M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017, on USB (2017)

Computational Magnetohydrodynamics Using the deal.II Finite Element Library

K.S. Han, B.H. Park, and A.Y. Aydemir

National Fusion Research Institute, Daejeon, Korea
hanks318@nfri.re.kr

Abstract - *The finite element library deal.II has been used for the numerical solution of partial differential equations in a wide-range of fields. In this work, we apply it to nuclear fusion research. To investigate its applicability, we start with a time-independent, equilibrium problem and solve the Grad-Shafranov equation. Comparisons with known analytic and other numerical solutions confirm the validity and accuracy of our calculations. To use it in experimentally relevant cases, profiles for the pressure and toroidal magnetic field are needed. With some assumptions, these two profiles can be obtained from the experimental data. Results of these realistic equilibria are also shown to be consistent with the available data. Our next goal is to solve the "free-boundary" problem where the plasma boundary comes out of a self-consistent calculation, starting with a given distribution of external currents. Our longer-term plans include stability analysis of KSTAR-relevant equilibria using various magnetohydrodynamic (MHD) models.*

I. INTRODUCTION

Computational magnetohydrodynamics (MHD) has long been an integral part of magnetic fusion research, making important contributions to nearly all aspects of confinement theory and experimental efforts. Equilibrium in toroidal systems, and investigation of the linear and nonlinear stability of such equilibria using physics models of various sophistication are still major areas of study in fusion research.

Computational tools used to study equilibrium and stability have traditionally been "custom built" by individuals or a small group of researchers, who have painstakingly constructed complex codes essentially starting from scratch to address their particular needs. This process is slow, error-prone, and typically involves years of development time.

Over the past decade or so, however, the general computational community has increasingly concentrated on developing "frameworks", sophisticated set of integrated, general purpose tools, that can be used to construct codes for specific tasks. Since they are developed by dedicated groups of experts in rigorous debugging and validation procedures, they tend to be robust and reliable platforms that make higher-level code development easier, faster, and less prone to errors.

In this work, we investigate using the deal.II finite element library [1], [2] in equilibrium and stability applications in magnetic fusion research. The deal.II library provides a wide array of tools to solve partial differential equations using adaptive finite elements in a nearly dimension-independent way on parallel machines. It has interfaces to various meshing libraries, linear algebra and visualization packages and thus appears to be a nearly complete framework for developing advanced computational tools in fusion research.

II. EQUILIBRIUM IN TOROIDAL GEOMETRY

Our initial application of deal.II is in toroidal equilibrium calculations. Equilibrium in axisymmetric toroidal geometry is described by the Grad-Shafranov equation[3],

$$-\Delta^* \psi = \mu_0 R^2 p'(\psi) + F(\psi)F'(\psi), \tag{1}$$

which directly follows from the force-balance condition $\boldsymbol{J} \times \boldsymbol{B} = \nabla p$, where $\boldsymbol{J} = \nabla \times \boldsymbol{B}/\mu_0$ is the current density and p the plasma pressure. In cylindrical (R, ϕ, Z) coordinates, using $\partial/\partial\phi = 0$, the operator $\Delta^*\psi$ is given by

$$\Delta^* \psi = R \frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial \psi}{\partial R}\right) + \frac{\partial^2 \psi}{\partial Z^2}. \tag{2}$$

Here ψ is the poloidal flux function, in terms of which the magnetic field can be written as $\boldsymbol{B} = \nabla\psi \times \nabla\phi + F\nabla\phi$. Primes in Eq. 1 denote differentiation with respect to ψ : $p' = \partial p/\partial\psi$.

III. NUMERICAL METHODS

Multiplying Eq. 1 with a test function φ that satisfies the appropriate boundary conditions, the problem is cast in the “weak from” needed for finite element methods:

$$\int \frac{1}{R}\nabla\psi \cdot \nabla\varphi d\Omega = \int \left[\mu_0 R p' + \frac{FF'}{R}\right]\varphi d\Omega. \tag{3}$$

Expanding ψ in terms of arbitrary shape functions φ_j,

$$\psi(R, Z) = \sum_{j=1}^{N} \Psi_j \varphi_j(R, Z), \tag{4}$$

and letting the test function be one of the basis functions φ_i results in the linear system

$$A_{ij}\Psi_j = R_i. \tag{5}$$

What kind of solver to use depends on the matrix A_{ij}. In this problem, A_{ij} comes from a Laplace-like equation with spatially variable coefficients that leads to Symmetric Positive Definite(SPD) matrix. Since the appropriate choice for SPD matrix is Conjugate Gradients(CG) solver with Symmetric Successive Over Relaxation(SSOR) preconditionor, we use it to solve Eq. 5.

In a realistic case, the right-hand-side of Eq. 1 is in general a nonlinear function of the independent variable ψ. Since it is a nonlinear problem, solving for $\psi(R, Z)$ within a closed curve $\partial\Omega$ in the (R,Z) plane requires an iterative procedure. We adopt the simple Picard iteration

$$-\Delta^*\psi^{n+1} = \mu_0 R^2 p'(\psi^n) + F(\psi^n)F'(\psi^n). \tag{6}$$

To reduce the iteration count, we need to use some iteration scheme. Using a "relaxation parameter" θ, we can write

$$-\Delta^*[\theta\psi^{n+1} + (1-\theta)\psi^n] = \mu_0 R^2 p'(\psi^n) + F(\psi^n)F'(\psi^n). \tag{7}$$

The parameter θ is chosen to minimize the iteration count. This is done by trial and error. This scheme greatly reduces the number of iterations. For instance, in our case, it is reduced from 2358 to 244 times when $\theta = 0.277$. A sample of the actual code is shown in the Appendix.

IV. ANALYTIC SOLUTION

A well-known analytic solution for Eq 1 is given by [4]

$$\psi = \frac{\psi_0 R^2}{R_0^4}(2R_0^2 - R^2 - 4\alpha^2 Z^2), \tag{8}$$

which corresponds to

$$F' = 0, \quad p' = \frac{8\psi_0}{\mu_0 R_0^4}(1+\alpha^2). \tag{9}$$

Here α is a parameter that determines the shape of the equilibrium flux surfaces.

Our solution of Eq. 1 with deal.II using Eq. 9 is shown in Fig. 1 for $\alpha = 1$. Fig. 1a shows the numerically obtained flux surfaces, whereas Fig. 1b shows the error, $||\psi_{num} - \psi_{analytic}||$ where ψ_{num} means numerical solution and $\psi_{analytic}$ analytic solution.

For scalar problems as in Eq. 1, continuous Lagrange elements are generally used. They are called as Q_p elements in deal.II where p indicates the order of elements. Higher p yields higher convergence rate. Fig. 2 shows the placement of nodes within quadrilateral elements in deal.II, as p increases.

From the theory of finite element method, the L_2-norm of difference between the analytic solution and the numerical solution is related to the mesh size and the H_{p+1}-norm of the analytic solution by

$$||\psi_{num} - \psi_{analytic}||_{L_2} \leq Ch^{p+1}||\psi_{analytic}||_{H_{p+1}}. \tag{10}$$

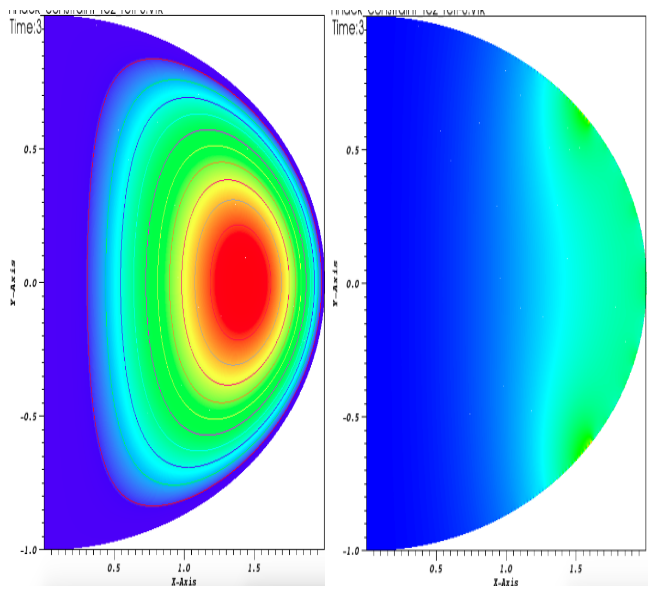
Here h represents mesh size. h is also related to N by

$$N \simeq \frac{|\Omega|}{(h/p)^d} = p^d\frac{|\Omega|}{h^d} \rightarrow h \simeq p\left(\frac{|\Omega|}{N}\right)^{1/d} \tag{11}$$

With Eq. 10 and Eq. 11, the error can be expressed as function of N and p

$$||\psi_{num} - \psi_{analytic}||_{L_2} \simeq C' p^{p+1} N^{-(p+1)/d}. \tag{12}$$

Here d represents the dimension of space (2 in our case). The convergence of $||\psi_{num} - \psi_{analytic}||_{L_2}$, in terms of N for $p = 1$, is shown in Fig. 3.



(a) ψ_{num} (b) $||\psi_{num} - \psi_{analytic}||$

Fig. 1: (a) Solution of Eq. 1 using the functions in Eq. 9 with zero-boundary condition. (b) The difference between the analytic solution in Eq. 8 and the numerical solution obtained using deal.II. The maximum error is approximately 0.01%

V. BOUNDARY AND PROFILE

1. Mesh and Boundary Condition

To solve the fixed-boundary equilibria, the boundary condition and profiles for $p(\psi)$, $F(\psi)$ in Eq. 1 should be specified within $\partial\Omega$. The several locations of plasma boundary can be measured in KSTAR. For the calculation, we use the actual boundary curve that comes from the experiment #13256 at 6.7*sec*. Within this boundary curve, the mesh can be easily generated by using Gmsh that is a finite element mesh generator. The created mesh is shown in Fig 4. On the boundary of this mesh, we set $\psi = 0$.

2. Profile for $p(\psi)$

Our model for $p(\psi)$ is as follows

$$p = p_0(1 - \rho^{\lambda_{p1}})^{\lambda_{p2}}, \tag{13}$$

Here ρ is the normalized radius of plasma. It is defined as

$$\rho = \sqrt{\frac{\psi - \psi_0}{\psi_l - \psi_0}} \tag{14}$$

where ψ_0 is poloidal flux on the magnetic axis and ψ_l on the boundary. Unfortunately profile for $p(\psi)$ can not be obtained in KSTAR. However, profile for ion temperature can be obtained very well from Charge Exchange Spectroscopy(CES). In the experiment #13256 at 6.7*sec*, CES data for ion temperature is shown in Fig. 5a. To estimate $\lambda_{p1}, \lambda_{p2}$, the ion temperature profile is used instead of pressure profile. The comparison between profile from CES and our model for

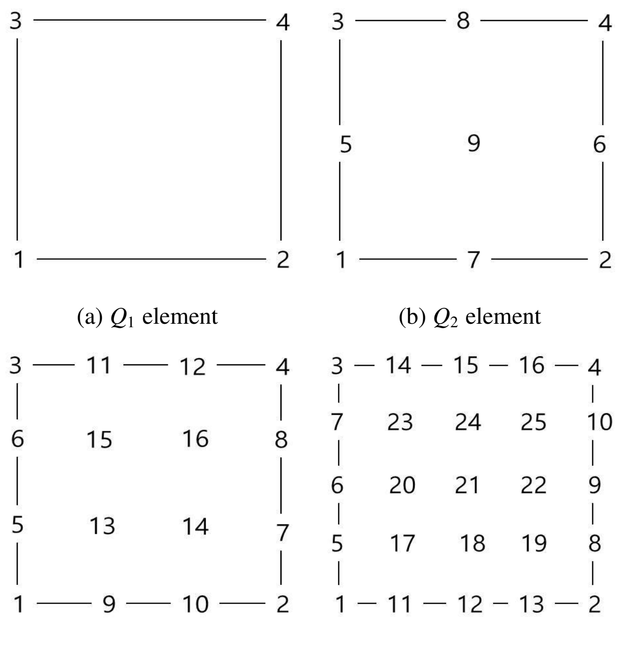


Fig. 2: Approximate picture for Q_p elements in 2D. N per cell is 4, 9, 16, and 25, respectively. That means ,as p increases, more finite element shape functions would be used in the calculation.

$\lambda_{p1} = 1.06, \lambda_{p2} = 0.63$ is shown in Fig. 5b. They look almost the same, so our model for $p(\psi)$ seems to be plausible.

The value of maximum on axis, p_0, can be calculated from β_T that can be measured experimentally. β_T is the ratio of the averaged plasma pressure to the magnetic pressure of toroidal magnetic field,

$$\beta_T = \frac{< p >}{B_\phi^2/2\mu_0}, \tag{15}$$

where

$$< p >= \int_0^1 p_0(1-\rho^{\lambda_{p1}})^{\lambda_{p2}}\,\mathrm{d}\rho. \tag{16}$$

Experimentally known value of β_T leads to

$$p_0 = \beta_T \frac{B_\phi^2}{2\mu_0}\left(\int_0^1 (1-\rho^{\lambda_{p1}})^{\lambda_{p2}}\,\mathrm{d}\rho\right)^{-1}, \tag{17}$$

Where B_ϕ is the magnetic field in toroidal direction. For simplicity, a choice for B_ϕ is usually the vacuum toroidal field at the geometric center of the chamber confining the plasma.

3. Profile for $F(\psi)$

Our model for $F(\psi)$ is as follows

$$F = F_{Amp}[1 + F_0(1-\rho^{\lambda_{F1}})^{\lambda_{F2}}]. \tag{18}$$

By definition of $F(\psi)$, it is related to the magnetic field in toroidal direction [3]. The profile of $F(\psi)$ can be obtained from the magnetic field in toroidal direction. However, it also

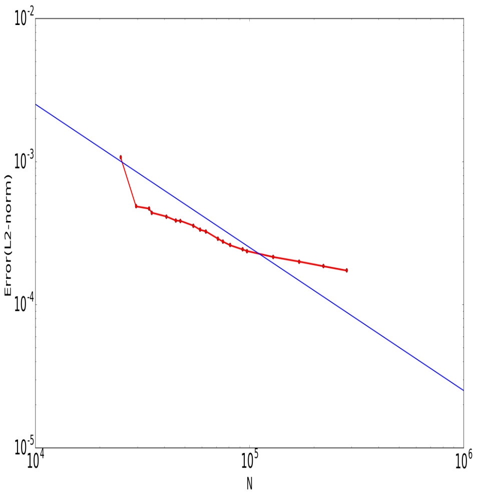


Fig. 3: Plot for $||\psi_{num} - \psi_{analytic}||_{L_2}$ in terms of N for $p = 1$. Since the plot is in log scale, the slope of trend line is very similar to the exponent of N in Eq. 12. The blue line indicates a line with a slope of -1 that is the exponent of N in this case.

can not be obtained in KSTAR. So we assume that $\lambda_{F1} \simeq \lambda_{p1}$, $\lambda_{F2} \simeq \lambda_{p2}$.

$F(\psi)$ is also related to the total poloidal current (plasma plus coil) by [4]

$$I_{polo}^{total} = \frac{2\pi}{\mu_0}F(\psi). \tag{19}$$

At $\rho = 1$, since the only contribution to I_{polo}^{total} is the current in Toroidal Field(TF) coils,

$$F_{Amp} = \frac{\mu_0}{2\pi}I_{TF}^{total}. \tag{20}$$

Total plasma current, that is usually denoted by I_p, can be measured experimentally. It also can be calculated from current denstiy in toroidal direction, J_ϕ. Since J_ϕ is related to ψ by $\mu_0 J_\phi = -\Delta^*\psi/R$ [3] where $-\Delta^*\psi$ can be calculated from $p(\psi)$ and $F(\psi)$, I_p can be obtained from $p(\psi)$ and $F(\psi)$ by

$$I_p = \int\left[Rp'(\psi) + \frac{F(\psi)F'(\psi)}{\mu_0 R}\right]\mathrm{d}R\,\mathrm{d}Z.$$

Substituting Eq. 18 into this equation yields

$$\begin{aligned} I_p &= \int Rp'(\psi)\,\mathrm{d}R\,\mathrm{d}Z \\ &+ \int \frac{A}{R}\rho^{\lambda_{F1}-2}(1-\rho^{\lambda_{F1}})^{\lambda_{F2}-1}[F_0 + F_0^2(1-\rho^{\lambda_{F1}})^{\lambda_{F2}}]\,\mathrm{d}R\,\mathrm{d}Z, \end{aligned} \tag{21}$$

where

$$A = \frac{1}{\mu_0}F_{Amp}^2\frac{\lambda_{F1}\lambda_{F2}}{2(\psi_0-\psi_l)}.$$

This is a quadratic equation for F_0. So the picard iteration should be used to calculate F_0. Applying the picard iteration to Eq. 21,

$$F_0^{n+1} = \frac{I_p - \int Rp'(\psi)\,\mathrm{d}R\,\mathrm{d}Z}{\int \frac{A}{R}\rho^{\lambda_{F1}-2}(1-\rho^{\lambda_{F1}})^{\lambda_{F2}-1}[1 + F_0^n(1-\rho^{\lambda_{F1}})^{\lambda_{F2}}]\,\mathrm{d}R\,\mathrm{d}Z}. \tag{22}$$

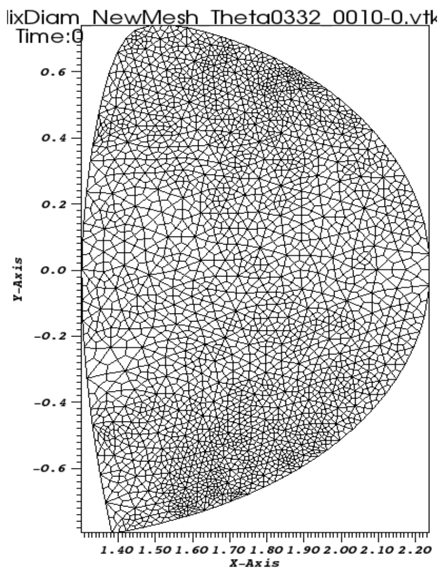


Fig. 4: Created mesh for the shot #13256 at $t = 6.7s$.

VI. RESULTS AND ANALYSIS

A solution using the actual boundary curve in Fig. 4 is shown in Fig. 6. We find that for this equilibrium $F_0 = 2.70 \times 10^{-3}$, implying the plasma is slightly paramagnetic. This is understandable because of the relatively low pressure of the plasma ($\beta_T = 1.33\%$).

The averaged magnitude of poloidal magnetic field on the boundary, denoted by $\overline{B_p}$, can be approximately calculated from κ(elongation) and I_p by

$$\overline{B_p} = \frac{\mu_0 I_p}{2\pi\overline{a}} \tag{23}$$

where

$$\overline{a} = a\sqrt{\frac{1+\kappa^2}{2}}\ (a = 0.5m\ in\ KSTAR).$$

The calculated $\overline{B_p}$ is $0.17T$. $\overline{B_p}$ also can be calculated from ψ by [3]

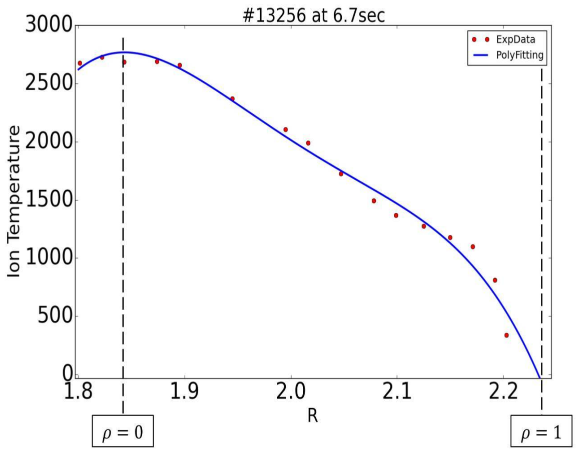
$$\overline{B_p} = \frac{\int_{\partial\Omega} |\psi|/R\,\mathrm{d}l}{\int_{\partial\Omega}\mathrm{d}l}. \tag{24}$$

The calculated value is $0.18T$, which agrees with $\overline{B_p}$ calculated from I_p, κ.

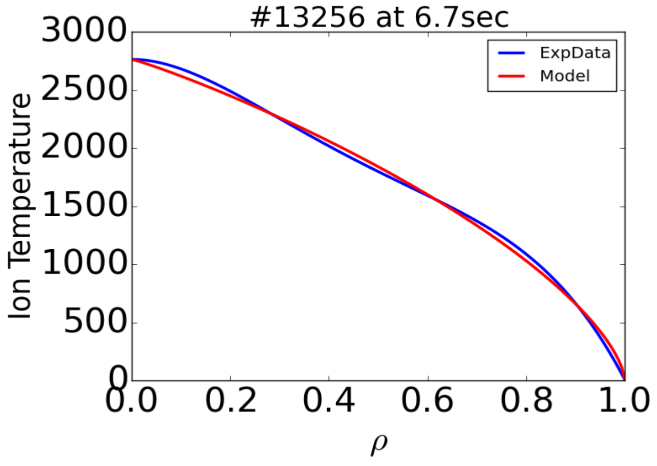
As it is mentioned before, the plasma is just slightly paramagnetic. Since F_0 is very small, the applied toroidal magnetic field is hardly affected by plasma. So the magnetic field in toroidal direction, on the major radius of KSTAR, would be $1.8T$ as it is applied. The toroidal magnetic field, on $Z = 0$, calculated from ψ is shown in Fig. 7. As it is expected, B_ϕ is almost $1.8T$ on $R = 1.8m$ that is the major radius.

In Eq. 16, plasma pressure is averaged over ρ. Now, since we know ψ, it can be averaged over the volume of plasma in the cylindrical coordinate system. So, instead of Eq. 16, the averaged pressure can be redefined as

$$< p >= \frac{\int_\Omega p_0(1-\rho^{\lambda_{p1}})^{\lambda_{p2}} R\,\mathrm{d}R\,\mathrm{d}Z}{\int_\Omega R\,\mathrm{d}R\,\mathrm{d}Z}. \tag{25}$$



(a)



(b)

Fig. 5: (a)Red dots represent measured experimental values that comes from #13256 at $6.7sec$. Based on these values, the blue line represents polynomial fitting. Radial position R would be normalized by $\rho \simeq R/R_l$ where R_l represents the radial position at the plasma boundary. (b)Comparison between polynomial fitting line and Eq. 13 for $\lambda_{p1} = 1.06, \lambda_{p2} = 0.63$.

By using this equation, $< p >$ calculated from ψ is $0.11atm$. It also can be calculated from experimental data β_T, B_ϕ, by the definition of β_T. The calculated $< p >$ is $0.17atm$. Both values look similar.

Like β_T, the poloidal beta is defined as

$$\beta_p = \frac{2\mu_0 < p >}{\overline{B_p^2}} \tag{26}$$

where

$$\overline{B_p^2} = \frac{\int_{\partial\Omega}(|\psi|/R)^2\,\mathrm{d}l}{\int_{\partial\Omega}\mathrm{d}l}.$$

In paramagnetic plasma, β_p must be smaller than 1 [4]. Using Eq. 25 and Eq. 26, the calculated β_p is 0.84 that is smaller than

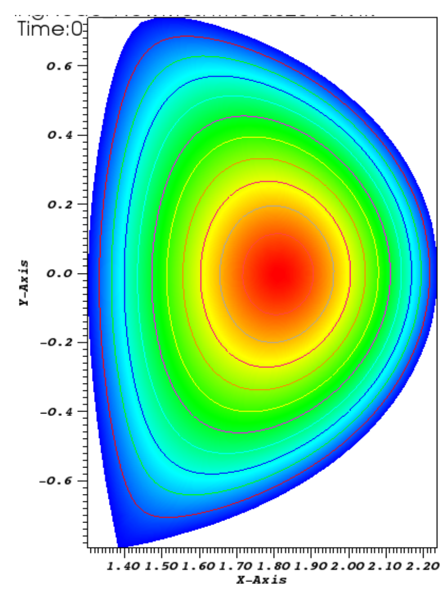


Fig. 6: Solution for $\psi(R, Z)$ of Eq. 1 for #13256 at $t = 6.7s$.

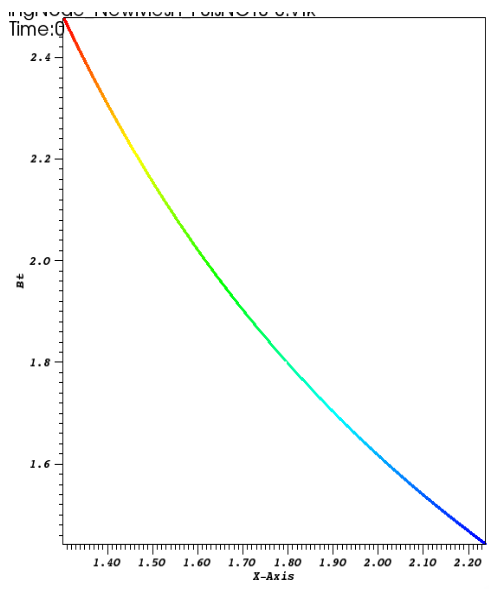


Fig. 7: Calculated B_ϕ from ψ on $Z = 0$

1 as expected. When β_T is increased from 1.33% to 1.73%, F_0 is almost zero that is 3.42×10^{-5}. In this case, since the plasma is neither paramagnetic nor diamagnetic, β_p should be 1. Our calculated value is 1.08 that is close to 1.

The simple calculations presented here demonstrate the relative ease with which a rather difficult nonlinear problem in a complex geometry can be tackled using deal.II.

VII. FUTURE WORK

Our immediate goal is to attack the more difficult "free-boundary" problem where the plasma boundary, instead of being specified a priori, comes out of a self-consistent calculation that makes use of the external poloidal field current data.

Studying the stability of these equilibria requires time-dependent calculations involving a set of hyperbolic equations. Here also deal.II provides the necessary tools, which we will be applying in the near future.

VIII. ACKNOWLEDGMENTS

This work was supported by MSIP, the Korean Ministry of Science, ICT and Future Planning, through the KSTAR project. The authors gratefully acknowledge the help of a referee whose extensive comments and criticism made essential contributions to the manuscript.

APPENDIX

Matrix A_{ij} and vector R_i in Eq. 5 would be assembled in the function below.

```
template <int dim>
void Step6<dim>::assemble_system ()
{
 const QGauss<dim> quadrature_formula(2);
 const RightHandSide<dim> right_hand_side;
 FEValues<dim> fe_values (fe,
                          quadrature_formula,
                          update_values |
                          update_gradients|
                    update_quadrature_points|
                          update_JxW_values);

 const unsigned int   dofs_per_cell
                        = fe.dofs_per_cell;
 const unsigned int   n_q_points
                   = quadrature_formula.size();

 FullMatrix<double> cell_matrix(dofs_per_cell,
                                dofs_per_cell);
 Vector<double>  cell_rhs (dofs_per_cell);

 std::vector<types::global_dof_index>
             local_dof_indices (dofs_per_cell);

 const Coefficient<dim> coefficient;
 std::vector<double> pre_sol(n_q_points);

 typename DoFHandler<dim>::
                    active_cell_iterator
 cell = dof_handler.begin_active(),
 endc = dof_handler.end();

 for (; cell!=endc; ++cell)
 {
   cell_matrix = 0;
   cell_rhs = 0;

   fe_values.reinit (cell);
```

```
    //Due to Picard iteration psi^n that
    //is previous solution would be used
    fe_values.get_function_values(
                            pre_solution,
                            pre_sol);

     for (unsigned int q_index=0;
          q_index<n_q_points; ++q_index)
     {
      //coefficient is 1/x in this problem
      const double current_coefficient
            = coefficient.value(fe_values.
                quadrature_point(q_index));
       for (unsigned int i=0;
             i<dofs_per_cell; ++i)
       {
        for (unsigned int j=0;
              j<dofs_per_cell; ++j)
        //matrix A_ij is assembled
        cell_matrix(i,j) +=
          (current_coefficient *
          fe_values.shape_grad(i,q_index) *
          fe_values.shape_grad(j,q_index) *
          fe_values.JxW(q_index));

        //vector R_i is assembled
        cell_rhs(i) +=
         (fe_values.shape_value(i,q_index) *
         right_hand_side.value(fe_values.
                 quadrature_point(q_index),
                 pre_sol[q_index],max_psi) *
                 fe_values.JxW(q_index));
         }
      }

     //Since A_ij is Sparse matrix, the
     //elements are calculated in local
     //matrix. They would be sended to global
     //matrix. So is R_i
     cell->get_dof_indices
          (local_dof_indices);
     constraints.distribute_local_to_global
                            (cell_matrix,
                             cell_rhs,
                        local_dof_indices,
                             system_matrix,
                             system_rhs);
  }
}
```

After A_{ij} and R_i are assembled, Ψ_j would be calculated by CG solver with SSOR preconditionor as follows.

```
template <int dim>
void Step6<dim>::solve ()
{
 SolverControl solver_control (2000,
                                1e-12);
 SolverCG<>  solver (solver_control);

 PreconditionSSOR<> preconditioner;
 preconditioner.initialize(system_matrix,
                              1.2);

 solver.solve (system_matrix, solution,
               system_rhs,preconditioner);

 constraints.distribute (solution);
}
```

The above two functions would be used for picard iteration with a "relaxation parameter" θ. The below part would be iterated until the solution converges.

```
//store PSI^n (previous solution)
pre_solution=solution;

//assemble matrix A_ij and vector R_i
assemble_system();

//calculate new RHS for PSI^(n+1)
//(1/theta)*[(theta-1)*A_ij*PSI^n+R_i]
system_matrix.vmult(tmp,pre_solution);
tmp *= (theta-1);
system_rhs += tmp;
system_rhs /= theta;

//solve PSI^(n+1)
solve ();
```

REFERENCES

1. W. BANGERTH, D. DAVYDOV, T. HEISTER, L. HELTAI, G. KANSCHAT, M. KRONBICHLER, M. MAIER, B. TURCKSIN, and D. WELLS, “The deal. II library, version 8.4,” *Journal of Numerical Mathematics*, **24**, *3*, 135–141 (2016).
2. W. BANGERTH, R. HARTMANN, and G. KANSCHAT, “deal. II–a general-purpose object-oriented finite element library,” *ACM Transactions on Mathematical Software (TOMS)*, **33**, *4*, 24 (2007).
3. J. P. FREIDBERG, *Ideal MHD*, Cambridge University Press (2014).
4. D. D. SCHNACK, *Lectures in magnetohydrodynamics: with an appendix on extended MHD*, vol. 780, Springer (2009).