less likely to crash. This is done rescaling the density of particles in the whole plasma by changing a switch in b2mn.dat. There will be some value with which the code will be able to run, so it is sufficient to start some runs with different values of the switch and find one that

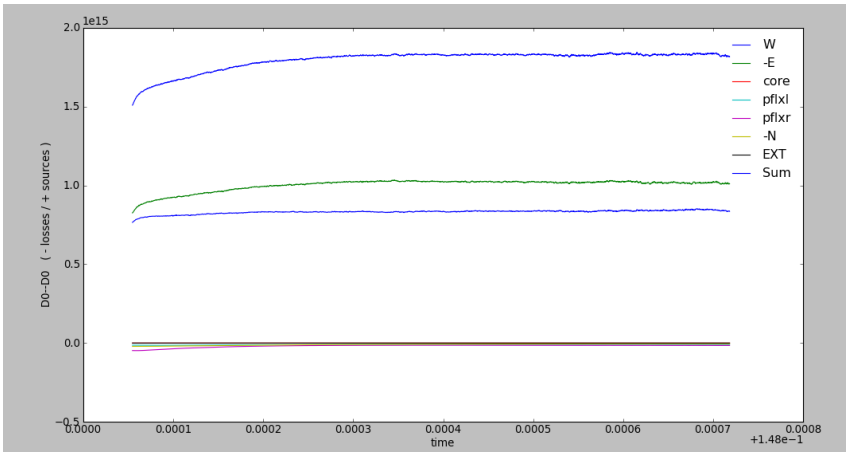

Figure 5.11: Fluid fluxes, ITER, SOLPS-ITER, neutral deuterium

does not crash.

This approach was taken thanks to the fact that just an example was needed for this work. Users obviously will have to be able to start from an arbitrary case. To find a stand-alone case from a working coupled case is possible, but it requires a little bit more of work. The first thing to do when a user wants to do so is to adjust the density of neutrals in the divertor area.

Substantially, it is necessary to do a calibration. There are two processes that can be used to control the density: puffs and pumps [15]. These two quantities are defined in $b2.boundary\_parameters$ and can be varied by the user. The idea is to vary the pumps and to introduce a new strong Deuterium puff. The Neon puff can be varied too. Those quantities are changed until the density of the stationary state is as close as possible to the density found in the coupled simulation. This is coherent with the need to have the coarse run as similar as possible to the fine one.

The most reliable way to check if the plasma has reached a stationary state after this kind of changes is to analyse the fluid fluxes (Figure 5.11). In a stationary state the fluxes entering the plasma must be equal to the ones that are exiting. A python routine allows the user to print the total fluid fluxes for every region and for every species, varying the time. It provides also the possibility to sum the fluid fluxes for various species.

This is useful because the total fluid fluxes have to be null both in total and for every atom, but not for every charged species. Considering deuterium, for example, the D^+ flux coming from the core will be neutralised in the divertor area and then pumped as D^0.

The calibration process has some problems. First of all, to reach a stationary state in an ITER case takes a lot of time. During the first part of the run the variables are changing very fast, because the plasma state is far from equilibrium, so it is quite likely that the run will crash. To overcome this problem it is necessary to

start more than one run and to keep the ones that do not crash. It is then possible to get bigger time steps, even fifty times bigger, but it still takes more than one week to reach a stationary state.

It can be asked what is the utility of the parareal if one needs a calibration that can last a week before being able to use it. The point is that usually SOLPS is used to do parameter scans, so a lot of very similar cases are runned at the same time. The calibration for all those cases should be done just one time, and then the parareal could turn a month in a week. Still, a more automatic way to calibrate the stand-alone would be a big improvement.

5.2.3 Tools developed

Managing three different cases and running tens of parareal simulations is too long to be done manually, both because is too time consuming and because human error can make the user lose a lot of time. A series of scripts were developed to automatize the process. The input files the user needs are just the folder where the serial run is ready and a folder called "$Input\_parareal$" where all the scripts required by the parareal are stored. The script automatically recognize the files that will need to be copied. It can work both with SOLPS-5 and SOLPS-ITER, but the user has to specify which code is using.

A folder called "$Base\_Run\_Parareal$" is created, and the user can make in it any change desired to customize the coarse run. This can include the calibration too, that has to be done separately. If they are needed the user will have to provide all the fort files specified before in the specific folders.

Then, two scripts allows the user to start the parareal runs. One is for both fine and coarse using the stand-alone version of SOLPS, while the other is for a fine coupled.

There are some options that can be specified. Those are

1. The maximum time length the simulation is allowed to run
2. The number of processors
3. The tolerance for the convergence
4. $ntim_F$, $ntim_G$, $dtim_F$, $dtim_G$
5. The dimensions of the grid
6. The kind of corrections desired. The user can choose from all the four options presented in 5.1.
7. Whether, just for the coupled fine version, the coarse has to be coupled or stand-alone.

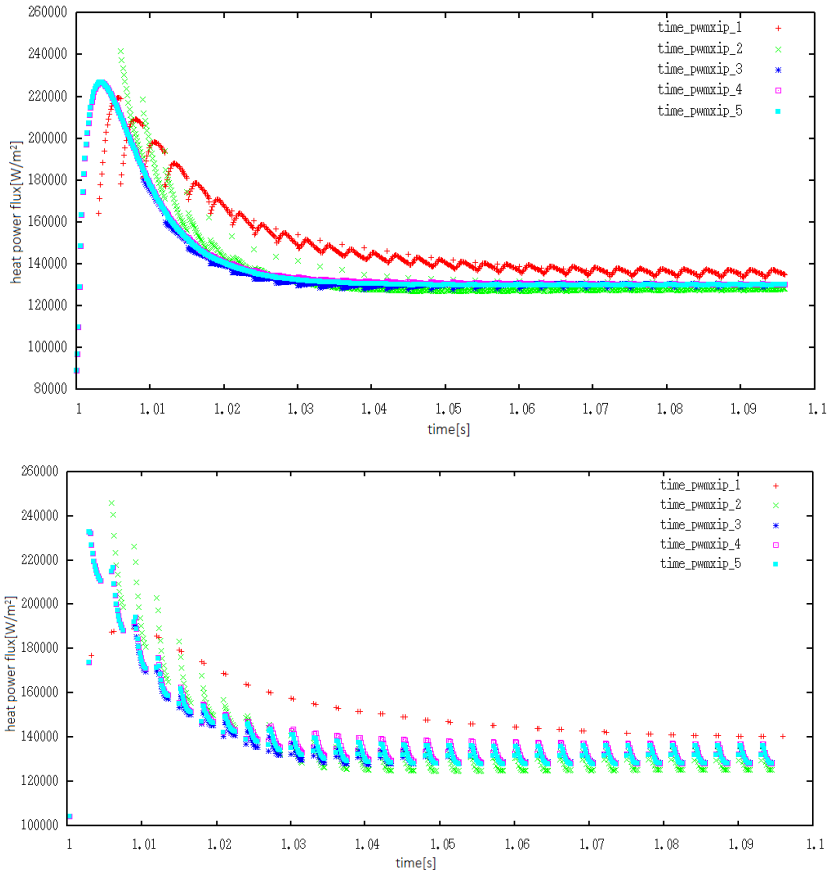

Figure 5.12: plot with all the iterations for a parareal run, fine slices on the right and coarse slices on the left. ASDEX, SOLPS-ITER, pwmxip, 64 processors

The script will automatically start a parareal run with the specified options and with the eventual changes made at the beginning. Along with it, the script starts a serial fine run and a serial coarse run. This allows the user to compare the two runs, to check convergence and to get an idea on the gain. If for some reason the parareal crashes, the default serial fine run will still be completed, providing results.

It already existed a script to plot a particular iteration of the parareal run once everything was finished. A new tool was developed in order to allow the user to plot all the iterations at the same time, both in a series of plot and in one plot for all the iterations (Figure 5.12).

The whole process is not perfect yet, it still is not tested for all the cases, but it allows the user to set up a parareal run in a fraction of the time.

Chapter 6

Conclusions

Four cases were analysed in this study. The first one was just an example, but it was useful to understand how the parareal works and served as a solid base to check the changes to the code and the development of the scripts. The data collected running the ASDEX case on SOLPS5 proved the functioning of the parareal algorithm, justifying the attempt to use it in SOLPS-ITER. The semi-automatic procedure that was developed can actually allow users who are running ASDEX cases with SOLPS5 to use the parareal and cut by a factor of three the time required for their runs.

A first optimization was done on the ratio $\frac{\tau_F}{\tau_G}$, showing the highest gain around twenty. The scaling with the number of processors was confirmed in this case too, with an increasing trend for all the simulations.

While the edge fusion community turns from SOLPS5 to SOLPS-ITER, the code will gain importance. The parareal can speed up the runs and be a useful tool for the users, with the implementation of the algorithm in the last version of the code being a step towards its possible diffusion. The compatibility was reached in this thesis, as proved by the results obtained in the ASDEX case. A gain of two with just 16 processors is promising and justifies further research.

In cases like ASDEX ones, that are not too complicated to run, there are chances that the parareal will work without further complications. The optimisation of the parameters is quite case dependent and will have to be done every time, or at least once for every class of cases that are fairly similar. The correction protocol was changed and tested without deterioration of the performance and now the user can choose the best method depending on the physics studied. It would be interesting to move forward in trying to use different corrections, to see if one of the new choices available can actually improve the gain.

Even more, a useful study would be to use the coupled version as a fine model and the stand-alone as a coarse. The two solutions, without making any other alteration to the coarse, will probably be far from each other. Knowing that an ASDEX case takes a much shorter time than an ITER one to reach a steady state, this could be

a perfect place to test and improve the calibration process. The ideas for the ITER case could be tried on this simpler example, to test their effectiveness before losing a lot more time on the slower cases.

The focus of any following work, however, should remain on the ITER case on SOLPS-ITER. It was studied and modified to increase stability, with operations that can be repeated on other ITER cases. The problems with the calibration process were noticed and solved, providing experience useful for future runs. A short manual was written on how to change the grid.

A few things can be still tried to enable the parareal to work. The first thing would be to download the latest version of the code and to modify the correction script to ensure compatibility. This alone should help mitigating the discontinuity in the residuals that was indeed observed. Then, a simple coarse with Eirene could be tested. If it is true, in fact, that a reduced grid does not change much the computational time for Eirene, is also true that there is a possible way to circumvent the problem.

It is possible to specify in the Eirene input files how many particles are launched for each strata. With a grid with cells double as big half of the particles would provide the same statistic error, so changing the grid and lowering the number of particles would effectively speed up the run and the performance would not deteriorate. Still, the most promising option is to use the stand-alone version with bigger time-steps as a coarse, because with this combination a $\frac{\tau_F}{\tau_G}$ of seventy or even more is easy to reach. This combination will have to be tried with the proper calibration and positive results can be expected. If even after those precautions the parareal runs crash, there are still more ideas that can be explored.

In the simplest one, again using b2yt, a coarse run with a reduced number of species could be tested by a method called charge bundling. With this approach, all the ionized species for each atom would be considered together as just one charged specie. This could be particularly useful when Tungsten is used, thanks to the fact that it can be ionized a lot of times. This would require a substantial update of the correction code, but it is possible. This new method could be used in connection with the other ones already studied.

Another big improvement for all the cases would be to make the parareal able to recover after the crash of just one slice. For example, the run could converge up to the slice that crashed and automatically start again from that point. This would require to change the python framework and would likely take some time, but would be extremely useful.

Once the parareal part is done nothing prevents to try to couple time parallelization with spatial parallelization, assuring an even higher gain. There is much more to do, but this thesis took some important first steps. Since a typical ITER simulation with SOLPS-ITER can take a few months, a gain by a factor of 3 is also a considerable improvement.

Bibliography

[1] History Database of the Global Environment (HYDE) for the historical data and UN(2008 Revision) for the projections.

[2] *http : //www.opec.org/opec_web/en/publications/*340.*htm* (2015) World Oil Outlook

[3] http://climate.nasa.gov/, official NASA website for climate change

[4] https://www.iter.org/, Official ITER website

[5] D. Reiter, H. Kever, et al. (1991) Helium removal from tokamaks. *Plasma Phys. and Contr. Fus.*, 33:1579.

[6] A. S. Kukushkin, et al. (2011) Finalizing the ITER divertor design: The key role of SOLPS modeling, *Fusion Engineering and Design* 86 (12) 2865.

[7] V. Kotov, D. Reiter, and A. Kukushkin. (2007) Numerical study of the ITER divertor plasma with the B2-EIRENE code package. *FZ-Julich Report JUEL-4257*

[8] Richard A. Pitts, thanks to AndreKukuskin, Philip Andrew, ITER Organisation. Tokamak edge physics and plasma surface interactions, *Centre de Recherches en Physique des Plasma Ecole Polytechnique Fédérale de Lausanne.*

[9] D.P Coster, X. Bonnin, (2015) SOLPS-ITER manual, available with the code.

[10] braginskii, S. I., (1965) Transport processes in a plasma, in Reviews of Plasma Physics, edited by LEONTOVICH, M. A., volume 1, pages 205–311, Consultants Bureau, New York.

[11] Mihailovskii, A. et al. (1984), Transport equations of plasma in a curvilinear magnetic field.

[12] http://www.eirene.de/html/manual.html, User Guide.

[13] http://solps-mdsplus.aug.ipp.mpg.de:8080/solps/Documentation/solps.pdf, User Guide.

[14] P. B. D. Reiter, M. Baelmans (2005) The EIRENE and B2-EIRENE Codes, *Fusion Science and Technology* 47 (2) 172.

[15] G.P. Maddison and D. Reiter. (1994) Recycling source terms for edge plasma fluid models and impact on convergence behaviour in the BRAAMS B2 code. *KFA-Julich Report Jul2872*

[16] X. Bonnin, H. Burbaumer, R. Schneider, et al. (2002) The Two-Mesh Grid Refinement Method for the B2 Code, *Contributions to Plasma Physics* 42 (2) 175.

[17] Notes on the workshop held at Cadarache, (April 2016).

[18] CARRE User Notes, SOLPS-ITER Release Workshop, ITER Organization, User Guide.

[19] Converting or resizing a case: b2yt, SOLPS-ITER Release Workshop, ITER Organization, User Guide.

[20] Triang user notes, SOLPS-ITER Release Workshop, ITER Organization, User Guide.

[21] A.S.Kukushkin (2005) Triangular mesh for Eirene: how to prepare and use it, User Guide.

[22] R. Schneider, et al. (2006) Plasma Edge Physics with B2-Eirene, *Contrib.Plasma Phys.* 46 (1-2) 3.

[23] stangeby, P. et al. (1995), The ion velocity (Bohm-Chodura) boundary condition at the entrance to the magnetic presheath in the presence of diamagnetic and exp drifts in the scrape-off layer.

[24] W. Fundamenski (2007) Scrape Off Layers transport on jet, Euratom, Culham Science Centre.

[25] Christoph Bergmeister, BSc Supervisor Dr. Debasmita Samaddar (2016). Time Parallelization of the SOLPS Code using the Parareal Algorithm.

[26] D. Samaddar, T. Casper, et al. (2010), Application of the parareal algorithm to advanced operation scenario simulations of ITER plasma using the CORSICA code, *Applied Mathematics and Computation (submitted).*

[27] Samaddar, D., Coster, D. P., Bonnin, X. et. all, (2016) Greater than 10x Acceleration of fusion plasma edge simulations using the Parareal algorithm. *Preprint submitted to Journal of Computational Physics*

[28] D. Samaddara, D.P. Costerb, X. Bonninc, C. Bergmeistera (2014) Temporal parallelization of edge plasma simulations using the parareal algorithm and the SOLPS code.