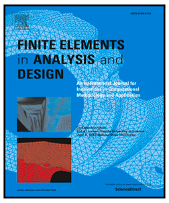
Finite Elements in Analysis and Design 204 (2022) 103755

Contents lists available at ScienceDirect

Finite Elements in Analysis & Design

journal homepage: www.elsevier.com/locate/finel



Modelling a permanent magnet synchronous motor in FEniCSx for parallel high-performance simulations

James McDonagh[a,*], Nunzio Palumbo[b], Neeraj Cherukunnath[b], Nikolay Dimov[a], Nada Yousif[a]

[a] *School of Physics, Engineering and Computer Science, University of Hertfordshire, Hatfield, AL10 9AB, United Kingdom*
[b] *Future Methods, Rolls-Royce plc, Derby, DE24 8BJ, United Kingdom*

ARTICLE INFO

Keywords:
Electric machine
Maxwell's equations
Finite element method
High-performance computing
Open-source software
FEniCS

ABSTRACT

There are concerns that the extreme requirements of heavy-duty vehicles and aviation will see them left behind in the electrification of the transport sector, becoming the most significant emitters of greenhouse gases. Engineers extensively use the finite element method to analyse and improve the performance of electric machines, but new highly scalable methods with a linear (or near) time complexity are required to make extreme-scale models viable. This paper introduces a three-dimensional permanent magnet synchronous motor model using FEniCSx, a finite element platform tailored for efficient computing and data handling at scale. The model demonstrates comparable magnetic flux density distributions to a verification model built in Ansys Maxwell with a maximum deviation of 7% in the motor's static regions. Solving the largest mesh, comprising over eight million cells, displayed a speedup of 198 at 512 processes. A preconditioned Krylov subspace method was used to solve the system, requiring 92% less memory than a direct solution. It is expected that advances built on this approach will allow system-level multiphysics simulations to become feasible within electric machine development. This capability could provide the near real-world accuracy needed to bring electric propulsion systems to large vehicles.

1. Introduction

The past decades have seen electrification emerge as a leading solution for reducing the environmental impact of transportation. Global electric car stock exceeded 10 million in 2020, fuelled by technological innovation, fiscal incentives, and ambitious policies such as the Paris Agreement, achieving a year-on-year growth rate of 43% [1]. This rapid adoption has aided in curbing the increase of global transport emissions to less than 0.5% in 2019, compared with 1.9% annually since 2000 [2]. However, heavy-duty vehicles and aviation are expected to become the most significant contributors to transport emissions without breakthroughs in technology [3]. The underlying challenge with these vehicles is that they require more efficient, power-dense, and lightweight electric motors than are currently available.

Permanent magnet synchronous motors (PMSMs) are extensively used within heavy-duty electric vehicles and electric demonstrator aircraft due to their inherent benefits over other topologies [4–8]. Namely, PMSMs offer high efficiency and high torque density within a small packaging volume, although they are not economically viable for many applications due to the frequent use of rare-earth elements within the magnets. Furthermore, designing high-performance electric motors involves finding the optimal compromise between many criteria, including mass, torque, losses, noise, reliability, and cost. Most of these specifications can be calculated analytically; however, to accurately predict the electromagnetic performance requires numerical simulations such as the finite element method.

Ideally, a great deal of detail in the geometry should be retained to ensure the solution matches the real-world behaviour of a PMSM system. The most fundamental attribute in building realistic models is likely the dimension of the simulation. Three-dimensional models become prohibitively expensive at scale, in terms of computation time, due to the at best $O(n^2)$ time complexity of sparse direct linear solvers, where n is the number of degrees of freedom (DOFs) [9]. Additionally, sparse direct solvers are known for being memory intensive, requiring $O(n^{4/3})$ memory.

To illustrate this, a previous study [10] considers directly solving a linear finite element problem using linear Lagrange elements. Achieving a 10 times reduction in the L^2-norm error requires the cell size to be reduced by a factor of $\sqrt{10}$; in two dimensions, this change increases the computation time by a factor of 31.6, in contrast to 10 000 for its three-dimensional counterpart [11]. Hence, two-dimensional and sector models are commonly employed to bypass this obstacle. Such simplifications come at the expense of accuracy. As system-level PMSM

* Corresponding author.
E-mail address: james.d.mcdonagh@outlook.com (J. McDonagh).

https://doi.org/10.1016/j.finel.2022.103755
Received 22 September 2021; Received in revised form 2 March 2022; Accepted 6 March 2022
Available online 6 April 2022

geometries are unlikely to have any symmetry due to the terminal block and housing body, only a full three-dimensional model can capture the resulting unsymmetrical behaviour. Therefore, methods with a linear (or near) time complexity and access to more computing power are essential to making large three-dimensional simulations feasible. Although less robust than direct solvers, preconditioned Krylov subspace methods are a promising candidate for delivering $O(n)$ complexity [10] and will be explored further in this work.

For more than 50 years, increased computing performance has been facilitated by the miniaturisation of semiconductor transistors [12]. As miniaturisation approaches its limits, massively parallel systems have satisfied the persistent demand for increased processing power. Instead of running a problem in serial, it is broken down into smaller tasks that are computed simultaneously. Unfortunately, many commercial finite element packages rely on robust direct solvers, which are challenging to parallelise. This paper presents the transient analysis of a PMSM using FEniCSx, an open-source platform for solving partial differential equations (PDEs) in parallel. FEniCSx is an incremental rewrite of the original FEniCS platform [13,14] that offers greater customisability, speed, and scalability [15]. FEniCSx comprises four internal components: DOLFINx, a Python/C++ finite element library; UFL (Unified Form Language) [16,17], a language in which weak formulations are written; FFCx (FEniCSx Form Compiler) [18–20], a library that generates efficient C code from the UFL form; and Basix [21], a library that creates finite element basis functions. In addition, the solving of linear systems is facilitated through PETSc [22–24] by default, a collection of data structures and linear algebra solvers for the scalable solution of PDEs. With further configuration, FEniCSx can interface with other linear algebra libraries such as Eigen and Trilinos if desired.

The advantages of parallelising electric machine analysis have been known for decades [25]. The greater compute power delivered by new platforms offers time and cost benefits for the iterative design process. Further, these resources are more accessible and flexible with the advent of cloud computing, the benefits of which have already been explored in the context of electromagnetic simulations [26]. Lower complexity and parallel finite-element solvers for large-scale 3D electromagnetic problems have been developed [11,27], though they have not been applied to models similar in complexity to a PMSM. Consequently, a small selection of commercial electromagnetic packages, which typically rely on direct solvers, remain dominant within industry. This work seeks to begin bridging this gap by demonstrating the scalable solution of a PMSM model that can exploit modern computing paradigms.

In this article, the following aspects of PMSM simulations in FEniCSx are conveyed. Beginning with a theoretical background, Section 2 presents the derivation of the governing equations and boundary conditions using the A-V formulation. The outlining of a conventional PMSM and its implementation into FEniCSx is detailed in Section 3. This section covers the core activities involved in building the model, including the mesh generation, weak formulation definition and PETSc solver settings. An equivalent model within a commercial electromagnetic package is also specified to verify the results of the FEniCSx model. In Section 4, the results of the model are presented and discussed. This involves analysing the solution of the primary variables, verifying the computed magnetic flux density, and reviewing the model's suitability for high-performance simulations. Finally, conclusions are drawn based on our findings, and suggestions for future work are proposed in Section 5.

2. Numerical formulation

The motor is modelled using the nodal A-V formulation in the time domain, which in essence couples the magnetic vector potential $\boldsymbol{A}$ with the electric scalar potential V. This formulation was chosen due to its prevalence and wide range of applications found in the literature [28–32]. A typical PMSM is composed of a stator, a rotor, permanent magnets, and coil windings which are denoted by Ω_s, Ω_r, Ω_{pm}, and Ω_c, respectively. The surrounding air domain is referred to by Ω_a.

2.1. Maxwell's equations

Deriving the formulation begins with Maxwell's equations, which are presented in their differential form as follows

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{1a}$$

$$\nabla \times \boldsymbol{H} = \frac{\partial \boldsymbol{D}}{\partial t} + \boldsymbol{J} \tag{1b}$$

$$\nabla \cdot \boldsymbol{B} = 0 \tag{1c}$$

$$\nabla \cdot \boldsymbol{D} = \rho \tag{1d}$$

where $\boldsymbol{E}$ is the electric field intensity, $\boldsymbol{H}$ is the magnetic field intensity, $\boldsymbol{B}$ is the magnetic flux density, $\boldsymbol{J}$ the current density, and $\boldsymbol{D}$ is the electric displacement field.

Maxwell's equations can be used in conjunction with the constitutive relations to capture the effects of particular material properties, including permeability μ, conductivity σ, and permittivity ϵ. In the simplest case, these relations are linear and are defined as

$$\boldsymbol{B} = \mu \boldsymbol{H} \tag{2a}$$

$$\boldsymbol{J} = \sigma \boldsymbol{E} \tag{2b}$$

$$\boldsymbol{D} = \epsilon \boldsymbol{E} \tag{2c}$$

Within low-frequency machines, the displacement current will have a negligible effect on the magnetic field close to the current sources [33–35]. Hence, the problem is simplified by omitting Eq. (1d), relation (2c), and asserting $\boldsymbol{D} = 0$ in Eq. (1b) to reduce it to

$$\nabla \times \boldsymbol{H} = \boldsymbol{J} \tag{3}$$

The A-V formulation begins by introducing the magnetic vector potential through the following definition

$$\boldsymbol{B} = \nabla \times \boldsymbol{A} \tag{4}$$

which is then substituted into Faraday's law (1a) to define the electric scalar potential

$$\boldsymbol{E} = -\frac{\partial \boldsymbol{A}}{\partial t} - \nabla V \tag{5}$$

The current density is split into two parts: the source current density $\boldsymbol{J}_s$ (flowing through the wires) and the eddy current density $\boldsymbol{J}_e$ (induced by a time-varying magnetic field). In the A-V formulation, the eddy current contributions are calculated by substituting Eq. (5) into Ohm's law (2b), as presented below

$$\boldsymbol{J} = \boldsymbol{J}_e + \boldsymbol{J}_s = \sigma \left(-\frac{\partial \boldsymbol{A}}{\partial t} - \nabla V \right) + \boldsymbol{J}_s \tag{6}$$

2.2. Modelling permanent magnets

The permanent magnets within the motor are modelled as neodymium iron boron (NdFeB), a ferromagnetic material. Ferromagnets exhibit hysteresis, a non-linear behaviour that can be described through methods such as the Jiles–Atherton model [36]. However, a linear model is a reasonable approximation for rare-earth permanent magnets as their performance is commonly described using the second quadrant of their B-H curve [37,38]. The linear approach will be implemented to reduce the model's complexity. In this case, the magnetic flux density is specified by the following unidirectional function

$$\boldsymbol{B} = \mu_0 \mu_r \boldsymbol{H} + \boldsymbol{B}_r \tag{7}$$

where μ_0 is the permeability of free space, μ_r is the relative permeability of the material, and $\boldsymbol{B}_r$ is the remanent magnetic flux density of the ferromagnet. The final quantity is equated to the magnetisation field to simplify the source expression in the model. The magnetisation field is linear from south to north in a saturated magnet, while the magnetic field intensity and magnetic flux density are both curved. The magnetisation $\boldsymbol{M}$ is calculated by

$$\boldsymbol{M} = \nu_0 \boldsymbol{B} \tag{8}$$

where ν_0 is the reluctivity of free space ($\nu_0 = \mu_0^{-1}$). Correspondingly, reluctivity is defined as $\nu = \mu^{-1}$. Substituting this into Eq. (7) with $\mu = \mu_0 \mu_r$ leads to

$$\boldsymbol{B} = \mu \boldsymbol{H} + \mu_0 \boldsymbol{M} \tag{9}$$

which will be used to describe the behaviour of the permanent magnets.

2.3. Modelling rotation

The rotation of the motor can be accounted for using the motion voltage term $\boldsymbol{v} \times \boldsymbol{B}$, where $\boldsymbol{v}$ is the velocity [33,39]. This approach is sometimes referred to as the multiple frame of reference scheme [40]. Another technique is the moving band principle [41,42], where the rotor physically shifts during the simulation time rather than the formulation reflecting the rotation. Once discretised into a mesh, the translation of the rotor stretches cells within the air gap, eventually requiring remeshing. The multiple frames of reference scheme will be employed as the additional meshing operations and cell quality checks required in the moving band principle would significantly increase the computational cost of the simulation. This method begins by defining a stationary reference frame $O(x, y, z)$ and rotating reference frame $O'(x', y', z')$. Through the assumption that the time t and t' are the same in both reference frames, the motion voltage term can be summed to the electric field intensity within the rotating reference frame to give

$$\boldsymbol{E}' = \boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B} \tag{10}$$

The magnetic field intensity, magnetic flux density, and current density are identical between reference frames. The current density expression in the rotor and permanent magnet domains, as observed from the stationary frame, can be revised to

$$\boldsymbol{J} = \sigma \left(-\frac{\partial \boldsymbol{A}}{\partial t} - \nabla V + \omega_v r \times \boldsymbol{B} \right) \quad \text{in } \Omega_r \cup \Omega_{pm} \tag{11}$$

where ω_v is the angular velocity and r is the distance from the motor's central axis.

2.4. Governing equations

Every aspect of the motor has now been considered, allowing the complete formulation to be assembled. Eqs. (3), (4), (6), (9), and (11) are substituted to reveal

$$\begin{aligned} \nabla \times (\nu \nabla \times \boldsymbol{A}) = &- \sigma \frac{\partial \boldsymbol{A}}{\partial t} - \sigma \nabla V + \sigma \omega_v r \times (\nabla \times \boldsymbol{A}) \\ &+ \nabla \times \left(\nu \mu_0 \boldsymbol{M} \right) + \boldsymbol{J}_s \end{aligned} \tag{12}$$

To ensure a unique solution of the magnetic vector potential, the Coulomb gauge must be satisfied [43]. The Coulomb gauge is defined as

$$\nabla \cdot \boldsymbol{A} = 0 \tag{13}$$

and using a vector identity alongside the gauge in Eq. (12) introduces the first governing equation:

$$\nu \nabla^2 \boldsymbol{A} = \sigma \left(\frac{\partial \boldsymbol{A}}{\partial t} + \nabla V - \omega_v r \times (\nabla \times \boldsymbol{A}) \right) - \nabla \times \left(\nu \mu_0 \boldsymbol{M} \right) - \boldsymbol{J}_s \tag{14}$$

Similarly, the conservation of charge must be enforced for the eddy currents in the formulation [44]. This relation is given as

$$\nabla \cdot \boldsymbol{J}_e = 0 \tag{15}$$

Applying this conservation law to Eq. (12) results in the second governing equation:

$$\sigma \nabla^2 V = \nabla \cdot \sigma \left(-\frac{\partial \boldsymbol{A}}{\partial t} + \omega_v r \times (\nabla \times \boldsymbol{A}) \right) \tag{16}$$

However, not every term is relevant for the entire motor and the governing equations will now be defined with respect to each of the motor's domains. The solution of the magnetic vector potential will be calculated using

$$\nu \nabla^2 \boldsymbol{A} = \sigma \frac{\partial \boldsymbol{A}}{\partial t} + \nabla V - \sigma \omega_v r \times (\nabla \times \boldsymbol{A}) \quad \text{in } \Omega_r \tag{17a}$$

$$\nu \nabla^2 \boldsymbol{A} = \sigma \frac{\partial \boldsymbol{A}}{\partial t} + \nabla V \quad \text{in } \Omega_s \tag{17b}$$

$$\nu \nabla^2 \boldsymbol{A} = -\boldsymbol{J}_s \quad \text{in } \Omega_c \tag{17c}$$

$$\nu \nabla^2 \boldsymbol{A} = -\nabla \times \left(\nu \mu_0 \boldsymbol{M} \right) - \sigma \omega_v r \times (\nabla \times \boldsymbol{A}) \quad \text{in } \Omega_{pm} \tag{17d}$$

and in the same manner, the electric scalar potential with the following

$$\sigma \nabla^2 V = \nabla \cdot \sigma \left(-\frac{\partial \boldsymbol{A}}{\partial t} + \omega_v r \times (\nabla \times \boldsymbol{A}) \right) \quad \text{in } \Omega_r \tag{18a}$$

$$\sigma \nabla^2 V = \nabla \cdot \left(-\sigma \frac{\partial \boldsymbol{A}}{\partial t} \right) \quad \text{in } \Omega_s \tag{18b}$$

$$\sigma \nabla^2 V = \nabla \cdot \left(\sigma \omega_v r \times (\nabla \times \boldsymbol{A}) \right) \quad \text{in } \Omega_{pm} \tag{18c}$$

By assuming that the outer boundary is located far from the motor, where the magnetic fields are negligible, the following homogeneous Dirichlet condition can be imposed

$$\boldsymbol{A} = \boldsymbol{0} \quad \text{on } \partial\Omega \tag{19}$$

The set of coupled governing equations and boundary condition derived in this section comprises the strong formulation of the problem. From this, a weak formulation is obtained to construct the finite element model. This process is carried out by multiplying the strong formulation by a test function, integrating it over the spatial domain, and performing integration by parts to terms with second-order derivatives. The unknown variables to be solved are represented by a trial function, and the trial and test functions must be defined in suitable function spaces, such as the Sobolev space.

2.5. Temporal discretisation

As the derived problem is transient, the governing equations must be discretised in both space and time. The backward Euler method was employed, and hence the time derivative in the strong formulation is discretised as follows

$$\frac{\partial \boldsymbol{A}}{\partial t} = \frac{\boldsymbol{A}_{n+1} - \boldsymbol{A}_n}{\Delta t_n} \tag{20}$$

where Δt_n is the time-step and $t_{n+1} = t_n + \Delta t_n$. The time interval of interest is $I = (0, t_N]$.

2.6. Weak formulation

The weak formulation of the problem reads: given $\boldsymbol{A}_n$ at time t_n, and $\boldsymbol{J}_{s,n+1}$ and $\boldsymbol{M}_{n+1}$ at time t_{n+1}, find $\boldsymbol{A}_{n+1} \in [Q_h]^3$ and $V_{n+1} \in Q_h$ such that

$$\begin{aligned} F_{\boldsymbol{A}}(\boldsymbol{A}_{n+1}; \boldsymbol{v}) := &\int_{\Omega} \nu \nabla \boldsymbol{A}_{n+1} \cdot \nabla \boldsymbol{v} \, dx + \int_{\Omega_{r,s}} \sigma \frac{\boldsymbol{A}_{n+1} - \boldsymbol{A}_n}{\Delta t_n} \cdot \boldsymbol{v} \, dx \\ &+ \int_{\Omega_{r,s}} \sigma \nabla V_{n+1} \cdot \boldsymbol{v} \, dx - \int_{\Omega_{r,pm}} \sigma \omega_v r \times (\nabla \times \boldsymbol{A}_{n+1}) \cdot \boldsymbol{v} \, dx \\ &- \int_{\Omega_c} \boldsymbol{J}_{s,n+1} \cdot \boldsymbol{v} \, dx - \int_{\Omega_{pm}} \nu \mu_0 \boldsymbol{M}_{n+1} \cdot \nabla \times \boldsymbol{v} \, dx = 0 \\ &\forall \boldsymbol{v} \in [Q_h]^3 \end{aligned} \tag{21a}$$

$$\begin{aligned} F_V(V_{n+1}; q) := &\int_{\Omega} \sigma \nabla V_{n+1} \cdot \nabla q \, dx - \int_{\Omega_{r,s}} \sigma \frac{\boldsymbol{A}_{n+1} - \boldsymbol{A}_n}{\Delta t_n} \cdot \nabla q \, dx \\ &+ \int_{\Omega_{r,pm}} \sigma \omega_v r \times (\nabla \times \boldsymbol{A}_{n+1}) \cdot \nabla q \, dx = 0 \quad \forall q \in Q_h \end{aligned} \tag{21b}$$

where Q_h is a finite element space and $Q_h \subset H^1(\Omega)$.

Table 1
Parameters and dimensions of the modelled motor.

Parameter	Value	Unit
Number of slots	12	–
Number of poles	10	–
Number of phases	3	–
Active axial length	80.0	mm
Stator outer radius	75.0	mm
Rotor outer radius	40.0	mm
Rotor inner radius	17.0	mm
Air gap	0.5	mm
PM width	10.0	mm
PM thickness	2.0	mm
PM embrace	0.7	mm
PM bridge	0.5	mm

3. FEniCSx implementation

3.1. Installation

To assess the parallel performance of the model, FEniCSx was installed on the University of Hertfordshire high-performance computing (UH HPC) cluster. FEniCSx is readily available through Docker, a platform that packages software within containers. These containers provide a convenient and portable method of deploying software, with research showing that they can achieve near bare-metal performance when properly tuned [45,46]. "Singularity", a container platform developed especially for HPC systems, is installed on the UH HPC cluster in place of Docker. Singularity is able to convert the FEniCSx Docker image into a compatible file.

Individual processes communicate with one another in parallel using the message passing interface (MPI) standard. When using containers, a separate MPI installation to the host machine is required. Container-host communication through MPI only properly functions when both MPI libraries are compatible. The FEniCSx Dockerfile was modified to use Open MPI v4.0.5 instead of MPICH as this resulted in the best performance with the MPI implementations already available on the UH HPC system. The core components of FEniCSx were not altered with DOLFINx, FFCx, Basix v0.1.0, and UFL v2021.1.0 used.

3.2. Motor specification

As with the formulation, the motor parameters are simplified to ease the development phase while still representing the fundamental physics of PMSMs. For example, defining the current density excitation within double-layer distributed windings would require a considerably more complex expression than single-layer concentrated windings while not improving the findings of this work. The complete parameters and dimensions of the motor are detailed in Table 1.

To aid in visualising the machine, Fig. 1 presents a cross-section view with each domain shown. The three phases of the supply current are represented by red, green, and blue colouring.

3.3. Mesh generation

FEniCSx natively supports geometry and mesh generation through the open-source library Gmsh. However, this is only accessible in FEniCSx through Gmsh's Python or C++ application programming interface. Despite being simplified, the shape of the PMSM is still rather sophisticated and suits the use of a computer-aided design package with an advanced graphical user interface. Ansys Maxwell, a commercial electromagnetic package, can fulfil this role via an internal tool called rotating machine expert (RMxprt). RMxprt allows users to build a range of machines through templates which can then be automatically converted into a three-dimensional Ansys Maxwell model. This process was used to produce the geometry and subsequent meshes of the motor.

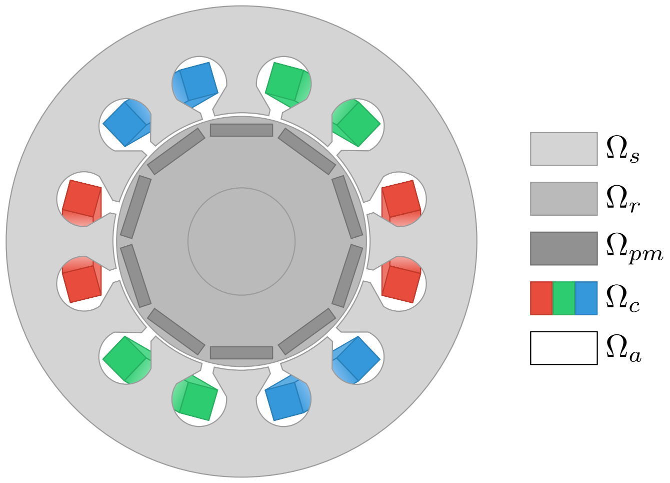


Fig. 1. Cross-section of the modelled motor. Ω_s denotes the stator, Ω_r the rotor, Ω_{pm} the permanent magnets, Ω_c the coil windings, and Ω_a the surrounding air. The coil windings are coloured to signify each phase; red for phase A, green for phase B, and blue for phase C.

The meshes were converted into a FEniCSx compatible format through a Python script using the h5py package [47]. The geometries generated by RMxprt are displayed in Fig. 2. The full motor model in Fig. 2(a) will be solved by FEniCSx. In contrast, the sector model in Fig. 2(b) is required to solve the verification model on a personal computer (PC) due to memory limitations.

Four unstructured meshes of the motor were created using first-order tetrahedral elements to understand the model's performance at increasing problem sizes. The maximum edge lengths for Mesh1 were determined by Ansys Maxwell after converting the RMxprt model and vary by domain. These lengths were halved between successive meshes and the statistics for each are described in Table 2.

Ansys Maxwell numbers the domains and facets of the geometry, which are stored within the mesh file. This data was retained during the conversion process and will be helpful when implementing the formulation in FEniCSx.

3.4. Problem definition

At this stage, the physical problem is ready to be implemented into the FEniCSx platform. The majority of the model's code was written using functionality from the DOLFINx and UFL libraries. Reading the mesh, assembling the system, and saving the solution were handled in parallel by DOLFINx and the implementation of the formulation was managed by UFL. However, before these steps, the material properties and simulation parameters had to be defined. The values used are summarised in Tables 3 and 4.

The rotational speed was set to the motor's synchronous speed, and the supply frequency was selected as many electric machines operate at 50 Hz due to standardisation. Further, the time-step must be small enough to provide adequate resolution of the sinusoidal supply current. In this case, the supply current has a period of 0.02 s, and with a time-step of 2 ms, the solution will be computed at eleven equidistant points per oscillation.

The coil and magnet excitations were created using distinct Python methods, which are known as expressions within FEniCSx. These expressions used the motor's geometry to define the source at every point before being interpolated by DOLFINx onto a particular mesh. Particular variables could be updated within the excitations, allowing the three phases within the coils to be updated based on the current simulation time. The excitations defined within the model are summarised in Table 5.

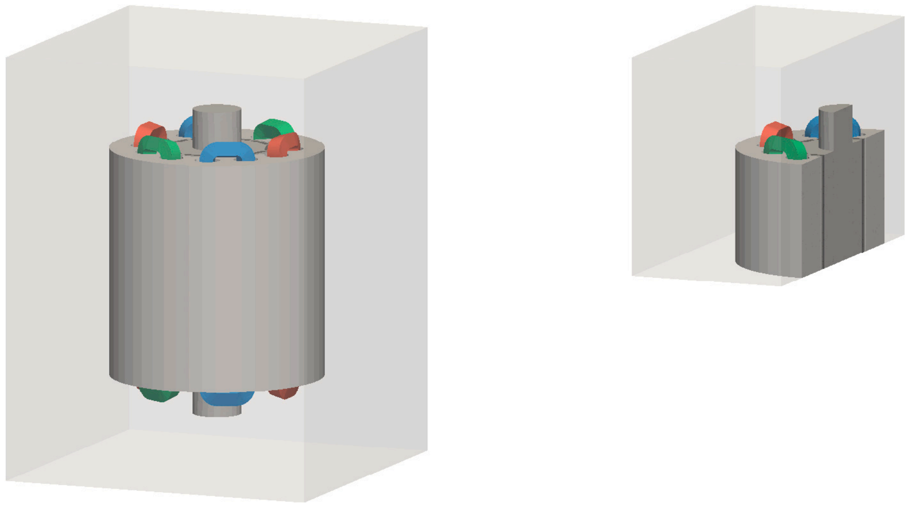


Fig. 2. Model geometries of the motor. (a) The full model contains the entire geometry of the motor. (b) The quarter sector model uses the motor's two planes of symmetry to reduce the size of the geometry.

Table 2
Mesh statistics for the full motor model.

Mesh name	Number of vertices	Number of cells	Number of DOFs	Maximum Edge Length (mm)		
				Ω_c	Ω_{pm}	$\Omega_s, \Omega_r, \Omega_a$
Mesh1	20 964	125 478	83 856	16.00	4.00	14.00
Mesh2	46 941	278 151	187 764	8.00	2.00	7.00
Mesh3	214 684	1 258 542	858 736	4.00	1.00	3.50
Mesh4	1 425 854	8 356 409	5 703 416	1.00	0.50	1.75

Table 3
Magnetic and electric material properties.

Material	Relative Permeability	Conductivity (S/m)	Domains
Steel	100	2.00×10^6	Ω_s, Ω_r
NdFeB	1.04457	6.25×10^5	Ω_{pm}
Copper	0.999991	5.80×10^7	Ω_c
Vacuum	1	0	Ω_a

Table 4
Simulation parameters.

Parameter	Value	Unit
Rotational speed	600	RPM
Supply current frequency	50	Hz
Time-step	2	ms
Final time	10	ms

Table 5
Excitations within the motor.

Excitation type	Magnitude	Unit	Domain
Magnetic coercivity	8.38×10^5	A/m	Ω_{pm}
Alternating current	35	A	Ω_c

Despite constructing the mesh using linear tetrahedral elements, FEniCSx allows the element type to be altered as long as the geometries match. For instance, second-order Nédélec elements could be used for this model without any modifications to the mesh file. A corresponding finite element function space was created, within which the trial and test functions are defined. The model used linear Lagrange elements, otherwise known as P1 elements, as shown by the trial and test function definitions in Fig. 3. For brevity, the code snippets highlighting essential sections of the model are condensed.

The weak formulation was derived from the coupled governing equations in the previous section. As presented in Fig. 4, the complete formulation was inputted into FEniCSx using the UFL library. This library offers operators and expressions that allow the coded form to remain close to mathematical notation. The domain numbers, inherited from the Ansys Maxwell model, were collected to simplify the form. The weak formulation was defined and then separated into the bilinear and linear form.

The boundary condition was enforced through DOLFINx, as demonstrated in Fig. 5. First, the DOFs on the outer boundaries were collected into an array using the facet numbering from the mesh file. Then, the three components were set to zero, constraining the magnetic vector potential to vanish at the boundary.

Before the problem was solved, the final step was to translate the form into the underlying linear algebra problem. This operation was carried out by the external library PETSc but is called using DOLFINx wrapper functions, as shown in Fig. 6.

The first two lines construct a matrix from the bilinear form with the Dirichlet boundary conditions imposed. The fourth line onwards constructs a vector from the linear form and applies a lift of the Dirichlet boundary conditions. Then, values at the ghost points are updated, and the boundary condition values are inserted into the locally owned vectors. To enable parallel computation, the vector is partitioned and distributed across many processes; ghost points are the bordering portions of the vector owned by neighbouring processes. Hence, this operation is required to update the ghost points with the correct values from the owning process.

3.5. *Solver configuration*

As previously stated, FEniCSx uses the linear solvers and preconditioners available from PETSc by default. This work concentrates on Krylov subspace iterative methods to achieve more efficient scaling compared to direct solvers. A range of solvers and preconditioners were trialled to understand their effectiveness for the given problem. As the system is unsymmetric, this included the generalised minimal residual

```
CG_V   = VectorElement('CG', cell, 1)
CG_F   = FiniteElement('CG', cell, 1)
CG_VF  = MixedElement([CG_V, CG_F])
V_VF   = FunctionSpace(mesh, CG_VF)

A, V = TrialFunctions(V_VF)
v, q = TestFunctions(V_VF)
```

Fig. 3. Trial and test functions expressed using UFL.

```
dx_air     = (dx(1)  + dx(2)   + dx(4)   + dx(23))
dx_rotor   = (dx(3)  + dx(12))
dx_stator  = (dx(5))
dx_coil    = (dx(6)  + dx(7)   + dx(8)   + dx(9)   + dx(10) + dx(11))
dx_magnet  = (dx(13) + dx(14)  + dx(15)  + dx(16)  + dx(17) + \
              dx(18) + dx(19)  + dx(20)  + dx(21)  + dx(22))

f_A = \
  + inner(nu_air*grad(A),grad(v))*dx_air \
  + inner(nu_stl*grad(A),grad(v))*dx_rotor \
  + inner(nu_stl*grad(A),grad(v))*dx_stator \
  + inner(nu_cop*grad(A),grad(v))*dx_coil \
  + inner(nu_mag*grad(A),grad(v))*dx_magnet \
\
  + inner(sigma_stl*((A-A0)/dt),v)*dx_rotor \
  + inner(sigma_stl*((A-A0)/dt),v)*dx_stator \
\
  + inner(sigma_stl*grad(V),v)*dx_rotor \
  + inner(sigma_stl*grad(V),v)*dx_stator \
\
  - inner(sigma_stl*cross(omega_v*r,curl(A)),v)*dx_rotor \
  - inner(sigma_mag*cross(omega_v*r,curl(A)),v)*dx_magnet \
\
  - inner(J_s,v)*dx_coil \
\
  - inner(nu_mag* mu_0 *M,curl(v))*dx_magnet \

f_V = \
  + inner(sigma_air*grad(V),grad(q))*dx_air \
  + inner(sigma_stl*grad(V),grad(q))*dx_rotor \
  + inner(sigma_stl*grad(V),grad(q))*dx_stator \
  + inner(sigma_cop*grad(V),grad(q))*dx_coil \
  + inner(sigma_mag*grad(V),grad(q))*dx_magnet \
\
  - inner(sigma_stl*((A-A0)/dt),grad(q))*dx_rotor \
  - inner(sigma_stl*((A-A0)/dt),grad(q))*dx_stator \
\
  + inner(sigma_stl*cross(omega_v*r,curl(A)),grad(q))*dx_rotor \
  + inner(sigma_mag*cross(omega_v*r,curl(A)),grad(q))*dx_magnet \

form = f_A + f_V
a, L = system(form)
```

Fig. 4. Derived A-V formulation expressed using UFL.

method (GMRES), bi-conjugate gradient stabilised method (BiCGSTAB), and transpose free quasi-minimal residual method (TFQMR) [48]. Similarly, the block Jacobi and various multigrid methods, including algebraic multigrid, were assessed to improve the rate of convergence. The methods were judged by the number of iterations, walltime, and accuracy compared to a direct solution. The GMRES solver exceeded the performance of BiCGSTAB in terms of walltime, and the TFQMR solver did not converge. Further, the block Jacobi preconditioner delivered greater accuracy to the direct solution than the trialled multigrid methods.

The complete parameters of the solver are provided in Fig. 7. The GMRES method computes orthogonal sequences, and all previous vectors in the sequences must be stored. This approach, therefore, requires memory proportional to the maximum number of iterations. The method was set to restart every 50 iterations within the solver parameters to prevent this, clearing the intermediate sequences. Restarting reduces the maximum memory needed to solve the problem and reduces computational work as the GMRES solver must orthogonalise against all of the previous vectors. Other restart values were tested but to no benefit from the initial selection.

The modified Gram–Schmidt algorithm is used to orthogonalise against the Krylov space. This algorithm offers better stability than the classical Gram–Schmidt but is more computationally expensive. The improved robustness of the solver was determined to be worth the additional cost, especially in the endeavour to reduce the reliance

```
u0 = Function(V_VF)
u0.vector.set(0.)
u0.vector.ghostUpdate(addv=PETSc.InsertMode.INSERT,
                      mode=PETSc.ScatterMode.FORWARD)

outer_faces = locate_dofs_topological(V_VF,
                  mesh.topology.dim - 1,
                  mt_facets.indices[np.logical_or.reduce((
                      mt_facets.values == 1,
                      mt_facets.values == 2,
                      mt_facets.values == 3,
                      mt_facets.values == 4,
                      mt_facets.values == 5,
                      mt_facets.values == 6))])

bcs.append( DirichletBC(u0.sub(0), outer_faces) )
```

Fig. 5. Derived boundary condition expressed using DOLFINx.

```
A = assemble_matrix(a, bcs)
A.assemble()

b = assemble_vector(L)
apply_lifting(b, [a], [bcs])
b.ghostUpdate(addv=PETSc.InsertMode.ADD,
              mode=PETSc.ScatterMode.REVERSE)
set_bc(b, bcs)
```

Fig. 6. System assembly using DOLFINx.

```
solver = PETSc.KSP().create(mesh.mpi_comm())
solver.setOptionsPrefix('AV_')
opts = PETSc.Options('AV_')
opts['ksp_type'] = 'gmres'
opts['ksp_gmres_restart'] = 50
opts['ksp_gmres_modifiedgramschmidt'] = None
opts['ksp_diagonal_scale'] = None
opts['ksp_rtol'] = 1e-08
opts['ksp_max_it'] = 50000
opts['pc_type'] = 'bjacobi'
solver.setFromOptions()
```

Fig. 7. Iterative solver configuration expressed using PETSc for Python.

on direct solvers in industry. It has been shown that row scaling can improve the convergence of nonsymmetric systems with discontinuous coefficients [49]. Diagonal scaling displayed similar benefits and was implemented as it is readily available through the PETSc solver parameters. The linear system does not need to be scaled back as it is reconstructed before the next time-step. The relative tolerance is set as the desired convergence criterion. The solver will continue iterating until the relative decrease in the residual norm drops below the given value. Another convergence criterion is the maximum number of iterations, which was increased from PETSc's default of 10 000. This change will ensure that the solution of each time-step completes due to satisfying the relative convergence tolerance rather than the maximum number of iterations.

Incomplete LU factorisation can significantly improve the convergence of GMRES [50]. However, its application is not scalable due to its inherently sequential process with strong data dependencies. This limitation can be overcome by partitioning the system using a block method such as block Jacobi [51], which is one of the simplest parallel preconditioners. The technique involves breaking down the linear system into independent sub-systems corresponding to the diagonal blocks in the reordered matrix. Each sub-system can be computed locally and, as there is no overlap, the application of the preconditioner requires no communication between blocks. Therefore, partitioning the problem into blocks equal to the number of processes achieves parallelisation without communication penalties. A well-known shortcoming is the number of iterations to converge typically increases with the number of blocks. Although this behaviour is far from ideal, the strategy leads to favourable performance if the benefits of parallelism and minimised communication exceed the increased number of iterations.

3.6. Verification model

An equivalent model is solved using a commercial electromagnetic solver to understand if the FEniCSx results are accurate for the given problem. Naturally, Ansys Maxwell was considered as the geometry was already available. The electromagnetic solvers within Maxwell were initially developed in the 1980s [52] and are now packaged within the Ansys Electronics Suite. Ansys Maxwell meets the criteria as a robust package extensively used to develop production-ready designs. Ansys Electronics version 2019 R3 was used to create and solve the verification model. The sector model generated in Section 3.3 was

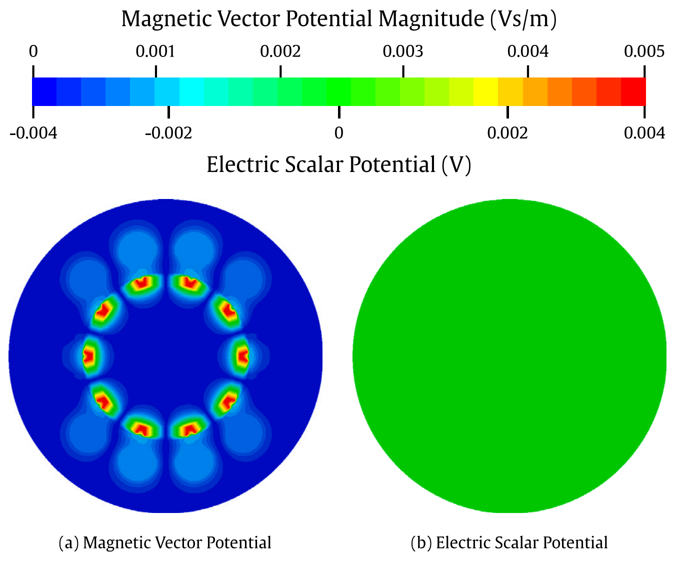


Fig. 8. Contour plots of the solution at $t = 0$ s.

modified to use the same maximum edge lengths, excitations, and material properties as in FEniCSx.

The solutions of the primary variables could not be compared as Ansys Maxwell implements the T-Ω formulation [53]. This formulation is derived in a similar way to the A-V formulation but instead implements the electric vector potential $\boldsymbol{T}$ and the magnetic scalar potential Ω. Given an adequately sized mesh, the crucial physical quantities calculated in the post-processing step are almost identical between formulations [54]. The magnetic flux density solution, wholly dependent on the magnetic vector potential, can be calculated via both formulations and was used to verify the FEniCSx model. Also, as the parallel performance of Ansys Maxwell was not measured, the model was solved using a direct solver. Although Ansys Maxwell has implemented iterative solvers, they are disabled for transient simulations [55]. Instead, to take advantage of HPC systems, the time decomposition method (TDM) is enabled for transient models. The TDM parallelises the problem by solving each time-step simultaneously, rather than sequentially [56]. This method has inherent limitations as the solution at any time-step cannot depend on a previous time-step. Therefore, the TDM will not be used in the analysis as it does not support hysteresis modelling or mechanical transient behaviours [57].

4. Results and discussion

The results from the model were post-processed using the open-source data visualisation tool ParaView [58]. The results presented in this section were computed using the largest mesh from Table 2, unless stated otherwise.

4.1. Primary solutions

The magnetic vector potential and electrical scalar potential solutions are displayed at the motor's mid-plane in Fig. 8. The outer domain is disregarded in the plots as both quantities are zero in the surrounding air.

The maximum values within the magnetic vector potential plot in Fig. 8(a) highlight the outer edges of the permanent magnets. The contribution from the coils is shown within eight slots of the motor, corresponding to the second and third phases. This distribution is expected as the first phase has no phase shift, and the current should equal zero at the first time-step. The value is negligible within the majority of the stator and inner region of the rotor. Fig. 8(b) shows that the electric scalar potential is zero at the motor's mid-plane.

Table 6
Walltime comparison for solving the full Mesh1 model in serial.

Model	DOFs	Solver	Processor	Walltime (s)
FEniCSx	83 856	Iterative	AMD EPYC 7452	19
Ansys Maxwell	45 871	Direct	Intel i5-3570K	59

4.2. Model verification

The first step before verifying the model is computing the magnetic flux density, a simple calculation through Eq. (4). This was performed through ParaView's Python calculator function for FEniCSx while Ansys Maxwell outputs this quantity by default. Fig. 9 presents this solution for both models at 0 and 0.1 s. The magnetic flux density is an essential quantity for engineers as it can be used to calculate the torque through approaches such as the Maxwell stress tensor method [41,59–61]. Ansys Maxwell uses the moving band principle to account for rotation as opposed to the motion voltage term [62]. As this method involves physically translating the mesh, the time-steps presented were chosen to ensure the rotors were in the same position across models. To achieve this, the model's final time from Table 4 was increased to 100 ms, with the remaining simulation parameters unchanged. The verification model was post-processed using tools built-in to Ansys Maxwell.

The FEniCSx results show a similar pattern overall to the verification model at the first time-step. The maximum magnetic flux density values for both are at the outer corners of the permanent magnets, although the magnitude is not as extreme in the FEniCSx model. The coarser result from Ansys Maxwell varies more and reaches a greater depth around the rotor. These trends remain relevant after a full rotation, with the most noticeable change being the more extensive distribution between the stator slots.

Due to the use of a sector model, the meshes are not identical between models, which can cause slight variations in the solution. However, as the same maximum edge lengths and meshing algorithm were used for both models, it is doubtful that differences in the mesh would significantly influence the results.

The magnetic flux density values along the x-axis at 0.1 s have been plotted in Fig. 10 to verify the model further. The magnitude in the static parts of the motor matches well, while more considerable differences are seen within the rotor, mirroring the trends in the contour plots. The maximum value experienced at the outer edge of the rotor is 7% higher in Ansys Maxwell. This trend reverses at the inner side of the permanent magnets where Ansys Maxwell predicts a value 34% lower than the FEniCSx model. Inspection of the results near material boundaries shows no sign of the Gibbs phenomenon. These differences would affect the torque of the motor predicted by both models. The variance could be explained by the different formulations or methods used to account for rotation as the static region remains within 0.1 T between both models.

Table 6 demonstrates the speed advantage of iterative solvers by comparing the solving step walltime in serial for six time-steps. Mesh1 was used to allow both models to solve the full model mesh and the different number of DOFs is a result of Ansys Maxwell's tuned implementation of the T-Ω formulation. Due to being solved on a single process, the processor's clock rate is the critical metric rather than the number of cores or threads. The FEniCSx model was solved using an AMD EPYC 7452, which has a maximum clock rate of 3.35 GHz, while the model in Ansys Maxwell was solved on an Intel i5-3570K with a higher maximum of 3.80 GHz. The maximum clock rates are referenced as the problems were solved in isolation on each processor. The iterative FEniCSx model achieved a 67.8% walltime reduction compared to the commercial direct solver despite solving a larger matrix on a slower processor (in terms of serial performance).

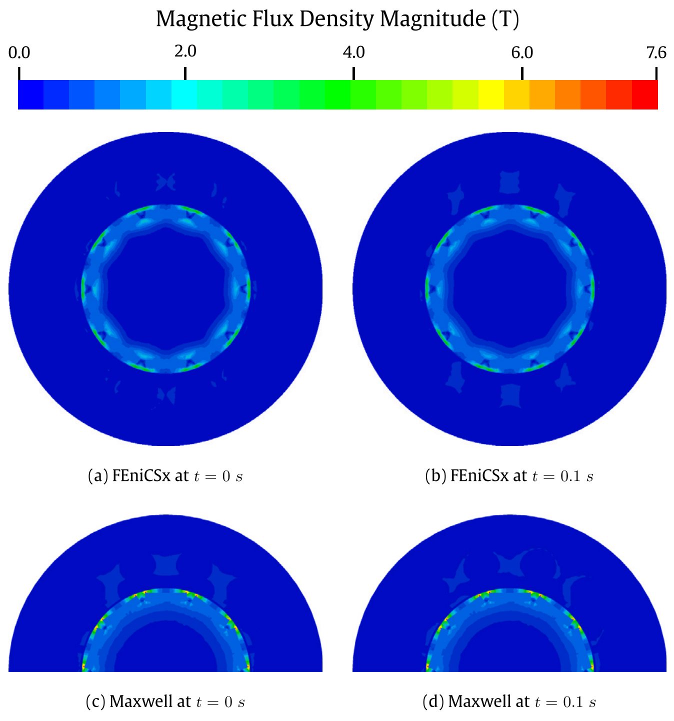


Fig. 9. Contour plots of the magnetic flux density solution.

4.3. *Suitability for high-performance simulations*

The problem was solved on dual-socket compute nodes featuring AMD EPYC 7452 processors with 256 GB of memory, FDR InfiniBand, and BeeGFS parallel file system. Each compute node has 64 cores and threads as simultaneous multithreading was disabled. Accordingly, inter-node communication only occurred on problems spanning more than 64 processes. The final time was reverted to the value from Table 4, and thus six time-steps were computed per run.

4.3.1. *Scaling*

The preconditioned GMRES solver remained stable in parallel, converging on identical solutions from serial execution to execution across the maximum number of processes. The impact of the number of cores on the overall walltime is shown in Fig. 11, and as the total problem size is fixed, the strong scaling is demonstrated. MPI processes were distributed across the entire compute node to allow the strong scaling to be assessed up to 768 processes. This approach can encounter memory-bandwidth constraints, which can be alleviated by partially occupying each compute node. For example, this model solved 14.3% quicker across 128 processes when occupying only half the compute nodes. However, this strategy required four compute nodes to be reserved, and for the same effort, running across all 256 processes achieved a more significant improvement in walltime of 39.5%. The most significant operations within the code were timed and highlighted separately, providing insights into how each scale. A brief description of each section is provided: “Read Mesh” comprises of reading, partitioning, and distributing the mesh data across each process; “Define Problem”, the initialisation of the UFL form, excitation vectors, and boundary conditions; “Assemble System”, the operations presented in Figure 6. “Solve”, the solution of the problem using PETSc; “Update Problem”, incrementing the simulation time, updating the source current excitation vector, and copying the solution vector for the magnetic vector potential; “Other”, the remaining operations in the model, such as initialising variables and writing the solution.

The chart illustrates that the model positively scaled up to 512 processes and that all but one operation took advantage of parallel processing. Positive scaling occurs when a problem’s walltime reduces as the number of processes increases and conversely, negative scaling when the walltime increases with the number of processes. Positive scaling was not achieved beyond 512 processes as the problem size per process likely became too small, despite using the largest mesh. This limit seemed to be around 11 000 DOFs per process. In this situation, the particular mesh has been excessively partitioned, and the simulation becomes dominated by the additional communication between processes. The reading of the mesh file is the only operation that negatively scaled, accounting for 54% of the total walltime at 768 processes. Every mesh algorithm implemented in FEniCSx, with a walltime exceeding one second, scaled positively. The overall negative scaling of the mesh reading portion is due to the external library PT-Scotch [63] which FEniCSx uses to partition and distribute the mesh for each process. A report by developers of FEniCSx shows negative scaling of the partition operation in the previous version, FEniCS [64]. However, one of the three systems trialled in the article scaled positively to 512 cores, demonstrating that it is achievable. At lower core counts, defining and updating the problem were the two most computationally expensive portions of the code. This is because of the excitation expressions, which involve looping over every point of the mesh. This operation could be optimised to improve the walltime at lower core counts, but this did not seem necessary as it efficiently scaled. It is expected that only iterating over the coil and magnet domains, and caching the coordinate lists, would reduce the combined walltime of these operations by more than 90%.

The trends discussed in the previous chart become more apparent when the speedup is plotted, as in Fig. 12. The following formula was used to calculate the speedup

$$\text{Speedup} = \frac{T(N,1)}{T(N,P)} \tag{22}$$

where T is the walltime, N is the size of the problem, and P is the number of processes. Therefore, the speedup is the ratio of the time taken to solve the problem in serial against the time taken in parallel across P number of processes. Due to operations that cannot be parallelised and increased communication at scale, the speedup is typically less than the number of processes [65]. Speedup that equals the number of processes is known as ideal scaling. Fig. 12(a) shows the maximum speedup of 198 achieved and that the scaling efficiency slowly decreases before reversing at 512 processes.

The scalability of the code sections in Fig. 12(b) helps to prioritise which operations should be focused on to improve the model’s parallel performance. The two most expensive portions discussed previously showed the best speedup and did not plateau. The longer walltime of these portions gives them a higher weighting towards the overall speedup, meaning that the overall speedup seen in Fig. 12(a) would be lower if their computational cost were reduced. Conversely, although declining from different points, the remaining operations could not scale efficiently beyond 512 processes. At 128 processes, the solving step was the second-least efficient parallel operation but achieved the third-best speedup at the largest core count. Although the “Other” section had the second-worst speedup at 768 processes, it only contributed to 3% of the overall walltime.

Although definite improvements could be made, these charts can understate the practical benefit of this scaling performance. The model took 13.5 h to solve in serial but only required four minutes across 512 processes. This capability dramatically reduces the turnaround time of the simulation, enabling designs to be improved at a much quicker pace. Comparing this performance with Ansys Maxwell reinforces the advantage of iterative methods, as the full Mesh4 model could not run on a PC with 24 GB of memory. The 535 GB of memory needed by FEniCSx to directly solve Mesh4 suggests it would not be possible with Ansys Maxwell on any engineering PC, usually equipped with 64 GB at most.

A white paper from Ansys describes the parallel solution of a two-dimensional transient traction motor using the distributed solve option

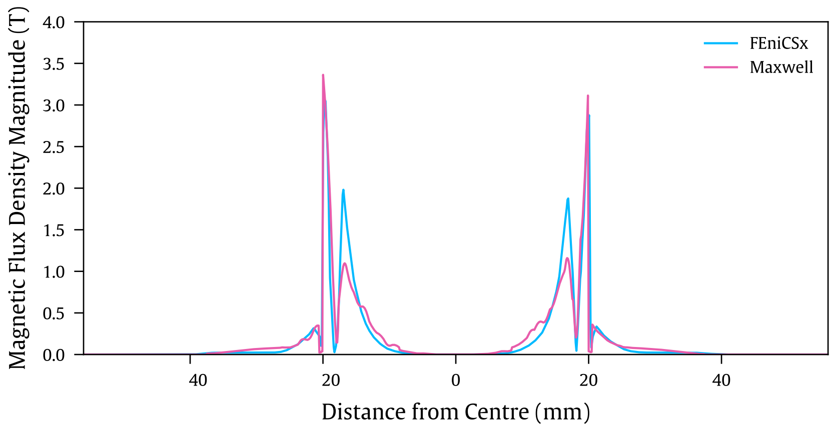


Fig. 10. Magnetic flux density along the motor's x-axis at $t = 0.1$ s.

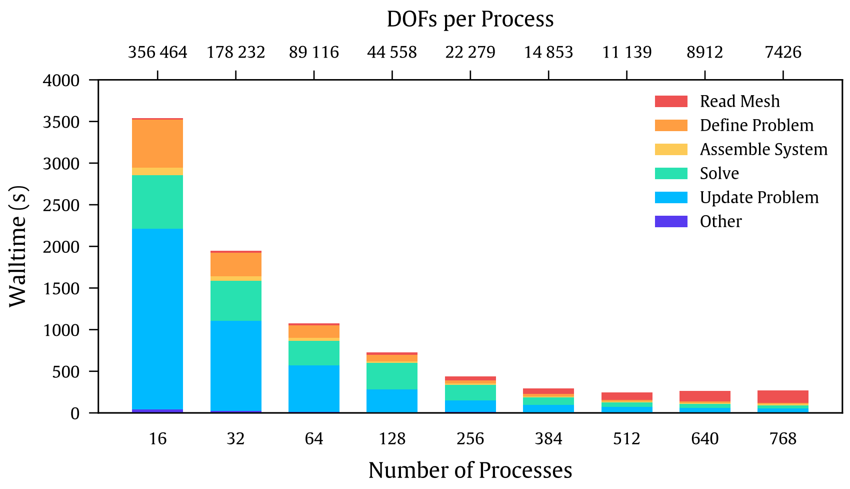


Fig. 11. Strong scaling on the UH HPC system.

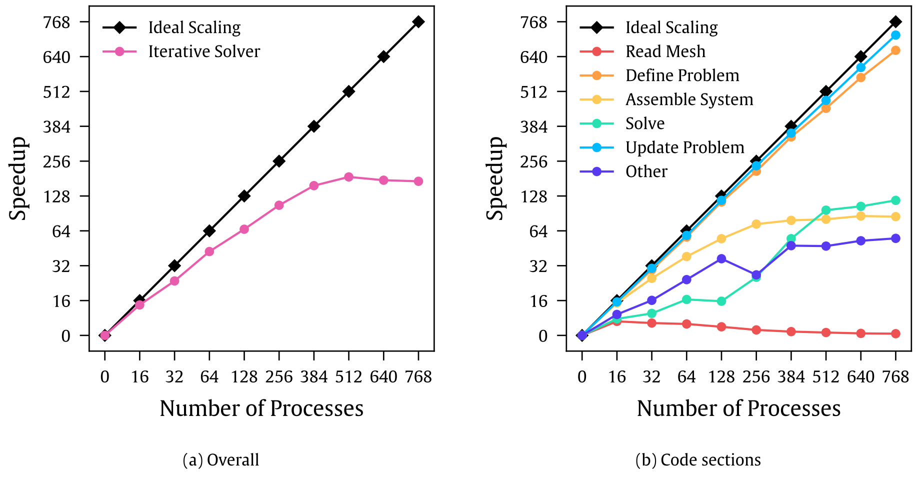


Fig. 12. Speedup of the model using an iterative solver.

(DSO) [66]. Ansys Maxwell achieved a speedup of 16 across 32 cores, 36% lower than the 25 times speedup obtained by the FEniCSx model at the same core count. The TDM is another approach implemented into Ansys Maxwell to take advantage of HPC systems. A three-dimensional PMSM model was simulated by the authors of the TDM, demonstrating a speedup of 13 at 256 processes [56]. Further, an Ansys white paper solving a three-dimensional induction motor shows that Ansys Maxwell's implementation of the TDM reached a speedup of 15 at 1024 cores [67]. These are both significantly lower than the maximum speedup displayed by the FEniCSx model at 512 processes.

The weak scaling performance of the model is demonstrated in Fig. 13, where the problem size increases proportionally with the number of processes. Due to hardware limitations, a mesh larger than Mesh4 could not be generated, and therefore the weak scaling is only

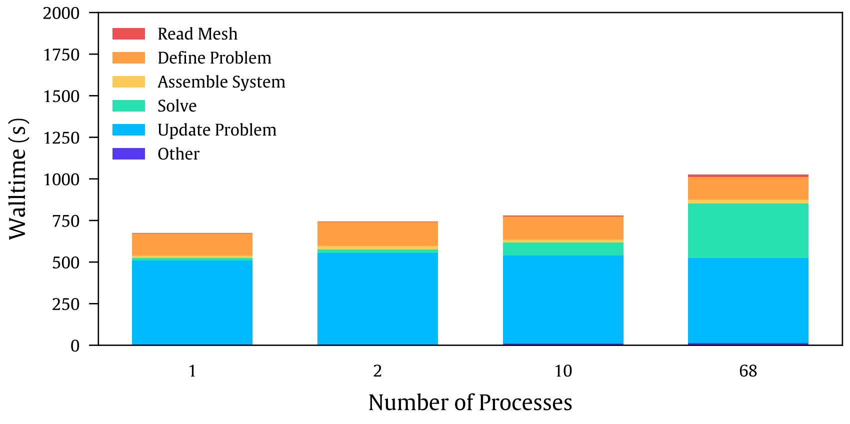


Fig. 13. Weak scaling on the UH HPC system.

Table 7
Weak scaling statistics.

Mesh name	Number of processes	DOFs per process	Number of iterations	Maximum memory per process (MB)
Mesh1	1	83 856	161	613
Mesh2	2	93 882	223	673
Mesh3	10	85 874	474	664
Mesh4	68	83 874	1255	671

assessed up to 68 processes. Ideally, the weak scaling across many compute nodes would be evaluated to better understand the model's performance at a greater scale. Each mesh was run with around 84 000 DOFs (±11%) per process. Additional statistics concerning the weak scaling are presented in Table 7.

Most of the code sections, particularly those managed entirely through FEniCSx, scaled efficiently with the problem size, demonstrating platform's excellent scalability. Solving the model across 68 processes led to just a 1.23% increase in the walltime of these sections compared to executing in serial. Reading the mesh and solving, primarily managed by the PT-Scotch and PETSc libraries, scaled poorly in comparison. The solving section displays the most dramatic change, with the walltime increasing by a factor of 12 from serial to 68 processes. This behaviour is partly due to the block Jacobi preconditioning, where the number of blocks, equivalent to the number of processes, is known for increasing the number of iterations to converge. Another factor to consider is that the model solved across 68 processes was the only model spanning more than one compute node. Hence, inter-node communication was not occurring during the other runs. The maximum memory required to run each model is an encouraging result, remaining bound with respect to the model size in parallel.

4.3.2. Memory usage

The memory benefit of iterative methods is demonstrated in Fig. 14. The maximum memory used by the model was compared using an iterative and direct solver. The direct solver uses the PETSc implementation of LU decomposition. The chart shows that the restarted GMRES iterative solver requires 42% less memory when solving the smallest mesh. This advantage exponentially grows to a 92% reduction for the largest mesh, only needing 45 GB of memory. It is clear from these results why iterative solvers are a leading candidate in making the solution of extreme-scale models feasible.

4.3.3. Storage usage

The resources required to store meshes and results is the final aspect of the model evaluated. Storing data affects how economical a simulation is, with larger files costing more to retain. A compromise exists between the cost of maintaining the results against solving the model again when needed.

The mesh and result data are stored using the HDF5 file format, as it allows for reading and writing in parallel [68]. The mesh files comprise five arrays: the mesh geometry and topology, the facet topology and numbering, and the domain numbering. The result files also include the mesh geometry and topology along with the nodal solution at each time-step. The result files could be output without the mesh data to avoid duplicating information; however, its inclusion simplifies the visualisation of the data through tools such as ParaView.

Figs. 15 and 16 illustrate how the model's raw and compressed file sizes grow with the mesh. The file format is space-efficient, with the largest file displayed only reaching 0.5 GB. Although writing compressed data in parallel is not yet possible through FEniCSx, it can be executed later using h5repack, a tool from the HDF group, to reduce the amount of cold storage required to store the model's data. Lossless compression was carried out using the gzip filter available through h5repack with the maximum compression level. This process was inexpensive, with all of the files compressed serially in under a minute.

The files show a linear (or near) relationship against the number of cells, and extrapolating the data suggests that a mesh with one billion cells would only require 45 GB of storage and 14 GB if compressed. Compression reduced the vector and scalar solution file sizes by around 50% and 64%, respectively. The result files shown contain the values for six time-steps. If many more time-steps were computed, the model could be modified to only save the results for certain time-steps instead.

5. Conclusions

This article demonstrates the scalable simulation of a PMSM through the FEniCSx platform. The derivation of the A-V formulation is presented along with its implementation in FEniCSx. The electromagnetic analysis of a three-phase PMSM at its synchronous speed is carried out to benchmark the model.

The solutions of the weak formulation are smooth and reflect the expected behaviour of the motor. The magnetic flux density was computed to verify against the result of an equivalent model from a commercial electromagnetic package. The results match well both qualitatively and quantitatively in the static regions, with the maximum values deviating by 7%. The variation increases in the rotor, with the verification model predicting a 34% lower magnitude than the FEniCSx

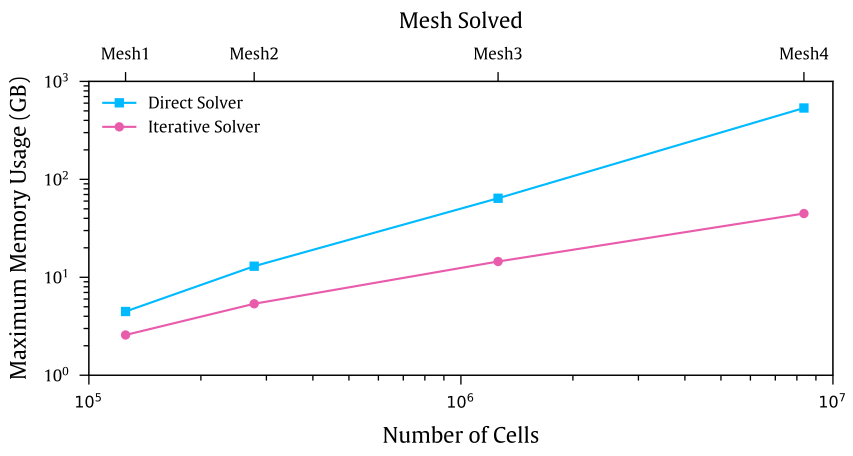


Fig. 14. Maximum memory usage for each mesh with both solvers.

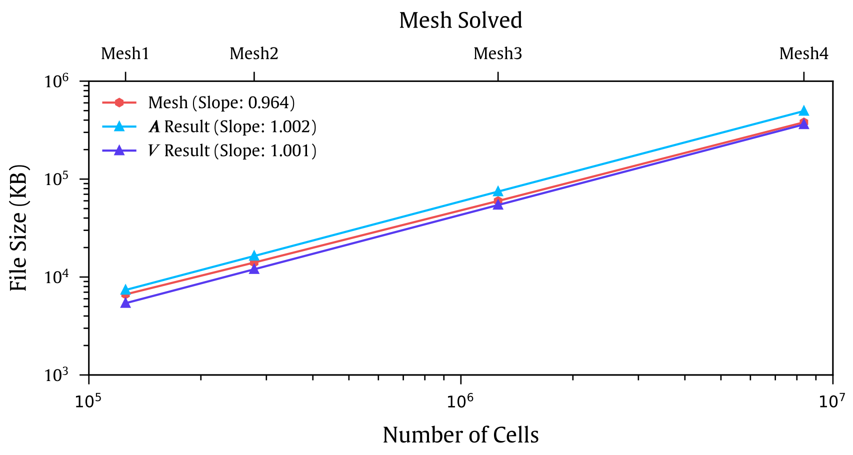


Fig. 15. File size for meshes and results.

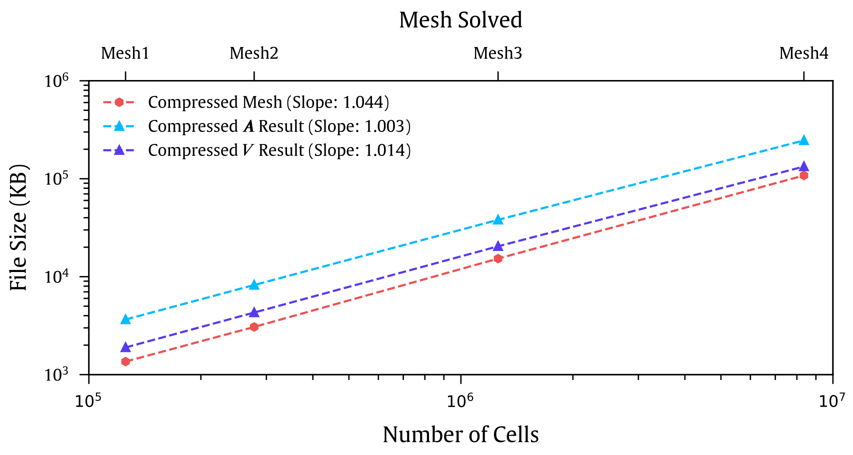


Fig. 16. File size for compressed meshes and results.

model. However, the commercial package uses a differing method to account for the motor's rotation. The iterative solver in FEniCSx demonstrated a 68% walltime reduction compared to Ansys Maxwell's direct solver when computing identical meshes.

The model's performance on HPC systems was investigated in three areas: scaling, memory usage, and storage usage. Solving the largest mesh, with over eight million cells, achieved a speedup of 198 across 512 processes. The speedup plateaued beyond this, likely due to communication dominating the simulation as the problem size per process further reduces. This limit was reached at around 11 000 DOFs per process and suggests that a larger model could achieve a greater speedup beyond 512 processes.

The speedup demonstrated by FEniCSx exceeded that of Ansys Maxwell using the DSO and TDM approaches. The FEniCSx model showed a 57% greater speedup than the Ansys Maxwell model solved using the DSO and a 16.5 times larger speedup than the TDM at identical core counts.

The weak scaling was assessed up to 68 processes with the problem size kept around 84 000 DOFs (±11%) per process. The results demonstrated efficient scaling for the code sections managed by FEniCSx. The walltime of these sections only increased by 1.23% when solved across 68 processes compared to the serial execution. Libraries external to FEniCSx primarily manage the mesh reading and solution stages, which did not scale as efficiently. The block Jacobi preconditioner implemented likely contributed to the increased walltime of the solving step as it typically requires more iterations to converge when distributed across more processes. The maximum memory used by the model remained bound in parallel at approximately 670 MB per process. Evaluating the weak scaling at higher core counts was not possible due to hardware limitations that restricted the maximum mesh size.

PETSc's implementation of the restarted GMRES iterative method significantly benefited memory usage compared to LU decomposition, a direct approach. Only 45 GB of memory was needed to solve the largest mesh, 92% less than was required when using the direct method. Finally, the model file sizes were analysed to understand the amount of space needed to store the data. This was another positive finding with the largest file, the vector result file for Mesh4, only requiring 0.5 GB of space.

There are many opportunities to develop the model further to improve its performance. The derivation of the T-Ω formulation and use of Nédélec edge elements could reduce the number of DOFs compared to the nodal A-V formulation used in this work. This endeavour requires more than modifying the UFL form as preconditioning $H(\text{curl})$ spaces is significantly more challenging than H^1 spaces [69]. This could improve the effectiveness of multigrid algorithms as a preconditioner, which can be shown to be asymptotically optimal for many problems [70]. Such algorithms have been developed for 3D Maxwell problems discretised using edge elements [27,71]. Resolutions to the negative scaling of the mesh operations may be found by substituting PT-Scotch with an alternate parallel graph partitioner. The latest version of FEniCSx has a functional interface to ParMETIS and KaHIP, and the potential benefits of these partitioners could be explored. Further, non-linear magnetic behaviour could be implemented to improve the model's accuracy to real-world data, and the benefits of higher-order elements could be investigated.

Enabling highly scalable simulations of electric machines is necessary to take advantage of ever-expanding HPC clusters. Developments upon the techniques explored in this paper could bring the capability to solve extremely large models that are currently not feasible. The data from these simulations would empower engineers to develop the high-performance machines needed to electrify the entire transport sector.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to express our gratitude to Sandeep Shrivastava and the Future Methods team at Rolls-Royce plc for their support and guidance. The use of the University of Hertfordshire's high-performance computing facility within this research is recognised.

References

[1] IEA, Global EV Outlook 2021, IEA, Paris, 2021, URL https://www.iea.org/reports/global-ev-outlook-2021.
[2] IEA, Tracking Transport 2020, IEA, Paris, 2020, URL https://www.iea.org/reports/tracking-transport-2020.
[3] R. Zhang, S. Fujimori, The role of transport electrification in global climate change mitigation scenarios, Environ. Res. Lett. 15 (3) (2020) 034019, http://dx.doi.org/10.1088/1748-9326/ab6658.
[4] S. Wolff, S. Kalt, M. Bstieler, M. Lienkamp, Influence of powertrain topology and electric machine design on efficiency of battery electric trucks—A simulative case-study, Energies 14 (2) (2021) 328, http://dx.doi.org/10.3390/en14020328.
[5] A. El-Refaie, M. Osama, High specific power electrical machines: A system perspective, in: 2017 20th International Conference on Electrical Machines and Systems, ICEMS, IEEE, 2017, pp. 1–6, http://dx.doi.org/10.1109/ICEMS.2017.8055931.
[6] A. Williams, Aviation electrification - choosing your motor topology and material advancements, in: AIAA Propulsion and Energy 2019 Forum, 2019, http://dx.doi.org/10.2514/6.2019-4480.
[7] M. Henke, G. Narjes, J. Hoffmann, C. Wohlers, S. Urbanek, C. Heister, J. Steinbrink, W.R. Canders, B. Ponick, Challenges and opportunities of very light high-performance electric drives for aviation, Energies 11 (2) (2018) 334, http://dx.doi.org/10.3390/en11020344.
[8] V. Madonna, P. Giangrande, M. Galea, Electrical power generation in aircraft: Review, challenges, and opportunities, IEEE Trans. Transp. Electrification 4 (3) (2018) 646–659, http://dx.doi.org/10.1109/TTE.2018.2834142.
[9] A. George, Nested dissection of a regular finite element mesh, SIAM J. Numer. Anal. 10 (2) (1973) 345–363, http://dx.doi.org/10.1137/0710032.
[10] C.N. Richardson, N. Sime, G.N. Wells, Scalable computation of thermomechanical turbomachinery problems, Finite Elem. Anal. Des. 155 (2019) 32–42, http://dx.doi.org/10.1016/j.finel.2018.11.002.
[11] B. Zhou, D. Jiao, A linear complexity direct finite element solver for large-scale 3-D electromagnetic analysis, in: 2013 IEEE Antennas and Propagation Society International Symposium, APSURSI, IEEE, 2013, pp. 1684–1685, http://dx.doi.org/10.1109/APS.2013.6711501.
[12] C.E. Leiserson, N.C. Thompson, J.S. Emer, B.C. Kuszmaul, B.W. Lampson, D. Sanchez, T.B. Schardl, There's plenty of room at the top: What will drive computer performance after moore's law? Science 368 (6495) (2020) eaam9744, http://dx.doi.org/10.1126/science.aam9744.
[13] M.S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, G.N. Wells, The FEniCS project version 1.5, Arch. Numer. Softw. 3 (100) (2015) http://dx.doi.org/10.11588/ans.2015.100.20553.
[14] A. Logg, K.-A. Mardal, G.N. Wells, et al., Automated Solution of Differential Equations by the Finite Element Method, Springer, 2012, http://dx.doi.org/10.1007/978-3-642-23099-8.
[15] M. Habera, J.S. Hale, C.N. Richardson, J. Ring, M.E. Rognes, N. Sime, G.N. Wells, FEniCSX: A sustainable future for the FEniCS project, in: SIAM PP20 Minisymposium: Improving Productivity and Sustainability for Parallel Computing Software, Seattle WA, US, 2020, http://dx.doi.org/10.6084/m9.figshare.11866101.
[16] M.S. Alnæs, A. Logg, K.B. Ølgaard, M.E. Rognes, G.N. Wells, Unified form language: A domain-specific language for weak formulations of partial differential equations, ACM Trans. Math. Software 40 (2) (2014) 1–37, http://dx.doi.org/10.1145/2566630.
[17] M.S. Alnæs, UFL: a finite element form language, in: A. Logg, K.-A. Mardal, G.N. Wells (Eds.), Automated Solution of Differential Equations By the Finite Element Method, Springer, 2012, pp. 303–338, http://dx.doi.org/10.1007/978-3-642-23099-8_17.
[18] R.C. Kirby, A. Logg, A compiler for variational forms, ACM Trans. Math. Software 32 (3) (2006) 417–444, http://dx.doi.org/10.1145/1163641.1163644.
[19] A. Logg, K.B. Ølgaard, M.E. Rognes, G.N. Wells, FFC: the FEniCS form compiler, in: A. Logg, K.-A. Mardal, G.N. Wells (Eds.), Automated Solution of Differential Equations By the Finite Element Method, Springer, 2012, pp. 227–238, http://dx.doi.org/10.1007/978-3-642-23099-8_11.
[20] K.B. Ølgaard, G.N. Wells, Optimisations for quadrature representations of finite element tensors through automated code generation, ACM Trans. Math. Software 37 (1) (2010) 1–23, http://dx.doi.org/10.1145/1644001.1644009.
[21] M.W. Scroggs, J.S. Dokken, C.N. Richardson, G.N. Wells, Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes, 2021, pp. 1–23, 1(1), arXiv:2102.11901, Preprint.
[22] S. Balay, S. Abhyankar, M.F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E.M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M.G. Knepley, F. Kong, S. Kruger, D.A. May, L.C. McInnes, R.T. Mills, L. Mitchell, T. Munson, J.E. Roman, K. Rupp, P. Sanan, J. Sarich, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, J. Zhang, PETSc web page, 2021, URL https://www.mcs.anl.gov/petsc.
[23] S. Balay, S. Abhyankar, M.F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E.M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M.G. Knepley, F. Kong, S. Kruger, D.A. May, L.C. McInnes, R.T. Mills, L. Mitchell, T. Munson, J.E. Roman, K. Rupp, P. Sanan, J. Sarich, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, J.

Zhang, PETSc/TAO Users Manual, ANL-21/39 - Revision 3.16, Argonne National Laboratory, 2021.

[24] S. Balay, W.D. Gropp, L.C. McInnes, B.F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202, http://dx.doi.org/10.1007/978-1-4612-1986-6_8.

[25] C.C. Chan, K.T. Chau, Design of electrical machines by the finite element method using distributed computing, Comput. Ind. 17 (4) (1991) 367–374, http://dx.doi.org/10.1016/0166-3615(91)90049-F.

[26] N. Simpson, P.H. Mellor, Exploiting cloud computing in the multi-physics design and optimisation of electromagnetic devices, in: 2015 IEEE 16th Workshop on Control and Modeling for Power Electronics, COMPEL, IEEE, 2015, pp. 1–8, http://dx.doi.org/10.1109/COMPEL.2015.7236508.

[27] G. Haase, M. Kuhn, U. Langer, Parallel multigrid 3D maxwell solvers, Parallel Comput. 27 (6) (2001) 761–775, http://dx.doi.org/10.1016/S0167-8191(00)00106-X.

[28] I.D. Mayergoyz, G. Bedrosian, Iterative solution of 3D eddy current problems, IEEE Trans. Magn. 29 (6) (1993) 2335–2340, http://dx.doi.org/10.1109/20.280859.

[29] D. Rodger, N. Allen, P.C. Coles, S. Street, P.J. Leonard, J.F. Eastham, Finite element calculation of forces on a DC magnet moving over an iron rail, IEEE Trans. Magn. 30 (6) (1994) 4680–4682, http://dx.doi.org/10.1109/20.334188.

[30] Y. Liu, A. Bondeson, R. Bergström, C. Johnson, M.G. Larson, K. Samuelsson, Eddy-current computations using adaptive grids and edge elements, IEEE Trans. Magn. 38 (2) (2002) 449–452, http://dx.doi.org/10.1109/20.996119.

[31] D. Rodger, H.C. Lai, P.J. Leonard, A comparison of finite-element models for 3-D rotating conductors, IEEE Trans. Magn. 38 (2) (2002) 537–540, http://dx.doi.org/10.1109/20.996141.

[32] M. Mirzaie, M. Yazdani-Asrami, A.A.S. Akmal, 3D electromagnetic study of transformers' flux line distribution and losses determination under harmonic distortion caused by electronic equipments, Sci. Res. Essays 6 (20) (2011) 4414–4420, http://dx.doi.org/10.5897/sre11.901.

[33] D. Marcsa, M. Kuczmann, Motional finite element simulation of a single-phase induction motor, Pollack Period. 4 (2) (2009) 57–66, http://dx.doi.org/10.1556/Pollack.4.2009.2.6.

[34] J.P.A. Bastos, N. Sadowski, Statics and quasistatics electromagnetics brief presentation, in: Magnetic Materials and 3D Finite Element Modeling, CRC Press, 2014, pp. 1–57, http://dx.doi.org/10.1201/b15558-1.

[35] A. Bermúdez, D. Gómez, P. Salgado, The eddy currents model, in: Mathematical Models and Numerical Simulation in Electromagnetism, Springer, 2014, pp. 183–216, http://dx.doi.org/10.1007/978-3-319-02949-8_10.

[36] D.C. Jiles, D.L. Atherton, Theory of ferromagnetic hysteresis, J. Magn. Magn. Mater. 61 (1) (1986) 48–60, http://dx.doi.org/10.1016/0304-8853(86)90066-1.

[37] Q. Zhang, S. Cen, The physics models, in: Multiphysics Modeling: Numerical Methods and Engineering Applications, Elsevier Inc., 2016, pp. 1–96, http://dx.doi.org/10.1016/B978-0-12-407709-6.00001-8.

[38] J.P.A. Bastos, N. Sadowski, Maxwell equations, electrostatics, magnetostatics and magnetodynamic fields, in: Electromagnetic Modeling By Finite Element Methods, CRC Press, 2003, pp. 27–121, http://dx.doi.org/10.1201/9780203911174-8.

[39] T. Lubin, A. Rezzoug, 3-D analytical model for axial-flux eddy-current couplings and brakes under steady-state conditions, IEEE Trans. Magn. 51 (10) (2015) 1–12, http://dx.doi.org/10.1109/TMAG.2015.2455955.

[40] Q. Zhang, S. Cen, Mesh controls for rotating machinery, in: Multiphysics Modeling: Numerical Methods and Engineering Applications, Elsevier Inc., 2016, pp. 164–166, http://dx.doi.org/10.1016/B978-0-12-407709-6.00004-3.

[41] N. Sadowski, Y. Lefevre, M. Lajoie-Mazenc, J. Cros, Finite element torque calculation in electrical machines while considering the movement, IEEE Trans. Magn. 28 (2) (1992) 1410–1413, http://dx.doi.org/10.1109/20.123957.

[42] J.P.A. Bastos, N. Sadowski, Movement modeling for electrical machines, in: Electromagnetic Modeling By Finite Element Methods, CRC Press, 2003, pp. 343–366, http://dx.doi.org/10.1201/9780203911174-12.

[43] J.P.A. Bastos, N. Sadowski, Using nodal elements with magnetic vector potential, in: Magnetic Materials and 3D Finite Element Modeling, CRC Press, 2014, pp. 243–259, http://dx.doi.org/10.1201/b15558-6.

[44] H. Song, N. Ida, An eddy current constraint formulation for 3D electromagnetic field calculations, IEEE Trans. Magn. 27 (5) (1991) 4012–4015, http://dx.doi.org/10.1109/20.104981.

[45] J.S. Hale, L. Li, C.N. Richardson, G.N. Wells, Containers for portable, productive, and performant scientific computing, Comput. Sci. Eng. 19 (6) (2017) 40–50, http://dx.doi.org/10.1109/MCSE.2017.2421459.

[46] O. Rudyy, M. Garcia-Gasulla, F. Mantovani, A. Santiago, R. Sirvent, M. Vázquez, Containers in HPC: A scalability and portability study in production biological simulations, in: 2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS, IEEE, 2019, pp. 567–577, http://dx.doi.org/10.1109/IPDPS.2019.00066.

[47] A. Collette, Python and HDF5, O'Reilly, 2013.

[48] A. Ghai, C. Lu, X. Jiao, A comparison of preconditioned Krylov subspace methods for large-scale nonsymmetric linear systems, Numer. Linear Algebra Appl. 26 (1) (2019) e2215, http://dx.doi.org/10.1002/nla.2215.

[49] D. Gordon, R. Gordon, Row scaling as a preconditioner for some nonsymmetric linear systems with discontinuous coefficients, J. Comput. Appl. Math. 234 (12) (2010) 3480–3495, http://dx.doi.org/10.1016/j.cam.2010.05.021.

[50] M. Wang, H. Klie, M. Parashar, H. Sudan, Solving sparse linear systems on NVIDIA Tesla GPUs, in: G. Allen, J. Nabrzyski, E. Seidel, G. Albada, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2009, 5544, Springer, 2009, pp. 864–873, http://dx.doi.org/10.1007/978-3-642-01970-8_87.

[51] W. Ma, Y. Hu, W. Yuan, X. Liu, Developing a multi-GPU-enabled preconditioned GMRES with inexact triangular solves for block sparse matrices, Math. Probl. Eng. 2021 (2021) 6804723, http://dx.doi.org/10.1155/2021/6804723.

[52] Z.J. Cendes, E.C. Smetak, D. Shenton, Mesh generation for modelling in magnetics, Math. Comput. Modelling 11 (1988) 192–196, http://dx.doi.org/10.1016/0895-7177(88)90478-5.

[53] Ansys, 3D transient excitations (sources), in: Maxwell Help, Electromagnetics Suite 2019 R3, Ansys, Inc., 2019, pp. 2928–2930.

[54] D. Marcsa, M. Kuczmann, Comparison of the $A^{*}-A$ and $T,\Phi-\Phi$ formulations for the 2-D analysis of solid-rotor induction machines, IEEE Trans. Magn. 45 (9) (2009) 3329–3333, http://dx.doi.org/10.1109/TMAG.2009.2022328.

[55] Ansys, Iterative matrix solver technical details, in: Maxwell Help, Electromagnetics Suite 2019 R3, Ansys, Inc., 2019, pp. 2925–2928.

[56] B. He, C. Lu, N. Chen, D. Lin, M. Rosu, P. Zhou, Time decomposition method for the general transient simulation of low-frequency electromagnetics, Prog. Electromagn. Res. 160 (2017) 1–8, http://dx.doi.org/10.2528/pier17072501.

[57] Ansys, Time decomposition method for maxwell transient designs, in: Maxwell Help, Electromagnetics Suite 2019 R3, Ansys, Inc., 2019, pp. 1223–1225.

[58] U. Ayachit, The ParaView Guide: A Parallel Visualization Application, Kitware, Inc., 2015.

[59] A. Bermúdez, D. Gómez, P. Salgado, Maxwell's equations in material regions, in: Mathematical Models and Numerical Simulation in Electromagnetism, Springer, 2014, pp. 77–113, http://dx.doi.org/10.1007/978-3-319-02949-8_6.

[60] K.J. Meessen, J.J.H. Paulides, E.A. Lomonova, Force calculations in 3-D cylindrical structures using Fourier analysis and the Maxwell stress tensor, IEEE Trans. Magn. 49 (1) (2013) 536–545, http://dx.doi.org/10.1109/TMAG.2012.2206821.

[61] G.-J. Park, Y.-J. Kim, S.-Y. Jung, Design of IPMSM applying V-Shape skew considering axial force distribution and performance characteristics according to the rotating direction, IEEE Trans. Appl. Supercond. 26 (4) (2016) 1–5, http://dx.doi.org/10.1109/TASC.2016.2543267.

[62] Ansys, Setting up motion for transient projects, in: Maxwell Help, Electromagnetics Suite 2019 R3, Ansys, Inc., 2019, pp. 1103–1111.

[63] C. Chevalier, F. Pellegrini, PT-Scotch: A tool for efficient parallel graph ordering, Parallel Comput. 34 (6) (2008) 318–331, http://dx.doi.org/10.1016/j.parco.2007.12.001.

[64] C.N. Richardson, G.N. Wells, Expressive and Scalable Finite Element Simulation Beyond 1000 Cores, University Distributed CSE Project Report, University of Cambridge, 2013, URL https://www.repository.cam.ac.uk/handle/1810/245070.

[65] D.L. Eager, J. Zahorjan, E.D. Lazowska, Speedup versus efficiency in parallel systems, IEEE Trans. Comput. 38 (3) (1989) 408–423, http://dx.doi.org/10.1109/12.21127.

[66] B. Smith, 16X Speedup in ANSYS Maxwell DSO on 32-Core High-Performance Compute Farm Doubles Traction Motor Design Productivity at General Motors, White paper, Ansys, Inc., 2014.

[67] Ansys, Maxwell Time Decomposition Method Accelerates Simulation of Transient Electromagnetic Fields with Intel Xeon Processor E5-2697 v4, White paper, Ansys, Inc., 2017.

[68] The HDF Group, Hierarchical data format, version 5, 1997-2022, URL https://www.hdfgroup.org/HDF5.

[69] D.N. Arnold, R.S. Falk, R. Winther, Multigrid in H(div) and H(curl), Numer. Math. 85 (2) (2000) 197–217, http://dx.doi.org/10.1007/PL00005386.

[70] E. Chow, R.D. Falgout, J.J. Hu, R.S. Tuminaro, U.M. Yang, A survey of parallelization techniques for multigrid solvers, in: M.A. Heroux, P. Raghavan, H.D. Simon (Eds.), Parallel Processing for Scientific Computing, Society for Industrial and Applied Mathematics, 2006, pp. 179–201, http://dx.doi.org/10.1137/1.9780898718133.ch10.

[71] R. Hiptmair, Multigrid method for Maxwell's equations, SIAM J. Numer. Anal. 36 (1) (1998) 204–225, http://dx.doi.org/10.1137/S0036142997326203.