Contents

1 The problem (Reisenegger 2007)

1.1 PDF and initial conditions

In this section, the problem outlined in (reference to Reisenegger 2007) is being solved, namely, the growth of perturbation in the weak poloidal field $\mathbf{B}_p$ in the presence of a strong background toroidal component $\mathbf{B}_t$. The approach of evolving $\mathbf{B}_p$ boils down to numerically solving the following equation:

$$\frac{\partial \mathbf{B}_p}{\partial t} = -\nabla \times \left[\frac{c}{4\pi n e} \left(\nabla \times \mathbf{B}_t \right) \times \mathbf{B}_p \right], \tag{1}$$

where the toroidal field $\mathbf{B}_t$ can be decomposed using basis vectors of the cylindrical coordinate system:

$$\mathbf{B}_t = B \cdot \nabla\varphi = \frac{B}{R} \cdot \mathbf{e}_\varphi, \tag{2}$$

The scalar function B in the expression above is defined as

$$B = \frac{b_0 R^4 n^2}{r_0^3 n_0^2}, \tag{3}$$

and the number density of electrons n is given by

$$n = n_0 \left(1 - \frac{r^2}{r_0^2} \right),$$

with R and r standing for cylindrical and spherical radial coordinates respectively. The constants b_0, n_0 and r_0 have the following values:

$$b_0 = 10^{14} \text{ G},$$
$$n_0 = 10^{36} \text{ cm}^{-3},$$
$$r_0 = 10 \text{ km}.$$

By introducing dimensionless quantities

$$\tilde{r} = r/r_0$$
$$\tilde{R} = R/r_0$$
$$\tilde{n} = n/n_0$$
$$\tilde{t} = t/\tau = t\frac{c b_0}{n_0 e r_0^2}$$
$$\tilde{\mathbf{B}}_p = \mathbf{B}_p/b_0, \quad \tilde{\mathbf{B}}_t = \mathbf{B}_t/b_0,$$

as well as derivatives along dimensionless spatial coordinates $\tilde{x}_i = x_i/r_0$ expressed in the new operator

$$\tilde{\nabla} = r_0 \nabla,$$

equation 1 is rewritten in its dimensionless form:

$$\frac{\partial \tilde{\mathbf{B}}_p}{\partial \tilde{t}} = -\tilde{\nabla} \times \left[\frac{1}{4\pi\tilde{n}} \left(\tilde{\nabla} \times \tilde{\mathbf{B}}_t \right) \times \tilde{\mathbf{B}}_p \right]. \tag{4}$$

The new timescale

$$\tau = \frac{n_0 e r_0^2}{c b_0} = 5.08 \text{ Myr}.$$

In the Cartesian coordinates' basis, the dimensionless toroidal field is presented in the form of

$$\tilde{\mathbf{B}}_t = -\tilde{y}\tilde{R}^2\tilde{n}^2 \cdot \mathbf{e}_x + \tilde{x}\tilde{R}^2\tilde{n} \cdot \mathbf{e}_y + 0 \cdot \mathbf{e}_z.$$

It is assumed that the toroidal field does not evolve with time so that the curl of $\mathbf{B}_t$ has to be computed just once, in the beginning of the simulation. In addition, by the weakness of $\mathbf{B}_p$ it is meant that $|\mathbf{B}_p| \ll |\mathbf{B}_t|$.
The initial configuration of the dimensionless poloidal perturbation is the following:

$$\tilde{\mathbf{B}}_p\Big|_{t=0} = 0 \cdot \mathbf{e}_R + 0 \cdot \mathbf{e}_\varphi + 10^{-7} \cdot \mathbf{e}_z. \tag{5}$$

The problem is solved on a 4π spherical shell-like manifold, where the radial variable $\tilde{r}$ takes values from $\tilde{r}_{\min} = 0.5$ to $\tilde{r}_{\max} = 0.9$. As can be seen from the equations above, all relevant physical constants equal to 1 and the problem is treated in dimensionless variables.

1.2 First step. Solving for velocities

Firstly, the dimensionless velocity field $\mathbf{u}$ owing to the background toroidal magnetic field $\tilde{\mathbf{B}}_t$ is calculated by merely taking curl of $\tilde{\mathbf{B}}_t$ and then dividing the result by the scalar function $4\pi\tilde{n}$. It is done in a bit sophisticated manner though. Namely, in the first part, an auxiliary vector field $\mathbf{A} = 4\pi\tilde{n} \cdot \mathbf{u}$ is introduced, and equation 6 below is solved:

$$4\pi n \cdot \mathbf{u} \equiv \mathbf{A} = -\left[\tilde{\nabla} \times \tilde{\mathbf{B}}_t\right]. \tag{6}$$

The calculation begins with use of a smooth *trial function* $\mathbf{w}$, whereby equation 6 is multiplied and then integrated over the whole domain. This is a classic procedure carried out in any numerical calculations that are somehow related to the finite element method. Henceforth the subscript t in the toroidal magnetic field $\tilde{\mathbf{B}}_t$ will be omitted so as to not confuse it with the vector components (i.e. $\tilde{B}_k$, where k are $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$). Taking into consideration the above said, equation 6 is rewritten in its integral form:

$$\int w_k A_k dV = -\int w_k e_{kij} \partial_i \tilde{B}_j dV, \tag{7}$$

where summation under the repeating indices is assumed, e_{kij} denotes the Levi-Civita tensor and ∂_i is a partial derivative with respect to the dimensionless coordinate $\tilde{x}_i$. After integrating by parts the right term one obtains

$$\int w_k A_k dV = -\int \tilde{B}_k e_{kij} \partial_i w_j dV + \int n_k e_{kij} w_i \tilde{B}_j dS. \tag{8}$$

The next step is to divide the manifold into cells (also called elements) and in each cell perform decomposition of $\mathbf{w}$ and $\mathbf{A}$ by means of *Lagrange interpolating polynomials*, which are uniquely associated with nodes of the cells. These Lagrange polynomials do not live on the real (physical) manifold of the problem, but lie on a so called reference unit domain, a unit cube in 3 dimensions. In the unit cube, the polynomials form a basis, by use of which any other function can be decomposed. A mapping between the unit cube and each cell of the physical manifold is then constructed and, as a next step following the decomposition, integration within each cell's volume is carried out. In the numerical code, the integration over a single cell requires the Jacobian of the mapping to relate physical coordinates to that of the unit cell. By summing up the contributions of integrals over individual cells, the whole manifold volume is eventually covered.

The decomposition process can be viewed as interpolation of $\mathbf{w}$ and $\mathbf{A}$, since the Lagrange interpolating polynomials have a fixed order. Namely, if by chance the exact solution is found at the nodal points, in between the numerical solution will still remain an approximation of the exact

solution, provided that the exact solution is not a polynomial of the same or lower order than the interpolating polynomials. Note that eq. 8 is exact.

In the simplest and most convenient form, the decomposition of $\mathbf{w}$ and $\mathbf{A}$ would look like

$$w_k(\tilde{\mathbf{x}}) = \sum_l N_{kl}(\xi(\tilde{\mathbf{x}}))c_{kl}, \quad A_k(\tilde{\mathbf{x}}) = \sum_l N_{kl}(\xi(\tilde{\mathbf{x}}))d_{kl}, \tag{9}$$

where $N_{kl}(\xi(\tilde{\mathbf{x}}))$ are the Lagrange polynomials of fixed order, c_{kl} and d_{kl} are weights of the decomposition and ξ stands for coordinates that chart the unit domain. Each Lagrange polynomial N_{kl} is associated with a certain *support point* $\tilde{\mathbf{x}}_l$, whereat $N_{kl}(\tilde{\mathbf{x}}_l) = 1$. In the sums l goes over all support points of a cell within which the integration is performed. For the linear order polynomials, the support points coincide with the vectices of the cell, while for the higher order polynomials, additional support points appear in between. Note that in eq. 9 there is no summation over k.

1.3 An aside: more on decomposition and integration

To better grasp the idea of how these all structures are realized in the code, let's consider a scalar problem in 2 dimensions where a hypothetical scalar field $\varphi(x, y)$ must be found. Let's also take the Lagrange polynomials of linear order (in each dimension), as it is more convenient for visualization. The series representation of the unknown scalar function $\varphi(x, y)$ and the corresponding trial function $w(x, y)$, which is obviously a scalar field as well, will be the following:

$$w(x, y) = \sum_l N_l(\xi_1(x, y), \xi_2(x, y))c_l, \quad \varphi(x, y) = \sum_m N_m(\xi_1(x, y), \xi_2(x, y))d_m. \tag{10}$$

As we are now working in 2D, the unit reference cell is given by a unit square. For the linear order polynomials, the index l (and m) in the sums 10 above covers four support points, which coincide with the vertices of the unit square. We will denote the vertices by a subscript $l = 0$ for $\xi_1, \xi_2 = 0$; $l = 1$ for $\xi_1 = 1, \xi_2 = 0$; $l = 1$ for a case $\xi_1, \xi_2 = 1$ and $l = 3$ if $\xi_1 = 0, \xi_2 = 1$. Each bilinear polynomial $N_l(\xi_1, \xi_2)$ is uniquely associated with one of the support points. To see this, we write $N_l(\xi_1, \xi_2)$ in the explicit form:

$$N_0(\xi_1, \xi_2) = (1 - \xi_1)(1 - \xi_2), \quad N_1(\xi_1, \xi_2) = \xi_1(1 - \xi_2),$$

$$N_2(\xi_1, \xi_2) = (1 - \xi_1)\xi_2, \quad N_3(\xi_1, \xi_2) = \xi_1\xi_2.$$

A few nice properties of $N_l(\xi_1, \xi_2)$ are clearly noticeable. First,

$$N_l(\xi_m, \xi_n) = \delta_{lm}\delta_{ln},$$

where ξ_m, ξ_n stand for ξ_1 and ξ_2 coordinates of the vertices of the unit square. Thus, $N_l(\xi_1, \xi_2)$ is tied to a support point, whereat $N_l = 1$. At all the other points $N_l = 0$.

The second property of $N_l(\xi_1, \xi_2)$ is

$$\sum_{l=0}^{3} N_l(\xi_1, \xi_2) = 1,$$

that gives us a tool to exactly represent constant functions in the decomposition. Of course, these properties also hold for higher order polynomials. For higher order polynomials the number of support points is always bigger than the number of vertices of the cell. Support points used in the code are always equidistant. The bilinear Lagrange polynomials are shown in figure 1.

Suppose now we need to compute an integral shown in the expression 11 below over a particular cell of the physical manifold, where the 2D problem is defined:

$$\int\limits_{(V_{\text{cell}})} w\frac{\partial\varphi}{\partial x}dV = \iint\limits_{(V_{\text{cell}})} w\frac{\partial\varphi}{\partial x}dxdy. \tag{11}$$

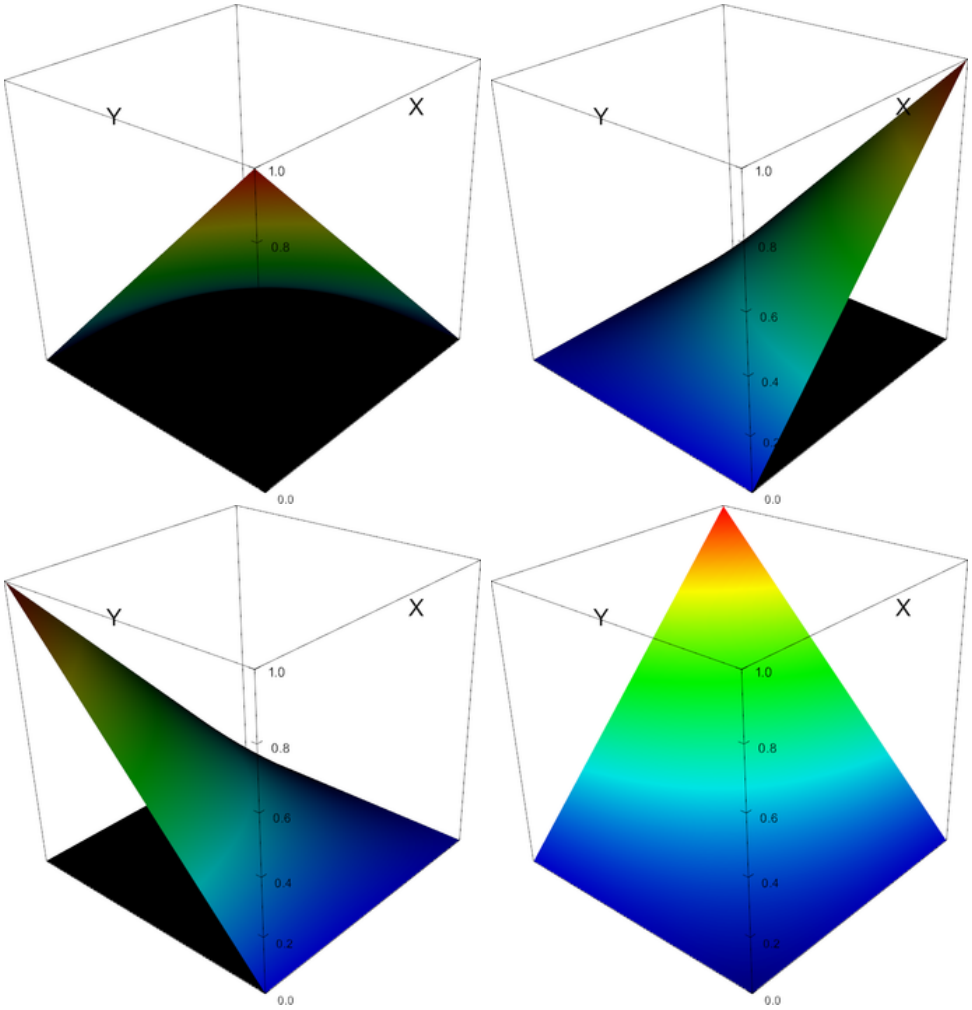

Figure 1: Bilinear decomposition polynomials in 2D for each of four vertices of the unit square.

To do so, we make use of the decomposition 10, where l and m go from 0 to 3, we also use the chain rule and the Jacobian of the smooth mapping $(\xi_1, \xi_2) \Rightarrow (x, y)$:

$$\iint\limits_{(V_{\text{cell}})} w\frac{\partial\varphi}{\partial x}dxdy = \sum_{l,m=0}^{3} \iint\limits_{(V_{\text{cell}})} N_l c_l \frac{\partial N_m}{\partial x} d_m dxdy = \sum_{l,m=0}^{3} c_l d_m \cdot \int_0^1 \int_0^1 N_l \frac{\partial N_m}{\partial \xi_i}\frac{\partial \xi_i}{\partial x}\left|\frac{\partial(x,y)}{\partial(\xi_1,\xi_2)}\right| d\xi_1 d\xi_2. \quad (12)$$

The integral on the right is what is actually computed in our code. The integration is fulfilled employing *the Gauss-Legendre quadrature rule*. The method has a parameter n, for which polynomial of the order $2n-1$ or less are integrated exactly along each dimension. The integration rule is expressed as follows:

$$\text{for } F(\xi_1,\xi_2) = N_l\frac{\partial N_m}{\partial \xi_i}\frac{\partial \xi_i}{\partial x}\left|\frac{\partial(x,y)}{\partial(\xi_1,\xi_2)}\right|, \quad \int_0^1\int_0^1 F(\xi_1,\xi_2)d\xi_1 d\xi_2 = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j F(\xi_1=\xi_i, \xi_2=\xi_j),$$

where w_i and w_j are weights of the quadrature rule. As can be clearly seen, the number of terms in the sum goes as n^{dim} in dim dimensions. The integration error is proportional to (the size of the cell)^{2n}.

1.4 The main problem: continue

After exploring the decomposition and integration routines of the code, we can come back to the main problem. We consider eq. 8 for a certain cell and plug the decomposed functions **w** and **A** into

it. Thus, we have

$$\sum_{l,m} \int\limits_{(V_{\rm cell})} N_{3l+k} N_{3m+k} c_{3l+k} d_{3m+k} dV = -\sum_{l} \int\limits_{(V_{\rm cell})} \tilde{B}_k e_{kij} N_{3l+j,i} c_{3l+j} dV + \sum_{l} \int\limits_{(S_{\rm cell})} n_k e_{kij} N_{3l+i} c_{3l+i} \tilde{B}_j dS. \tag{13}$$

In the expression the indices l and m cover support points of the cell, whereas k goes over components of the vectors. Note that the sum over k in the first term on the left is also implied and a comma in $N_{3l+j,i}$ represents a derivative of N_{3l+j} with respect to the ith spatial coordinate of the physical manifold.

It is suitable to define a new index $a = 3l + k$ (and $b = 3m + k$), whose maximum value is the number of degrees of freedom of the cell = the total number of support points $\times$ the number of vector components. This step allows us to introduce two new objects, a local matrix $K^{\rm local}_{ab}$ and a local vector $F^{\rm local}_a$, using the expression below:

$$K^{\rm local}_{ab} = \int\limits_{(V_{\rm cell})} N_a N_b dV, \quad F^{\rm local}_a \equiv F^{\rm local}_{3l+j} = -\int\limits_{(V_{\rm cell})} \tilde{B}_k e_{kij} N_{3l+j,i} dV + \int\limits_{(S_{\rm cell})} n_k e_{kji} N_{3l+j} \tilde{B}_i dS.$$

Here and further in the text, *local* refers to a single cell, while *global* refers to the whole manifold. By means of $K^{\rm local}_{ab}$ and $F^{\rm local}_a$ we can rewrite 9 in the matrix-vector form:

$$c_a K^{\rm local}_{ab} d_b = c_a F^{\rm local}_a.$$

This equation, however, is valid just for a single cell. Therefore, to proceed further we also introduce the global matrix K_{ab} and vector F_a. The difference from the local ones is that now in the definition of $a = 3l + k$, l covers not the support points of a specific cell but that of the whole manifold. Essentially, K_{ab} and F_a are assembled from $K^{\rm local}_{ab}$ $F^{\rm local}_a$. Note that by definition the matrix K_{ab} is sparse and symmetric. The global analogue of eq. 13 corresponding to the integration over the whole physical manifold is given by the following equation

$$\sum_{\rm cells} \sum_{l,m} \int\limits_{(V_{\rm cell})} N_{3l+k} N_{3m+k} c_{3l+k} d_{3m+k} dV = -\sum_{\rm cells} \sum_{l} \int\limits_{(V_{\rm cell})} \tilde{B}_k e_{kij} N_{3l+j,i} c_{3l+j} dV + \\ + \sum_{\rm cells} \sum_{l} \int\limits_{(S_{\rm cell})} n_k e_{kij} N_{3l+i} c_{3l+i} \tilde{B}_j dS,$$

where the first sum in each term implies that we add the contributions from all cells of the manifold. By making use of the matrix-vector notations introduced above, we are able to rewrite this expression as

$$\sum_{\rm cells} c_a({\rm cell}) K^{\rm local}_{ab}({\rm cell}) d_b({\rm cell}) = \sum_{\rm cells} c_a({\rm cell}) F^{\rm local}_a({\rm cell}),$$

where it is highlighted that all the objects are functions of cell of the manifold. In the global matrix-vector notations, the last expression becomes

$$c_a K_{ab} d_b = c_a F_a \tag{14}$$

with c_a and d_a being global vectors. As this equation has to hold for any trial function $\mathbf{w}$ (i.e. for any coefficients c_a), it immediately yields

$$\boxed{K_{ab} d_b = F_a} \tag{15}$$

The boxed equation is solved iteratively in the code and the coefficients d_i are found. Following the calculation of the coefficients, the vector field $\mathbf{A}$ is constructed and the velocity field $\mathbf{u} = \mathbf{A}/(4\pi\tilde{n})$ is obtained. As calculated by our numerical code, the velocity field $\mathbf{u}$ is depicted in figure 2.

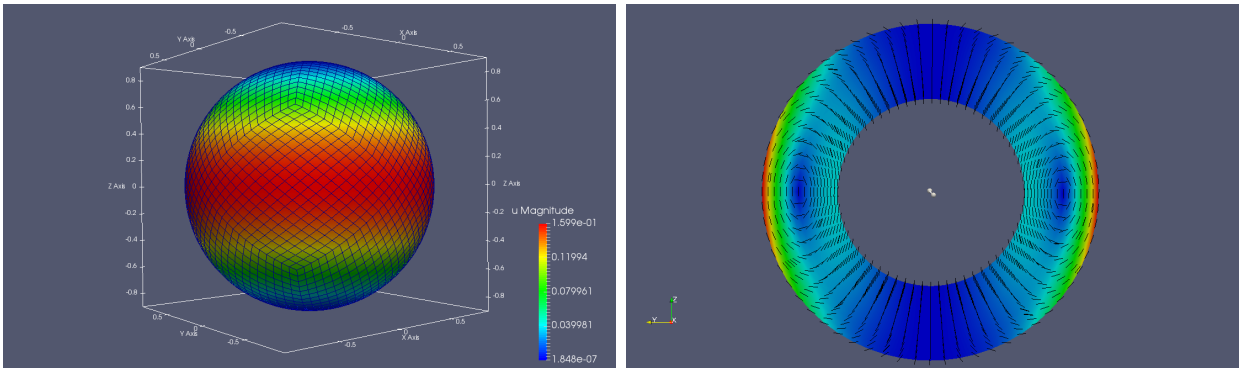

Figure 2: *(left)* Distribution of velocity field $\mathbf{u}$ on the outer surface of the spherical shell-like manifold. *(right)* A plane cut $\tilde{x} = 0$ of the same manifold, where the velocity field is shown. The color describes the magnitude of the field (for both left and right panels).

1.5 Second step. Solving for poloidal perturbation

Now we seek for the solution of the partial differential equation

$$\frac{\partial \tilde{\mathbf{B}}}{\partial \tilde{t}} = \tilde{\nabla} \times \left[\mathbf{u} \times \tilde{\mathbf{B}} \right], \tag{16}$$

where from now on $\tilde{\mathbf{B}}$ denotes the poloidal perturbation, not the toroidal magnetic field. Taking the same steps as in the subsection 1.2 we get

$$\int w_k \frac{\partial \tilde{B}_k}{\partial \tilde{t}} dV = \int w_i e_{ijk} \partial_j (e_{klm} u_l \tilde{B}_m) dV.$$

The term on the right hand side is integrated by parts

$$\int w_k \frac{\partial \tilde{B}_k}{\partial \tilde{t}} dV = \int (e_{ilm} u_l \tilde{B}_m) e_{ijk} \partial_j w_k dV - \int n_i e_{ijk} w_j (e_{klm} u_l \tilde{B}_m) dS, \tag{17}$$

and the decomposition of $\mathbf{B}$ and $\mathbf{w}$ is fulfilled:

$$w_k(\tilde{\mathbf{x}}, \tilde{t}) = \sum_l N_{kl}(\xi(\tilde{\mathbf{x}})) c_{kl}(\tilde{t}), \quad \tilde{B}_k(\tilde{\mathbf{x}}, \tilde{t}) = \sum_l N_{kl}(\xi(\tilde{\mathbf{x}})) d_{kl}(\tilde{t}).$$

Note, that here weights $c_{kl}(\tilde{t})$ and $d_{kl}(\tilde{t})$ are time-dependent.

The series representations of $\tilde{\mathbf{B}}$ and $\mathbf{w}$ are then plugged back into 17 and the equation is rewritten in the following form

$$\sum_{\text{cells}} \sum_{p,s} \int_{V_{\text{cells}}} N_{3p+k} N_{3s+k} c_{3p+k} \dot{d}_{3s+k} dV = \sum_{\text{cells}} \sum_{p,s} \int_{V_{\text{cells}}} (e_{ilm} u_l N_{3s+m} d_{3s+m}) e_{ijk} N_{3p+k,j} c_{3p+k} dV -$$

$$- \sum_{\text{cells}} \sum_{p,s} \int_{S_{\text{cells}}} n_i e_{ijk} N_{3p+j} c_{3p+j} (e_{klm} u_l N_{3s+m} d_{3s+m}) dS, \tag{18}$$

with a dot over d denoting a time derivative (with respect to the dimensionless time). As the subsequent step, similarly to what has been done in the previous subsections, we write 18 in the matrix-vector notation, introducing local matrices M^{local}_{ab} and K^{local}_{ab} as

$$M^{\text{local}}_{ab} = \int_{V_{\text{cell}}} N_a N_b dV, \;\; K^{\text{local}}_{ab} \equiv K^{\text{local}}_{3s+m\, 3p+k} = \int_{V_{\text{cell}}} e_{ilm} u_l N_{3s+m} e_{ijk} N_{3p+k,j} dV - \int_{S_{\text{cell}}} n_i e_{ikj} N_{3p+k} e_{jlm} u_l N_{3s+m} dS.$$

Then by adding up entires of M_{ab}^{local} and K_{ab}^{local} from all the cells, the matrices global M_{ab} and K_{ab} are assembled. Analogously to 14, the equation below has to be true independently of the coefficients c_a

$$c_a M_{ab} \dot{d}_b = c_a K_{ab} d_b,$$

that leaves us with the final matrix-vector equation:

$$\boxed{M_{ab}\dot{d}_b = K_{ab}d_b.} \tag{19}$$

1.6 Third step. Evolution in time

Equation 19 now needs to be discretized in time. For this purpose, the backward Euler is employed. The time then takes only discrete values:

$$d_b(\tilde{t}) \approx d_b(\tilde{t}_n) \equiv d_b^n,$$

where the upper index denotes a point along the time axis at which d_b is evaluated. Thus, at some point $\tilde{t} = \tilde{t}_n$ we have

$$M_{ab}^{n+1} \dot{d}_b^{n+1} = K_{ab}^{n+1} d_b^{n+1},$$

where it is assumed that both matrices are time-dependent. The last steps are

$$\dot{d}_b^{n+1} \approx \frac{d_b^{n+1} - d_b^n}{\Delta\tilde{t}}$$

$$M_{ab}^{n+1} \frac{d_b^{n+1} - d_b^n}{\Delta\tilde{t}} = K_{ab}^{n+1} d_b^{n+1}$$

$$\boxed{(M_{ab}^{n+1} - \Delta\tilde{t} K_{ab}^{n+1}) d_b^{n+1} = M_{ab}^{n+1} d_b^n} \tag{20}$$

The equation 20 is solved iteratively in the code, the set of coefficients d_b is found and the poloidal perturbation is evolved in time by $\Delta\tilde{t}$. So, by numerically solving eq. 20 n times we obtain the solution at $\tilde{t} = \tilde{t}^{n+1}$ provided that the initial conditions are specified

1.7 The code

The code is written in C++ programming language and currently under further development. The Finite element method is realized within the framework of the deal.II library (see *https://www.dealii.org*). The code is parallelized with use of Message Passing Interface (MPI). The parallelization has a great effect on the assembly of global matrices M_{ab}, K_{ab} and a global vector F_a, since in the MPI mode, the problem's domain is divided among a certain predefined number of processors participating in the computation so that each processor carries out the assembly only on its own part of the domain. An illustration on how the domain in our problem is divided among 6 processors is given in figure 3. For a bigger number of processors the time to solve equations 15 and 20 shortens as well.

1.8 Scaling of the code

The important feature of the code is that MPI is optimized by the deal.II library; and therefore, the scaling of the problem

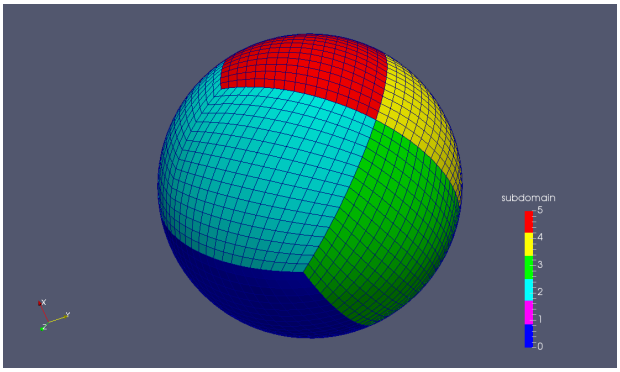

Figure 3: An illustration of utilization of parallel computing in the Finite element method. The manifold of the problem divined into 6 subdomains whereat 6 processors operate.

1.9 Results

To evolve the poloidal perturbation $\mathbf{B}_p$ a mesh of 11408 elements was used. The first order Lagrange interpolation polynomials were employed, which resulted in the total number of degrees of freedom in the problem = 156774. Among 6 processes used in the computing, the degrees of freedom were distributed as 28203, 27336, 26304, 25704, 26112 and 23115. The constant dimensionless timestep $\Delta\tilde{t}$ was set to 0.005, which in dimensional units equals 0.025 Myr. Figure 4 shows the evolution of $\mathbf{B}_p$ on a plane cut $x = 0$ at four consecutive time steps $t \equiv \tau\tilde{t} = 6.6, 12.9, 25.6$ and 39.4 Myr.

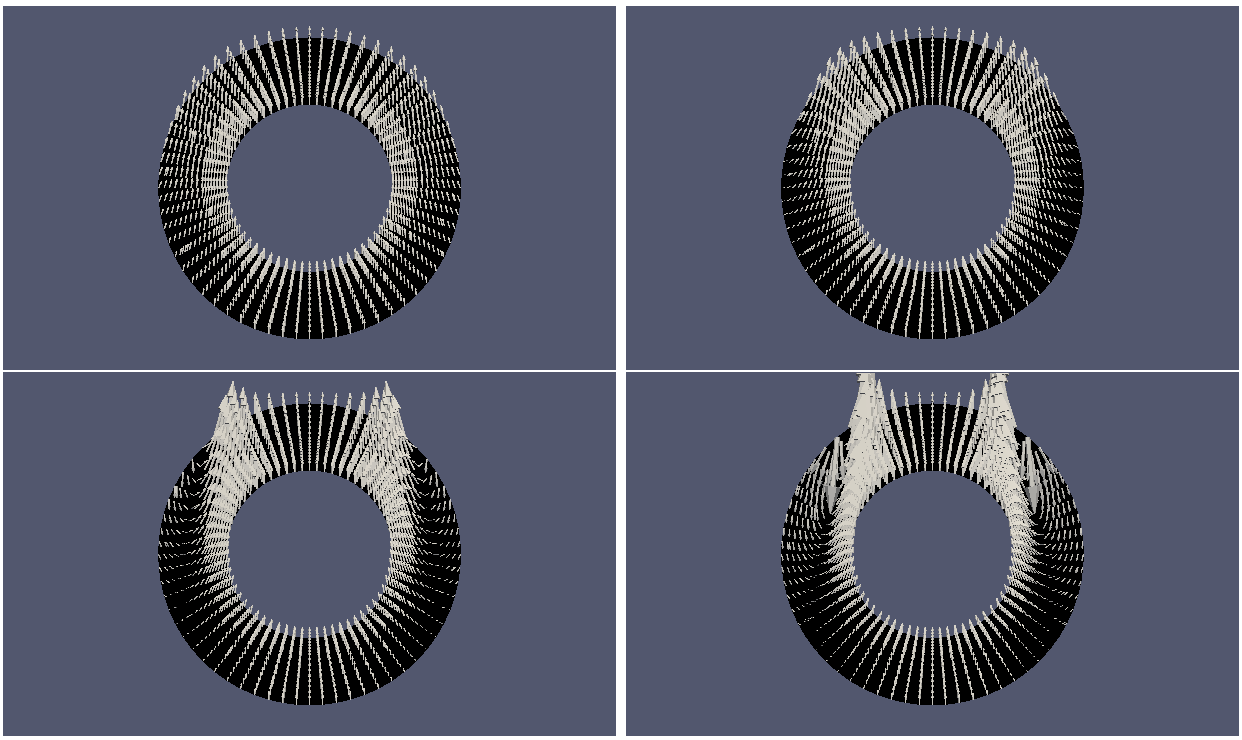

Figure 4: Four consecutive steps of evolution of the weak poloidal perturbation $\mathbf{B}_p$ in the presence of the background toroidal component $\mathbf{B}_t$. The poloidal component $\mathbf{B}_p$ on a plane cut $x = 0$ is shown. The arrows illustrate the vectors tangential to the field lines. Their size is linearly proportional to the magnitude of $\mathbf{B}_p$. The time $t \equiv \tau\tilde{t}$ has the following values: 6.6 Myr (top left panel), 12.9 Myr (top right panel), 25.6 Myr (bottom left panel) and 39.4 Myr (bottom right panel).