Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Geometric Aspects of Code Coupling in Magnetic Fusion Applications

Ishaan Desai

Computational Science and Engineering (International Master's Program)

Technische Universität München

Master's Thesis

Geometric Aspects of Code Coupling in Magnetic Fusion Applications

Author:	Ishaan Desai
First examiner:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Second examiner:	Prof. Dr. Frank Jenko
First supervisor:	Dr. rer. nat. Benjamin Uekermann
Second supervisor:	Dr. rer. nat. Andreas Stegmeir
Submission Date:	October 28th, 2020

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

October 28th, 2020 Ishaan Desai

Acknowledgments

I wish to deeply thank my supervisors Andreas Stegmeir and Benjamin Uekermann. Andreas made me feel at home at the Institute of Plasma Physics and I will always remember the long fruitful discussions we had over the course of this work. Benjamin has been an inspiration to me not only during this thesis but throughout my Masters. His firm pragmatism and scientific rigor helped me in gaining clarity when I needed it the most.

I recognise the pivotal roles my mother and brother have played in the journey so far. Their support and contribution in my life is of a scale that humbles me.

I am grateful to have friends and colleagues who supported me in every way they could. It is because of them that every endeavour is more enjoyable.

Lastly I thank Datta Dandge for igniting the flame of greater pursuits in me.

Abstract

Nuclear fusion technology is projected to play a major role as a source of clean and safe energy in the future. The immediate challenge is to develop sustainable fusion reactors. In the process of converting complex physical theories to working engineering applications, modelling and simulation assumes a vital position. While simulating nuclear fusion devices, the physical and geometrical complexity arising from different scales and physical regimes needs to be addressed. Specifically for tokamak devices, the regimes are broadly classified into core and edge regions. Simulating both regions in a single software is a laborious task and mostly segregated analysis is pursued.

The edge and core regions can be coupled in a way that the individual analysis remains the same and some form of data communication across a physical boundary takes place. To perform this coupling, a partitioned black-box approach is pursued using the open-source coupling library preCICE. A model diffusion problem is simulated in the edge physics code GRILLIX having a Cartesian grid and a core physics code having a polar coordinate system. The edge region is simulated by the GRILLIX code and the core region is simulated by a custom-built code as a part of this thesis.

A coupling in which the core is modelled with a polar coordinate system and the edge with a Cartesian grid is shown to be first order convergent. Global and local Radial-basis function mapping schemes available in preCICE are tested. A comparative analysis of mapping entities within GRILLIX and doing the same operation with preCICE is shown. In the last part, a strategy for coupling with diverted geometries in cylindrical and curvilinear coordinate systems is presented.

Contents

List of Figures

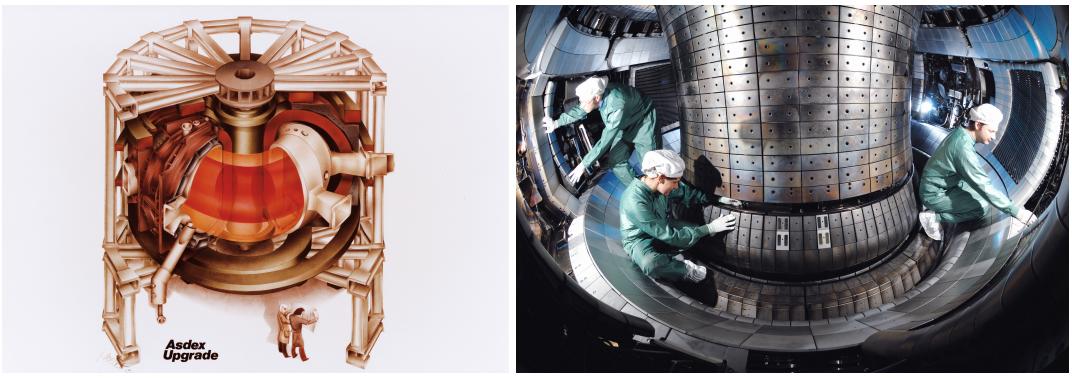
1 Introduction

Simulations play a critical role in modern scientific methods by being the third pillar to theoretical and experimental analysis. Simulation science is becoming even more relevant in fields where experiments on a large scale are becoming increasingly unfeasible. Examples of such fields range from aerospace engineering, large structural systems to applications of nuclear physics in power generation. In the near future fusion reactors are predicted to become reliable machines for power generation [10]. Fusion reactors hold considerable advantages over conventional nuclear fission reactors. Nuclear waste with long decay times is not generated in fusion reactors and they are passively safe to operate. Fusion reactors can be operated with sustainable sources of fuel, for example hydrogen as source and helium as residual ash.

Plasma can be broadly characterized as consisting of charged particles that respond to electromagnetic forces in a collective manner. When light nuclei, like hydrogen, are in an environment of very high temperature (15 million degrees Celsius) there is a possibility of them fusing. At high velocities they may overcome their electrostatic repulsive force and fuse. The entities resulting from the fusion do not have the same rest mass as the constituent colliding atoms. This minimal difference of rest mass when multiplied by the square of the speed of light releases a high amount of energy ($E = mC^2$). A well-known selection of elements for fusion reactors is deuterium (D) and tritium (T) nuclei fusing to form one helium (He) nucleus, one neutron and a high amount of energy. The ionised fuel (deuterium and tritium) and helium ash form the plasma. This plasma can be magnetically confined in a doughnut shaped reactor. The neutron does not have any charge and escapes from the magnetized plasma with a high amount of kinetic energy. The walls of the reactor absorb these neutrons and convert their kinetic energy to heat energy. This heat energy is then used to convert water into steam on which turbines connected to generators are run to produce electricity.

The Max Planck Institute of Plasma Physics (IPP) hosts two experimental facilities, the ASDEX-UPGRADE [8] (Axially Symmetric Divertor Experiment) and WENDELSTEIN-7X [15]. ASDEX-UPGRADE is a divertor tokamak type reactor (fig. 1.1). A tokamak is a doughnut shaped device used for magnetic confinement of plasma in the shape of a torus. WENDELSTEIN-7X is a stellarator type reactor. The experimental branch at IPP is strongly complemented by the development of software and libraries to simulate reactor physics.

This thesis focuses on tokamak style reactors and henceforth when a reactor is mentioned it is assumed to be a tokamak style reactor. Modelling the entire domain of a reactor with



(a) Schematic Diagram (from IPP Online Archive)

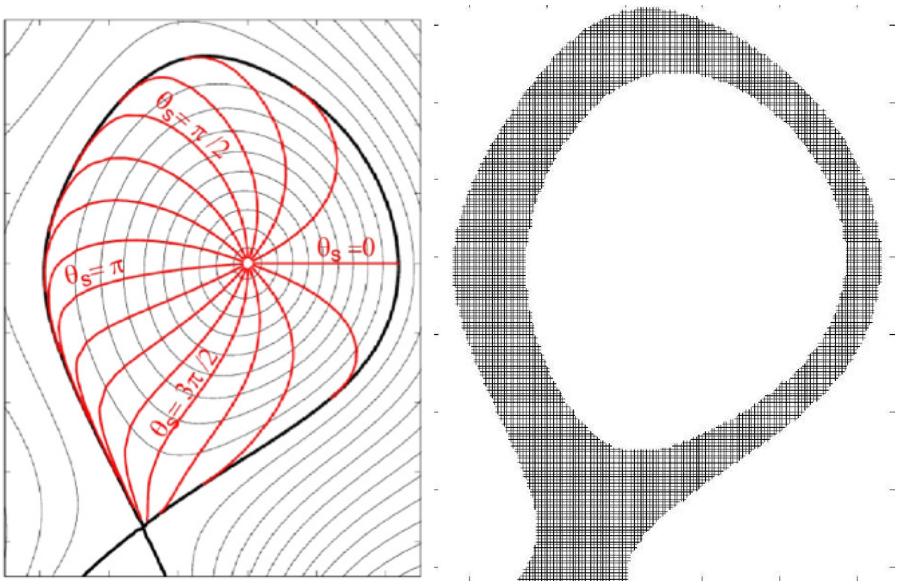
(b) Photo of Torus Chamber (from IPP Online Archive)

Figure 1.1: Schematic Diagram and view of Torus Chamber in ASDEX-UPGRADE

a single set of equations and numerical schemes is not feasible for various reasons. Experiments have shown strong evidence[5] that the plasma close to the walls of the reactor behaves differently than close to the toroidal axis. A surface with normal $\vec{n}$ is defined as a flux surface (fig. 1.3) of a vector field $\vec{B}$ if $\vec{B} \cdot \vec{n} = 0$. It can be deducted from the Hairy-Ball theorem that for a non-vanishing vector field in three dimensions the only possible closed flux surface is a topological toroid. This fact is the fundamental basis behind designing toroidal magnetic confinement devices. For the description of a fusion plasma it is advantageous to use coordinate systems that are aligned to these flux surfaces or even the magnetic field, since the dynamics is strongly anisotropic (fast along magnetic field lines and slow in the perpendicular cross-section). This alignment is characterized as flux-aligned coordinates. In diverted magnetic fusion devices the flux/field aligned coordinates become singular at the separatrix. The separatrix is a flux surface that separates the inner toroidally nested flux surfaces from the outer open flux surfaces that are in contact with target plates. As a broad distinction the entire modelling domain can be differentiated into the *core* and *edge* regions. While there is no exact differentiation of the core and edge regions they can be loosely characterized by the following properties:

The core region exhibits closed flux surfaces, where it is advantageous to use flux-aligned coordinates. However, due to a coordinate singularity this description becomes problematic towards the separatrix, for which a flux-independent description for the edge region has been developed.

The need to have field- or flux-aligned coordinates in the core region and non-aligned coordinates in the edge region is one of the fundamental driving forces of separating the complete domain into these two parts. This differentiation makes it increasingly challenging to simulate the full reactor using a single set of equations and numerical schemes. In collaborations with partners all over the world, scientists at IPP have developed the GENE



(a) Contours of flux coordinates commonly employed in plasma core codes. Solid black line is the separatrix where the field flux aligned coordinate system becomes singular

(b) Cylindrical grid (poloidal cut-section is Cartesian) employed in the edge code GRILLIX

Figure 1.2: Grids in Core and Edge Regions of a Tokamak

(Gyrokinetic Electromagnetic Numerical Experiment) [6] [1] code to simulate regimes with a field-aligned coordinate system. For edge modelling the GRILLIX [13] code is developed which relies on a non-aligned coordinate system and the flux-coordinate independent approach [7]. This thesis explores geometric aspects of coupling these two regions, specifically mapping data across domains having fundamentally different coordinate systems for their geometric representation.

In theory a single software code can be developed to simulate both the core and edge regions, but several limitations come forth. Not all physical models are valid or appropriate in both regions, and modelling the most complex equations and finest mesh resolutions for the entire domain is not feasible from a resource and processing time standpoint. As the core and edge regions need different coordinate systems, the numerical schemes and discretization methods used to solve equations are different in both regions. A possible solution to handle this within a single software is to have two sub packages which handle the regions separately and then join the two packages in some form. This joining can be stated more scientifically as ”coupling” the packages. The physical domain of the problem

[1] http://www.genecode.org/

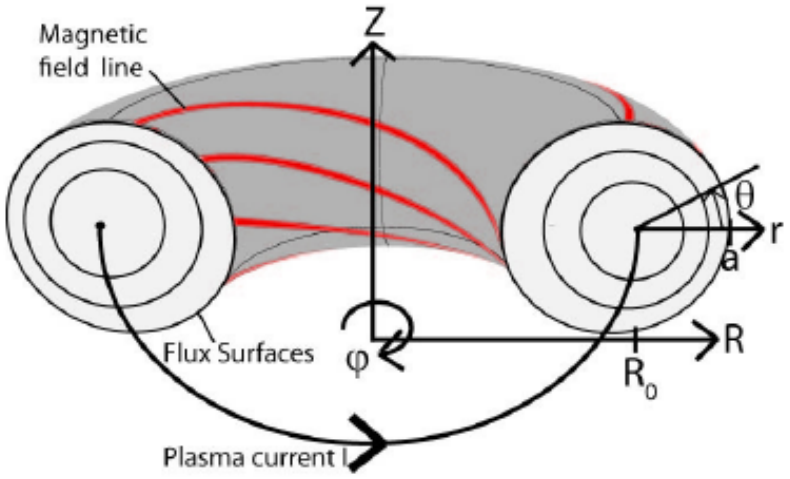


Figure 1.3: Flux Surfaces in a Magnetically Confined Plasma [12]

can be divided into parts and solved individually by different software packages which are then coupled. The connection of the divided domains can be along an interface or a volume. It is not necessary that packages or tools simulating parts of a physical problem need to be under the same roof. Such packages can be completely different and coupling can still be achieved if a sufficiently sophisticated intermediate tool is available. This is especially relevant if the packages already exist as sophisticated software codes concentrating on specific regimes or types of physical scenarios. The use of an interface platform which couples packages without any need for structural modifications to the packages themselves is called *black-box* or *partitioned* coupling. Simulation of a fusion device having core and edge region segregation is a case where such a partitioned coupling strategy can be used to achieve a consistent working model. The codes GENE and GRILLIX already exist and have been developed over many years. Coupling these codes in a black-box manner is a clean and efficient solution as the tedious alternative is to develop a new software from scratch.

In this thesis the partitioned coupling is done by using the open-source library preCICE [1]. A full GENE-GRILLIX coupling framework which can effectively simulate a tokamak device is a much larger and complex problem to solve, which is beyond the scope of this thesis. For this work a simplified model problem, i.e. 2D diffusion perpendicular to magnetic field lines, is selected, and the focus is on the geometric aspects of coupling a prototype core code working in a flux-aligned description with an edge code working in a flux-independent (Cartesian) description. For the core participant a new code is developed from scratch, featuring only the most important geometric aspects of a typical core code, like GENE. For the edge participant, the GRILLIX code is employed, using only a reduced model with respect to its full functionality. Additional modifications to both the edge and core codes are necessary to use the preCICE library and further functionality to map data.

Section 2 presents the preCICE library and some of its features. Additionally the software codes which are coupled to present the geometrical aspects of the coupling are also described in Section 2. Section 3 describes the diffusion equation in two dimensions as the model problem. Section 4 describes the coupling strategies and different stages of modelling towards a realistic coupling. A coupling between a Cartesian grid system and a polar grid system is achieved and verified. Section 5 shows that the developed coupling framework can also be used to couple core and edge regions having diverted geometries. This section is especially important because real world tokamak designs have diverted geometries, which is eventually the motivation for using different geometric descriptions. The last section discusses outlooks and the realm of possibilities for applying partitioned coupling in fusion applications.

2 Introduction to Software Packages

To explore coupling in fusion applications the primary requirements are two packages simulating the core and edge region and a coupling tool to handle the coupling physics. This section describes the coupling library preCICE, the edge modelling software GRILLIX and the custom built code to model the core.

2.1 Coupling library preCICE

preCICE[1] (Precise Code Interaction Coupling Environment) is a library for performing coupled simulations of partitioned multi-physics applications. preCICE being a *library* does not have an executable file of its own. The codes being coupled are modified in a minimal way to include preCICE API function calls. The codes are executed individually which is no different than the case when the codes are being used as solitary packages. After execution preCICE takes over steering and concluding the coupled simulation. Partitioned coupling means that software codes which solve a part of the complete problem are linked without the codes themselves being aware of the coupling. The core algorithm and numerical scheme in each code remains the same. Before starting the simulation each code must define a coupling mesh for preCICE. The coupling mesh is the only part of the entire domain which is visible to preCICE. Within each time iteration, the code must provide values of quantities on the coupling mesh. preCICE reads these values, maps them to the other participant and returns new values on the same coupling mesh. The code has to apply the new values in its data structures. It is important to note that the physical dimensions the coupling mesh on several connected participants have to be same but the mesh itself can be different. The coupling mesh initialization, reading and writing is done via preCICE API function calls. The API functions need to be called at particular instances so that preCICE is initialized and takes over steering. The code snippet below shows a skeleton solver code with the API function calls included.

preCICE handles all communication and data mapping between the codes being coupled. It is common practice to call the codes of a coupled simulation *participants* of coupling. If one or all of the codes are implemented in parallel then peer-to-peer (P2P) communication is done. During initialization it is determined from the coupling mesh information which ranks on either side of the coupling need to communicate further. After this step preCICE works in a fully distributed way. The distributed P2P approach means preCICE scales

well for massively parallel applications. This introduction aims to give a brief insight into preCICE as a versatile coupling library. A host of features and capabilities are available which are beyond the scope of discussion here. Only the features relevant to the coupling pursued later on are discussed in this section.

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  // Define the coupling interface
3  precice::SolverInterface precice("SolverName","config.xml",rank,size);
4
5  int dim = precice.getDimensions();
6  int meshID = precice.getMeshID("CouplingMesh");
7  int vertexSize; // number of vertices on coupling mesh (user defined)
8  double* coords = new double[vertexSize*dim]; // coords of coupling vertices
9  int* vertexIDs = new int[vertexSize];
10 precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);
11
12 int rID = precice.getDataID("ReadVariable", meshID);
13 int wID = precice.getDataID("WriteVariable", meshID);
14 double* readData = new double[vertexSize*dim];
15 double* writeData = new double[vertexSize*dim];
16
17 double dt; // solver timestep size
18 double precice_dt; // maximum precice timestep size
19 precice_dt = precice.initialize();
20 while (not simulationDone()){ // time loop
21     precice.readBlockVectorData(rID, vertexSize, vertexIDs, readData);
22     setReadData(readData);
23     dt = beginTimeStep(); // e.g. compute adaptive dt
24     dt = min(precice_dt, dt);
25     computeTimeStep(dt);
26     computeWriteData(writeData);
27     precice.writeBlockVectorData(wID, vertexSize, vertexIDs, writeData);
28     precice_dt = precice.advance(dt);
29     endTimeStep(); // e.g. update variables, increment time
30 }
31 precice.finalize(); // frees data structures and closes communication channels
32 turnOffSolver();
```

Figure 2.1: Example of Code adapted for coupling using preCICE

API commands and solver commands from the adapted code example have the following functionality:

- `precice("SolverName","config.xml",...)` (line 4): Define the preCICE interface for a solver code with the name *SolverName*. `config.xml` is a configuration file needed by preCICE which defines the quantities to be mapped, method of communication and choice of coupling scheme to be used. An example of a configuration

file is shown later when the coupling setup is discussed.

- `precice.setMeshVertices(...)` (line 13): Define the coupling mesh in preCICE. The mesh defined here is the coupling mesh for preCICE over the course of the simulation.
- `precice.initialize(...)` (line 24): Initializes the coupling interface.
- `precice.readBlockVectorData(...)` (line 26): Read a set of vector data from preCICE. The data returned in the fourth argument `readData` is in the same order as the coupling mesh vertices defined before. A similar command exists to read scalar data (`precice.readBlockScalarData(...)`).
- `setReadData(...)` (line 27): Apply the data read from preCICE to the data structures of the solver.
- `computeTimeStep(dt)` (line 30): Main numerical scheme and updating mechanism of the solver algorithm.
- `computeWriteData(...)` (line 31): Collect modified values on the coupling mesh vertices for writing them to preCICE.
- `precice.writeBlockVectorData(...)` (line 32): Write previously collected values to preCICE. A similar command exists to write scalar data (`precice.writeBlockScalarData(...)`).
- `precice.advance(dt)` (line 33): Advance the coupling by one iteration. This function maps data between coupling meshes and communicates data between coupled solvers.

preCICE is written in C++ and the API is available in `C`, `Python`, `Fortran 90/95` and `Fortran 2003`. In this work the Fortran and Python API are used because to match the parent languages of the codes being coupled. In the above code skeleton two functions: `setReadData(readData)` and `computeWriteData(writeData)` are important because they need to be manually implemented in the code only for the purpose of coupling. These functions can be bundled together with the corresponding preCICE API function calls: `precice.readBlockVectorData()` and `precice.writeBlockVectorData()` respectively to form elaborate functions specifically for coupling cases. In a sufficiently complicated case of coupling, such functions can be made generic and packaged in a way to make them usable for multiple applications. The preCICE project takes this idea further and develops *adapters* for solver codes which are coupled frequently and have a wide range of applications.

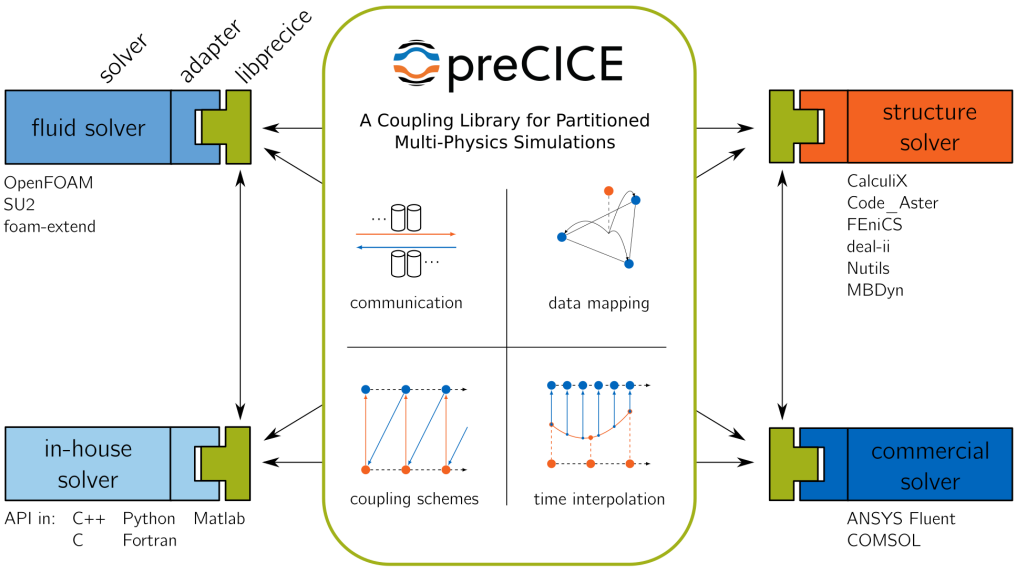


Figure 2.2: Features of coupling library preCICE [1]

Coupling Schemes

preCICE distinguishes between two types of coupling schemes, *explicit* and *implicit*. In explicit coupling both participants are executed once per coupling time step. In an implicit scheme both participants are evaluated multiple times until convergence is achieved within a coupling time step. Furthermore a coupling scheme can be *serial* or *parallel*. In a serial coupling scheme the participants are executed one after the other and in a parallel setting the participants are executed simultaneously. The choice of coupling schemes is critical for numerical stability and accuracy of the coupling. In this thesis only serial coupling schemes are used.

For representation in pseudo code the Core participant is C and the Edge participant is E. Core participant calculates entity c and Edge participant calculates entity e. A serial explicit coupling scheme time loop is shown below. Here n is current time step and nt are the total number of time steps.

```
e = e(0) // initialization
for n = 0 to nt do
    solve C(e) = c
    solve E(c) = e
end for
```

Figure 2.3: Pseudo Code for Explicit Coupling Scheme

In implicit coupling both participants are executed multiple times and a residual (`r`) is calculated. Each time step is re-calculated till the residual drops below a user defined threshold (`eps`). A serial implicit coupling scheme time loop is shown below.

```
while (r > eps) do
    solve C(e) = c
    solve E(c) = e_ref
    r = e_ref - e  // Calculate residual
    e = A(e_ref)  // Acceleration Step
    k = k + 1
end while
```

Figure 2.4: Pseudo Code for Implicit Coupling Scheme

Explicit coupling is computationally less expensive than implicit coupling but is limited in numerical accuracy. In many cases implicit coupling becomes mandatory as explicit coupling cannot guarantee stability. Implicit coupling demands more computational time but produces stable and more accurate results. One important aspect of configuration an implicit coupling scheme is setting up the `Acceleration Step` scheme which is already shown in the above algorithm. Further explanation of implicit coupling can be found in [14].

Data Mapping Methods

Data mapping between coupling meshes is a critical operation in the coupling pipeline and selecting the correct mapping scheme affects numerical accuracy and processing time. The mapping schemes are as follows:

- **Nearest Neighbour Mapping**: Points on the target mesh of a mapping get data values from the closest point on the source mesh. This mapping is first order accurate.
- **Nearest Projection Mapping**: The points of the target mesh are projected onto mesh elements of the source mesh. At the projection point, values from the source mesh are interpolated and then these values are copied to the target mesh. The mesh elements on the source mesh need to be defined by the user. This method is second order if the distance between both meshes is much smaller than the source mesh resolution.
- **Radial-Basis Function Mapping**: In this mapping interpolation of data on non-matching meshes is done using radial-basis functions. A global interpolant is computed on the mesh from which data is being sent. This interpolant is then evaluated on the mesh on which data is being received [9]. The interpolant is formed by linear combination of radially symmetric functions with the sending mesh point at their center. A variant of this mapping which computes the interpolant using local basis

functions is also available. The global variant is computationally more expensive as the mesh resolution increases. Both the variants are evaluated at a later point in this thesis.

Dirichlet and Neumann Coupling

In most coupling applications the data mapping is done along a common boundary. The boundary is a point in one dimension, a curve in two dimensions and a surface in three dimensions. When one participant receives values of quantities from the other participant, they are applied as boundary conditions along the interface. If the received values are the physical values at grid points then applying them is nothing but the equivalent of applying a Dirichlet boundary condition at the interface. If the received values are gradients of the physical values at grid points then applying them is the equivalent of applying a Neumann boundary condition. The participant which receives the Dirichlet values is typically referred to as the Dirichlet participant and similarly there is also a reference to a Neumann participant. If both participants send physical values along the interface then both are Dirichlet participants and the coupling is referred to as a *Dirichlet-Dirichlet* coupling. If one participant sends physical values along the interface and the other sends gradients back, then such a coupling is referred to as a *Dirichlet-Neumann* coupling.

Uni-directional and Bi-directional Coupling

The scenario where one participant sends data and the other participant receives data is known as uni-directional coupling. Uni-directional coupling is often the first step of attempting coupling between two software packages because it helps to adapt the codes and establish a mechanism for collecting data and allocating it along the coupling interface. The scenario where both participants send and receive data is called bi-directional coupling. Each participant of the coupling has to assemble data along the coupling interface and send it to the other participant. Similarly the other participant has to receive this data and send updated data in the reverse direction.

2.2 GRILLIX: Edge Modelling

GRILLIX [13] is a 3D fluid turbulence code based on the flux-coordinate independent approach [7] which enables the simulation of turbulence around the separatrix and in the scrape-off layer regions of a fusion reactor in realistic diverted geometry. Discretization of grid points along magnetic field lines by a field line mapping technique (fig. 2.5) is a hallmark of this software. The use of a Cartesian coordinate grid (non-aligned) in GRILLIX is specifically for simulating of the edge region of a fusion reactor.

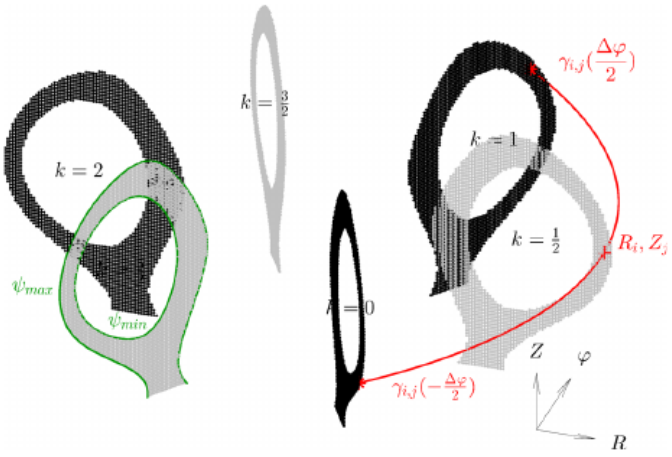


Figure 2.5: Field Line Mapping in GRILLIX [13]

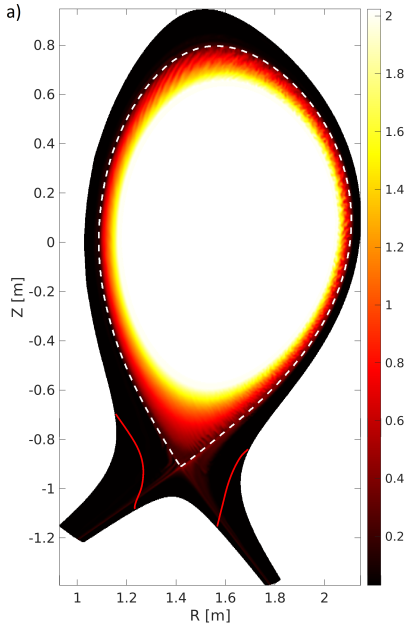


Figure 2.6: Normalised particle Density Plot of GRILLIX Simulation [4]

The figure 2.6 shows a snapshot of density (units are 10^{19} particles per cubic meter) from an ASDEX-UPGRADE simulation done using GRILLIX. The simulation is actually 3D and the snapshot shows a single poloidal plane cut-section. The temperature and hence plasma pressure have a similar shape, high at the core and dropping towards the wall. As the plasma edge is simulated and scrape-off layer is reached, there is a boundary towards the plasma core where both density and temperature are simply prescribed. The fluid

model being is no longer valid inwards and this is generally the coupling boundary later on. As part of this work a model problem is solved using a simplified version of the GRILLIX code in the form of a library. The solver is written as a `Fortran` code which uses functionality from GRILLIX. The full GRILLIX turbulence model is not used directly in this work, but rather a simplified model problem (chapter 3) is solved using finite differences. An example GRILLIX code is shown below:

```
program grillix_schematic
    ! Import functions from PARALLAX Library ---------------
    ...
    ! Define all variables ---------------------------------
    ...
    call MPI_init(ierr)
    ! read parameters from file -----------------------------
    open(unit=33, file = 'simple_params.in', status = 'old')
    ...
    ! select equilibrium (magnetic geometry)
    call create_equilibrium(equi, geometry, 'config.in')
    ...
    ! Build mesh -------------------------------------------
    call mesh%initialize(equi,...)
    ...
    ! set up map and operators -----------------------------
    call create_map_matrix(qmap_minus, ...)
    call create_map_matrix(qmap_plus, ...)
    ...
    ! Initialize the system to a desired value -------------
    call fields%init(equi, mesh,...)
    ...
    ! Time Loop --------------------------------------------
    time=0
    do t = 1, n_t
            ! Advance fields by one time-step
        call timestep(equi, mesh, fields, dt,...)
        time = time + dt

        ! Output field
        if (mod(t,nt_snaps)==0)
            write_snaps(time, fields,...)
        end if
    end do
    call MPI_finalize(ierr)
end program
```

Figure 2.7: Example Code for GRILLIX Software

The code snippet in figure 2.7 shows a typical GRILLIX workflow to simulate equations

in a toroidal geometry. The creation of a equilibrium (line 11) and mesh initialization (line 14) are necessary to define the geometry of the reactor. Map matrices are created (line 17 and 18) for field line mapping. The solution is computed within the time loop (line 27).

2.3 Custom Core Modelling Code

A code simulating the core region of a reactor is necessary for coupling to the edge code GRILLIX. GENE is widely used to simulate the core region relying on field aligned coordinates. With its capability to compute gyro-radius scale fluctuations, GENE is an ideal candidate to simulate the core region of a fusion reactor. However, coupling an edge code directly to GENE is a big step with many complexities. As an alternative to GENE, a code capable of solving equations on a flux-aligned coordinate system grid is developed from scratch. This code is hosted publicly on GitHub[1]. It is envisioned that successful coupling between this custom-built code and GRILLIX lays the foundation for a full GENE-GRILLIX coupling. This thesis focuses initially (chapter 3) on a simplified 2D geometry to study the coupling in a numerically rigorous manner. Toroidicity effects are neglected and flux surfaces are assumed circular, for which flux-aligned coordinates simplify to polar coordinates (r, θ). This code generates a mesh in polar coordinates for a circular cross-section and solves the model problem (chapter. 3) using finite differences on this mesh. The path to realistic toroidal and diverted geometries is presented in chapter 5. An adapted variant of the code which includes all preCICE API function calls is provided to allow for coupling. The custom-built code developed for core region simulation is referred to as **core code** from this point onward.

The core code is written in `Cython` (`Python` + `C`) to utilize the simplicity of `Python` and also maintain the performance of `C` data structures. A modular structure is adopted to promote re-usability and ease of understanding. The main modules of the core code are as follows:

- **Configuration**: Geometric and physical parameters are provided through a user written JSON file. The core code provides the class `Config` whose object is used to access variables from the configuration file. Variable names for coupling such as participant name, name of the data being read and the data being written is also provided through this file.
- **Mesh Generation**: The mesh is generated in the polar coordinate system based on user defined geometric dimensions. The core code works with two dimensions, hence for the polar grid the radial r and angular θ directions are used. The `Mesh` class object generates the mesh and provides various `get_` functions to access mesh details.

[1] https://github.com/IshaanDesai/fusion-core-coupling

- **Boundary Conditions**: Dirichlet and Neumann boundary conditions are available and can be imposed at the physical boundaries of the domain (r minimum and maximum). A boundary condition defined with a certain value can later be modified to a new value. This is essential in coupling.
- **Numerical Scheme**: The polar code is first order Euler explicit in time. A stability condition is derived and checked for the given input parameters. To solve the model problem finite differences are implemented along the grid points. The geometric configuration is periodic in θ direction and the numerical scheme handles this periodicity.
- **Method of Manufactured Solutions**: This module provides functionality to run a validation scheme based on a prospective ansatz function. The workings of this module are explained in detail in a later section on validation of the core code.
- **Output**: Files are output in VTK format for visualization purposes. Data for analysis is available in CSV format.

3 Model Problem: 2D Diffusion in Circular Domain

The aim of this thesis is to investigate the geometrical aspects of coupling in fusion applications. A GENE-GRILLIX coupling would involve turbulence modelling at gyroradius-scale with highly non-linear fluid and kinetic models. These models are critical to simulate the functioning of a fusion reactor but at the same time are very complex itself. To eliminate these complexities and concentrate solely on the geometrical aspects of coupling, the choice of the physical problem is the diffusion model.

Tokamak plasmas are three dimensional anisotropic mediums. The flux surfaces and magnetic field lines in the plasma have a structure that is represented by a toroidal geometry. As explained earlier at edge region of tokamak plasmas is modelled using cylindrical coordinates. Considering a two dimensional cut-section of a tokamak plasma, the toroidal coordinates remain the same and the cylindrical coordinates are converted to Cartesian coordinates. The geometry is further simplified by converting the cut-section to a circular shape and ignoring the toroidicity. This simplifies the toroidal grid to a polar coordinate grid. A two dimensional circular shaped section with polar coordinates in the core region and Cartesian coordinates in the edge region is the model configuration for coupling. As geometric configurations are still different in the core and edge regions, the complexity along the coupling interface is still retained.

$$\partial_t f = Z_\perp \nabla_\perp^2 f, \tag{3.1}$$

where $Z_\perp$ is the diffusion coefficient in the 2D circular section. In this section, the diffusion equation is stated as an initial value problem with prescribed boundary conditions. The diffusion equation is discretized using finite differences. The discretization is formulated on a Cartesian grid and a polar grid. In the last part of this section the model problem is used to verify the custom built core code using the method of manufactured solutions.

The approach used in GRILLIX to solve an equation in a toroidal geometry is to simulate cross-section areas only at certain toroidal angles. Quantities are evaluated along these cross-sections using a field line mapping where finite differences are evaluated in the toroidal direction as shown in the figure 2.5.

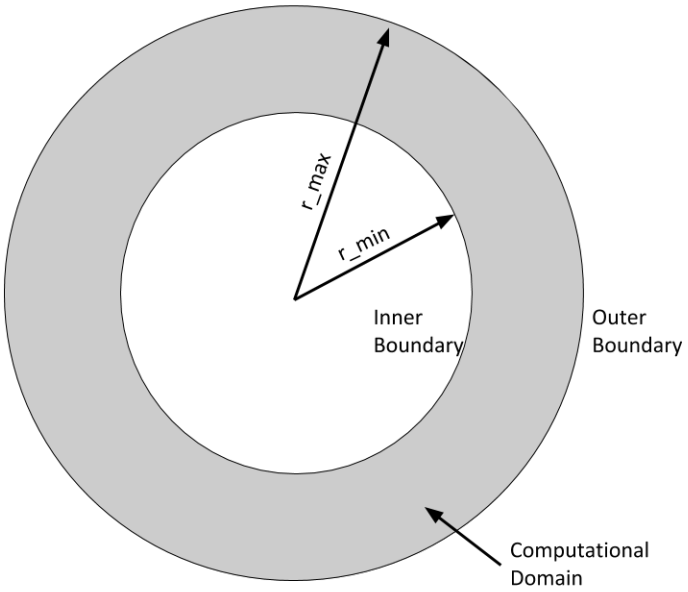


Figure 3.1: Circular Cross-Section for Model Problem

Boundary Conditions

Boundary conditions need to be defined at the inner and outer boundary of the circular cross-section. The outer-most points which are the points along the circumference are referred as *outer* boundary points. Each circular cross-section also has a hole in the middle and the points at the minimum radius are referred to as *inner* boundary points. The boundary points and computational domain are shown in figure 3.1. Dirichlet or Neumann boundary conditions can be imposed on the inner and outer boundaries of the cross-section planes. Dirichlet boundary condition is applied by imposing a specific value at the boundary points. Neumann boundary condition means specifying the flux $\partial_n f$ with respect to the normal n to the boundary surface.

Time Stepping

The diffusion equation is evaluated in time using an explicit Euler scheme. If a time step δt is chosen then the update scheme is as follows:

$$f^{t+1} = f^t + \delta t Z_\perp \nabla_\perp^2 f \tag{3.2}$$

Explicit Euler scheme is first order convergent. A higher order scheme is not chosen because later on only spatial convergence is checked. For an explicit Euler scheme in time stability issues are well known. A CFL condition is devised to check the stability of the numerical scheme:

$$\frac{Z_\perp \cdot dt}{dx^2} \leq \frac{1}{2} \tag{3.3}$$

Here dx is the mesh width. This corresponds to dr or $rd\theta$ if a polar grid is being employed. If a sufficiently small time stepp dt is used it can be guaranteed that the time stepping does not affect overall convergence and stability.

3.1 Discretization in Cartesian Coordinate System

The $\nabla^2_\perp$ operator is evaluated by a second order central difference on a uniform 2D Cartesian grid with a five point stencil. Let $f_{x,y}$ be the value of field f at a point (x, y) on the Cartesian grid. The five point stencil is formulated as follows:

$$\nabla^2_\perp f = \frac{f_{x-1,y} + f_{x+1,y} + f_{x,y-1} + f_{x,y+1} - 4f_{x,y}}{dx^2} \tag{3.4}$$

The simplification of using a single grid spacing dx is followed throughout this thesis as the two dimensional Cartesian grid is a uniform grid.
The update scheme is:

$$f^{t+1}_{x,y} = f^t_{x,y} + \frac{dtZ_\perp}{dx^2}(f^t_{x-1,y} + f^t_{x+1,y} + f^t_{x,y-1} + f^t_{x,y+1} - 4f^t_{x,y}) \tag{3.5}$$

3.2 Discretization in Polar Coordinate System

A two dimensional polar coordinate system has the radial component denoted by r and the angular component denoted by θ. Before the model problem can be discretized in these coordinates, the original diffusion equation has to be transformed from the Cartesian coordinates to the polar coordinates. The coordinate transformation is as follows:

$$r^2 = x^2 + y^2, x = r \cdot \cos\theta, y = r \cdot \sin\theta \tag{3.6}$$

The original diffusion equation in Cartesian coordinates is:

$$\partial_t f = Z_\perp \nabla^2_\perp f = Z_\perp(\partial_{xx} f + \partial_{yy} f) \tag{3.7}$$

The terms $\partial_x\theta$ and $\partial_y\theta$ can be formulated using the original coordinate transformations as:

$$\partial_x\theta = -\frac{\sin\theta}{r} \tag{3.8}$$

$$\partial_y\theta = \frac{\cos\theta}{r} \tag{3.9}$$

$\partial_x f$ and $\partial_y f$ are computed by using the chain rule as follows:

$$\partial_x f = \partial_r f \partial_x r + \partial_\theta f \partial_x \theta = \partial_r f \cos(\theta) - \partial_\theta f \frac{\sin\theta}{r} \tag{3.10}$$

$$\partial_y f = \partial_r f \partial_y r + \partial_\theta f \partial_y \theta = \partial_r f \sin(\theta) + \partial_\theta f \frac{\cos\theta}{r} \tag{3.11}$$

Similarly the second order derivatives $\partial_x^2 f$ and $\partial_y^2 f$ are calculated by applying the product rule to the evaluations of $\partial_x f$ and $\partial_y f$:

$$\partial_x^2 f = \partial_x(\partial_r f \partial_x r + \partial_\theta f \partial_x \theta) = \partial_x \partial_r f \partial_x r + \partial_r f \partial_x^2 r + \partial_x \partial_\theta f \partial_x \theta + \partial_\theta f \partial_x^2 \theta \tag{3.12}$$

Substituting the value of $\partial_x\theta$ from eq. (3.8):

$$\begin{aligned} \partial_x^2 f = \partial_r^2 f \cos(\theta)^2 - 2\partial_r \partial_\theta f \frac{\cos(\theta)\sin(\theta)}{r} + \partial_\theta^2 f \frac{\sin(\theta)^2}{r^2} \\ + 2\partial_\theta \frac{\cos(\theta) sin(\theta)}{r^2} + \partial_r f \frac{\sin(\theta)^2}{r} \end{aligned} \tag{3.13}$$

$$\partial_y^2 f = \partial_y(\partial_r f \partial_y r + \partial_\theta f \partial_y \theta) = \partial_y \partial_r f \partial_y r + \partial_r f \partial_y^2 r + \partial_y \partial_\theta f \partial_y \theta + \partial_\theta f \partial_y^2 \theta \tag{3.14}$$

Substituting the value of $\partial_y\theta$ from eq. (3.9):

$$\begin{aligned} \partial_y^2 f = \partial_r^2 f \sin(\theta)^2 + 2\partial_r \partial_\theta f \frac{\cos(\theta)\sin(\theta)}{r} + \partial_\theta^2 f \frac{\cos(\theta)^2}{r^2} \\ - 2\partial_\theta \frac{\cos(\theta)\sin(\theta)}{r^2} + \partial_r f \frac{\cos(\theta)^2}{r} \end{aligned} \tag{3.15}$$

Substituting the formulations of $\partial_x^2 f$ and ∂_y^2 into equation (3.7):

$$\partial_t f = Z\left(\partial_r^2 f + \frac{\partial_r f}{r} + \frac{\partial_\theta^2 f}{r^2}\right) \tag{3.16}$$

The above equation can be re-written as:

$$\partial_t f = Z(\frac{\partial_r (r f_r)}{r} + \frac{f_{\theta\theta}}{r^2}) \tag{3.17}$$

Discretized form of the equation is formulated by using finite differences over a staggered grid. If $f^t_{r,\theta}$ is value of a the field variable f on point (r, θ) at time t, the discrete form of the equation is as follows:

$$\partial_t f = Z(\frac{1}{r_{r,\theta} dr}([r\partial_r f^t]_{r+\frac{1}{2}} - [r\partial_r f^t]_{r-\frac{1}{2}}) + \frac{f^t_{r,\theta+1} + f^t_{r,\theta-1} - 2f^t_{r,\theta}}{r^2 d\theta^2}) \tag{3.18}$$

$$\begin{aligned} f^{t+1}_{r,\theta} = f^t_{r,\theta} + dtZ\Bigg(&\frac{1}{r_{r,\theta} dr}([r_{r+\frac{1}{2}}\partial_r f^t_{r,\theta}] - [r_{r-\frac{1}{2}}\partial_r f^t]) \\ &+ \frac{f^t_{r,\theta+1} + f^t_{r,\theta-1} - 2f^t_{r,\theta}}{r^2 d\theta^2}\Bigg) \end{aligned} \tag{3.19}$$

The update scheme is:

$$\begin{aligned} f^{t+1}_{r,\theta} = f^t_{r,\theta} + dtZ\Bigg(&\frac{1}{r_{r,\theta} dr}([r_{r+\frac{1}{2}} \frac{f_{r+1,\theta} - f_{r,\theta}}{dr}] - [r_{r-\frac{1}{2}} \frac{f_{r,\theta} - f_{r-1,\theta}}{dr}]) \\ &+ \frac{f^t_{r,\theta+1} + f^t_{r,\theta-1} - 2f^t_{r,\theta}}{r^2 d\theta^2}\Bigg) \end{aligned} \tag{3.20}$$

The gradients specified at the Neumann boundary points are incorporated into the internal grid point values by a second order evaluation. The evaluation is shown below for a polar coordinate system. A similar evaluation for Cartesian coordinates can be derived but for this work, flux computations only on the polar grid are relevant. Second order evaluations of gradients are computed from the Taylor expansions of two neighbouring grid points in the interior of the computational domain. If dr is the grid spacing in the radial direction then calculation for outer boundary is as follows:

$$f_{r-dr,\theta} = f_{r,\theta} - drdf_{r,\theta} + \frac{dr^2}{2} d^2 f_{r,\theta} + O(dr^3)$$
$$f_{r-2dr,\theta} = f_{r,\theta} - 2drdf_{r,\theta} + \frac{dr^2}{2} d^2 f_r + O(dr^3)$$

The second order term $d^2 f_r$ can be eliminated and the equations reformulated to get the value at the boundary point ($r = r_{\max}$):

$$f_{rmax,\theta} = \frac{4}{3} f_{rmax-dr,\theta} - \frac{1}{3} f_{rmax-2dr,\theta} + \frac{2}{3} dr df_{rmax,\theta} \tag{3.21}$$

Analogously the update scheme for inner boundary is as follows:

$$f_{r+dr,\theta} = f_{r,\theta} + dr df_{r,\theta} + \frac{dr^2}{2} d^2 f_{r,\theta} + O(dr^3)$$
$$f_{r+2dr,\theta} = f_{r,\theta} + 2dr df_{r,\theta} + \frac{dr^2}{2} d^2 f_{r,\theta} + O(dr^3)$$

Analogously the evaluation done for outer boundary points leads to the update scheme:

$$f_{rmin,\theta} = \frac{4}{3} f_{rmin+dr,\theta} - \frac{1}{3} f_{rmin+2dr,\theta} - \frac{2}{3} dr df_{rmin,\theta} \tag{3.22}$$

3.3 Verification of Custom Core Code

The method of manufactured solutions [11] is used to verify the core code. In this method an ansatz function is chosen to represent the field at every point in the domain and at every instant of time. For this case the ansatz function is of the type $f = F(r, \theta, t)$. The ansatz function is chosen in such a way that it is easy to differentiate with all variables. The ansatz is then inserted into the differential equation and the result is stored as a source term S_{mms}. The numerical scheme is evaluated by choosing the initial state and the boundary conditions according to the ansatz function. The source term is added to the updated field values in each time iteration. At the end of the simulation the original ansatz function is evaluated at every grid point and compared to the result of the numerical scheme. The error between these two values is compared for different mesh resolutions to determine the convergence order of the numerical scheme. A common choice for ansatz functions is a combination of trigonometric entities $\sin()$ and $\cos()$ which are easy to differentiate and evaluate. sin and cos are periodic and hence have bounded limiting values which is also helpful in computations. The choice made for this case is as follows:

$$f = \sin(A)\cos(t)\cos(\theta) \tag{3.23}$$

where,

$$A = 2\pi \frac{r - r_{min}}{r_{max} - r_{min}} \tag{3.24}$$

radial points	theta points	dr	dtheta	dt
50	262	6.0E-03	6.0E-03	5.0E-04
100	524	3.00E-03	3.00E-03	1.25E-04
200	1048	1.50E-03	1.50E-03	3.125E-05
400	2096	7.50E-04	7.49E-04	7.8125E-06

Figure 3.2: Parameter Values for Core Code Convergence Study

The diffusion equation is a homogeneous partial differential equation, so the source term S_{mms} can be easily evaluated by moving all partial derivatives to one side.

$$S_{mms} = \partial_t f - Z(f_{rr} + \frac{f_r}{r} + \frac{f_{\theta\theta}}{r^2}) \tag{3.25}$$

Inserting the above ansatz into the source term formulation, the source term is determined as:

$$\begin{aligned} S_{mms} = &- \sin(A)\sin(t)\cos(\theta) + Z\cos(t)\cos(\theta)\left(\sin(A)(\partial_r A)^2\right. \\ &\left. - \frac{\cos(A)\partial_r A}{r} + \frac{\sin(A)}{r^2}\right) \end{aligned} \tag{3.26}$$

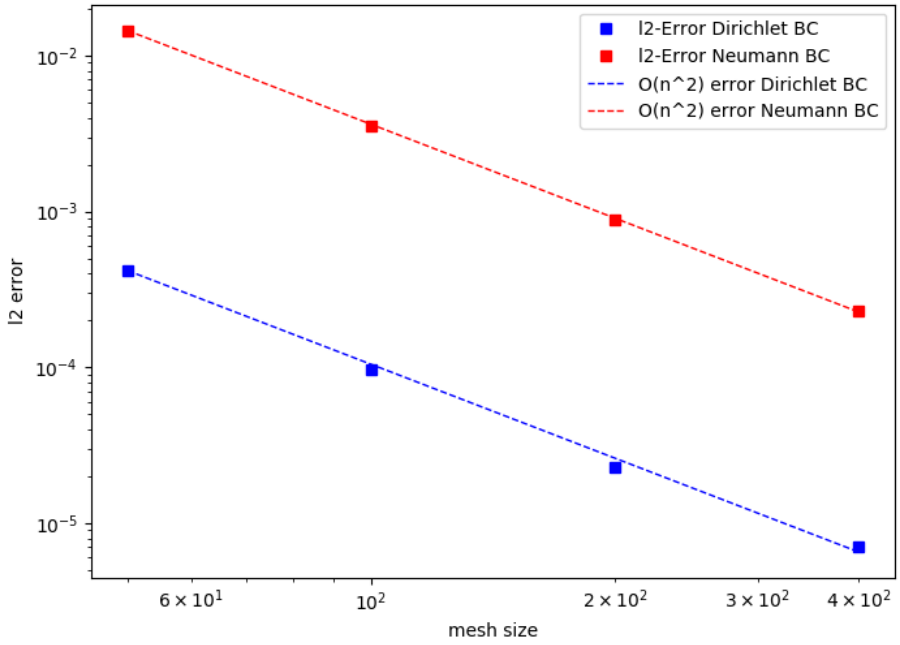
Initial condition of the ansatz problem is obtained by inserting $t = 0$ in eq.3.23:

$$f_{init} = \sin(A)\cos(\theta) \tag{3.27}$$

Evaluation of the initial condition at boundary points ($r = r\_min$ and $r = r\_max$) is $f = 0$. The relative l^2 error between the ansatz solution and solution obtained by the discretization scheme is compared. Let t_n be the total number of time steps calculated, $f^A_{r,\theta,t}$ be the ansatz function evaluation at a certain r, θ and t and correspondingly $f^D_{r,\theta,t}$ be the solution of the numerical scheme. The error is computed as follows:

$$l^2 Error = \sqrt{\frac{\sum_{r,\theta} |f^A_{r,\theta,t_n} - f^D_{r,\theta,t_n}|^2}{\sum_{r,\theta} |f^D_{r,\theta,t_n}|^2}} \tag{3.28}$$

The error is computed for various mesh sizes (fig. 3.2) to check for the order of convergence. The mesh size in the plot (fig. 3.3) is the number of radial points. The mesh size is only an indicating factor that the grid is refined by a factor of two in each direction. To capture second order spatial convergence the time step is reduced by a factor of four when the grid is refined by a factor of two. This needs to be done because the time stepping is



Mesh size are the number of grid points along one axis. For each experiment the mesh size is refined by a factor of two and time step is refined by a factor of four.

Figure 3.3: Convergence Order (spatial) for Verification of Custom Core Code

explicit Euler scheme which is first order convergent. The custom core code developed as part of this thesis is shown to be second order convergent in space. The numerical scheme used to discretize and solve the diffusion equation is also second order accurate. The verification of the core code is complete.

4 Polar-Cartesian Geometry Coupling

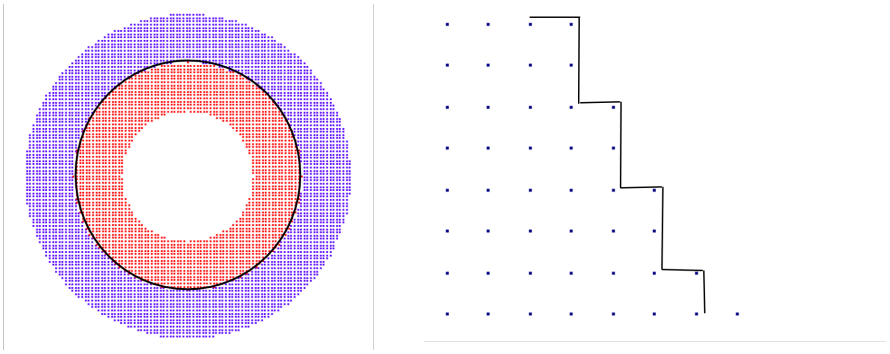
The edge code and core codes to be used in coupling are fully described at this point. The core code is built from scratch and verified in the previous section. These geometric configurations are selected such that the edge region of a reactor is modelled using non-aligned coordinates and the core region is modelled using flux-aligned coordinates. The edge participant is simulated using a Cartesian coordinate system and the core participant is simulated using a polar coordinate system. The primary tasks before a Polar-Cartesian coupling can be done are adapting both core and edge codes to use the preCICE library and formulating methods to map data between the codes when coupling is going on.

In the first step 4.1 a Cartesian-Cartesian coupling is performed. In this coupling both the core and edge participants are simulated with GRILLIX using a uniform Cartesian grid. Similar geometric configurations are naturally easier to couple, but several intricacies still arise and are discussed below. An additional purpose of this coupling is to adapt the GRILLIX code for coupling with preCICE. Functions for collecting data to be written to preCICE and assigning data which is obtained from preCICE are written and discussed. Using the functionality developed in the Cartesian-Cartesian coupling, a polar-Cartesian coupling is constructed. This coupling is the first step in coupling participants with non-similar geometric configurations and hence in this chapter this coupling is studied in greater detail.

4.1 First Step: Cartesian-Cartesian Coupling

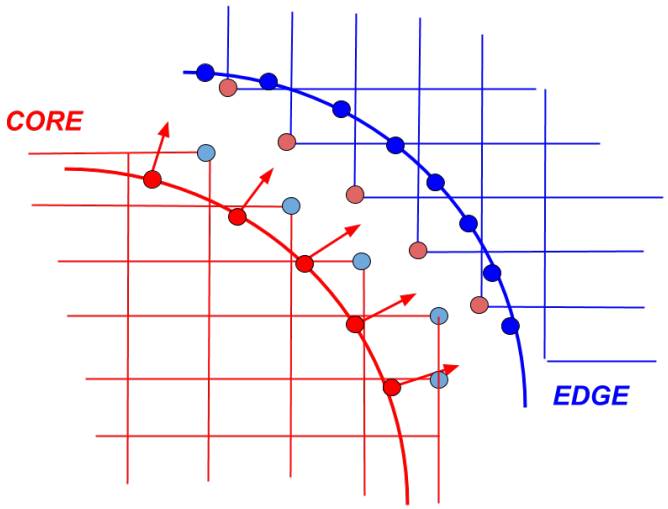
As a first step in setting up a coupling model a uni-directional coupling is implemented. In a uni-directional coupling the core sends Dirichlet values to the edge which are applied as boundary conditions on the edge. An interesting finding of the uni-directional coupling is that a circular cross-section when represented by a Cartesian grid produces a step style representation of a circle (fig. 4.1). This introduces an error because it is unclear from which grid points along this step shape should values be read and to which grid points should values be written. Using the nearest-neighbor mapping is not accurate enough as there is a loss of data due to the uncertainty of the step shape. This problem is countered by introducing an additional circular shaped grid on top of the Cartesian grid. This circular grid has a width of a single grid point and is used only for data mapping purposes.

Based on the insights from uni-directional coupling a more realistic bi-directional coupling is implemented. The problem of having a step style shaped boundary at the coupling



(a) Core and edge grids (b) Step-grid at coupling boundary

Figure 4.1: Mesh Details of Cartesian-Cartesian Coupling



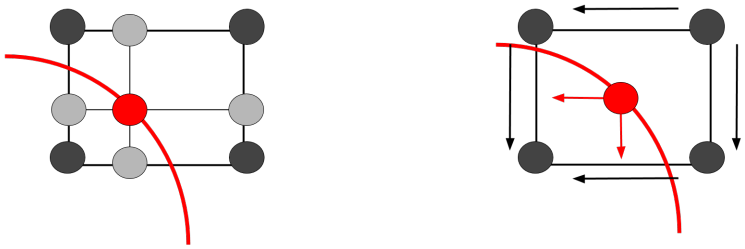
Red and blue circular meshes are actually at the same physical location but are shown separately here to clarify the use of circular mesh. Red arrows are the fluxes mapped from the core to the light red points boundary points of the edge. Blue points on the edge are values mapped to the light blue boundary points of the core.

Figure 4.2: Use of additional Circular Mesh in Cartesian-Cartesian Coupling

interface is described in unidirectional coupling. An additional circular mesh having only one grid point width is added at the boundary of the edge. This circular mesh is used purely for data mapping and does not interfere with the Cartesian grid of the edge on

which the numerical scheme is solved. The bi-directional coupling is a Dirichlet-Neumann coupling. The data mapping (fig. 4.4) in one time window is done by the following steps:

- **Step 1**: The core participant solves the numerical scheme and updates values on the grid points of its Cartesian mesh. If this is the first time step then the values at the grid points are initialized by a pre-defined initial state. Gradient values are computed for every cell of the Cartesian grid on the coupling boundary by applying finite differences to the nodal values at the four corner grid points (fig: 4.3). The points on the circular mesh which are lying in this cell are identified and the gradient values are copied to those points.
- **Step 2**: The fluxes on the circular mesh of the core are mapped to the Cartesian grid points of the edge using radial-basis function (RBF) mapping. The mapped fluxes are applied as a Neumann boundary condition on the coupling boundary points of the edge. The mapped fluxes are used to compute nodal values at the boundary points of the edge. The edge participant then solves the numerical scheme and updates values on all the grid points of its Cartesian mesh.
- **Step 3**: The values from the Cartesian mesh of the edge are interpolated on the circular mesh of the edge using bilinear interpolation (fig. 4.3). In this bilinear interpolation, four corner values of a cell of the Cartesian grid are used to obtain the value on the circular mesh point which lies in this cell.
- **Step 4**: The values on the circular mesh of the edge are mapped to the Cartesian grid points of the core using RBF mapping. The mapped values are applied as a Dirichlet boundary condition on the core mesh boundary points. The core participant can now solve the numerical scheme again for the next time step.



(a) Bilinear interpolation from Cartesian grid to circular mesh

(b) Finite difference approximation of fluxes from the Cartesian mesh to circular mesh

Figure 4.3: Interpolation between Cartesian and Circular Mesh

Each operation described in the steps above is done by developing additional functional-

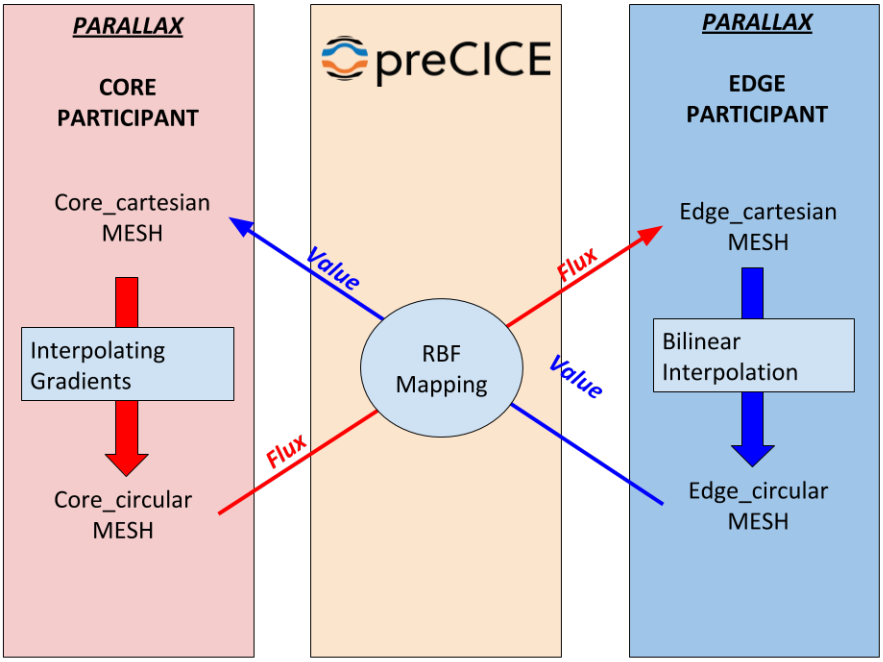


Figure 4.4: Data Mapping Mechanism for Cartesian-Cartesian Coupling

ity on top of the original model problem code. The radial-basis function (RBF) mapping is done via preCICE but the internal mapping between the Cartesian mesh and circular mesh of a participant is done within the coupled codes. The internal mapping of fluxes is done using first order finite differences and the internal mapping of values is done using bilinear interpolation. The RBF mapping is done using preCICE and is configured in the XML configuration file. Radial-basis functions of type thin-plate-splines are used for mapping. In the code snippet below the mapping is defined from the perspective of the core participant. The core participant writes fluxes from its circular mesh to the Cartesian mesh of the edge using RBF mapping. The core then reads values from the circular mesh of the edge to its own Cartesian mesh again using RBF mappings. The same mapping configuration can be done from the perspective of the edge participant too.

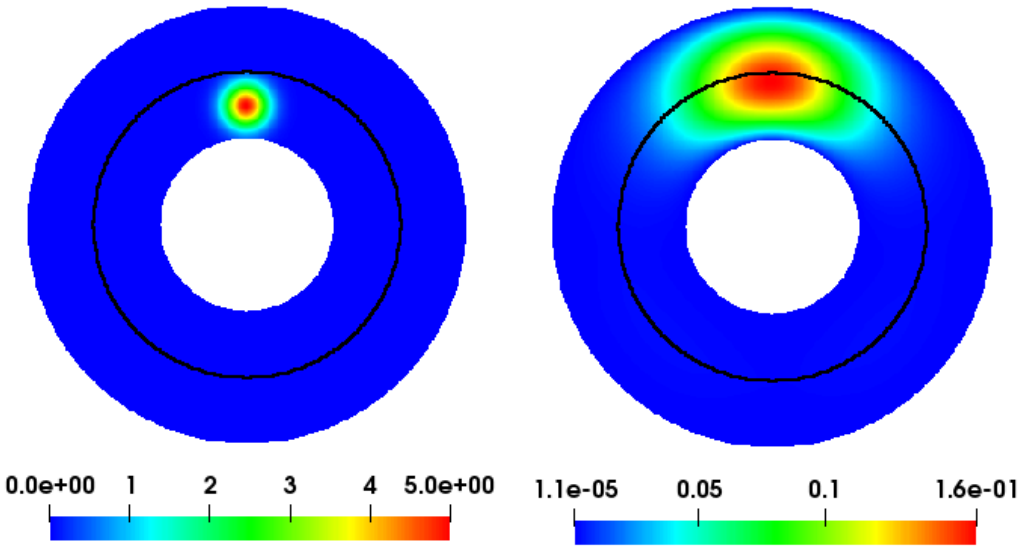
The data mapping scheme shown above is just one combination of using the Cartesian and circular meshes on either side. With two entities (values and fluxes) being mapped using two meshes (Cartesian and circular) on either side there are several possible combinations of data mappings. It is not mandatory that the edge participant transfers values to the core and the core sends back fluxes. Transferring fluxes from edge to core and values from core to edge is also a valid mapping configuration and is shown to be a preferred combination in the next section.

```
<participant name="Core">
  <use-mesh name="core_cartesian" provide="yes"/>
  <use-mesh name="core_circular" provide="yes"/>
  <use-mesh name="edge_cartesian" from="Edge"/>
  <use-mesh name="edge_circular" from="Edge"/>
  <write-data name="flux" mesh="core_circular"/>
  <read-data name="value" mesh="core_cartesian"/>
  <mapping:rbf-thin-plate-splines direction="write"
    from="core_circular" to="edge_cartesian"
    constraint="consistent"/>
  <mapping:rbf-thin-plate-splines direction="read"
    from="edge_circular" to="core_cartesian"
    constraint="consistent"/>
</participant>

<participant name="Edge">
  <use-mesh name="edge_cartesian" provide="yes"/>
  <use-mesh name="edge_circular" provide="yes"/>
  <read-data name="flux" mesh="edge_cartesian"/>
  <write-data name="value" mesh="edge_circular"/>
</participant>
```

Figure 4.5: Configuration of Mapping in preCICE for Cartesian-Cartesian Coupling

There are various ways to perform mapping between core and edge. Availability of several combinations show that the coupling mechanism developed is versatile and there is scope to use better interpolation techniques and mapping methods to improve performance and numerical accuracy. The flexibility of preCICE allows the user to switch between mappings on different meshes with little effort.
The model problem is simulated using the bi-directional Cartesian-Cartesian coupling. A Gaussian blob is initialized in the core domain and the diffusing field is observed through the coupled core and edge domains. Zero valued Dirichlet boundary conditions are applied at the inner boundary of the core and outer boundary of the edge. The coupling interface is marked in a black circle over the field images (fig. 4.6) to show that the blob diffuses smoothly over the interface.

The intention of performing a Cartesian-Cartesian coupling as a first step is to establish the problems in data mapping and provide mechanisms to address these problems. This coupling also provided the foundation of writing code with PARALLAX which includes the preCICE API. A working diffusion model for this coupling is considered as sufficient evidence of working framework and no further analysis is done. The primary objectives of the Cartesian-Cartesian coupling are satisfied.



(a) t = 0 (b) t = 1

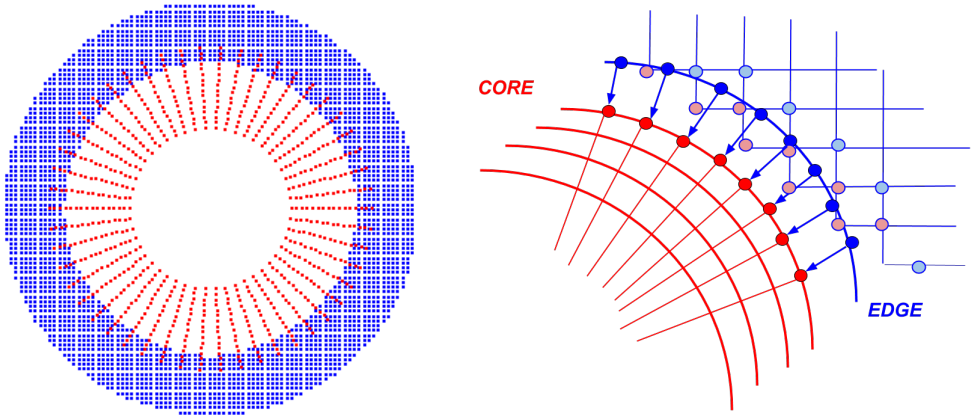
Black line is the coupling interface. The Gaussian blob diffuses from the core to the edge

Figure 4.6: Diffusion of Gaussian Blob in Cartesian-Cartesian Coupling

4.2 Mapping Data in Polar-Cartesian Configuration

The Cartesian-Cartesian coupling done in the previous section is a coupling between similar geometric configurations. In spite of the similarity an additional circular mesh is necessary to map data accurately. The additional circular mesh defined for data mapping purposes is already a step in the direction of a Polar-Cartesian coupling. The circular mesh captures the characteristic shape of the original geometry to ensure data is transferred consistently at the coupling interface. It is already discussed in the introduction that the core region of a reactor can be modelled by field-aligned coordinates which are aligned not only to the flux surfaces but also resemble the physical cross-sectional geometry. The core region is now modelled by the custom built polar core code which uses a polar coordinate system (fig. 4.7).

The edge participant is still modelled with the code used in the Cartesian-Cartesian coupling. An additional circular mesh is defined for the edge participant which is used only for data mapping purposes. The main advantage of having a polar grid in the core is that the coupling boundary of the core is now perfectly aligned with the circular mesh of the edge. By using matching meshes the grid points on both these meshes can be perfectly aligned and data can be transferred in an accurate way. Figure 4.7 (right figure) shows the data mapping implemented in polar-Cartesian coupling. The following steps are executed



(a) Polar grid in the core and Cartesian grid in the edge. The Cartesian grid is a uniform grid. There is an overlap between core and edge regions which is explained later on.

(b) The outermost layer of the core mesh is at the same physical location as the circular mesh of the edge but is shown separately for representation. Arrows from the edge to core are fluxes being mapped and light red points on the edge are the values mapped from the core

Figure 4.7: Grid and Mapping Mechanism in Polar-Cartesian Coupling

in one time window:

- **Step 1**: The edge participant solves the numerical scheme and updates values on the grid points of its Cartesian mesh. If this is the first time window then the values at the grid points are initialized by a predefined initial state. Gradient values are computed for every cell of the Cartesian grid on the coupling boundary by applying finite differences to the nodal values at the four corner grid points. The points on the circular mesh which are lying in this cell are identified and the gradients values are copied to those points.
- **Step 2**: The fluxes on the circular mesh of the edge are directly mapped to the coupling boundary points of the polar mesh of the core using nearest-neighbor mapping. The mapped fluxes are applied as a Neumann boundary condition on the coupling boundary points of the core participant. The mapped fluxes are used to compute nodal values at the boundary points of the core. The core participant then solves the numerical scheme and updates values on all the grid points of its polar mesh.
- **Step 3**: The new values on the coupling boundary of the polar mesh of the core participant are mapped to the boundary points of the Cartesian mesh of the edge using radial-basis function mapping. The edge participant can now solve the numerical scheme again for the next time step.

The mapping of values from core to edge is done in the same way as for Cartesian-Cartesian coupling the difference being that now the core has a polar mesh which can directly provide values on a circular shaped mesh for the RBF mapping. For the steps described above the preCICE configuration for data mapping looks as follows:

```
<participant name="Core">
  <use-mesh name="core_polar" provide="yes"/>
  <use-mesh name="edge_cartesian" from="Edge"/>
  <use-mesh name="edge_circular" from="Edge"/>
  <write-data name="value" mesh="core_write_mesh"/>
  <read-data name="flux" mesh="core_read_mesh"/>
  <mapping:rbf-thin-plate-splines direction="write"
    from="core_polar" to="edge_cartesian"
    constraint="consistent"/>
  <mapping:nearest-neighbor direction="read" from="edge_circular"
    to="core_polar" constraint="consistent"/>
</participant>

<participant name="Edge">
  <use-mesh name="edge_cartesian" provide="yes"/>
  <use-mesh name="edge_circular" provide="yes"/>
  <read-data name="value" mesh="edge_cartesian"/>
  <write-data name="flux" mesh="edge_polar"/>
</participant>
```

Figure 4.8: Configuration of Mapping in preCICE for Polar-Cartesian Coupling

Once Polar-Cartesian coupling framework is finalized, some form of verification is necessary. To predict the spatial convergence order of the coupling the individual orders of all mapping schemes and the coupling participants themselves are taken into consideration. Verification of participants as standalone codes and literature supporting the mapping schemes give the following details:

- **Standalone Edge Code**: Edge code uses the PARALLAX library. Dirichlet and Neumann boundary conditions in PARALLAX are first order convergent. Hence the overall order of the edge participant is first order.
- **Standalone Core Code**: In Chapter 3 the custom built core code is verified to be second order convergent for Dirichlet and Neumann boundary conditions.
- **Mapping Fluxes within Core**: Flux values on points of the circular mesh of the edge are computed by first order finite differences on the Cartesian grid point values. This internal computation in the core code is first order convergent.
- **Mapping Fluxes from Edge to Core**: The fluxes computed on the circular mesh of the edge are mapped to the boundary of the core using nearest-neighbor mapping. Nearest-neighbor mapping is second order convergent if mapping meshes are used.

- **Mapping Values from Core to Edge**: Values are mapped from core to edge using radial-basis function mapping which is second order convergent [9].

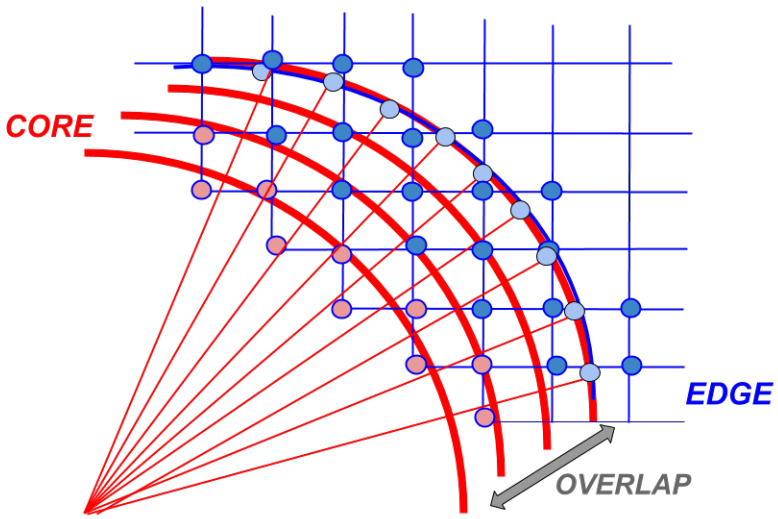
The overall coupling convergence is restricted to first order as the edge participant and several mapping schemes are first order. The boundary conditions of the edge code and the first order mapping schemes can be improved to higher order schemes. This is discussed later on as part of future work.
Both the edge and core codes are explicit in time. In the update scheme the right hand side consists of entities which are only from the previous time step. Solving such a fully explicit algorithm using an implicit coupling scheme does not make sense as every implicit time iteration will contribute to no improvement in the state of the solution. Hence an explicit coupling scheme is used for polar-Cartesian coupling.

4.3 Data Initialization and Overlapping Domains

For a Dirichlet-Neumann coupling, initialization of boundaries to zero values and zero fluxes does not work. preCICE offers functionality to initialize the quantities along the coupling boundary to non-standard values. If this functionality is not used then the default initialization values are zero. One of the participants starts the simulation. This participant is called the *first* participant. Normally it starts with zero values initialized at the coupling boundary. If an initial state is prescribed then the zero value initialization does not work. In such cases the second participant defines values and writes them to preCICE. preCICE supplies the values to the first participant before it starts.

Overlapping the domains of the two participants can be understood by studying the data accessing and updating of one of the participant. In the scenario of this work, two participants having fully explicit time stepping are being coupled with a Dirichlet-Neumann coupling and an explicit coupling scheme. Considering the core participant: as soon as the time loop starts data (Neumann fluxes) is read from preCICE and applied as boundary conditions along the interface points. Then the numerical scheme is solved and all internal grid points (excluding interface boundary points) are updated to the new values for the next time step. Note that the boundary points still have the values of the old time step. If data (Dirichlet values) is now collected from the boundary points of the core and written to preCICE, then the old time step values would be written back to the edge. This is incorrect as the edge participant expects updated values along its interface points.

Overlapping the domains (fig. 4.9) by a certain amount can resolve this problem. When domains are overlapped, the data transfer happens at two interfaces rather than a single one. Considering the data transfer from the core to the edge, the boundary values needed for the edge are now generated using internal grid point data of the core. The internal grid point values are updated by the numerical scheme and hence the data collected is already updated to new values. The overlap width is selected to be two times the mesh width



Edge reads data on red points of its Cartesian mesh. Core reads data on light blue points of its polar mesh

Figure 4.9: Overlapping Domains for Polar-Cartesian Coupling

of the edge participant. This is done because during the flux computation on the edge participant, entire cells are necessary as the flux on the $edge\_circular$ mesh points is calculated using four corner values of the encompassing cell. These cells need to be completely isolated from the boundary points of the edge which are updated using the data received from the core. An overlap width of two times the mesh width of the edge Cartesian grid ensures this and hence is a natural choice. When the domains are overlapped the core code has to maintain two coupling meshes, one for reading data and one for writing data. The overlapped domain is solved by both participants and data exchange takes place at the two boundaries of this overlapping domain. The configuration for preCICE is then as follows:

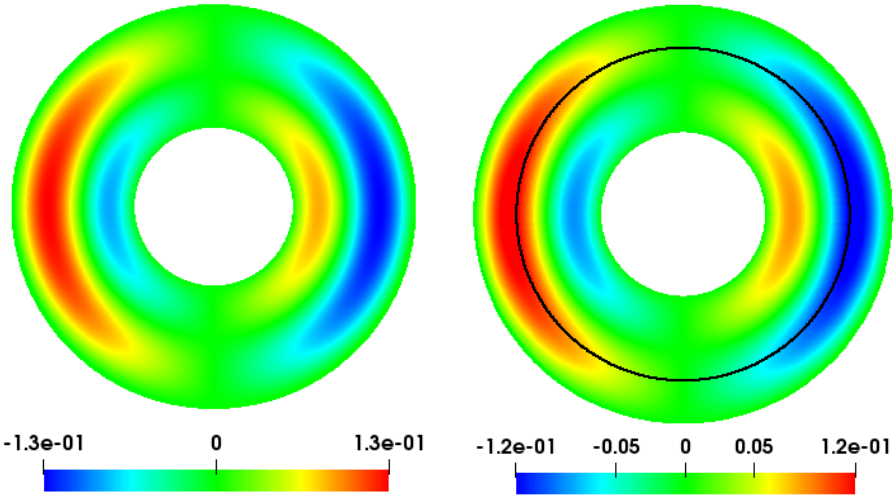
```
<participant name="Core">
  <use-mesh name="core_read_mesh" provide="yes"/>
  <use-mesh name="core_write_mesh" provide="yes"/>
  <use-mesh name="edge_cartesian" from="Edge"/>
  <use-mesh name="edge_circular" from="Edge"/>
  <write-data name="value" mesh="core_write_mesh"/>
  <read-data name="flux" mesh="core_read_mesh"/>
  <mapping:rbf-thin-plate-splines direction="write"
    from="core_write_mesh" to="edge_cartesian"
    constraint="consistent"/>
  <mapping:nearest-neighbor direction="read" from="edge_circular"
    to="core_read_mesh" constraint="consistent"/>
</participant>

<participant name="Edge">
  <use-mesh name="edge_cartesian" provide="yes"/>
  <use-mesh name="edge_circular" provide="yes"/>
  <read-data name="value" mesh="edge_cartesian"/>
  <write-data name="flux" mesh="edge_circular"/>
</participant>
```

Figure 4.10: Configuration of Mapping in preCICE with Overlapping Domains

4.4 Verification of Polar-Cartesian Coupling

The Polar-Cartesian coupling framework is verified using a strategy similar to the earlier explained method of manufactured solutions. The core and edge participants are initialized to an ansatz function of the form $f = sin(r)cos(\theta)cos(t)$ (fig. 4.11). The discretized form of the diffusion equation is solved up to $t = 0.5$ for the coupled problem with varying mesh resolutions. In the method of manufactured solutions a source term is calculated by plugging the ansatz in the original equation but here a different strategy is pursued. The source term would need to be calculated for the edge and core regions separately and plugged into the respective update schemes. To avoid the increase in computation time the coupled solution is compared to a reference solution. The reference result is obtained by simulating the entire domain using the core code at a very fine mesh resolution (*1.3 Million grid points*). The solution of the reference result is interpolated on the grid points of the coupled solution to be compared using cubic splines. The error is calculated by taking the l^2 norm of the difference between the solution and reference at each grid point. The error is plotted against mesh resolutions to show first order convergence (fig. 4.12).

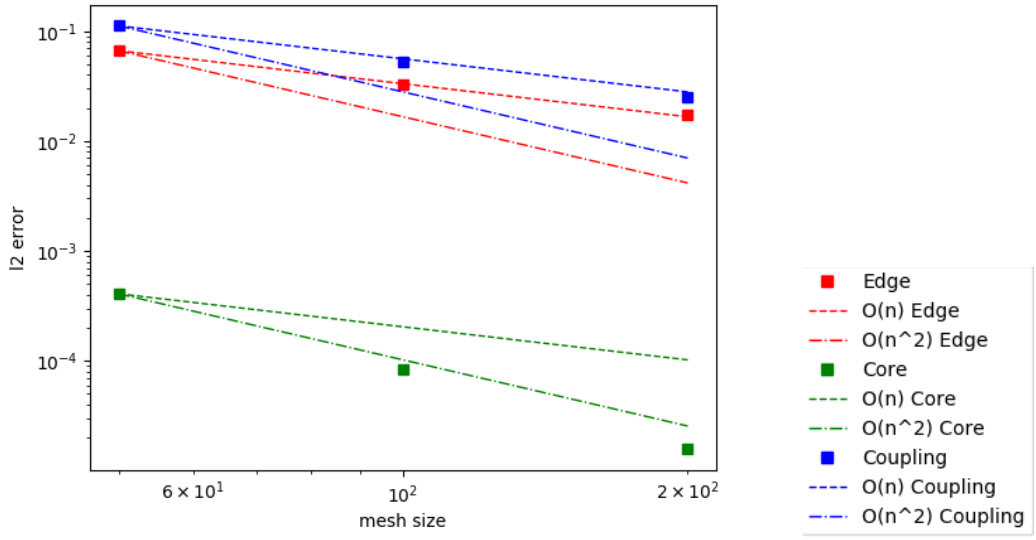
The core and edge codes are individually run to simulate the entire domain with the mesh resolution of the core and edge regions in each coupling case. The individual results are compared to the same reference result. The overall error of the coupling case can be com-



(a) Reference Solution (b) Polar-Cartesian Coupled Solution

Figure 4.11: Comparison of Monolithic Ansatz Solution vs. Coupled Ansatz Solution

pared with the the errors of the core and edge codes individually.



(a) (b)

Figure 4.12: Convergence Order of Polar-Cartesian Coupling vs. Core and Edge

The polar-Cartesian coupling is shown to be first order convergent. This result is expected as the GRILLIX edge code and multiple data mapping steps are first order convergent. The polar code is perfectly aligned with the boundaries and is second order accurate. Therefore, the polar code performs significantly better. The relative error of the coupled solution is a little worse than the edge code. This gap can be predicted to the interpolation error in data mapping between geometrically different configurations.

4.5 Analysis of Data Mapping Methods

Comparing various RBF Mapping Techniques

preCICE offers two types of radial-basis functions, one with global support (fig. 4.13) and one with local support (fig. 4.14). In RBF mapping a global interpolant is formed on the mesh from which data is being sent. This interpolant is then evaluated on the mesh on which the data is received. The interpolant is formed by a linear combination of radial-basis functions centered at each vertex and is further enriched by a linear global polynomial. For the evaluation of each target vertex value, all the source vertices are taken into consideration for local and global basis functions. Both functions lead to a linear system with the difference being that for global basis functions the system matrix is dense and for local basis functions the matrix is sparse. The global basis functions can be problematic due to the algorithmic complexity and lack of scalability for big problems. Such a RBF mapping is configured as follows:

```
<mapping:rbf-thin-plate-splines direction="write"
    from="core_write_mesh" to="edge_cartesian"
    constraint="consistent"/>
```

Figure 4.13: Configuration of RBF Mapping in preCICE with Global Basis Functions

The alternative to the global approach is a local approach where an additional entity is defined as a cut-off distance from the vertex on which the RBF is centered. No vertex beyond this cut-off distance is considered in the evaluation. An example of the additional entity in the definition is defining a *support radius* for the compact thin plate splines mapping which is the radial cut-off distance from the vertex being evaluated. A reasonable value for the support radius is such that five vertices in each direction are considered during evaluation. The configuration is as follows:

The RBF mapping is used in the polar-Cartesian coupling when the Dirichlet values are mapped from the boundary of the core code to the Cartesian mesh of the edge code. Using the verification technique from the previous section the two RBF mapping techniques are

```
<mapping:rbf-compact-tps-c2 direction="write"
    from="core_write_mesh" to="edge_cartesian"
    constraint="consistent" support-radius="0.1"/>
```

Figure 4.14: Configuration of RBF Mapping in preCICE with Local Basis Functions

evaluated. The error with respect to the reference result and total computational time are compared for varying mesh resolutions.

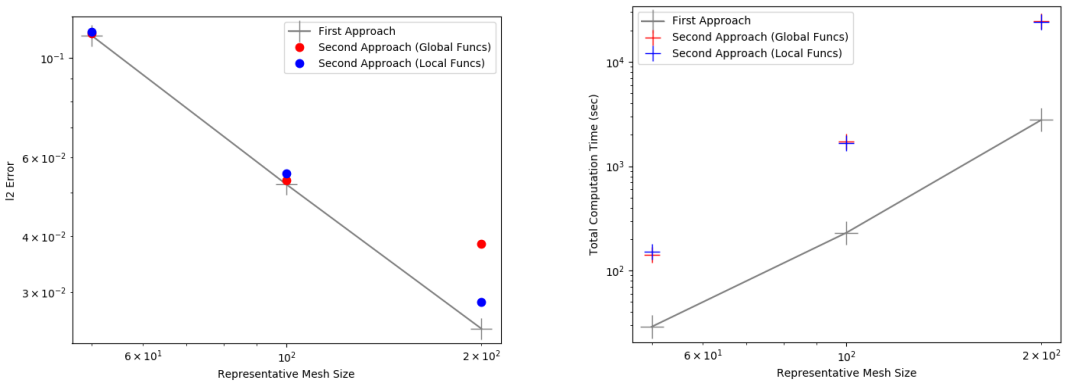
Results of l^2 error and computation time for varying mesh resolutions show that using the global or local variants in one of the mapping schemes of a polar-Cartesian coupling does not affect the numerical accuracy or performance. It can be predicted that for bigger problem sizes local supported RBF mappings perform better than the global variants.

Mapping in GRILLIX vs. Mapping with preCICE

In the Polar-Cartesian coupling the mapping of fluxes is done in two stages. In the first stage the fluxes are approximated from the Cartesian mesh of the edge to the circular mesh of the edge. Once the fluxes are available on the circular mesh of the edge they are mapped to the coupling boundary of the core using nearest-neighbor mapping. As an alternative fluxes can be mapped directly from the Cartesian mesh of the edge to the coupling boundary of the core using mapping functionality from preCICE. In general, these approaches can be described as follows:

- **First Approach (Internal Mapping in Edge + Mapping with preCICE)**: Data from a non-aligned coordinate mesh (Cartesian mesh) in the edge is mapped internally to a mesh having the shape of the flux-aligned coordinates (circular). The data is then mapped from this flux-aligned coordinate shaped mesh to the core.
- **Second Approach (Direct Mapping with preCICE)**: Data from a non-aligned coordinate mesh (Cartesian mesh) in the edge region is mapped directly to the flux-aligned coordinate mesh in the core region using mapping functionality from preCICE.

The first approach is already described in the Polar-Cartesian coupling (fig. 4.2). The second approach is simulated by using radial basis function (RBF) mapping in preCICE. Within the RBF mapping two types of mappings are tested, one with local-basis functions and one with global-basis functions. Data is mapped from the Cartesian mesh of the edge directly to the polar mesh of the core using RBF mapping. Plotting the l^2 error (fig. 4.15) shows that there is no significant gain in numerical accuracy when the second approach is used. In fact for a large mesh the RBF mapping with global-basis is worse than the internal mapping. The RBF mapping with local basis performs a bit better than the global-basis but is still worse than internal mapping. This needs to be investigated further. A



(a) (b)

Figure 4.15: Comparison of Mapping in GRILLIX vs. Mapping using preCICE

comparison of the computation time (fig. 4.15) of the two approaches shows that the RBF mapping from a Cartesian mesh to a polar mesh is more expensive than combined cheaper operations of internal mapping and nearest-neighbor mapping. The computation time of RBF mapping depends on the mesh size from which the data is being mapped. In the second approach the source mesh consisting of vertices in particular band of radius values on the Cartesian grid of the edge. The number of points in this mesh band increase faster than the corresponding increase on a circular mesh. This can possibly explain the slowing down of RBF mapping as against internal mapping for large meshes. The difference in total computation time of the two approaches needs to be investigated further.

5 Diverted Geometry Coupling

Polar-Cartesian coupling shows that a methodology can be developed to couple two participants having different geometric configurations. The GRILLIX edge is already used to simulate edge physics in tokamak reactors. For the flux-aligned coordinates in the core, the simplest choice of geometry that can be made is a polar coordinate system. For more realistic simulations in magnetic fusion applications, the interface coupling methodology needs to be extended to handle diverted geometries.

To model realistic geometries in fusion applications, equations are commonly solved in curvilinear coordinate system. Several coordinate systems like spherical, cylindrical and toroidal systems are curvilinear coordinate systems. Common coordiantes used for the tokamak core region are toroidal-like coordinate systems with the flux surface label ρ, a poloidal angle θ and a toroidal angle ϕ. The flux surface label is a function $\rho(R, Z)$ depending only on R, the distance to the symmetry axis of the torus, and the vertical coordinate Z, but ρ is independent of the toroidal angle ϕ. The usage of flux-aligned coordinates become problematic in diverted geometries towards the separatrix due to a coordinate singularity at the X-point. GRILLIX overcomes this by working in the cylindrical coordinate system (R, Z, φ). The scope of this section is to show the path how to couple a code working in flux-aligned coordinates with GRILLIX working in cylindrical coordinates.

5.1 Analysis in Curvilinear Coordinates

To model a differential equation in a curvilinear coordinate system, an introduction to basic concepts such as covariant and contravariant components of a vector, metric coefficients and their use needs to be given.

Covariant and Contravariant Components of a Vector

A vector space can be defined using coordinate surfaces and coordinate curves. Surfaces can encompass the entire space and curves are defined as intersection of these surfaces. In a three dimensional vector space, three surfaces and three curves exist which can be used to fully define the space. To define a vector in such a space, basis vectors based on the coordinate surfaces and coordinate curves need to be defined. Basis vectors e_i's are defined as tangent vectors to the coordinate curves. Basis vectors e^i's are vectors perpendicular to the

coordinate surfaces. Vectors e_i's are referred to as *tangent basis vectors* and vectors e^i's are referred to as *reciprocal basis vectors*. The nomenclature of using superscript for reciprocal basis vectors and subscript for tangent basis vectors is purely for notation purposes and is generally followed in literature.

A vector D in a 3D vector space can be represented by the tangent and reciprocal basis vectors in the following way:

$$D = (D \cdot e_1)e^1 + (D \cdot e_2)e^2 + (D \cdot e_3)e^3 \tag{5.1}$$

$$D = (D \cdot e^1)e_1 + (D \cdot e_3)e^3 + (D \cdot e_3)e^3 \tag{5.2}$$

The terms $D \cdot e_i$'s are known as the *covariant* components of vector D. The terms $D \cdot e^i$'s are known as the *contravariant* components of vector D. In 3D Euclidian space the covariant and contravariant components of vectors are the same, but need to be considered for curvilinear coordinates.

The toroidal coordinate system is a curvilinear coordinate system having coordinates ρ, θ and ϕ, the tangent basis vectors are e_ρ, e_θ and e_ϕ. The reciprocal basis vectors for the same system are e^ρ, e^θ and e^ϕ. It is important to note the nomenclature by which covariant components are represented by subscript and contravariant components are represented by superscript.

The basis vectors can also be defined by a coordinate transformation. If a coordinate transformation G is defined from the Cartesian coordinate system (x, y, z) to a general curvilinear coordinate system having coordinates c^1, c^2, c^3 then the basis vectors can be formulated as follows:

$$e_i = \frac{\partial G}{\partial c^i} \tag{5.3}$$

$$e^i = \nabla c^i \tag{5.4}$$

Metric Coefficients g_{ij} and g^{ij}

The metric coefficients provide a method to calculate the dot and cross product of vectors in a general curvilinear coordinate system. The metric coefficients g_{ij} are defined as the dot product of the tangent basis vectors e_i and e_j. The metric coefficients g^{ij} are defined as the dot products of the reciprocal basis vectors e^i and e^j. With some additional reformulation and derivations [3] it is shown that the metric coefficients can be used to interchange between covariant and contravariant components. The interchange is as follows:

$$e_i = g_{ij}e^j \tag{5.5}$$

$$e^i = g^{ij} e_j \tag{5.6}$$

The metric coefficients are determined from the coordinate transformations required to fully define a system in the curvilinear coordinate system. The problem needs to be transformed to a curvilinear coordinate system, specifically the toroidal coordinate system.

Jacobian of Curvilinear Coordinate System

The Jacobian in a curvilinear coordinate system is defined using the earlier stated transformation G as follows:

$$\begin{vmatrix} \frac{\partial x}{\partial c^1} & \frac{\partial x}{\partial c^2} & \frac{\partial x}{\partial c^3} \\ \frac{\partial y}{\partial c^1} & \frac{\partial y}{\partial c^2} & \frac{\partial y}{\partial c^3} \\ \frac{\partial z}{\partial c^1} & \frac{\partial z}{\partial c^2} & \frac{\partial z}{\partial c^3} \end{vmatrix}$$

The Jacobin can be reformulated with the transformation G itself as:

$$J = \frac{\partial G}{\partial c^1} \cdot \frac{\partial G}{\partial c^2} \times \frac{\partial G}{\partial c^3} \tag{5.7}$$

The relation between the Jacobian and the determinant of the tensor of the metric coefficients ($g = det[g_{ij}]$) is [3]:

$$g = det[g_{ij}] = \left(\frac{\partial G}{\partial c^1} \cdot \frac{\partial G}{\partial c^2} \times \frac{\partial G}{\partial c^3} \right)^2 = (J)^2 \tag{5.8}$$

5.2 Model Problem in Curvilinear Coordinate System

To simulate a cross-section of a reactor with a diverted toroidal geometry, the geometric configurations of both the core and edge regions change. The core region changes from a polar coordinate system to a toroidal coordinate system. The edge region changes from Cartesian coordinate system to a cylindrical coordinate system.

The model problem for the core region is modified to be solved in a toroidal coordinate system. The numerical method is still finite differences. To formulate any differential equation in curvilinear coordinates, the metric coefficients g^{ij} need to be resolved. The metric coefficients represent a mapping between vector spaces. Hence the full set of metric coefficients is in fact a tensor. For the toroidal coordinates e_ρ, e_θ and e_ϕ the tensor is as follows:

$$\begin{bmatrix} g^{\rho\rho} & g^{\rho\theta} & g^{\rho\phi} \\ g^{\theta\rho} & g^{\theta\theta} & g^{\theta\phi} \\ g^{\phi\rho} & g^{\phi\theta} & g^{\phi\phi} \end{bmatrix}$$

As the transformation between different coordinate systems is known, the metric coefficients can be algebraically evaluated, For such a system the metric coefficient tensor is symmetric, that is $g^{ij} = g^{ji}$. Due to axis-symmetry $g^{\rho\phi} = g^{\theta\phi} = 0$, and the metric coefficient tensor simplifies to:

$$\begin{bmatrix} g^{\rho\rho} & g^{\rho\theta} & 0 \\ g^{\rho\theta} & g^{\theta\theta} & 0 \\ 0 & 0 & g^{\phi\phi} \end{bmatrix}$$

Divergence of a vector u in a general curvilinear coordinate system is defined as [3]:

$$\nabla u = (\frac{1}{J})\partial_i(J(\nabla u)^i) \tag{5.9}$$

It is important to note that $(\nabla u)^i$ is the contravariant component of ∇u. The model problem to be solved is the diffusion equation which can be written as:

$$\partial_t u = \nabla \cdot \nabla u \tag{5.10}$$

Substituting the definition of divergence in curvilinear coordinates:

$$\partial_t u = \frac{1}{J}\partial_i(J(\nabla u)^i) \tag{5.11}$$

Contravariant component of divergence of a vector $(\nabla u)^i$ can be written in its covariant component using the metric coefficients as follows:

$$(\nabla u)^i = g^{ij}\partial_j u \tag{5.12}$$

Formulating the model problem using covariant components of a vector gives:

$$\partial_t u = \frac{1}{J}\partial_i(Jg^{ij}\partial_j u) \tag{5.13}$$

In the following we reduce the problem to 2D by assuming that the field u is axisymmetric, i.e. $\partial_\phi u = 0$, and expanding this equation in ρ and θ coordinates yields[1]:

$$\partial_t u = \frac{1}{J}[\partial_\rho(Jg^{\rho\rho}\partial_\rho u) + \partial_\theta(Jg^{\theta\theta}\partial_\theta u) + \partial_\rho(Jg^{\rho\theta}\partial_\theta u) + \partial_\theta(Jg^{\theta\rho}\partial_\rho u)] \tag{5.14}$$

[1] Note that $g^{\phi\phi}$ still enters the Jacbian J

The terms containing diagonal metric coefficients $g^{\rho\rho}$ and $g^{\theta\theta}$ are discretized in a staggered grid fashion:

$$\partial_\rho(Jg^{\rho\rho}\partial_\rho u) = \frac{1}{d\rho}(J_{i+\frac{1}{2},j}g^{\rho\rho}_{i+\frac{1}{2},j}\frac{u_{i+1,j}-u_{i,j}}{d\rho} - J_{i-\frac{1}{2},j}g^{\rho\rho}_{i-\frac{1}{2},j}\frac{u_{i,j}-u_{i-1,j}}{d\rho}) \tag{5.15}$$

Each entity evaluated at half a grid index length is further evaluated by averaging values at neighbouring grid points:

$$\begin{aligned}\partial_\rho(Jg^{\rho\rho}\partial_\rho u) = \frac{1}{d\rho}(&\frac{J_{i+1,j}+J_{i,j}}{2}\frac{g^{\rho\rho}_{i+1,j}+g^{\rho\rho}_{i,j}}{2}\frac{u_{i+1,j}-u_{i,j}}{d\rho} - \\ &\frac{J_{i,j}+J_{i-1,j}}{2}\frac{g^{\rho\rho}_{i,j}+g^{\rho\rho}_{i-1,j}}{2}\frac{u_{i,j}-u_{i-1,j}}{d\rho})\end{aligned} \tag{5.16}$$

Similarly the second term associated with diagonal metric coefficient $g^{\theta\theta}$ is discretized as:

$$\begin{aligned}\partial_\theta(Jg^{\theta\theta}\partial_\theta u) = \frac{1}{d\theta}(&\frac{J_{i,j+1}+J_{i,j}}{2}\frac{g^{\theta\theta}_{i,j+1}+g^{\theta\theta}_{i,j}}{2}\frac{u_{i,j+1}-u_{i,j}}{d\theta} - \\ &\frac{J_{i,j}+J_{i,j-1}}{2}\frac{g^{\theta\theta}_{i,j}+g^{\theta\theta}_{i,j-1}}{2}\frac{u_{i,j}-u_{i,j-1}}{d\theta})\end{aligned} \tag{5.17}$$

The terms containing off-diagonal metric coefficients $g^{\rho\theta}$ and $g^{\theta\rho}$ are discretized using finite differences:

$$\partial_\rho(Jg^{\rho\theta}\partial_\theta u) = \frac{1}{2d\rho}(J_{i+1,j}g^{\rho\theta}_{i+1,j}(\partial_\theta u)_{i+1,j} - J_{i-1,j}g^{\rho\theta}_{i-1,j}(\partial_\theta u)_{i-1,j}) \tag{5.18}$$

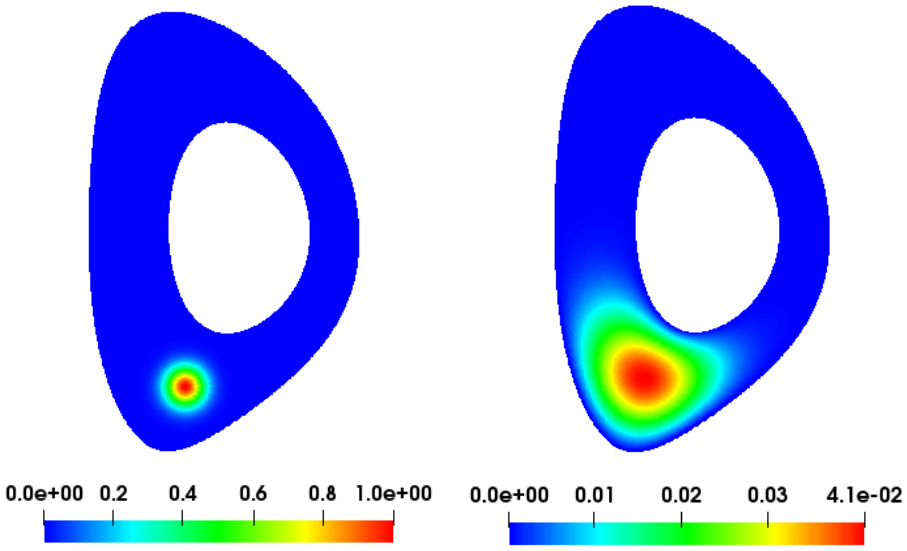
$$\partial_\rho(Jg^{\rho\theta}\partial_\theta u) = \frac{1}{2d\rho}(J_{i+1,j}g^{\rho\theta}_{i+1,j}\frac{u_{i+1,j+1}-u_{i+1,j-1}}{2d\theta} - J_{i-1,j}g^{\rho\theta}_{i-1,j}\frac{u_{i-1,j+1}-u_{i-1,j-1}}{2d\theta}) \tag{5.19}$$

Similarly for second term associated with $g^{\theta\rho}$:

$$\partial_\theta(Jg^{\theta\rho}\partial_\rho u) = \frac{1}{2d\theta}(J_{i,j+1}g^{\theta\rho}_{i,j+1}\frac{u_{i+1,j+1}-u_{i-1,j+1}}{2d\rho} - J_{i,j-1}g^{\theta\rho}_{i,j-1}\frac{u_{i+1,j-1}-u_{i-1,j-1}}{2d\rho}) \tag{5.20}$$

The core code is modified to solve the diffusion equation in toroidal coordinates. The core code still works in two dimensions but now the effect of toroidicity needs to be considered

in the 2D cross-section. The metric coefficients and Jacobian values for each grid point of the geometric shape are computed by an independent package within GRILLIX. The data from this package is read by the core code and used to solve the stencil derived above. Dirichlet boundary conditions are implemented at the inner and outer edges of the geometry.



(a) t = 0 (b) t = 1

Figure 5.1: Diffusion of Gaussian Blob in Diverted Geometry Setup

A Gaussian blob diffusing in a diverted geometry domain looks visually reasonable (fig. 5.1). Verification of the core code for diverted geometries is not done in this thesis.

5.3 Coupling with Diverted Geometry

The coupling framework developed for Polar-Cartesian coupling can be used for diverted geometries with some modifications. The critical difference between a circular geometry and diverted geometry is the computation and application of fluxes at the coupling boundary. The coupling boundary contour is no longer orthogonal to the poloidal grid lines. The normal vector to the coupling boundary needs to be calculated separately and based on that the flux calculation is done. This added complexity is not pursued in this thesis and is discussed further as part of future work.

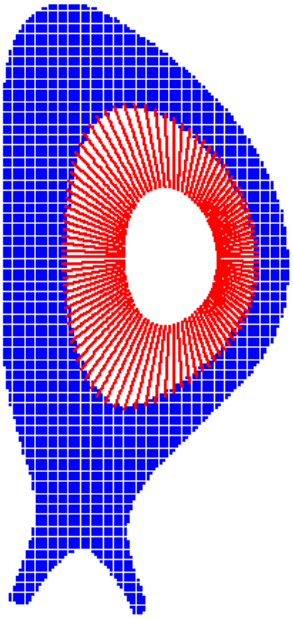
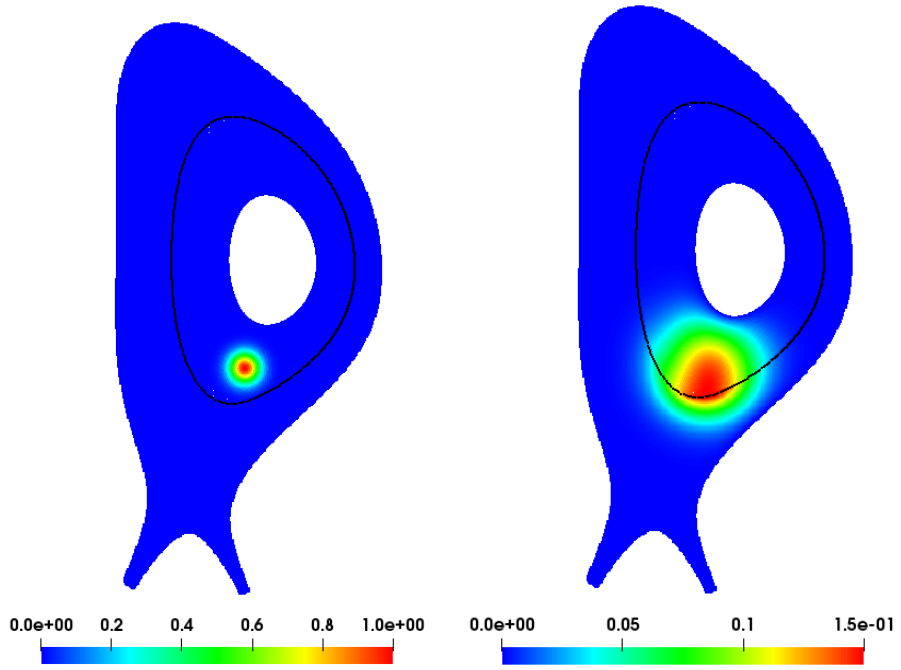


Figure 5.2: Schematic of Geometric Configurations in Coupled Diverted Geometry

To show that the coupling framework developed in this thesis can be directly applied to transfer data across diverted geometries, a uni-directional coupling is solved (fig. 5.3). In this coupling, Dirichlet values from the core region boundary are transferred to the edge region at the coupling boundary. A uni-directional coupling does not provide physically valid results. This experiment is done only to show that the existing coupling framework can be easily modified to handle diverted geometries.

The geometry is chosen from [2]. A code package in GRILLIX is used to compute the metric coefficients and Jacobian values for the toroidal grid points of the core. The edge region is simulated using a uniform Cartesian grid. For diverted geometries the edge region needs to solve using cylindrical coordinates. The change from Cartesian to cylindrical coordinates is handled within GRILLIX and no additional modifications are required. The



(a) t = 0 (b) t = 0.5

The black line is the coupling interface. The interior domain is the core and the exterior domain is the edge

Figure 5.3: Diffusion of Gaussian Blob in Coupled Diverted Geometry Setup

fig. 5.3 shows a Gaussian blob initialized in the core region diffuses into the edge region.

6 Conclusions

6.1 Summary of the Thesis

This thesis explores geometric aspects of coupling codes simulating the core and edge regions of a tokamak fusion reactor. A black-box partitioned coupling approach is pursued using the coupling library preCICE. The physics in a fusion reactor is simplified to a diffusion problem. The geometry is simplified from 3D toroidal and cylindrical coordinates to 2D polar and Cartesian coordinates. These simplifications allow to investigate the coupling in detail. A differential equation is solved on a Cartesian and polar grid separately which are then coupled. The simplifications of physics and geometry are done in a way that the generality of the coupling itself is still retained. This thesis addresses the complexities of mapping data across different geometric configurations and attempts to build a framework for coupling in fusion applications.

A code is developed from scratch to simulate the core region. This core code is verified for second order spatial convergence. The core code is structured in a modular fashion to enable re-usability in the future. The code is designed in a way to allow for further modifications to use diverted geometries, higher order boundary condition schemes and better initialization methods. The edge region is simulated using GRILLIX. The GRILLIX code is adapted to use the preCICE library. The adapted form of the edge code is also modular in fashion. The data mapping functions are bundled as generic functions which can be used for future applications.

A Cartesian-Cartesian coupling with both core and edge simulated with GRILLIX is done. A Dirichlet-Neumann coupling is constructed. This coupling highlights the problem that a Cartesian grid cannot fully capture the shape of a flux-aligned geometry. An additional internal mapping scheme is developed to address this problem. In this internal mapping the edge code stores an extra mesh which has the shape of the flux-aligned geometry. This internal mesh is used to map data consistently to the core region. Using this idea an additional circular shaped mesh is defined within the edge code. This internal mapping is a critical aspect of achieving accurate data mapping. In Polar-Cartesian coupling the custom built core code is coupled to GRILLIX. It is found that overlapping the domains of core and edge is necessary to achieve a physically valid coupling. The resulting Polar-Cartesian coupling with overlap is shown to be spatially first order convergent. This is expected because the flux mapping and boundary conditions in GRILLIX are first order convergent. The internal mapping done in the edge code can be skipped if preCICE is

employed to map data directly between the non-aligned edge mesh and flux-aligned core mesh. Use of internal mapping is compared against using a radial-basis function mapping in preCICE. The comparison shows that internal mapping is computationally cheaper than using radial-basis function mapping with global basis functions. At higher mesh resolutions the RBF mapping has a higher error than the internal mapping. Behavior of RBF mapping for large meshes needs to be investigated further.

The framework developed for Polar-Cartesian coupling is extended to handle diverted geometries under certain restrictions. The core region is now modelled with toroidal coordinates and the edge region with cylindrical coordinates. Switching from Cartesian to cylindrical coordinates in edge is handled internally in GRILLIX. The core code models the diffusion equation in two dimensional toroidal geometry. Implementing Neumann boundary conditions along the coupling boundary is as straight-forward as for a polar mesh because the poloidal grid lines are no longer orthogonal to the interface. This is not pursued further in this thesis. A first test case of uni-directional coupling in which Dirichlet values are exchanged from core to edge shows promising results. Coupling with diverted geometry could not be studied in more detail within the scope of this thesis.

Partitioned coupling methodology has the capability to provide efficient and robust solutions for coupling in fusion applications. A flexible tool like the preCICE library along with adapted codes for different regimes or physics is path worth exploring for fusion research.

6.2 Future Challenges

This thesis highlights many future paths for black-box coupling in fusion applications. The coupling framework developed in this work can be expanded in several ways to move closer to realistic coupled simulations for fusion reactors. Some of the possible avenues are discussed below:

- **Diverted Geometry Coupling**: The diverted geometry coupling shown in this thesis is very preliminary. Neumann boundary conditions along the coupling interface are essential to have a physically valid coupling. The core code can be to be developed further to map fluxes computed by GRILLIX which are not orthogonal to the interface.
- **Improving Overall Spatial Convergence Order**: The spatial convergence of the Polar-Cartesian coupling is shown to be first order. By using higher order boundary conditions in GRILLIX and a higher order scheme to map fluxes the spatial convergence would mostly improve to second order. The explicit Euler time stepping scheme is first order convergent and using higher order time stepping is a way to improve overall convergence.
- **Volume Coupling**: This thesis explores interface coupling. For two dimensions the

coupling interface is a curve and for three dimensions it is a surface. There are cases in fusion simulations where different models are applicable over the same domain but simulating the models together is difficult. Individual codes simulating each model solve the entire domain space and preCICE is used to couple all the grid points in the domain. In such a coupling, spatial convergence and performance are critical aspects which could be handled in the coupling library.

- **Coupling Advanced Physics**: In this thesis the model problem is simplified to the diffusion equation. In realistic plasma fusion scenarios many complex models are needed simultaneously to describe the behaviour of the system. Using a robust coupling framework, coupling of sophisticated highly non-linear plasma turbulence models can be pursued. A kinetic model can be coupled to a fluid model with interface coupling. A plasma model can be coupled with neutral gas model using volume coupling.

Bibliography

[1] Hans-Joachim Bungartz, Florian Lindner, Bernhard Gatzhammer, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. preCICE – a fully parallel library for multi-physics surface coupling. *Computers and Fluids*, 141:250–258, 2016. Advances in Fluid-Structure Interaction.

[2] Antoine Cerfon and Jeffrey Freidberg. "one size fits all" analytic solutions to the grad–shafranov equation. *Physics of Plasmas*, 17:032502–032502, 03 2010.

[3] W. D. D'haeseleer. *Flux coordinates and magnetic field structure: a guide to a fundamental tool of plasma structure*. Springer-Verlag, 1991.

[4] W. Zholobenko et al. Electric field and turbulence in global braginskii simulations across the asdex upgrade edge and scrape off layer. *submitted to Plasma Physics and Controlled Fusion*, 2020.

[5] O.E Garcia, J Horacek, R.A Pitts, A.H Nielsen, W Fundamenski, V Naulin, and J. Juul Rasmussen. Fluctuations and transport in the TCV scrape-off layer. *Nuclear Fusion*, 47(7):667–676, jun 2007.

[6] T. Görler, X. Lapillonne, S. Brunner, T. Dannert, F. Jenko, F. Merz, and D. Told. The global version of the gyrokinetic turbulence code GENE. *Journal of Computational Physics*, 230(18):7053–7071, 2011.

[7] F. Hariri and M. Ottaviani. A flux-coordinate independent field-aligned approach to plasma turbulence simulations. *Computer Physics Communications*, 184(11):2419 – 2429, 2013.

[8] Albrecht Herrmann and Otto Gruber. Chapter 1: Asdex upgrade - introduction and overview. *Fusion Science and Technology*, 44(3):569–577, 2003.

[9] Florian Lindner, Miriam Mehl, and Benjamin Uekermann. Radial basis function interpolation for black-box multi-physics simulations. 05 2017.

[10] Jef Ongena and Yuichi Ogawa. Nuclear fusion: Status report and future prospects. *Energy Policy*, 96:770 – 778, 2016.

[11] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions . *Journal of Fluids Engineering*, 124(1):4–10, 11 2001.

[12] Andreas Stegmeir. *GRILLIX: A 3D turbulence code for magnetic fusion devices based on a*

field line map. PhD thesis, Technische Universität München, 01 2015.

[13] Andreas Stegmeir, David Coster, Alexander Ross, Omar Maj, Karl Lackner, and Emanuele Poli. Grillix: A 3d turbulence code based on the flux-coordinate independent approach. *Plasma Physics and Controlled Fusion*, 60, 12 2017.

[14] Benjamin Uekermann. *Partitioned Fluid-Structure Interaction on Massively Parallel Systems*. PhD thesis, Technische Universität München, 10 2016.

[15] R. C. Wolf, C. D. Beidler, A. Dinklage, P. Helander, H. P. Laqua, F. Schauer, T. Sunn Pedersen, and F. Warmer. Wendelstein 7-x program—demonstration of a stellarator option for fusion energy. *IEEE Transactions on Plasma Science*, 44(9):1466–1471, 2016.