SCALABILITY ANALYSIS OF DIRECT AND ITERATIVE SOLVERS USED TO MODEL CHARGING OF NON-INSULATED SUPERCONDUCTING PANCAKE SOLENOIDS

M. MOHEBUJJAMAN, S. SHIRAIWA, B. LABOMBARD, J. C. WRIGHT, AND K. UPPALAPATI

Abstract. A mathematical model for the charging simulation of non-insulated superconducting pancake solenoids is presented. Numerical solutions are obtained by the simulation model implemented on the Petra-M FEM platform using a variety of solvers. A scalability analysis is performed for both direct and preconditioned iterative solvers for four different pancakes solenoids with varying number of turns and mesh elements. It is found that even with two extremely different time scales in the system an iterative solver combination (FGMRES-GMRES) in conjunction with the parallel Auxiliary Space Maxwell Solver (AMS) preconditioner outperforms a parallelized direct solver (MUMPS). In general, the computational time of the iterative solver is found to increase with the number of turns in the solenoids and/or the conductivity assumed for the superconducting material.

Key Words: Direct Solver, Iterative Solver, Scalability Analysis, Superconductor, Non-insulated Superconductor

1. Introduction. The discovery of high-temperature superconductors (HTS) [14, 26] opens a new chapter in both scientific and engineering fields for producing high-field superconducting magnets. Non-insulated superconductors in particular show great promise [13]. However, the numerical simulation of a non-insulated superconducting magnet is computationally expensive due to its highly non-linear electromagnetic behavior [35]. Though for simple geometries, the analytical solutions are possible to find under uniform external magnetic field [9, 22], time-dependent magnetic field simulation still remains a challenge to the scientific community even with the advanced computing facilities. This is because of (i) huge simulation domain as it includes air region along with the magnet that can produce multi-billions degrees of freedom (dofs) even taking advantage of symmetries (ii) complex magnet and conductor path geometry (iii) materials with widely disparate conductivity that can include an air region, \( \sigma_{\text{air}} = 1.0 \text{ S/m} \), in direct contact with a superconducting region, \( \sigma_{\text{hts}} = 10^{15} \text{ S/m} \), producing a system matrix in a finite element discretization of the PDEs that can be highly ill-conditioned, and (iv) null-space of the curl-curl operator in the governing Maxwell equations. Consider a sparse linear system

\[
Ax = b \quad \text{with} \quad A \in \mathbb{R}^{N \times N},
\]

where \( N \) is the order of the matrix \( A \). The sparse direct solvers usually compute the \( LU \) decomposition (or its variant) of the system matrix \( A \) using its sparsity pattern in an efficient way so that it becomes easier to compute and store the factors. The direct solver inverts the system exactly, provides a very robust solution, and easy to use. For solving a typical PDE using a sparse direct solver, the computational cost grows as \( O(N^2) \) in 2D and \( O(N^{7/3}) \) in 3D, and memory requirement grows as \( O(N \log N) \) in 2D and \( O(N^{4/3}) \) in 3D [5, 24]. Over the last few decades, due to the massive
improvement in the sparse direct solvers, e.g., MULTifrontal Massively Parallel sparse direct solver (MUMPS) \(^1\), PARDISO \(^2\), STRUMPACK \(^3\), SuperLU \(^4\), or UMFPACK \(^5\) have become popular among the researchers. Notwithstanding the advent of robust direct solvers, as the problem size and complexity increase, they require increasing computational time and computer memory and thus often fail to produce sufficiently resolved long range time-dependent solutions due to the limit of simulation time and memory constraint. Iterative solvers, e.g., CG \(^6\), BiCGSTAB \(^7\), MINRES \(^8\), or GMRES \(^9\) based on the Krylov subspace with appropriate preconditioners outperform \(^10\) over the direct solver techniques in many complex problems. The computational cost can be of \(O(N)\) or worse, and linear memory requirement \(O(N)\) for the choice of the iterative solver and the preconditioner type \(^11\). The iterative solver does not solve the system exactly; it starts with an initial guess and continues to improve the solution in each iteration until the error/residual is less than a specified tolerance. The availability of good preconditioners is one of the main hurdles while using iterative solvers. Without good preconditioners, the issue of convergence arises, and the use of the iterative solver is not a good idea.

In this work, we explore numerical solution methods to simulate the charging behavior of non-insulated superconducting pancake solenoids. The mathematical model considers the magnetic field due to the current flow in complex 3D geometries in which a superconducting wire or coil is co-wound with a conducting metal, rather than an insulator, separating the turns. The superconductor is extremely thin with very high conductivity compared to the physical dimensions and conductivities of co-wound metals, and air. This introduces two extremely different time scales in the system which makes harder for the solvers to produce desire solutions. The objective of this research work is to find efficient solvers for the present work that can scale to multi-billions degrees of freedom anticipated in future applications. The finite element solutions for the fully discrete scheme are obtained by using an iterative solver combination (FGMRES-GMRES) with AMS preconditioner \(^12\), \(^13\) which is built in the hypre library \(^14\), and the direct solver MUMPS. A scalability analysis for both direct and iterative solvers is presented. The scalability of the AMS preconditioner with conjugate-gradient (CG) method for low conductive materials is presented in \(^15\). To the best of our knowledge, the use of the robust FGMRES-GMRES solver with AMS preconditioner in simulating magnetic field with widely disparate material properties is new. All of the computational experiments presented in this paper are done in Petra-M \(^16\).

The paper is organized as follows: In Section \(^2\) we present mathematical modeling for the governing equations of magnetic field simulation. A brief description of the computational platform Petra-M is given in Section \(^3\). In Section \(^4\) we present the physical domains of the non-insulated superconducting pancake solenoids considered, provide a details description of the numerical experiment techniques, represent the scalability analysis for both direct and iterative solvers. Here the principal finding of this work is shown in both tabular and graphical forms, that the iterative solver with AMS preconditioner outperforms over the direct solver even with two extremely different time scales present in the system. In Section \(^5\) we present conclusions and future research directions. Finally, in the appendix, we show the Biot-Savart magnetic field computation for the spiral coils.

---

\(^1\)http://mumps.enseeiht.fr/
\(^2\)https://www.pardiso-project.org/
2. Mathematical Modeling. In this paper, we consider a time varying magnetic field which is excited by imposing voltage to a body of conducting coil. The conducting coil is not strictly contained in the computational domain Ω as its two disjoint electric ports Γ_E and Γ_J touch the domain boundary ∂Ω. We present the mathematical modeling of the time-dependent magnetic field following the frequency domain modeling in [33]. We neglect the displacement current density because we are concerned with diffusive and not with wave timescales. Therefore, the Ampere’s law in differential form [7]:

\[ \nabla \times \mathbf{H} = \mathbf{J} = \sigma \mathbf{E}, \]  

(2.1)

where \( \mathbf{H} \) the magnetic field strength, \( \mathbf{J} \) the current density, \( \mathbf{E} \) the electric field strength, and \( \sigma \) the conductivity. For simplicity we consider \( \sigma \) as a positive constant in this paper rather than a tensor. Dividing both sides by \( \sigma \), and taking curl operator

\[ \nabla \times \frac{1}{\sigma} \nabla \times \mathbf{H} = \nabla \times \mathbf{E}. \]  

(2.2)

Faraday’s law of induction:

\[ \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} = -\mu_0 \frac{\partial \mathbf{H}}{\partial t}, \]  

(2.3)

for the magnetic flux density \( \mathbf{B} = \mu_0 \mathbf{H} \), where \( \mu_0 \) is a material independent parameter called the permeability constant. The equation (2.3) ensures that if the initial magnetic field strength is divergence free then \( \nabla \cdot \mathbf{H} = 0 \) holds for all time. We assume Γ_E and Γ_J are perfect conductors, and thus the tangential component of the electric field vanishes there. Thus, we consider the following no-flux boundary conditions [8, 33]

\[ \mathbf{E} \times \hat{n} = \frac{1}{\sigma} \nabla \times \mathbf{H} \times \hat{n} = 0, \quad \text{in} \quad \Gamma_E \cup \Gamma_J \times (0, T], \]  

(2.4)

\[ \mu_0 \mathbf{H} \cdot \hat{n} = 0, \quad \text{in} \quad \partial \Omega \times (0, T], \]  

(2.5)

where \( \hat{n} \) is the outward unit normal vector to the boundary, and \( T \) the simulation time. We apply a fixed non-zero voltage \( V \) at \( \Gamma_J \) and zero voltage at \( \Gamma_E \), this potential difference drives the current to pass along the coil. Taking the dot product of \( \hat{n} \) on both sides of (2.3) and using (2.5) we have \( \nabla \cdot (\mathbf{E} \times \hat{n}) = 0 \) on \( \partial \Omega \). Assuming the boundary \( \partial \Omega \) a simply-connected surface, there exists a surface potential \( v \) such that

\[ \mathbf{E} \times \hat{n} = \frac{1}{\sigma} \nabla \times \mathbf{H} \times \hat{n} = \nabla v \times \hat{n} \quad \text{on} \quad \partial \Omega, \]  

(2.6)

where \( v|_{\Gamma_J} = V \) is a non-zero constant and \( v|_{\Gamma_E} = 0 \). Combining the equations (2.2) - (2.3), and (2.6) we have the following time-dependent governing equations for the magnetic field simulation of the voltage excitation problem

\[ \frac{\partial \mathbf{H}}{\partial t} + \frac{1}{\mu_0} \nabla \times \frac{1}{\sigma} \nabla \times \mathbf{H} = 0, \quad \forall (x, t) \in \Omega \times (0, T], \]  

(2.7)

\[ \frac{1}{\sigma} \nabla \times \mathbf{H} \times \hat{n} = \nabla v \times \hat{n}, \quad \forall (x, t) \in \partial \Omega \times (0, T], \]  

(2.8)

\[ \mathbf{H}(x, 0) = \mathbf{0}, \quad \forall x \in \Omega, \]  

(2.9)

where \( v|_{\Gamma_J} = V \) and \( v|_{\Gamma_E} = 0 \).
We note that this is a diffusion equation for the magnetic field with characteristic time given by $t_d = \mu_0 \sigma l^2$, where $l$ is a characteristic length scale. In the simulations shown in the following sections, the time is normalized by this diffusion time, where $l = 1.14 m$ and $\sigma = 2 \times 10^6 S/m$ are used. Additionally, the magnetic fields are normalized by the field produced by a single turn coil with same current ($I = 1000 A$), defined as $B_0 := \mu_0 I / l$.

3. Software and Computational Facility. The time-dependent equations (2.7)-(2.9) was discretized fully by the backward-Euler timestepping scheme and solved on the Petra-M finite element analysis platform. This open source platform allows for constructing a geometry, creating a mesh, assembling and solving the finite element linear system, and solving and visualizing the results using a user friendly graphical interface (GUI). Petra-M uses various open source software. In particular, it uses MFEM modular finite element library for the FEM linear system assembly. The weakform PDE interface in Petra-M allows for defining a mixed form PDE system we solved in this paper by choosing form integrators available in the MFEM library from menus. Petra-M combines GUls with the python scripting, allowing for rapidly developing the simulation model with very little coding effort. For example, the inner solver with pre-conditioner is defined in the code segment is given Fig. 3.1. Then, the outer solver is configured to use this inner solver using GUI. All simulations are done on the ‘Engaging’ cluster computer at the Massachusetts Institute of Technology, in which one node consists of 32 cores and 512 GB RAM memory.

Fig. 3.1: Petra-M code segment defines preconditioned inner solver.

4. Numerical Experiments. For our simulations, we consider four different geometries for the charging pancake solenoids: One turn single pancake (T-1), ten turns double pancakes (T-10) with each pancake has five turns, twenty turns double pancakes (T-20) with each pancake has ten turns, and thirty turns double pancakes (T-30) with each pancake has fifteen turns. The cross-section of the HTS spiral coils are rectangular. The thickness and height of the HTS coil are 0.16 cm and 0.3 cm, respectively for all of the four models. In each of the geometry, the HTS coil is co-wound with copper with a rectangular cross-section, which is then co-wound with stainless steel. The copper and stainless steel co-wounds help not to arise numerical issues as they prevent the superconducting coil from directly touching the air. The copper co-wound spiral coil preserves the spiral pattern of the HTS coil, keeping the uniform gap between turn to turn. A sufficiently large air domain is considered surrounding the stainless steel body so that the magnetic field lines do not go beyond it. Finally the terminals of the copper co-wound HTS coil are extended to the air boundary by solid copper rectangular bars. The two copper terminals are extended to
the boundary so that external voltage can be applied to one of them, and the voltage drives the current to flow along the spiral coil and the magnetic field is produced. Thus, in this work, we consider forced voltage excitation coil \cite{27,33}, and the voltage is adjusted to have the same current for all four models \cite{10}.

The physical properties of the different models are described below:

**4.1. T-1 Model.** In the single pancake T-1 model solenoid, the HTS coil has only one turn with center at the origin. A 1m$^3$ box centered at the origin, is considered as the air boundary. The Figures 4.1-4.2 show the shape of different material components in the T-1 model.

**4.2. T-10 Model.** The T-10 model geometry is a double pancake solenoid. Each of the pancakes has five HTS turns. The turns in the upper pancake spiral in and the turns in the lower pancake spiral out so that the current direction remains same, and their innermost turns are connected by a vertical joggle. The gap between two turns is 1.32 cm. The HTS coil and its copper co-wound with two electric ports are shown in Fig. 4.3. The electric ports are extended to the air boundary. The copper co-wound HTS is again co-wound with stainless keeping the air gap between two pancakes as shown in Figure 4.4. For the finite element simulations, we consider a cylindrical air domain of base diameter 100cm, height 114cm, and its axis is parallel to the pancakes so that the magnetic field lines remain inside the computational domain.

**4.3. T-20 Model.** The T-20 model geometry is also a double pancake solenoid and its physical properties are similar to the T-10 model except each of the pancakes
has ten turns HTS spiral coil. The gap between two turns is 0.65 cm. The HTS coil with two electric ports of the T-20 model is showing in Fig. 4.5. The physical properties of the air domain are the same as those in the T-10 model.

4.4. T-30 Model. The T-30 model is a double pancake charging solenoid where each pancake has a fifteen turns HTS coil. Dimensions of each component of the T-30 model are kept almost the same as those in the T-10 and T-20 models. The gap between two turns is 0.45 cm. The HTS coil and copper co-wound HTS coil with electric ports in the T-30 model are showing in Figure 4.6. The physical properties
4.5. Iterative Method. For the iterative solver, we consider the flexible inner-outer Krylov subspace methods [36, 40] which allow varying the preconditioner from one iteration to another. Using Krylov methods, instead of solving the linear system in equation (1.1), we solve a modified system such as the following right preconditioned equation

\[ AM^{-1}Mx = b, \]  

into two steps as

\[ AM^{-1}y = b, \quad \text{with} \quad Mx = y, \]  

having a fixed preconditioner \( M \). In flexible inner-outer method, we are allowed to use a different matrix, say \( M_j \), at each iteration. In inner-outer approach, the preconditioner itself can be a Krylov subspace method, and thus very appealing.

FGMRES [36] is used as the outer solver and GMRES as the inner solver. FGMRES stands for flexible GMRES, which is a variant of GMRES method but more robust. FGMRES algorithm is essentially the GMRES algorithm but with variable preconditioning. Except for the variable preconditioning, the only difference from the standard GMRES is that in FGMRES the preconditioned vectors are saved and used them to update the solution. For constant preconditioner, the FGMRES and GMRES algorithms are mathematically equivalent. In this paper, we use the AMS preconditioner to the GMRES solver and GMRES solver preconditioner to the FGMRES solver.

4.6. Iterative Solver Selection. In this experiment, we show the performance of FGMRES, GMRES, and BiCGSTAB as inner or outer solver with AMS and GS (Gauss-Seidel) preconditioners. We consider T-1 model with 6137426 dofs, simulation end time \( T = 0.5 \), time step size \( \Delta t = 0.5 \) (that is, a single time step solve), 4 nodes, 64 cores, and the parameters in the inner-outer solvers are given in Table 4.1.

We record the number of iterations and solving time in Table 4.2 taken by a combination of solvers and preconditioners. We observe, FGMRES as the outer solver and GMRES as the inner solver with AMS preconditioner takes 4 iterations with least solving time 31.43s, and thus choose this combination for all simulations with the iterative solver.
As a direct solver, we use MUMPS. The lowest order edge element is used for all simulations. The scalability analysis is performed with the iterative and direct solvers for T-1, T-10, T-20, and T-30 models. We used the following parameters values: $\sigma_{hts} = 1.0 \times 10^{15} S/m$, $\sigma_{cu} = 4.01 \times 10^{8} S/m$, $\sigma_{fe} = 2.0 \times 10^{6} S/m$, and $\sigma_{air} = 1.0 S/m$, as the conductivity of HTS, copper, stainless steel, and air, respectively, and the magnetic permeability constant $\mu_0 = 4\pi \times 10^{-7} H/m$.

![Plot of the cross section along the xy-plane of $B_z$ in T-10 model (lower view).](image)

**Table 4.1:** Parameters those are used in the flexible inner-outer method.

| Outer Solver | Inner Solver | Preconditioner | Iteration | Time   |
|--------------|--------------|----------------|-----------|--------|
| FGMRES       | GMRES        | GS             | Does not converge | –      |
| FGMRES       | BiCGSTAB     | AMS            | 3         | 44.30s |
| FGMRES       | GMRES        | AMS            | 4         | 31.43s |
| FGMRES       | FGMRES       | AMS            | 4         | 40.09s |
| GMRES        | GMRES        | AMS            | 5         | 49.34s |
| GMRES        | GMRES        | AMS            | 4         | 34.21s |

**Table 4.2:** Performance of different solvers in the Inner-Outer method.

4.7. **Code Verification and Model Comparison.** To verify our finite element method (FEM) code and to compare the four different models, we compute $B$ at the origin. We compare the stationary (fully charged) phase of $B$ with the magnetic field computed by the Biot-Savart law. We also compare the magnetic field profiles for all four models.

The voltage excitation problem is solved using FEM so that the same current $I = 1000A$ remains on the ports for all the models. We used coarse meshes of the
domains, a normalized timestep size $\Delta t = 3.06$, and ran the simulations until the normalized end time $T = 460$ for each of the models. For all these simulations, we used the direct solver.

The analytical formula for $B$ due to the current flow in the spiral HTS coil is derived by assuming the coil as filament, representing it as a vector equation and using the Biot-Savart law. In the Biot-Savart law, the uniform current $I = 1000\, A$ is assumed everywhere in the filament. The Biot-Savart computation of the magnetic field for the spiral coils are presented in Appendix A.

In Table 4.3, we present the FEM magnetic field at the stationary phase and the magnetic field computed from the Biot-Savart law. We observe a good agreement among the results of the two methods. Assuming the Biot-Savart magnetic fields as a benchmark, we computed the relative errors of the FEM magnetic fields and presented them in the last row. All the relative errors are less than 3%. The tabular values clearly show that as the number of turns increases the field strength gets stronger. Moreover, it is observed that the field strengths in T-20, and T-30 models are approximately double and triple as found in the T-10 model, which seems consistent as the field strength should be proportional to the number of turns. Since in T-1 model, the HTS does not have a full turn, we found a weaker field strength than what it supposed to be for a full one-turn model.

| Method      | Model | T-1 | T-10 | T-20 | T-30 |
|-------------|-------|-----|------|------|------|
| Biot-Savart |       | 3.02| 49.71| 98.07| 148.60|
| FEM         |       | 2.94| 50.98| 99.25| 150.14|
| Relative Error |     | 2.63%| 2.55%| 1.20%| 1.04% |

Table 4.3: For each model, the fully charged magnetic field strength at the origin.

For each of the model, the magnetic field profile is computed, normalized and presented altogether in Fig. 4.8. These profiles give us the full charging time of the models and observed that as the number of turns increases, the charging time increases as well. That is, the more turns in the coil the model has higher inductance.
4.8. Number of turns versus computational time. Now we want to compare the four different models in terms of the required computational time by the iterative solver for a single timestep solve in the FEM simulations. Since the inductances of the models are not the same, for a fair comparison, first we determine the appropriate timestep size for each of them. From the computed fully charged value of $\|B\|$ at the origin, we determine its 90% growth time, and divide it by a fixed number and set it as the timestep size $\Delta t$ of a model. We use the fixed number equal to 100 in this experiment. We find $\Delta t = 0.0098, 0.1439, 0.5572$, and $1.4083$ for the T-1, T-10, T-20, and T-30 model, respectively. We generate meshes for each of the models so that they all provide closely to 2.6 million dofs. With the above stated timestep sizes, we run the simulations of the respective model, use the proposed iterative solver with a single core processor to avoid communication time and record the single timestep solving time in Table 4.4. From the fourth column in Table 4.4 we see that as the number of turns increases the solving time increases. As the number of turns increases, the length of the HTS coil becomes longer, and consequently, its contribution to the system matrix gets stronger. Thus, the system matrix becomes more ill-conditioned and it becomes harder to solve for the iterative solver.

We note that for all these four simulations we do the profiling of our code, and find that the sparse matrix-vector multiplication in the iterative process is a major time consuming step.

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Model} & \text{dofs} & \Delta t & \text{Total Wall Clock Time} \\
\hline
T-1 & 2751768 & 0.0098 & 418.75s \\
T-10 & 2664168 & 0.1439 & 8860.03s \\
T-20 & 2680759 & 0.5572 & 10108.69s \\
T-30 & 2624896 & 1.4083 & 23990.02s \\
\hline
\end{array}
\]

Table 4.4: Time step used in simulation and solving time required by the iterative solver for four models. Note that time step is chosen to be 1% of the charging time.

4.9. Computational time versus conductivity. In this section, we investigate how the computational time of the iterative solver varies as the assumed conductivity of the HTS increases. We consider the T-30 model with a problem size of 330963 dofs, use 1 node with 1 core, and normalized time stepsize $\Delta t = 1.4083$. We vary the conductivity from $10^7$ to $10^{20}$ uniformly and run the simulations for a single time step solve, that is, $T = 1.4083$. The computational times are recorded in Table 4.5 and represented in Fig. 4.9. We observe that the solving time increases sharply as the conductivity of the HTS increases from $10^7$ to $10^{13}$. If we increase the conductivity further, the computational time remains almost of the same order. As the assumed conductivity of HTS increases, it starts introducing two extremely different time scale in the system, and thus the system matrix becomes more ill-conditioned and consequently, it becomes harder for the iterative solver to solve the system.

4.10. Parallel Scalability Analysis. Parallel scalability analysis is widely used to measure the performance of parallel codes as the problem size and the number of computer processor cores increase. It helps to predict the performance of a large number of cores on a large problem size based on the performance of a small number of cores on small problem size. In this section, we investigate the parallel scalability for the two solvers, the direct solver MUMPS, and the precondi-
SCALABILITY ANALYSIS OF DIRECT AND ITERATIVE SOLVERS

Table 4.5: Solving time increases as the conductivity of HTS does.

| $\sigma_{hts} (S/m)$ | $10^7$ | $10^8$ | $10^9$ | $10^{10}$ | $10^{11}$ | $10^{12}$ | $10^{13}$ |
|----------------------|--------|--------|--------|------------|------------|------------|------------|
| Wall Clock Time (s)  | 347.25 | 353.67 | 395.91 | 471.44 | 730.44 | 1053.17 | 1172.45 |

Table 4.6: Weak scaling: T-1 model.

| Nodes | Cores | dofs     | MUMPS  | WSE    | FGMRES-GMRES | WSE    |
|-------|-------|----------|--------|--------|--------------|--------|
| 1     | 2     | 88437    | 14.35s |        | 7.77s        |        |
| 1     | 16    | 687425   | 77.03s | 18.63% | 16.00s       | 48.56% |
| 4(5)  | 128   | 5415890  | 787.48s| 1.82%  | 43.38s       | 17.91% |

Table 4.6 shows the weak scaling performances of the direct and iterative solvers with the T-1 model. At first, we solve the problem of size 88437 dofs with 1 node.
and 2 cores. In the next refinement, the problem size is of 687425 dofs which is approximately 8 times the previous problem’s dofs. To keep the workload the same on each core, we increase the number of cores to 16, solve the problem, and record the solving time. We repeated the same procedure in the next refinement when the problem size increased to 5415890 dofs. In this case, the direct solver could not solve the problem without using less than 5 nodes while the iterative solver used only 4 nodes, that is, the iterative solver required less memory than the direct solver. The computational time taken by the solvers to solve a problem was represented as wall clock time. We plotted the wall clock time versus the dofs for both the direct and iterative solvers in Fig. 4.10. It is observed that the direct solver computational time remains always higher than the computational time of the iterative solver, that is, there is no crossing point. Also, as the problem size increases the computational time of the direct solver becomes much higher than that of the iterative solver.

The weak scaling efficiency is calculated as defined above and plotted against the number of cores employed in Fig. 4.11. We observe the weak scaling efficiency of the iterative solver is higher than that of the direct solver. As the problem size increases, the direct solver weak scaling efficiency drops to a factor of 10 while the iterative solver weak scaling efficiency drops only to a factor of 3.

| Nodes | Cores | dofs   | MUMPS  | WSE  | FGMRES-GMRES | WSE  |
|-------|-------|--------|--------|------|--------------|------|
| 1     | 5     | 273837 | 25.77s | 93.14s |              |      |
| 1     | 16    | 853921 | 56.07s | 45.96% | 130.90s      | 71.21% |
| 2     | 34    | 1860729| 83.82s | 30.74% | 164.82s      | 56.51% |
| 2     | 40    | 2168158| 89.38s | 28.83% | 221.17s      | 42.11% |
| 4     | 124   | 6771172| 275.80s| 9.34%  | 512.87s      | 18.16% |
| 10    | 316   | 17289872| 1002.37s| 2.57%  | 576.32s      | 16.16% |

Table 4.7: Weak scaling: T-10 model.

To study the weak scaling of the solvers with the T-10 model, we solve the problem with 273837, 853921, 1860729, 2168158, 6771172, and 17289872 dofs employing 5, 16, 34, 40, 124, and 316 cores, respectively so that the workload on each core remains same. We recorded the computational time and WSE for both solvers and represented
SCALABILITY ANALYSIS OF DIRECT AND ITERATIVE SOLVERS

Fig. 4.12: Measured computational
time versus degrees of freedom.

in Table 4.7. We plotted the wall clock time versus dofs in Fig. 4.12. We notice that for problems with lower dofs, the direct solver outperforms over the iterative solver, but for higher dofs the preconditioned iterative solver solves the problem much faster than the direct solver. And at around 10 million dofs there is a crossing point where the direct solver computational time exceeds the iterative solver time.

We also plotted the WSE versus the number of cores employed in Fig. 4.13. We observe that as we increase the number of cores employed, the iterative solver WSE remains always higher than that of the direct solver.

| Nodes | Cores | dofs      | MUMPS | WSE  | FGMRES-GMRES | WSE  |
|-------|-------|-----------|-------|------|--------------|------|
| 1     | 5     | 1094858   | 186.39s|      | 301.85s     |      |
| 2     | 25    | 5418938   | 446.72s| 41.72% | 550.87s | 54.80% |
| 8(12) | 86    | 18628074  | 1175.38s| 15.86% | 730.84s | 41.30% |

Table 4.8: Weak scaling: T-20 model.

Table 4.8 represents the weak scaling performance with the T-20 model. In this case, the initial problem was a size of 1094858 dofs, and we solved it with 1 node
and 5 cores. In the next refinement, the dofs increases to 5418938 which is 5 times more than before, and thus we solve this using 2 nodes and 25 cores. We repeat this process one more time and record the solving time. That is 5418938/1094858 ≈ 25/5 and 18628074/5418938 ≈ 86/25. When the problem gets 18628074 dofs, the minimum memory requirement for the direct solver is 12 nodes while the iterative solver can solve it using only 8 nodes. We plotted the solving time versus dofs in Fig. 4.14. It shows, for low dofs, the computational time taken by the direct solver is less than that of the iterative solver. But as the problem complexity in terms of the dofs increases, the direct solver computational time becomes much higher than that of the iterative solver. We observed that the direct solver wall clock time exceeds the iterative solver wall clock time at around 6 million dofs.

From Fig. 4.15, we also observe, the weak scaling efficiency of the iterative solver is much higher than that of the direct solver. As the dofs increases the direct solver weak scaling efficiency drops at a higher rate compared to the iterative solver.

| Nodes | Cores | dofs    | MUMPS | WSE   | FGMRES-GMRES | WSE   |
|-------|-------|---------|-------|-------|-------------|-------|
| 1     | 5     | 330963  | 77.89s|       |             | 334.74s|
| 1     | 27    | 1768029 | 175.02s| 44.50%| 384.18s     | 87.13%|
| 2     | 41    | 2624896 | 408.32s| 19.08%| 415.15s     | 80.63%|
| 8     | 203   | 13031432| 1146.46s| 6.79% | 751.22s     | 44.56%|

Table 4.9: Weak scaling: T-30 model.

We also studied the weak scaling with the T-30 model and presented the results in Table 4.9. In this case, we consider 5 cores as unit cores and solve the problem with a coarse mesh of 330963 dofs using both direct and iterative solver. We observe the direct solver is faster than the iterative solver. For the finer meshes as the number of dofs increases, we increase the number of cores so that the workload on each core remains the same. Which is done as keeping the ratios in the degrees of freedom the same as the ratios in the number of cores, i.e. 1768029/330963 ≈ 27/5, 2624896/330963 ≈ 41/5, and 13031432/330963 ≈ 203/5. We plotted the solving time versus dofs in Fig. 4.16 and weak scaling efficiency versus the number of cores in Fig. 4.17. We observe that for coarse mesh the direct solver outperform over the iterative solver but as the mesh becomes finer and consequently the dofs increases the computational time of the direct solver increases faster than that of the iterative solver. Also, there is a crossing point on the graph, which is at around 3 million dofs, after that the direct solver time graph overshoots the iterative solver time graph. The weak scaling efficiency of the iterative solver always lies above the direct solver weak scaling efficiency.

Finally, from Fig. 4.13, 4.14 and 4.16 we observe that as the number of turns increases the crossing point (where the direct solver computational time exceeds the iterative solver time) abscissa reduces. That is, if the number of turns of a coil increases, even the problem size remains small, the preconditioned iterative solver wins over the direct solver.

4.10.4. Strong Scaling. We study the strong scalability performance of the direct and iterative solvers by keeping fixed the problem size in terms of the dofs and increasing the number of cores used to solve it.
4.10.5. Speedup ($S_p$) and Strong Scaling Efficiency (SSE). We use the most commonly used metric, speedup, for the strong scaling analysis. Among several types of speedup metric, we use the fixed-size speedup, where the problem size remains fixed and determine how faster the problem can be solved. If the computational time to solve a problem of size $N$ with one unit core(s) is $C_1(N)$, and the computational time to solve the same problem with $p$ units cores is $C_p(N)$, then $S_p$ and SSE are defined as $S_p := \frac{C_1(N)}{C_p(N)}$ and $SSE := \frac{C_1(N)}{p C_p(N)} \times 100\%$, respectively. In an ideal case, the $S_p$ and the number of unit cores employed are linearly related.

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{Cores} & \text{MUMPS} & \text{S}_p & \text{SSE} & \text{FGMRES-GMRES} & \text{S}_p & \text{SSE} \\
\hline
5 & 5597.97s & & & 273.96s & \\
10 & 2542.43s & 2.20 & 110\% & 122.14s & 2.24 & 112\% \\
15 & 1822.59s & 3.07 & 102\% & 87.19s & 3.14 & 105\% \\
20 & 1520.46s & 3.68 & 92\% & 69.43s & 3.95 & 99\% \\
25 & 1324.26s & 4.23 & 85\% & 59.35s & 4.62 & 92\% \\
30 & 1171.39s & 4.78 & 80\% & 49.35s & 5.55 & 93\% \\
35 & 1101.91s & 5.08 & 73\% & 48.23s & 5.68 & 81\% \\
40 & 1197.96s & 4.67 & 58\% & 42.21s & 6.49 & 81\% \\
\hline
\end{array}
$$

Table 4.10: Strong scaling: T-1 model with dofs = 5415890.

4.10.6. Strong Scaling Results and Discussions. In Table 4.10 we represent the results of the strong scaling with the T-1 model. In this case, we monitor the computational time corresponds to a mesh that provides 5415890 dofs, while progressively increasing the number of cores. Here, one unit core has 5 cores. To test the scalability, we vary the number of units of cores as 1, 2, 3, 4, 5, 6, 7, and 8 which are corresponding to 5, 10, 15, 20, 25, 30, 35, and 40 cores. As shown in Fig. 4.19, the iterative solver speedup scales almost linearly up to 30 cores, having about 180,000 dofs per core, whereas the direct solver speedup scales almost linearly up to 20 cores, having about 270,000 dofs. After the linear scalability, even the parallel performances of both solvers reduce, the iterative solver speedup remains bigger than that of the direct solver. We observe that the direct solver strong scaling efficiency drops to 58\%
whereas the iterative solver drops to 81%. In Fig. 4.18 we plotted the solving time versus the number of cores for both the solvers, which shows how faster the problem can be solved by the direct and the iterative solver. The direct solver computational time is much higher than that of the iterative solver.

![T-1 Model Strong Scaling](image)

Fig. 4.18: Measured computational time versus number of cores.

![T-1 Model Ideal](image)

Fig. 4.19: Strong scaling for direct and iterative solver, speedup versus number of cores.

To study the strong scaling of the direct and iterative solver with the T-10 model, we consider a mesh that gives a total of 1860729 dofs, and solve the problem using 2, 8, 14, 20, 26, 32, 38, and 44 cores. The recorded computational time, computed speedup and SSE are presented in Table 4.11. We observe that as the number of cores increases, the iterative solver SSE does not decrease much, however, the direct solver SSE drops significantly. We plotted the computational time versus the number of cores employed in Fig. 4.20. From Fig. 4.20 we observe that the direct solver shows a poor scaling while the iterative solver shows a good scaling. Since as we increase the number of cores, the direct solver fails to reduce the solving time significantly, whereas the iterative solver solving time drops exponentially. However, the iterative solver solving time is higher than that of the director solver, this is supported by the weak scaling, and since the dofs under consideration is not too high.

We also plotted the speedup versus the number of processor cores employed graph for both solvers and the ideal case scenario in Fig. 4.21. The iterative solver speedup graph changes almost linearly as the number of cores varies, which also supports
the good scaling of the iterative solver, and clearly the graph shows a poor scaling phenomenon of the direct solver.

| Cores | MUMPS | $S_p$ | SSE   | FGMRES-GMRES | $S_p$ | SSE   |
|-------|-------|-------|-------|--------------|-------|-------|
| 1     | 346.25s | 1.84  | 36.87%| 4748.42s     |       |       |
| 5     | 187.83s | 2.46  | 24.62%| 1228.23s     | 3.87  | 77.32%|
| 10    | 140.66s | 2.84  | 18.96%| 783.14s      | 6.06  | 60.63%|
| 15    | 121.75s | 2.60  | 12.99%| 551.07s      | 8.62  | 57.44%|
| 20    | 133.25s | 2.60  | 12.99%| 483.96s      | 9.81  | 49.06%|

Table 4.12: Strong scaling: T-20 model with dofs = 1349812.

We monitor the strong scaling of the two solvers with the T-20 model and represent the data in Table 4.12. A mesh that provides a total of 1349812 dofs is considered. The problem is solved using 1, 5, 10, 15, and 20 cores, and while SSE drops to 12.99% and 49.06%, for the direct and iterative solvers, respectively. The computational time for a single time step solve versus the number of cores employed relation is plotted in Fig. 4.22. We observe that as we progressively increase the number of cores, the iterative solver computational time shows an exponential drop and while the direct solver computational time drop is not that much. That is, for the T-20 model the iterative solver shows a better strong scaling than that of the direct solver.

We also plot the speedup versus the number of cores employed relation in Fig. 4.23. Clearly, the iterative solver speedup remains bigger than that of the direct solver for all our experiments. For the direct solver the maximum $S_p$ we found is 2.84 using 15 cores if we further increase the number of cores to 20, the $S_p$ decreases to 2.60. That is, in all the scenarios we considered, the direct solver solved the problem 2.84 times faster than the direct solver with a single core. On the other hand, the $S_p$ for the iterative solver increases monotonically as we progressively increase the number of cores, and the maximum $S_p$ we found is 9.81 which corresponds to 20 cores. That is, the iterative solver solves the problem 9.81 times faster using 20 cores than it does using a single core.

Finally, we investigate the strong scaling of the direct and iterative solvers with the
M. Mohebujjaman, S. Shiraiwa, B. LaBombard, J. C. Wright, and K. Uppalapati

T-20 Model Strong Scaling

Fig. 4.22: Measured computational time versus number of cores.

| Cores | MUMPS  | $S_p$ | SSE   | FGMRES-GMRES | $S_p$ | SSE   |
|-------|--------|-------|-------|--------------|-------|-------|
| 8     | 1420.11s | 1.29  | 64.67% | 7455.98s     |       |       |
| 16    | 1097.91s | 1.60  | 40.07% | 3658.53s     | 2.04  | 101.90% |
| 32    | 885.91s  | 1.75  | 14.62% | 2228.31s     | 3.35  | 83.65%  |
| 96    | 809.37s  | 1.75  | 14.62% | 1147.35s     | 6.50  | 54.15%  |
| 192   | 770.10s  | 1.84  | 7.68%  | 751.43s      | 9.92  | 41.34%  |
| 256   | 788.62s  | 1.80  | 5.63%  | 720.42s      | 10.35 | 32.34%  |
| 288   | 744.80s  | 1.91  | 5.30%  | 683.46s      | 10.91 | 30.30%  |
| 304   | 780.06s  | 1.82  | 4.79%  | 668.46s      | 11.15 | 29.35%  |

Table 4.13: Strong scaling: T-30 model with dofs = 10479901.

T-30 model. In this case, we consider a mesh which provides a total of 10479901 dofs. We solve the problem employing 8, 16, 32, 96, 192, 256, 288, and 304 cores using both the direct and iterative solver for a single time step and records the computational time in Table 4.13. In this case, $C_1(10479901) = 8$, and $p$ varies as 2, 4, 12, 24, 32, 36, and 38. We compute $S_p$ and SSE for both the solvers. From our observations, we found the maximum $S_p$ equals to 1.91 and 11.15 for the direct and iterative solver, respectively. That is, we solve the problem 1.91 times faster by the direct solver and 11.15 times faster by the iterative solver progressively increasing the number of cores. We plot the speedup versus the number of employed cores for the T-30 model in Fig. 4.25. We observe the iterative solver speedup is higher than the direct solver speedup. We plot the computing time versus the number of cores employed for both solvers in Fig. 4.23. We observe a drastic drop in the computational time with the iterative solver than that with the direct solver. This indicates the iterative solver provides a better scaling than the direct solver.

5. Conclusion. Numerical solutions of complex PDEs are often one of the most challenging tasks, which involve finding solutions of systems of linear equations with multi-billions degrees of freedom. In this paper, we have presented a voltage excited magnetic field formulation for the charging simulation of non-insulated superconducting pancake solenoids. The numerical difficulty for the simulation of the problem arises
from the fact that the HTS coil is very thin but with excessively high conductivity whereas its surrounding materials have the opposite properties.

In this paper, we have explored an iterative solver for Maxwell’s equation simulation, which is a combination FGMRES and GMRES solver with a parallel AMS Solver preconditioner. We have investigated the scalability performance of the direct solver, MUMPS, and the iterative solver, FGMRES-GMRES with AMS preconditioner, on 1, 10, 20, and 30 turns pancake solenoids. The direct solver is efficacious for problems with fewer degrees of freedom and provides an additional advantage when the conductivity of the diffusion equation is independent of time. Since in this case, the system matrix remains same for all time step but the right hand side vector changes and the direct solver factorizes the system matrix only once and repeatedly use them for the further timesteps. On the other hand, the iterative solver is faster, uses less memory, and scale better than the direct solver, especially to the problem that has complex geometry and/or higher degrees of freedom. This iterative solver along with the AMS preconditioner can successfully solve the problem of size with at least 48 million degrees of freedom.

As a benchmark, we have computed the Biot-Savart fields and compared them with the FEM results, and found excellent agreement. It is observed that if we increase the number of turns in a charging solenoid, the charging time of the model and the computational time of the iterative solver both increase. Finally, we have also found that as the conductivity of the superconductor increases the problem becomes harder and the iterative solver needs longer time to converge.

We believe the most important future research avenue would be the incorporation of quench analysis in our mathematical model and explore the charging behavior of non-insulated superconducting pancake solenoids. We will use the proposed efficient solver, FGMRES-GMRES, for problems with large degrees of freedom, and will continue to investigate further for a better preconditioner.

Apart from the voltage excitation problem, we plan to investigate a current intensity excitation modeling and explore the charging behavior ramping up the current slowly. In a current intensity excitation problem, the voltage remains an unknown.

Instead of $H$-formulation, we plan to investigate the eddy current simulation on a multiply connected region based on $H-\Phi$ field formulation following the work in [11].
The magnetic field \( \mathbf{H} \) will be computed using the iterative solver in the conducting region and the multi-valued magnetic scalar potential \( \Phi \) will be the solution of a diffusion BVP in the non-conduction region. Finally, both solutions will be assembled together using the appropriate tangential continuity condition across the interfaces between different tetrahedral elements. This approach will potentially reduce the cost of computing the magnetic field in the air domain.

We plan to explore the stability and error estimate analysis for the fully discrete scheme used in this paper following the works in [2, 17]. The study of ensemble magnetic field calculations [28] with the advantage of block linear solver can also be a new research avenue. We also want to explore the reduced order modeling of the magnetic field simulation in a non-insulated superconducting pancake solenoids following the analysis and experiments given in [29].

Acknowledgment. The work was supported by Commonwealth Fusion Systems grant RPP002. We thank Dan Brunner for his constructive comments and suggestions, which significantly improved the quality of the manuscript.

Appendix A. Biot-Savart Field Computation. For the spiral models (T-10, T-20, and T-30), we assume \( x \) be the uniform gap between two turns, and the innermost radius of the filament is \( r \). Then the radius of a spiral changes as

\[
R = r + \frac{x}{2\pi} \theta, \quad 0 \leq \theta \leq 2n\pi, \tag{A.1}
\]

where \( n \in \mathbb{N} \), represents the number of turns. The position vector that represents the spiral filament, can be written as

\[
s(\theta) = < R \cos \theta, R \sin \theta, d >. \tag{A.2}
\]

Therefore, the magnetic field at the origin produced by the current flow in the lower pancake spiral filament is given by

\[
\mathbf{B}_L(\mathbf{0}) = -\frac{\mu_0 I}{4\pi} \begin{pmatrix}
\int_0^{2n\pi} \frac{2x}{2\pi} \sin \theta + d(\frac{1}{2\pi} + \theta) \cos \theta \{ (\frac{1}{2\pi} + \theta)^2 + d^2 \}^{3/2} d\theta \\
\int_0^{2n\pi} -\frac{2x}{2\pi} \cos \theta + d(\frac{1}{2\pi} + \theta) \sin \theta \{ (\frac{1}{2\pi} + \theta)^2 + d^2 \}^{3/2} d\theta \\
\int_0^{2n\pi} -\frac{4x}{2\pi} (\frac{1}{2\pi} + \theta)^2 \{ (\frac{1}{2\pi} + \theta)^2 + d^2 \}^{3/2} d\theta
\end{pmatrix}. \tag{A.3}
\]

Since the upper pancake lies on the \( xy \)-plane, we have in this case \( d = 0 \), and thus the magnetic field contribution at the origin, is given by

\[
\mathbf{B}_U(\mathbf{0}) = < 0, 0, \frac{\mu_0 I}{2x} \ln \left( 1 + \frac{nx}{r} \right) >. \tag{A.4}
\]

Finally, we consider the vector that represents the connection line between the two pancakes, is given by

\[
s_{con} = < r, 0, z >, \quad \text{where} \quad d \leq z \leq 0,
\]
and compute the magnetic field at the origin as

$$B_{\text{con}}(\hat{0}) = \frac{\mu_0 I}{4\pi} < 0, \frac{-d}{\sqrt{r^2 + d^2}}, 0 >.$$  (A.5)

The resultant magnetic field at the origin is then given by

$$B(\hat{0}) = B_L(\hat{0}) + B_U(\hat{0}) + B_{\text{con}}(\hat{0}),$$  (A.6)

and thus its strength is $\|B(\hat{0})\|$. The fully charged magnetic field strength at the origin $B_0^{\text{max}} = \|B(\hat{0})\|$, when $B(\hat{0})$ reaches its statistically steady-state. The computation of $B_0^{\text{max}}$ for the T-1 model using the Biot-Savart law is straightforward, and thus omitted.

REFERENCES

[1] MFEM: Modular finite element methods library. [https://mfem.org/]
[2] M. Akbas, S. Kaya, M. Mohebujjaman, and L. Rebholz. Numerical analysis and testing of a fully discrete, decoupled penalty-projection algorithm for mhd in elsässer variable. Int. J. Numer. Anal. Model., 13(1):90–113, 2016.
[3] P. R. Amestoy, I. S. Duff, and J. L’Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. Computer methods in applied mechanics and engineering, 184(2-4):501–520, 2000.
[4] P. R. Amestoy, I. S. Duff, J. L’Excellent, and J. Koster. MUMPS: A general purpose distributed memory sparse solver. Proceedings of the PAR. A PARA 2000. Lecture Notes in Computer Science, 1947:121–130, 2000.
[5] W. Bangerth. Finite element methods in scientific computing. Lecture 34: What solver to use. [https://www.math.colostate.edu/~bangerth/videos/676/slides.34.pdf]
[6] K. R. Bhattarai, K. Kim, K. Kim, K. Radcliff, X. Hu, C. Im, T. Painter, I. Dixon, D. Labelester, S. Lee, and S. Hahn. Understanding quench in no-insulation (ni) rebco magnets through experiments and simulations. Superconductor Science and Technology, 33(3):035002, 2020.
[7] O. Biro. Edge element formulations of eddy current problems. Computer methods in applied mechanics and engineering, 169(3-4):391–405, 1999.
[8] A. Bossavit. Most general ‘non-local’ boundary conditions for the maxwell equation in a bounded region. COMPEL, 19(2):239–245, 2000.
[9] E. H. Brandt. Superconductors of finite thickness in a perpendicular magnetic field: Strips and slabs. Physical Review B, 54(6):4246–4264, 1996.
[10] S. By, J. V. Rispoli, S. Cheshkov, I. Dimitrov, J. Cui, S. Seiler, S. Goudreau, C. Malloy, S. M. Wright, and M. P. McDougall. A 16-channel receive, forced current excitation dual-transmit coil for breast imaging at 7T. PloS one, 9(11), 2014.
[11] T. A. Davis. Algorithm 832: UMFPACK v4.3—an unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software (TOMS), 30(2):196–199, 2004.
[12] R. D. Falgout and U. M. Yang. hypre: A library of high performance preconditioners. International Conference on Computational Science, pages 632–641, 2002.
[13] P. Ghysels, X. S. Li, F. Rouet, S. Williams, and A. Napov. An efficient multicore implementation of a novel has-structured multifrontal solver using randomized sampling. SIAM Journal on Scientific Computing, 38(5):S358–S384, 2016.
[14] D. M. Ginsberg. Physical properties of high temperature superconductors III, volume 3. World Scientific, 1992.
[15] S. Hahn, D. K. Park, J. Bascunan, and Y. Iwasa. HTS pancake coils without turn-to-turn insulation. IEEE transactions on applied superconductivity, 21(3):1592–1595, 2010.
[16] B. He, C. Lu, N. Chen, D. Lin, and P. Zhou. An efficient parallel computing method for the steady-state analysis of electric machines using the woodbury formula. IEEE Transactions on Magnetics, 56(2):1–4, 2020.
[17] T. Heister, M. Mohebujjaman, and L. G. Rebholz. Decoupled, unconditionally stable, higher order discretizations for mhd flow simulation. Journal of Scientific Computing, 71(1):21–43, 2017.
[18] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

[19] R. Hiptmair, G. Widmer, and J. Zou. Auxiliary space preconditioning in $H_0$(curl; $\Omega$). *Numerische Mathematik*, 103(3):435–459, 2006.

[20] R. Hiptmair and J. Xu. Auxiliary space preconditioning for edge elements. *IEEE Transactions on Magnetics*, 44(6):938–941, 2008.

[21] Z. Huo, G. Mei, G. Casolla, and F. Giampaolo. Designing an efficient parallel spectral clustering algorithm on multi-core processors in julia. *Journal of Parallel and Distributed Computing*, 2020.

[22] T. V. Kolev and P. S. Vassilevski. Parallel auxiliary space AMG for $H$ (curl) problems. *Journal of Computational Mathematics*, pages 604–623, 2009.

[23] A. Kuzmin, M. Luisier, and O. Schenk. Fast methods for computing selected elements of the greens function in massively parallel nanoelectronic device simulations. In *European Conference on Parallel Processing*, pages 533–544. Springer, 2013.

[24] U. Langer and M. Neumüller. Direct and iterative solvers. In *Computational Acoustics*, pages 205–251. Springer, 2018.

[25] X. S. Li, J. Demmel, J. Gilbert, L. Grigori, and M. Shao. SuperLU. *Encyclopedia of Parallel Computing*, pages 1955–1962, 2011.

[26] D. Liu, H. Yong, and Y. Zhou. Analysis of charging and sudden-discharging characteristics of no-insulation recomo coil using an electromagnetic coupling model. *AIP Advances*, 7(11):115104, 2017.

[27] S. Meng, B. Zhu, Z. Zhuang, X. Chen, and C. Tang. Effect of excitation coil voltage on TiAlSiN coating on 42Crmo steel surface. *Materials Research Express*, 7(5):056519, 2020.

[28] M. Mohebujjaman and L. G. Rebholz. An efficient algorithm for computation of mhd flow ensembles. *Computational Methods in Applied Mathematics*, 17(1):121–137, 2017.

[29] M. Mohebujjaman, L. G. Rebholz, X. Xie, and T. Iliescu. Energy balance and mass conservation in reduced order models of fluid flows. *Journal of Computational Physics*, 346:262–277, 2017.

[30] K. Ono, T. Kato, S. Oshshima, and T. Nani. Scalable direct-iterative hybrid solver for sparse matrices on multi-core and vector architectures. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, pages 11–21, 2020.

[31] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on numerical analysis*, 12(4):617–629, 1975.

[32] L. Prigozhin. The bean model in superconductivity: Variational formulation and numerical solution. *Journal of Computational Physics*, 129(1):190–200, 1996.

[33] A. A. Rodríguez and A. Valli. *Eddy current approximation of Maxwell equations: Theory, algorithms and applications*, volume 4. Springer Science & Business Media, 2010.

[34] F. Rouet, X. S. Li, P. Ghysels, and A. Napov. A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. *ACM Transactions on Mathematical Software (TOMS)*, 42(4):1–35, 2016.

[35] D. Ruiz-Alonso, T. Coombs, and A. M. Campbell. Computer modelling of high-temperature superconductors using an A–V formulation. *Superconductor Science and Technology*, 17(5):S305, 2004.

[36] Y. Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[37] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[38] S. Shiraiwa, J. C. Wright, P. T. Bonoli, T. Kolev, and M. Stowell. RF wave simulation for cold edge plasmas using the MFEM library. *EPJ Web of Conferences*, 157(03048), 2017.

[39] H. Shoukourian, T. Wilde, A. Auweter, and A. Bode. Predicting the energy and power consumption of strong and weak scaling hpc applications. *Supercomputing frontiers and innovations*, 1(2):20–41, 2014.

[40] V. Simoncini and D. B. Szyld. Flexible inner-outer krylov subspace methods. *SIAM Journal on Numerical Analysis*, 40(6):2219–2239, 2003.

[41] J. Smajic, M. K. Bucher, C. Jäger, and R. Christen. Treatment of multiply connected domains in time-domain discontinuous galerkin $H – \Phi$ eddy current analysis. *IEEE Transactions on Magnetics*, 55(6):1–4, 2019.

[42] X. Sun and D. T. Rover. Scalability of parallel algorithm-machine combinations. *IEEE Transactions on parallel and Distributed Systems*, 5(6):599–613, 1994.

[43] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical ....*
Computing, 13(2):631–644, 1992.