ON TECHNIQUES FOR BARELY COUPLED MULTIPHYSICS

Rainald Löhner[1], Harbir Antil[2] and Sebastian Schöps[3]

[1]*Center for Computational Fluid Dynamics*
George Mason University, Fairfax, VA 22030-4444, USA
[2]*Center for Mathematics and Artificial Intelligence*
George Mason University, Fairfax, VA 22030-4444, USA
[3]*Computational Electromagnetics Group*
TU Darmstadt, Darmstadt, Germany

A technique to combine codes to solve barely coupled multiphysics problems has been developed. Each field is advanced separately until a stop is triggered. This could be due to a preset time increment, a preset number of timesteps, a preset decrease of residuals, a preset change in unknowns, a preset change in geometry, or any other physically meaningful quantity. The technique allows for a simple implementation in coupled codes using the loose coupling approach. Examples from evaporative cooling of electric motors, a problem that has come to the forefront with the rise of electric propulsion in the aerospace sector (drones and air taxis in particular) shows the viability and accuracy of the proposed procedure.

I. INTRODUCTION

Over the last five decades computational mechanics has matured rapidly. In each of the core disciplines - fluid dynamics, structural dynamics, combustion, heat transfer, acoustics, electromagnetics, mass transfer, control, etc. - robust and efficient numerical techniques have been developed, and a large code base of academic, open source and commercial codes is available. The acquisition of many of these commercial codes by the leading CAD-vendors attests to the desire to streamline the typical computational mechanics workflow (CAD, definition of boundary conditions, loads and physical parameters, solution, mesh adaptation, post-processing) by integrating all parts into a single application.

The ability to obtain accurate and timely results in each of the core disciplines or metiers has prompted the desire to reach the same degree of simplicity in computing multi-physics problems. Considering the different approximation levels possible in each metier, any multidisciplinary capability must have the ability to quickly switch between approximation levels, abstractions, models, grids, and, ultimately, codes. It is clear that only those approaches that allow a maximum of flexibility, i.e.:

- Linear and nonlinear metier-specific models adapted to the problem at hand;

- Different, optimally suited discretizations for each discipline;

- Modularity and extendibility in models and codes;

- Fast multidisciplinary problem definition; and

- Fully automatic grid generation for arbitrary geometrical complexity

will survive in the long term.

A large class of coupled problems exhibits large disparity of timescales. Examples include evaporative cooling (where the flowfield may be established in seconds while the temperature field requires minutes), sedimentation of rivers and estuaries (where the flowfield is established in seconds while the filling up of a channel takes weeks), deposition of cholesterol in arteries (where the flowfield is established after two heartbeats while the deposition can take years), the wear of semi-autogenous grinding (SAG) mills (where the movement of steel balls and mineral-rich rocks and mud is established in minutes while the wear of the liners can take hours), and many others. We denote this class of problems as 'barely coupled'. In each of these cases a coupling is clearly present. However, due to physics and nonlinear effects one can not simply run with a fully coupled time discretization using very large timesteps. This would lead to incorrect results. On the other hand, it becomes very costly (and in many cases impossible due to constraints in computing time) to run in a strictly time-accurate manner. The recourse advocated here is to run each problem in a controlled way to a quasi steady-state, and to couple the different disciplines in a loose manner. This approach also offers the possibility of easily coupling different software packages (in-house, open source or commercial).

II. BARELY COUPLED TIMESTEPPING

The two most popular ways of marching in time when coupling different codes are the so-called loose[2, 5, 6, 13, 14, 23, 24, 31, 33] and tight[2, 7, 11, 29] coupling. Both are shown diagrammatically in Figure 2.1, where the term 'app' stands for application or physics/code.
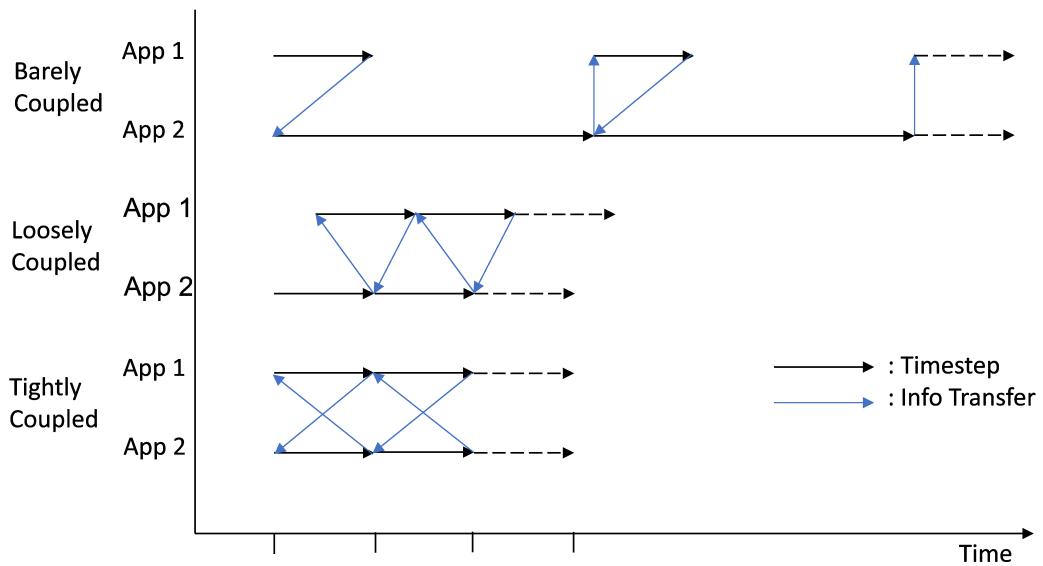

Figure 2.1: Advancement of Two Coupled Codes in Time

Loose coupling advances each code independently until the time of each code exceeds the time of the other code(s). Once an individual code reaches this time, the information is passed to the other code(s) and they advance in turn. Loose coupling has been used extensively for fluid-structure and fluid-structure-thermal interaction, particularly when explicit timestepping is required for one of the applications due to the physics.[13, 23, 24, 31]

Tight coupling, on the other hand, seeks to synchronize in a precise way the timesteps and the associated information transfer between the codes. Within each timestep, the codes are advanced several times until the iteration converges to an equilibrium state at the end of the timestep.[2, 7, 11, 29]

Both of these procedures are suboptimal for applications where the timescales at which the physically relevant processes occur are vastly different, yet a detailed, time-accurate simulation is required for any of the codes. There are numerous applications where this is the case: in fact, one could surmise that there are many more applications that fall under this category than those amenable to the previously mentioned forms of coupling. What all of these applications have in common is the establishment of a quasi-steady/quasi-periodic pattern

American Institute of Aeronautics and Astronautics

in one of the codes that affects the much slower evolution of the other code(s). Running all codes throughout the complete time required is neither efficient, nor (and more importantly): necessary (!). Which results in a new form of coupling: **barely coupled** multiphysics. The code with the 'fast transient' is run until a quasi-steady/quasi-periodic pattern is observed. This information (which may be transient data, i.e. not just a simple time-averaged field) is then passed to the 'slow transient' code(s). This code then advances in time until the geometry, the physics, or any other measure that would exceed an error tolerance is reached. The information at this time is then passed to the 'fast transient' code. The 'fast transient' code then continues from this point in time, i.e. this code **ignores all the intermediate time that has passed since the last update**. The sequence is shown at the top of Figure 2.1.

The implementation of barely coupled timestepping may be summarized as follows:

For Each Iteration/Timestep:

- Do: Loop Over the Codes Used

- Import All Relevant Information:

  Updated Geometry, Velocities, Temperatures,

  Forces, Heat Fluxes, ...

- Run Code to Completion/Quasi Steady-State

- Export All Relevant Information

  Forces, Heat Fluxes,

  Updated Geometry, Velocities, Temperatures, ...

- EndDo: Loop Over the Codes Used

The implicit assumption made here is that each individual code only imports and exports information from a coupling code or library that does all the required interpolation/projection operations.

The most important research question is how to detect the completion of each physics portion/code. Options that immediately come to mind include:

- Preset time increments (this could still imply many timesteps);

- Preset decrease of residuals (useful for steady state cases);

- Preset quasi-periodic behaviour (useful for cyclic loads);

- Preset energy balance / tolerance;

- Preset change in unknowns; and

- Preset change in geometry.

III.   EVAPORATIVE COOLING OF ELECTRICAL MOTORS

The modeling of evaporative cooling is a good example for barely coupled multiphysics: the high-fidelity modeling requires the solution of the Maxwell equations to obtain the heat sources, a mix of continuum and particle methods for the fluids (which reaches a quasi steady-state in less than a second), as well as a heat conduction solver for the rotor and stator (where the quasi-steady temperature field establishes itself in minutes). The problem has come to the forefront with the rise of electric propulsion in the aerospace sector (drones and air taxis in particular). Optimization of electrical motors (miniaturization, weight reduction, increase efficiencies) in some cases has led to overheating. Evaporative cooling is a possibility that has been actively pursued.[8] The PDEs and ODEs used to describe and compute the multiphase flow- and heat-fields have been summarized in Appendices 1,2.

American Institute of Aeronautics and Astronautics

In the sequel we consider two scenarios for electrical motors that are cooled evaporatively via particles that are injected in the narrow gap between the rotor and the stator. This is a typical barely coupled multiphysics case: the flowfield in the gap is established in less than a second, while the heating of the motor can take minutes.

3.1 <u>Simple Gap Region</u>

The geometry chosen, as well as the general boundary conditions, are shown in Figure 3.1a. The physical parameters were set as follows (all units in cgs):
Geometry of the section computed:

- Inner radius: 2.0

- Outer radius: 2.1

- Outermost radius: 4.0

- Depth(z): 2.0

Flow:

- Inflow: air: 100cm/sec, te=300K, dens=0.00122, p=1.0e6

- Rotation of inner part: 600 rpm

- Viscosity: 0.1850E-03

- Conductivity: 0.2400E+04

- Specific heat coefficient: 0.1000E+08

- Particles: water, d=0.01

Solid/Heat:

- Density: 7.85

- Specific heat coefficient: 420.0e+4

- Conductivity: k=50.0e+5

- Volumetric heat load: 2.0e+7

The discretizations used are shown in Figures 3.1b,c. Note that optimal discretizations have been employed for each field, and the grids do not match at the interface. Based on previous scoping runs, during each barely coupled step the flowfield was advanced for 500 timesteps with a timestep of approximately $\Delta t = 0.3 \cdot 10^{-4}$, which corresponds to an advective Courant-number of $C = 0.8$. This number of timesteps is required for the flowfield to reach a quasi steady state. The temperature field, on the other hand, was advanced for 10 timesteps with a timestep of $\Delta t = 10$. Figures 3.1d-f show the temperature field obtained, the velocity in the gap, the velocity of the flowfield and the particles, as well as the temperature of the flow and the particles. Note that evaporation of particles does occur. Figures 3.1g-i show the sum of the differences in temperature at the interface between the fluid and solid domains, the heat loads seen by each of the domains (CFD: computational fluid dynamics, i.e. the flow domain, CTD: computational thermo-dynamics, i.e. the solid region) in the gap region, and the minimum and maximum temperature seen on the gap surface by the two codes/domains. Note the 'transient beginning', indicative of the strong coupling of the problem at hand, and the subsequent 'settling to steady state'.

American Institute of Aeronautics and Astronautics
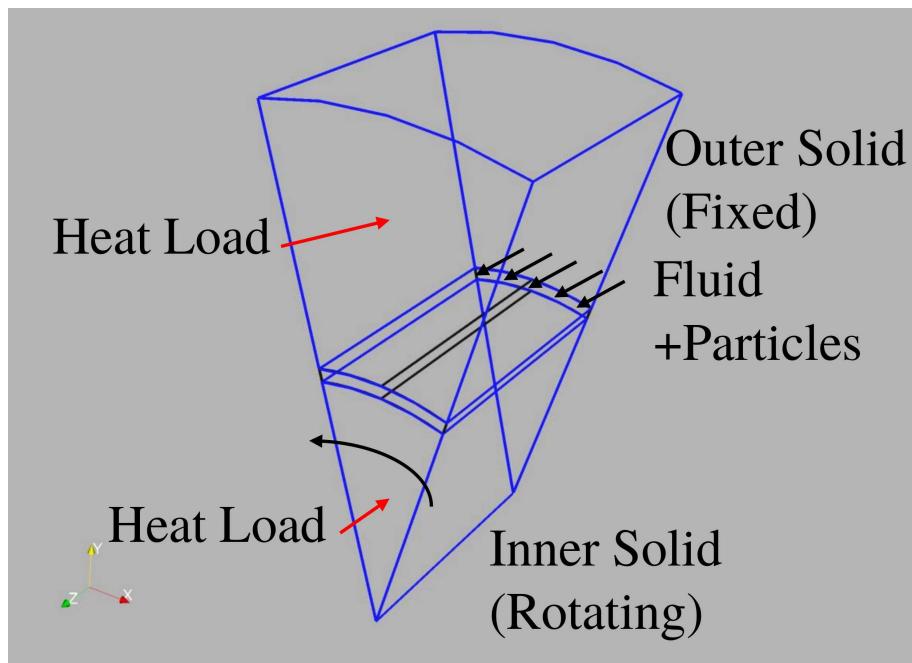
Figure 3.1a E-Motor: Problem Definition
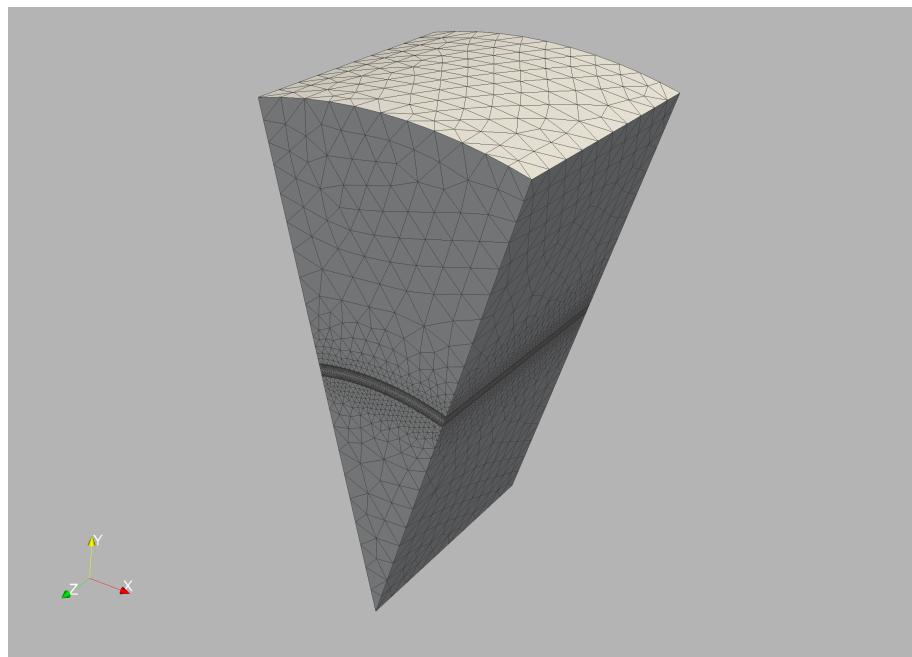
Figure 3.1b E-Motor: Grids Used
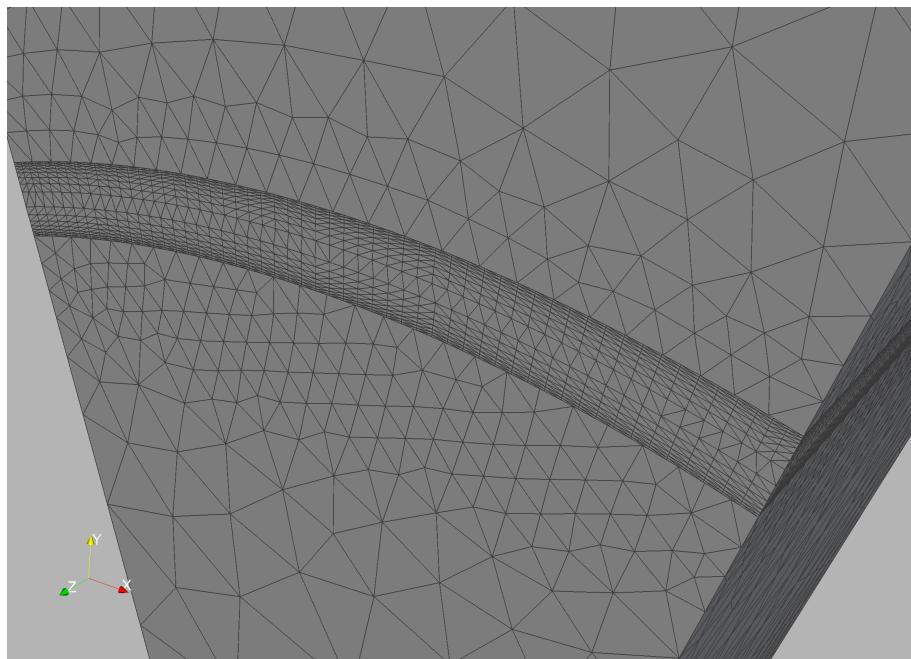
Figure 3.1c E-Motor: Grids Used (Detail)
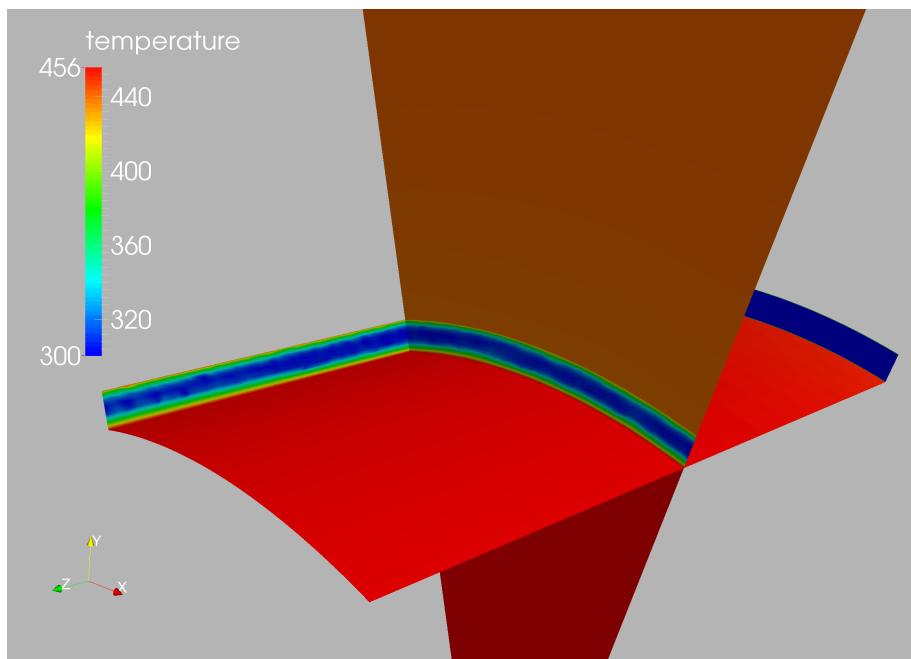
Figure 3.1d E-Motor: Temperature Field Obtained
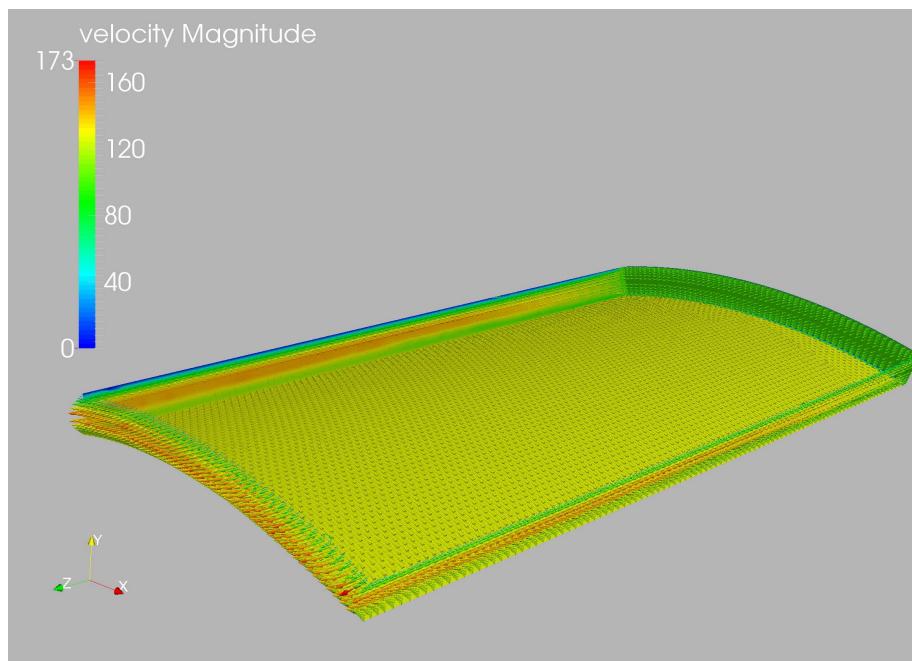
Figure 3.1e E-Motor: Velocity Field Obtained
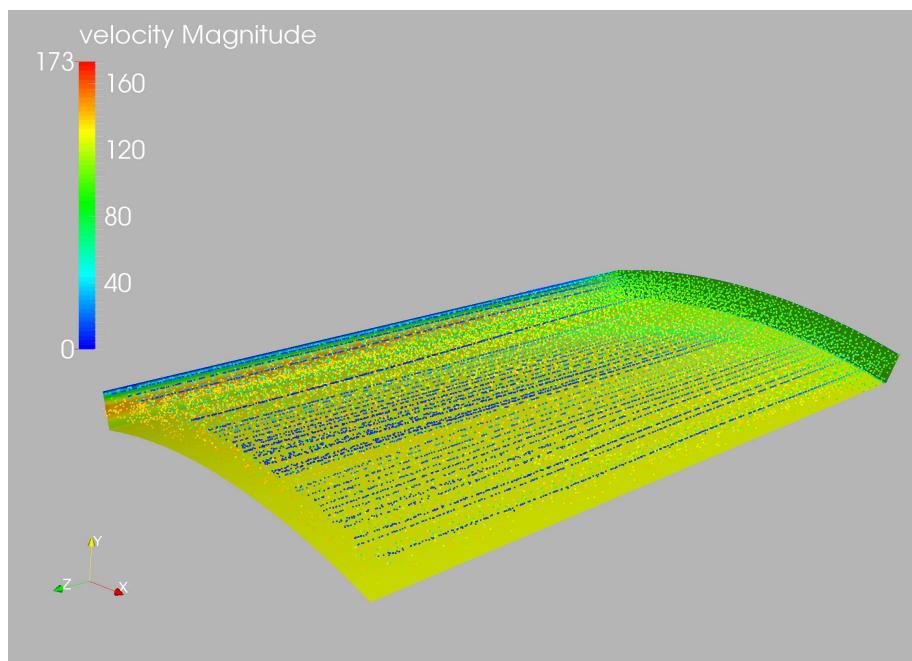
Figure 3.1f E-Motor: Velocity Field of Gas (Surface) and Particles (Volume)

American Institute of Aeronautics and Astronautics
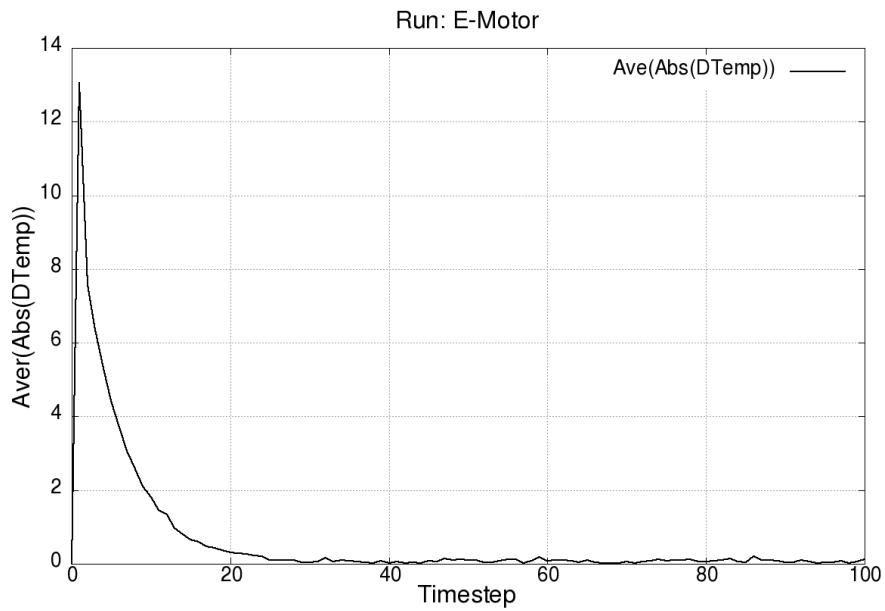
Figure 3.1g E-Motor: Difference in Surface Temperatures Between Flow and Heat Solvers at the Surface of the Gap
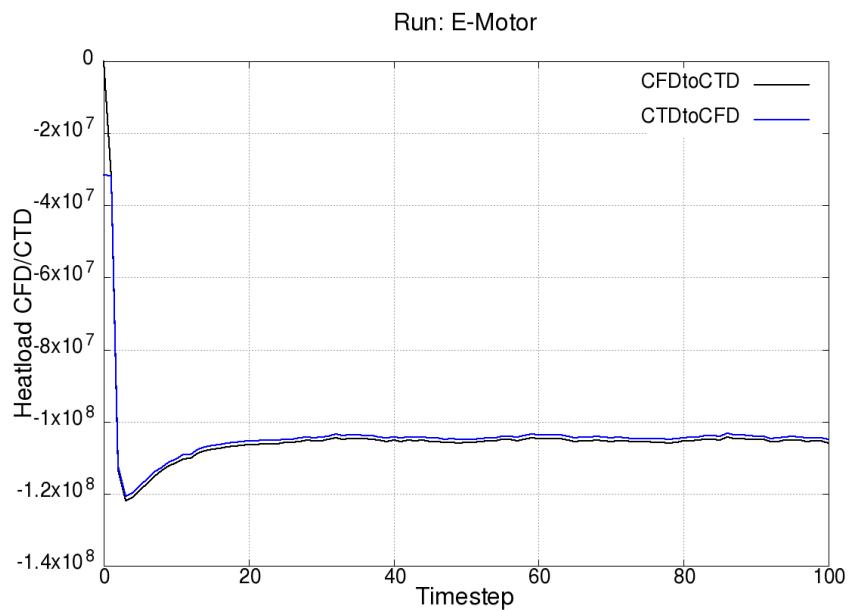
Figure 3.1h E-Motor: Heat Loads Seen by Flow and Heat Solvers at the Surface of the Gap

American Institute of Aeronautics and Astronautics
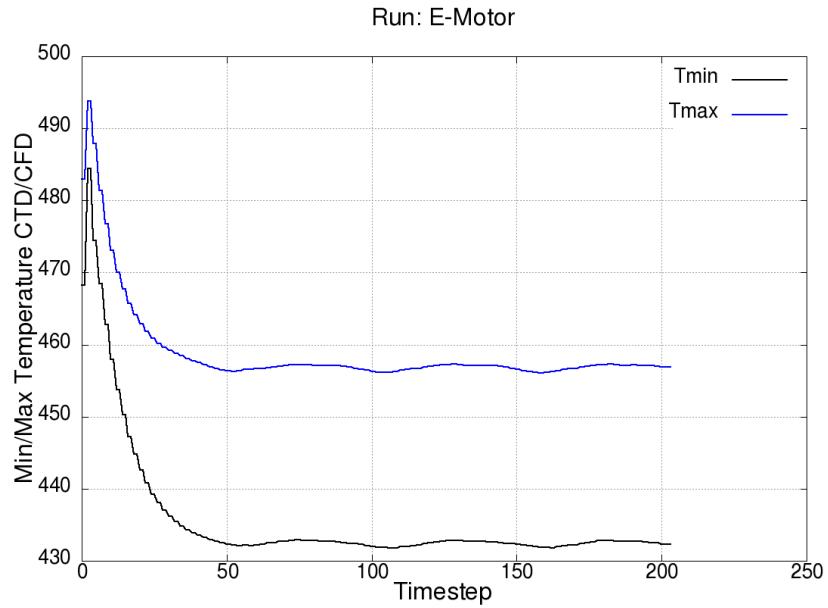
Figure 3.1g E-Motor: Minimum and Maximum Temperature at the Surface of the Gap

3.2 Complete Motor

The geometry chosen is shown in Figure 3.2a,b, and is taken from the repository than be accessed under.[9] Notice the arrangement of copper coils on both sides of the stator. In this case the particles are injected in the (axial) middle of the gap, then travel in the gap towards both ends, and are then injected with a centrifugal direction into the flow. The aim is for the particles to remove heat both in the gap and as they come into contact with the coils. The physical parameters were set as follows (all units in cgs):
Geometry of the section computed:

- Inner radius: 6.240

- Outer radius: 6.285

- Outermost radius: 10.0

- Length: 10.0

- Radius of Casing: 13.0

Flow:

- Inflow: air: 100cm/sec, te=300K, dens=0.00122, p=1.0e6

- Rotation of inner part: 600 rpm

- Viscosity: 0.1850E-03

- Conductivity: 0.2400E+04

- Specific heat coefficient: 0.1000E+08

- Particles: water, d=0.01

Solid/Heat: Magnets

- Density: 7.85

- Specific heat coefficient: 420.0e+4

American Institute of Aeronautics and Astronautics

- Conductivity: k=50.0e+5

Solid/Heat: Copper Coils in Stator and Outside Stator

- Density: 8.94

- Specific heat coefficient: 385.0e+4

- Conductivity: k=385.0e+5

Solid/Heat: Steel for Rotor, Shaft and Stator

- Density: 7.85

- Specific heat coefficient: 420.0e+4

- Conductivity: k=50.0e+5

Heat Loads:

- Volumetric Heat for Steel in Rotor: 2.0e+7

- Volumetric Heat for Steel in Stator: 1.0e+6

The discretizations used are shown in Figures 3.2c-e. The solid was discretized with about 4.3 Mtets, while the fluid field exceeded 100 Mtets. The boundary layers of the flowfield were properly discretized with elements of considerable stretching. As before, optimal discretizations have been employed for each field, and the grids do not match at the interface. Based on previous scoping runs, during each barely coupled step the flowfield was advanced for 1,500 timesteps. This number of timesteps was required for the flowfield to reach a quasi steady state. The temperature field, on the other hand, was advanced for 10 timesteps with a timestep of $\Delta t = 10$. Figures 3.2f-i show the temperature field obtained, the velocity in the gap, the velocity of the flowfield and the particles, as well as the temperature of the flow and the particles. The run was carried out in mixed OMP/MPI mode on a machine with 8 nodes of 128 cores each.
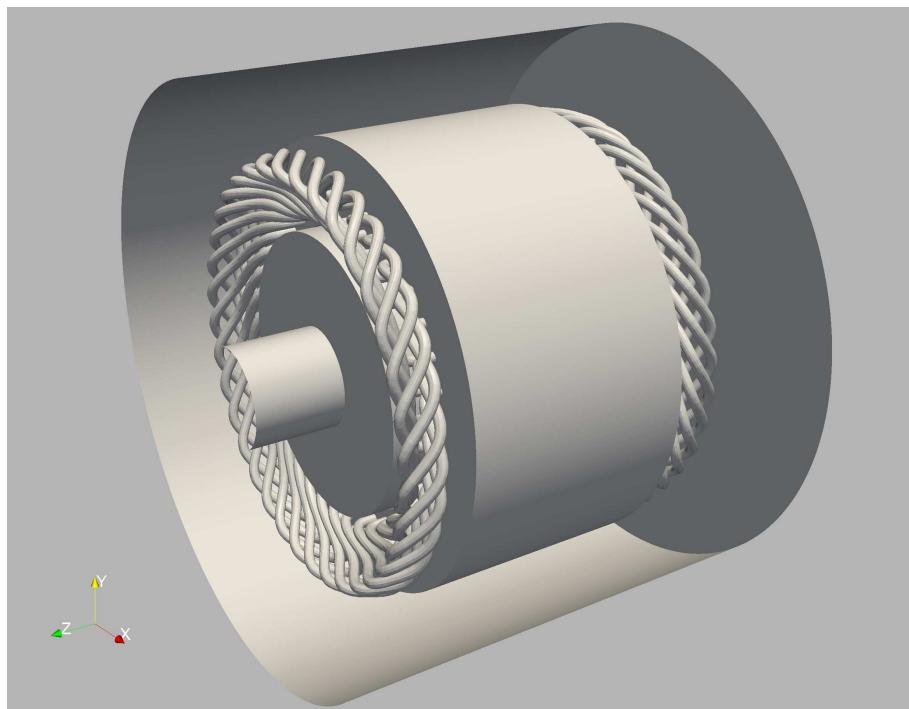
Figure 3.2a E-Motor: Motor in Casing

American Institute of Aeronautics and Astronautics
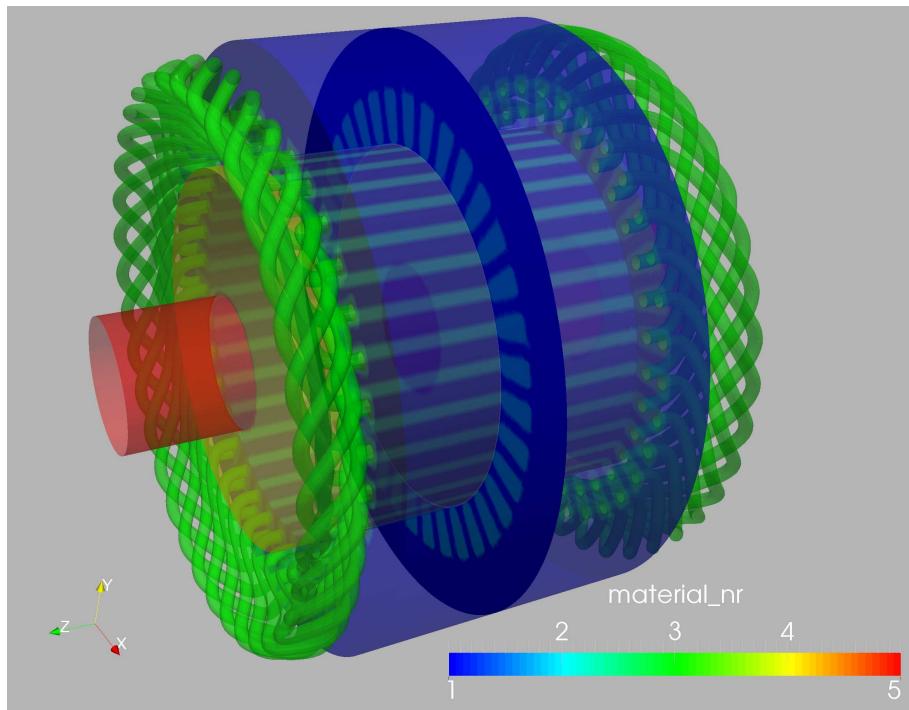
Figure 3.2b E-Motor: Materials Used
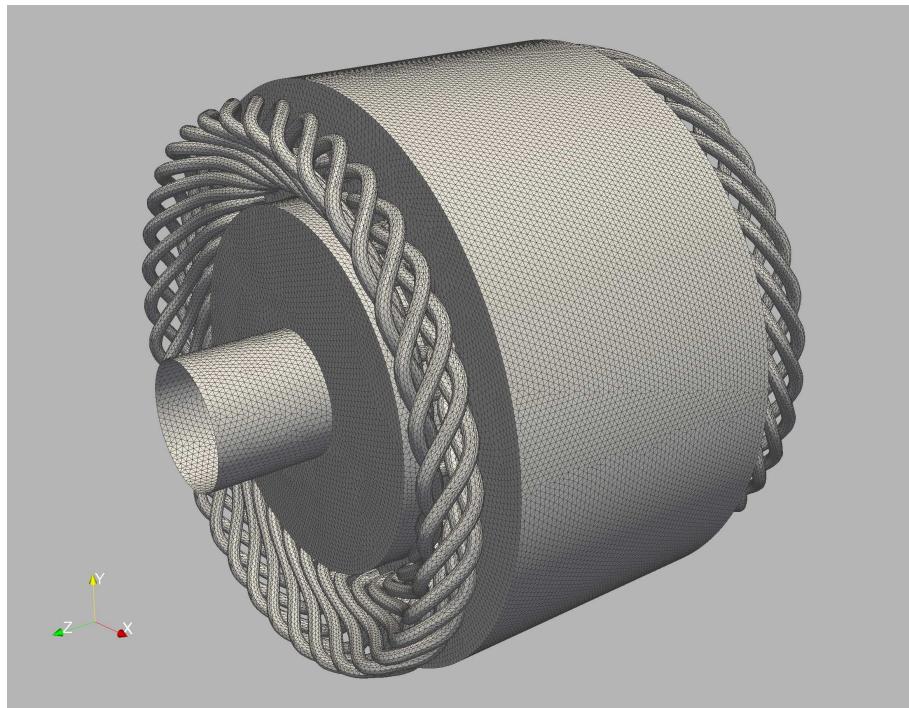
Figure 3.2c E-Motor: Surface Mesh of FEHEAT Region
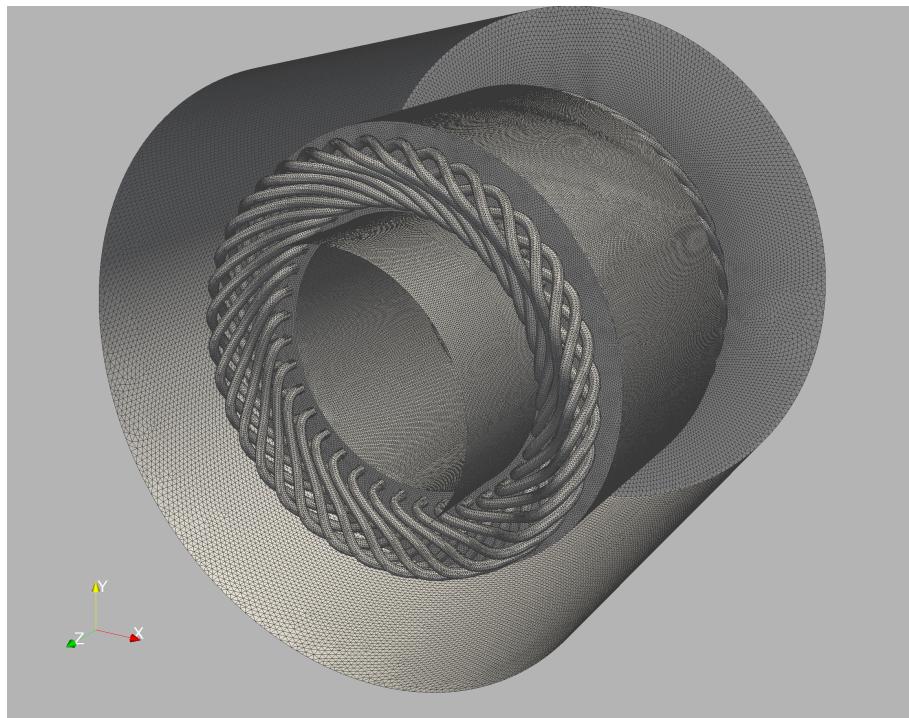
Figure 3.2d E-Motor: Surface Mesh of FEFLO Region
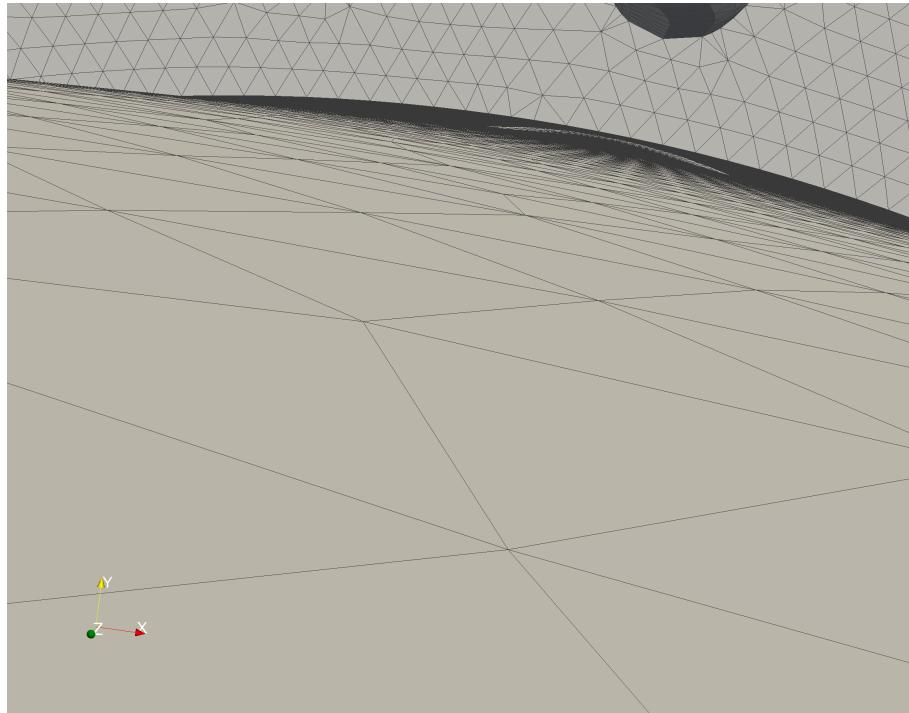
Figure 3.2e E-Motor: Gap Region (Detail)

American Institute of Aeronautics and Astronautics
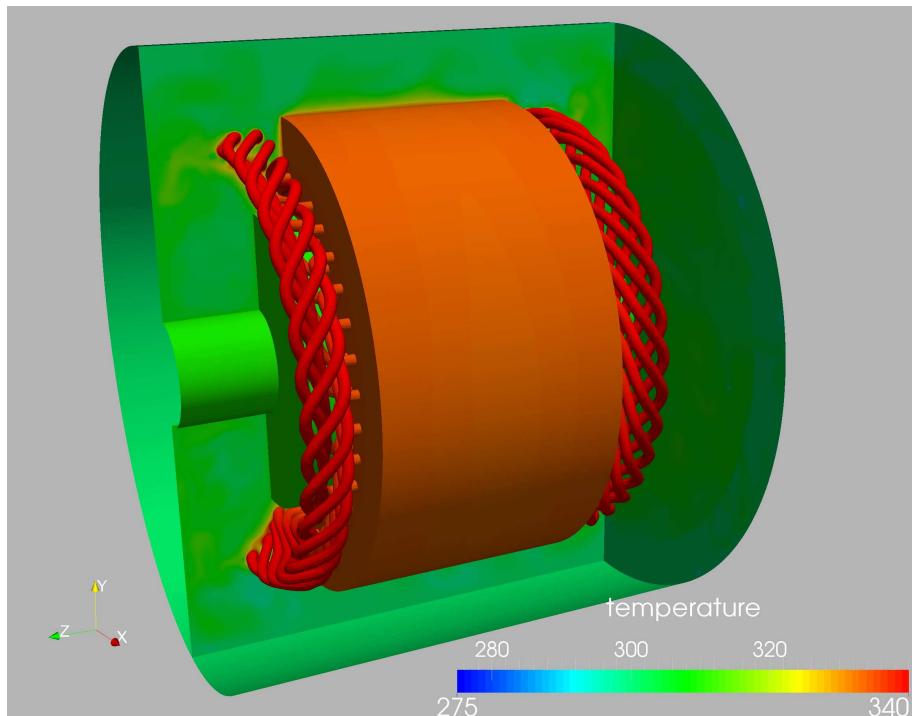
Figure 3.2f E-Motor: Temperature Field
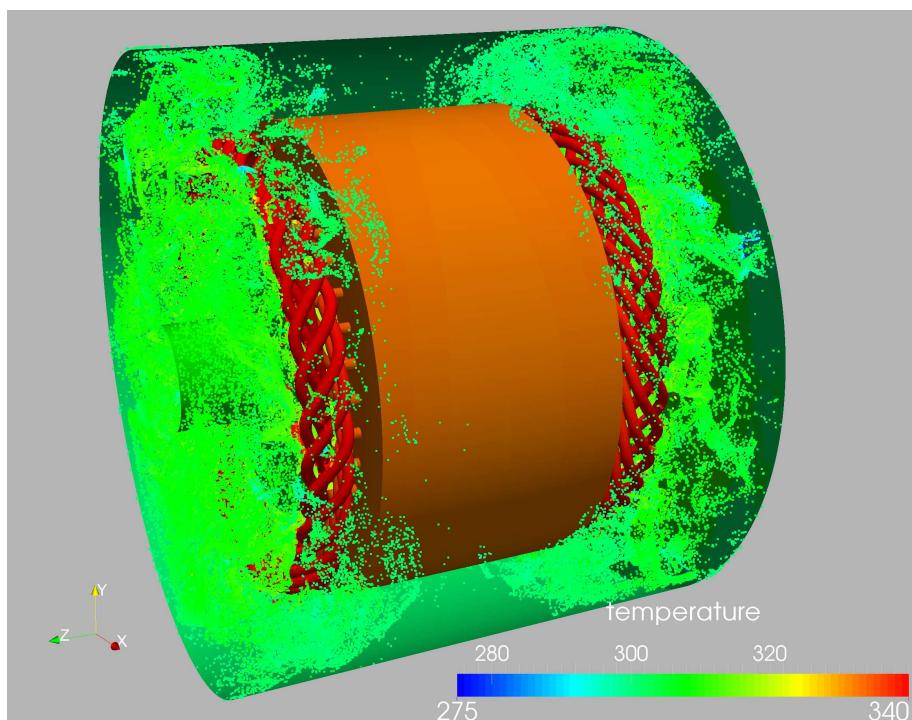
Figure 3.2g E-Motor: Temperature Field for Solid and Particles
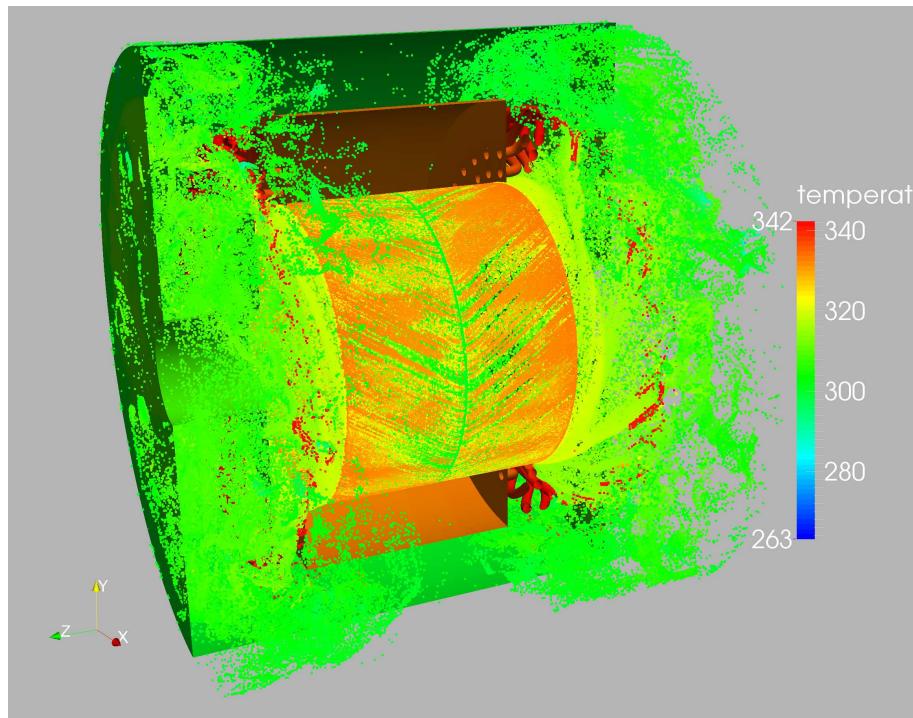
Figure 3.2h E-Motor: Temperature Field for Solid and Particles
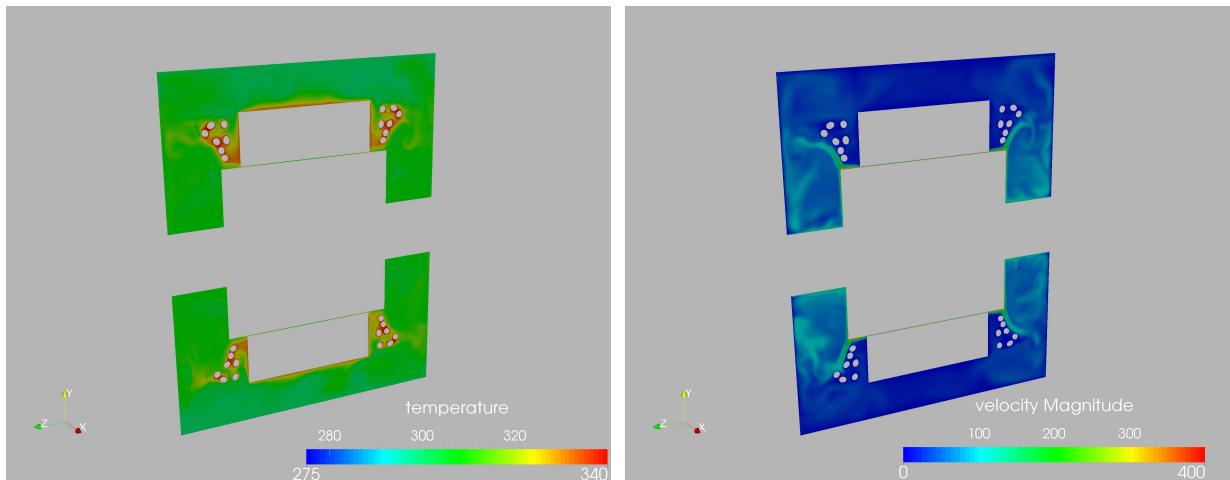
Figure 3.2i E-Motor: Temperature and Velocities in Plane $x = 0$

IV.   CONCLUSIONS AND OUTLOOK

A technique to combine codes to solve barely coupled multiphysics problems has been developed. Each field is advanced separately until a stop is triggered. This could be due to a preset time increment, a preset number of timesteps, a preset decrease of residuals, a preset change in unknowns, a preset change in geometry, or any other physically meaningful quantity. The technique allows for a simple implementation in coupled codes using the loose coupling approach.

Examples from evaporative cooling of electric motors shows the viability and accuracy of the proposed procedure.

Open questions include:

- The optimal increase ('ramping up') of time-increments/timesteps used in the individual discipline/metier codes;

- The automatic detection of when to stop/exit the individual discipline/metier codes and couple back;

- The implementation of barely coupled timestepping in optimization loops/packages; and

- The derivation of adjoints for this class of problems (different grids, different timescales, etc.).

V.   ACKNOWLEDGEMENTS

American Institute of Aeronautics and Astronautics

References

[1] R. Aubry, F. Mut, R. Löhner and J. R. Cebral - Deflated Preconditioned Conjugate Gradient Solvers for the Pressure-Poisson Equation; *J. Comp. Phys.* 227, 24, 10196-10208 (2008).

[2] H.-J. Bungartz and M. Schäfer (eds.) *Fluid-Structure Interaction: Modeling, Simulation, Optimization*; Springer Lecture Notes in Computational Science and Engineering, Springer (2006).

[3] F. Camelli and R. Löhner - Assessing Maximum Possible Damage for Contaminant Release Events; *Engineering Computations* 21, 7, 748-760 (2004).

[4] A. Corrigan, F. Camelli, R. Löhner and F. Mut - Semi-Automatic Porting of a Large-Scale Fortran CFD Code to GPUs; *Int. J. Num. Meth. Fluids* 69, 2, 314-331 (2012).

[5] J.R. Cebral and R. Löhner - Conservative Load Projection and Tracking for Fluid-Structure Problems; *AIAA J.* 35, 4, 687-692 (1997).

[6] J.R. Cebral and R. Löhner - On the Loose Coupling of Implicit Time-Marching Codes; *AIAA*-05-1093 (2005).

[7] M. Clemens, S. Schöps, C. Cimala, N. Gödel, S. Runke, D. Schmidthäusler and T. Timm - Aspects of Coupled Problems in Computational Electromagnetics Formulations; *ICS Newsletter (International Compumag Society)* 19, 2, 3-12, November (2012). https://arxiv.org/pdf/2303.08590

[8] P.-O. Gronwald and T.A. Kern - Traction Motor Cooling Systems, A Literature Review and Comparative Study; *IEEE Transactions on Transportation Electrification*, TTE-Reg-2020-12-1260.R2 doi: https://doi.org/10.1109/TTE.2021.3075844 (2021).

[9] K. Heidarikani - 'CREATOR Case: Induction Motor Data; Graz University of Technology (2024). Available at: https://doi.org/10.3217/kh3v7-dhn98.

[10] A. Jameson, W. Schmidt and E. Turkel - Numerical Solution of the Euler Equations by Finite Volume Methods using Runge-Kutta Time-Stepping Schemes; *AIAA*-81-1259 (1981).

[11] U. Küttler and W. Wall - Fixed-Point Fluid-Structure Interaction Solvers with Dynamic Relaxation; *Computational Mechanics* 43, 1, 6172 (2008).

[12] R. Löhner and J. McAnally - Transient and Steady Heat Conduction Using an Adaptive Finite Element CAD-Based Approach; *Int. J. Heat and Fluid Flow* 4, 311-327 (1994).

[13] R. Löhner, C. Yang, J.R. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA*-95-2259 [Invited] (1995).

[14] R. Löhner, C. Yang, J.R. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA*-98-2419 [Invited] (1998).

[15] R. Löhner - Multistage Explicit Advective Prediction for Projection-Type Incompressible Flow Solvers; *J. Comp. Phys.* 195, 143-152 (2004).

[16] R. Löhner, Chi Yang, J.R. Cebral, F. Camelli, O. Soto and J. Waltz - Improving the Speed and Accuracy of Projection-Type Incompressible Flow Solvers; *Comp. Meth. Appl. Mech. Eng.* 195, 23-24, 3087-3109 (2006).

[17] R. Löhner - *Applied CFD Techniques, Second Edition*; J. Wiley & Sons (2008).

[18] R. Löhner, F. Mut, J.R. Cebral, R. Aubry and G. Houzeaux; Deflated Preconditioned Conjugate Gradient Solvers for the Pressure-Poisson Equation: Extensions and Improvements; *Int. J. Num. Meth. Eng.* 87, 1-5, 2-14 (2011).

[19] R. Löhner, F. Camelli, J.D. Baum, F. Togashi and O. Soto - On Mesh-Particle Techniques; *Comp. Part. Mech.* 1, 199-209 (2014).

[20] R. Löhner and H. Antil - High Fidelity Modeling of Aerosol Pathogen Propagation in Built Environments With Moving Pedestrians; *Int. J. Num. Meth. Biomed. Engng.* 37, 3 (2021). https://doi.org/10.1002/cnm.3428

[21] R. Löhner - *FEFLO Users Manual*, GMU (2024).

[22] R. Löhner - *FEHEAT Users Manual*, GMU (2024).

[23] A. Michalski, P.D. Kermel, E. Haug, R. Löhner, R. Wüchner, K.-U. Bletzinger - Validation of the Computational FluidStructure Interaction Simulation at Real-Scale Tests of a Flexible 29 m Umbrella in Natural Wind Flow; *J. Wind Eng. Ind. Aerodyn.* 99, 400413 (2011).

[24] F. Perazzo, R. Löhner, F. Labbea, F. Knop, P. Mascaro - Numerical Modeling of the Pattern and Wear Rate on a Structural Steel Plate Using DEM; *Minerals Engineering* 137, 290-302 (2019). doi: https://doi.org/10.1016/j.mineng.2019.04.012

[25] R. Ramamurti and R. Löhner - Simulation of Flow Past Complex Geometries Using a Parallel Implicit Incompressible Flow Solver; pp. 1049,1050 in *Proc. 11th AIAA CFD Conf.*, Orlando, FL, July (1993).

[26] R. Ramamurti and R. Löhner - A Parallel Implicit Incompressible Flow Solver Using Unstructured Meshes; *Computers and Fluids* 5, 119-132 (1996).

[27] R. Ramamurti, W.C. Sandberg and R. Löhner - Computation of Unsteady Flow Past Deforming Geometries; *Int. J. Comp. Fluid Dyn.*, 83-99 (1999).

[28] H. Schlichting - *Boundary Layer Theory*; McGraw-Hill (1979).

[29] M. Schäfer and S. Turek (eds.) - *Proc. Int. Workshop on Fluid-Structure Interaction: Theory, Numerics and Applications*, Herrsching (Munich), Germany, Sept. 29 - Oct 1 (2008).

[30] D. Sharov, H. Luo, J.D. Baum and R. Löhner - Implementation of Untructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-Based Parallel Computers; *AIAA*-00-0927 (2000).

[31] O.A. Soto, J.D. Baum and R. Löhner - An Efficient Fluid-Solid Coupled Finite Element Scheme for Weapon Fragmentation Simulations; *Eng. Fract. Mech.* 77, 549-564 (2010).

[32] R. Tilch, A. Tabbal, M. Zhu, F. Decker and R. Löhner - Combination of Body-Fitted and Embedded Grids for External Vehicle Aerodynamics; *Engineering Computations* 25, 1, 28-41 (2008).

[33] Chi Yang, R. Löhner and H. Lu - An Unstructured-Grid Based Volume-of-Fluid Method For Extreme Wave and Freely-Floating Structure Interactions; *J. of Hydrodynamics* 18, 3, 1, 415-422 (2006).

American Institute of Aeronautics and Astronautics

VI.   Appendix 1: Physical Modeling of Droplet Propagation and Evaporation

When solving the two-phase equations, the air, as a continuum, is best represented by a set of partial differential equations (the Navier-Stokes equations) that are numerically solved on a mesh. Thus, the gas characteristics are calculated at the mesh points within the flowfield. The droplets/particles, are modeled using a Lagrangian description, where individual particles (or groups of particles) are monitored and tracked in the flow, allowing for an exchange of mass, momentum and energy between the air and the particles.

A.   Equations Describing the Motion of the Air

As seen from the experimental evidence, the velocities of the air in the gap region never exceed a Mach-number of $Ma = 0.1$. Therefore, the air may be assumed as a Newtonian, incompressible liquid. Given the narrow gap and the short timescales, buoyancy and the effect of gravity may be neglected. The equations describing the conservation of momentum, mass and energy for incompressible, Newtonian flows may be written as

$$\rho \mathbf{v}_{,t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p = \nabla \cdot \mu \nabla \mathbf{v} + \mathbf{s}_v \quad , \tag{A.1.1}$$

$$\nabla \cdot \mathbf{v} = 0 \quad , \tag{A.1.2}$$

$$\rho c_p T_{,t} + \rho c_p \mathbf{v} \cdot \nabla T = \nabla \cdot k \nabla T + s_e \quad . \tag{A.1.3}$$

Here $\rho, \mathbf{v}, p, \mu, T, c_p, k$ denote the density, velocity vector, pressure, viscosity, temperature, specific heat coefficient and conductivity respectively, and $\mathbf{s}_v, s_e$ momentum and energy source terms (e.g. due to particles or external forces/heat sources).

B.   Equations Describing the Motion of Particles/Droplets

In order to describe the interaction of particles/droplets with the flow, the mass, forces and energy/work exchanged between the flowfield and the particles must be defined. As before, we denote for **fluid (air)** by ρ, p, T, k, v_i, μ and c_p the density, pressure, temperature, conductivity, velocity in direction x_i, viscosity, and the specific heat at constant pressure. For the **particles**, we denote by $\rho_p, T_p, v_{pi}, d, c_{pp}$ and Q the density, temperature, velocity in direction x_i, equivalent diameter, specific heat coefficient and heat transferred per unit volume. In what follows we will refer to droplet and particles collectively as particles.

Making the classical assumptions that the particles may be represented by an equivalent sphere of diameter d, the drag forces $\mathbf{D}$ acting on the particles will be due to the difference of fluid and particle velocity:

$$\mathbf{D} = \frac{\pi d^2}{4} \cdot c_d \cdot \frac{1}{2} \rho |\mathbf{v} - \mathbf{v}_p| (\mathbf{v} - \mathbf{v}_p) \quad . \tag{A.2.1}$$

The **drag coefficient** c_d is obtained empirically from the Reynolds-number Re:

$$Re = \frac{\rho |\mathbf{v} - \mathbf{v}_p| d}{\mu} \tag{A.2.2}$$

as (see, e.g.[28]):

$$c_d = max \left(0.1, \frac{24}{Re} \left(1 + 0.15 Re^{0.687} \right) \right) \tag{A.2.3}$$

The lower bound of $c_d = 0.1$ is required to obtain the proper limit for the Euler equations, when $Re \to \infty$. The heat transferred between the particles and the fluid is given by

$$Q = \frac{\pi d^2}{4} \cdot \left[h_f \cdot (T - T_p) + \sigma^* \cdot (T^4 - T_p^4) \right] \quad , \tag{A.2.4}$$

where h_f is the film coefficient and σ^* the radiation coefficient. For the class of problems considered here, the particle temperature and kinetic energy are such that the radiation coefficient σ^* may be ignored. The film coefficient h_f is obtained from the Nusselt number Nu:

$$Nu = 2 + 0.459 Pr^{0.333} Re^{0.55} \quad , \tag{A.2.5}$$

American Institute of Aeronautics and Astronautics

where Pr is the Prandtl number of the gas

$$Pr = \frac{k}{\mu} \quad , \tag{A.2.6}$$

as

$$h_f = \frac{Nu \cdot k}{d} \quad . \tag{A.2.7}$$

Having established the forces and heat flux, the particle motion and temperature are obtained from Newton's law and the first law of thermodynamics. For the particle velocities, we have:

$$\rho_p \frac{\pi d^3}{6} \cdot \frac{d\mathbf{v}_p}{dt} = \mathbf{D} \quad . \tag{A.2.8}$$

This implies that:

$$\frac{d\mathbf{v}_p}{dt} = \frac{3\rho}{4\rho_p d} \cdot c_d |\mathbf{v} - \mathbf{v}_p|(\mathbf{v} - \mathbf{v}_p) = \alpha_v |\mathbf{v} - \mathbf{v}_p|(\mathbf{v} - \mathbf{v}_p) \quad , \tag{A.2.9}$$

where $\alpha_v = 3\rho c_d/(4\rho_p d)$. The particle positions are obtained from:

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p \quad . \tag{A.2.10}$$

The temperature change in a particle is given by:

$$\rho_p c_{pp} \frac{\pi d^3}{6} \cdot \frac{dT_p}{dt} = Q \quad , \tag{A.2.11}$$

which may be expressed as:

$$\frac{dT_p}{dt} = \frac{3k}{2c_{pp}\rho_p d^2} \cdot Nu \cdot (T - T_p) = \alpha_T (T - T_p) \quad , \tag{A.2.12}$$

with $\alpha_T = 3k/(2c_{pp}\rho_p d^2)$. Equations (A.2.9, A.2.10, A.2.12) may be formulated as a system of Ordinary Differential Equations (ODEs) of the form:

$$\frac{d\mathbf{u}_p}{dt} = \mathbf{r}(\mathbf{u}_p, \mathbf{x}, \mathbf{u}_f) \quad , \tag{A.2.13}$$

where $\mathbf{u}_p, \mathbf{x}, \mathbf{u}_f$ denote the particle unknowns, the position of the particle and the fluid unknowns at the position of the particle.

C. Numerical Integration of the Motion of the Air

The last six decades have seen a large number of schemes that may be used to solve numerically the incompressible Navier-Stokes equations given by Eqns.(A.1.1-A.1.3). In the present case, the following design criteria were implemented:

- Spatial discretization using **unstructured grids** (in order to allow for arbitrary geometries and adaptive refinement);

- Spatial approximation of unknowns with **simple linear finite elements** (in order to have a simple input/output and code structure);

- Edge-based data structures (for reduced access to memory and indirect addressing);

- Temporal approximation using **implicit integration of viscous terms and pressure** (the interesting scales are the ones associated with advection);

- Temporal approximation using **explicit, high-order integration of advective terms**;

- **Low-storage, iterative solvers** with **deflation-based preconditioning**[1,18] for the resulting systems of equations (in order to solve large 3-D problems); and

American Institute of Aeronautics and Astronautics

- Steady results that are **independent from the timestep** chosen (in order to have confidence in convergence studies).

The resulting discretization in time is given by the following projection scheme:[15, 16]

- <u>Advective-Diffusive Prediction</u>: $\mathbf{v}^n, p^n \rightarrow \mathbf{v}^*$

$$\mathbf{s}' = -\nabla p^n + \rho\mathbf{g} + \beta\rho\mathbf{g}(T^n - T_0) + \mathbf{s}_v \quad , \qquad (A.3.1)$$

$$\mathbf{v}^i = \mathbf{v}^n + \alpha^i\gamma\Delta t \left(-\mathbf{v}^{i-1} \cdot \nabla\mathbf{v}^{i-1} + \nabla \cdot \mu\nabla\mathbf{v}^{i-1} + \mathbf{s}'\right) \quad ; \quad i = 1, k-1 \quad ; \qquad (A.3.2)$$

$$\left[\frac{1}{\Delta t} - \theta\nabla \cdot \mu\nabla\right]\left(\mathbf{v}^k - \mathbf{v}^n\right) + \mathbf{v}^{k-1} \cdot \nabla\mathbf{v}^{k-1} = \nabla \cdot \mu\nabla\mathbf{v}^{k-1} + \mathbf{s}' \quad . \qquad (A.3.3)$$

- <u>Pressure Correction</u>: $p^n \rightarrow p^{n+1}$

$$\nabla \cdot \mathbf{v}^{n+1} = 0 \quad ; \qquad (A.3.4)$$

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t} + \nabla(p^{n+1} - p^n) = 0 \quad ; \qquad (A.3.5)$$

which results in

$$\nabla^2(p^{n+1} - p^n) = \frac{\nabla \cdot \mathbf{v}^*}{\Delta t} \quad ; \qquad (A.3.6)$$

- <u>Velocity Correction</u>: $\mathbf{v}^* \rightarrow \mathbf{v}^{n+1}$

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \Delta t\nabla(p^{n+1} - p^n) \quad . \qquad (A.3.7)$$

θ denotes the implicitness-factor for the viscous terms ($\theta = 1$: 1st order, fully implicit, $\theta = 0.5$: 2nd order, Crank-Nicholson). α^i are the standard low-storage Runge-Kutta coefficients $\alpha^i = 1/(k + 1 - i)$. The $k - 1$ stages of Eqn.(A.3.2) may be seen as a predictor (or replacement) of $\mathbf{v}^n$ by $\mathbf{v}^{k-1}$. The original right-hand side has not been modified, so that at steady-state $\mathbf{v}^n = \mathbf{v}^{k-1}$, preserving the requirement that the steady-state be independent of the timestep Δt. The factor γ denotes the local ratio of the stability limit for explicit timestepping for the viscous terms versus the timestep chosen. Given that the advective and viscous timestep limits are proportional to:

$$\Delta t_a \approx \frac{h}{|\mathbf{v}|} \quad ; \quad \Delta t_v \approx \frac{\rho h^2}{\mu} \quad , \qquad (A.3.8)$$

we immediately obtain

$$\gamma = \frac{\Delta t_v}{\Delta t_a} \approx \frac{\rho|\mathbf{v}|h}{\mu} \approx Re_h \quad , \qquad (A.3.9)$$

or, in its final form:

$$\gamma = min(1, Re_h) \quad . \qquad (A.3.10)$$

In regions away from boundary layers, this factor is $O(1)$, implying that a high-order Runge-Kutta scheme is recovered. Conversely, for regions where $Re_h = O(0)$, the scheme reverts back to the usual 1-stage Crank-Nicholson scheme. Besides higher accuracy, an important benefit of explicit multistage advection schemes is the larger timestep one can employ. The increase in allowable timestep is roughly proportional to the number of stages used (and has been exploited extensively for compressible flow simulations[10]). Given that for an incompressible solver of the projection type given by Eqns.(A.3.1-A.3.7) most of the CPU time is

American Institute of Aeronautics and Astronautics

spent solving the pressure-Poisson system Eqn.(A.3.6), the speedup achieved is also roughly proportional to the number of stages used.

At steady state, $\mathbf{v}^* = \mathbf{v}^n = \mathbf{v}^{n+1}$ and the residuals of the pressure correction vanish, implying that the result does not depend on the timestep Δt.

The spatial discretization of these equations is carried out via linear finite elements. The resulting matrix system is re-written as an edge-based solver, allowing the use of consistent numerical fluxes to stabilize the advection and divergence operators.[17]

The energy (temperature) equation (Eqn.(A.3.3)) is integrated in a manner similar to the advective-diffusive prediction (Eqn.(A.3.2)), i.e. with an explicit, high order Runge-Kutta scheme for the advective parts and an implicit, 2nd order Crank-Nicholson scheme for the conductivity.

D. Numerical Integration of the Motion of Particles/Droplets

The equations describing the position, velocity and temperature of a particle (Eqns. A.2.9, A.2.10, A.2.12) may be formulated as a system of nonlinear Ordinary Differential Equations of the form:

$$\frac{d\mathbf{u}_p}{dt} = \mathbf{r}(\mathbf{u}_p, \mathbf{x}, \mathbf{u}_f) \quad . \tag{A.4.1}$$

They can be integrated numerically in a variety of ways. Due to its speed, low memory requirements and simplicity, we have chosen the following k-step low-storage Runge-Kutta procedure to integrate them:

$$\mathbf{u}_p^{n+i} = \mathbf{u}_p^n + \alpha^i \Delta t \cdot \mathbf{r}(\mathbf{u}_p^{n+i-1}, \mathbf{x}^{n+i-1}, \mathbf{u}_f^{n+i-1}) \quad , \quad i = 1, k \quad . \tag{A.4.2}$$

For linear ODEs the choice

$$\alpha^i = \frac{1}{k + 1 - i} \quad , \quad i = 1, k \tag{A.4.3}$$

leads to a scheme that is k-th order accurate in time. Note that in each step the location of the particle with respect to the fluid mesh needs to be updated in order to obtain the proper values for the fluid unknowns. The default number of stages used is $k = 4$. This would seem unnecessarily high, given that the flow solver is of second-order accuracy, and that the particles are integrated separately from the flow solver before the next (flow) timestep, i.e. in a staggered manner. However, it was found that the 4-stage particle integration preserves very well the motion in vortical structures and leads to less 'wall sliding' close to the boundaries of the domain.[19] The stability/ accuracy of the particle integrator should not be a problem as the particle motion will always be slower than the maximum wave speed of the fluid (fluid velocity).

The transfer of forces and heat flux between the fluid and the particles must be accomplished in a conservative way, i.e. whatever is added to the fluid must be subtracted from the particles and vice-versa. The finite element discretization of the fluid equations will lead to a system of ODE's of the form:

$$\mathbf{M}\Delta\mathbf{u} = \mathbf{r} \quad , \tag{A.4.4}$$

where $\mathbf{M}$, $\Delta\mathbf{u}$ and $\mathbf{r}$ denote, respectively, the consistent mass matrix, increment of the unknowns vector and right-hand side vector. Given the 'host element' of each particle, i.e. the fluid mesh element that contains the particle, the forces and heat transferred to $\mathbf{r}$ are added as follows:

$$\mathbf{r}_D^i = \sum_{el\ surr\ i} N^i(\mathbf{x}_p)\mathbf{D}_p \quad . \tag{A.4.5}$$

Here $N^i(\mathbf{x}_p)$ denotes the shape-function values of the host element for the point coordinates $\mathbf{x}_p$, and the sum extends over all elements that surround node i. As the sum of all shape-function values is unity at every point:

$$\sum N^i(\mathbf{x}) = 1 \quad \forall \mathbf{x} \quad , \tag{A.4.6}$$

this procedure is strictly conservative.

From Eqns.(A.2.9, A.2.10, A.2.12) and their equivalent numerical integration via Eqn.(A.4.2), the change in momentum and energy for one particle is given by:

American Institute of Aeronautics and Astronautics

$$\mathbf{f}_p = \rho_p \frac{\pi d^3}{6} \frac{\left(\mathbf{v}_p^{n+1} - \mathbf{v}_p^n\right)}{\Delta t} \quad , \tag{A.4.7}$$

$$q_p = \rho_p c_{pp} \frac{\pi d^3}{6} \frac{\left(T_p^{n+1} - T_p^n\right)}{\Delta t} \quad . \tag{A.4.8}$$

These quantities are multiplied by the number of particles in a packet in order to obtain the final values transmitted to the fluid. Before going on, we summarize the basic steps required in order to update the particles one timestep:

- Initialize Fluid Source-Terms: $\mathbf{r} = 0$

- `DO`: For Each Particle:

  - `DO`: For Each Runge-Kutta Stage:

    - Find Host Element of Particle: `IELEM`, $N^i(\mathbf{x})$

    - Obtain Fluid Variables Required

    - Update Particle: Velocities, Position, Temperature, ...

  - - `ENDDO`

  - Transfer Loads to Element Nodes

- `ENDDO`

1. Particle Parcels

For a large number of very small particles, it becomes impossible to carry every individual particle in a simulation. The solution is to:

  a) Agglomerate the particles into so-called packets of N_p particles;

  b) Integrate the governing equations for one individual particle; and

  c) Transfer back to the fluid N_p times the effect of one particle.

Beyond a reasonable number of particles per element (typically > 8), this procedure produces accurate results without any deterioration in physical fidelity.[19]

2. Other Particle Numerics

In order to achieve a robust particle integrator, a number of additional precautions and algorithms need to be implemented. The most important of these are:

- Agglomeration/Subdivision of Particle Parcels: As the fluid mesh may be adaptively refined and coarsened in time, or the particle traverses elements of different sizes, it may be important to adapt the parcel concentrations as well. This is necessary to ensure that there is sufficient parcel representation in each element and yet, that there are not too many parcels as to constitute an inefficient use of CPU and memory.

- Limiting During Particle Updates: As the particles are integrated independently from the flow solver, it is not difficult to envision situations where for the extreme cases of very light or very heavy particles physically meaningless or unstable results may be obtained. In order to prevent this, the changes in particle velocities and temperatures are limited in order not to exceed the differences in velocities and temperature between the particles and the fluid.[19]

- Particle Contact/Merging: In some situations, particles may collide or merge in a certain region of space.

- Particle Tracking: A common feature of all particle-grid applications is that the particles do not move far between timesteps. This makes physical sense: if a particle jumped ten gridpoints during one timestep, it would have no chance to exchange information with the points along the way, leading to serious errors. Therefore, the assumption that the new host elements of the particles are in the vicinity of the current ones is a valid one. For this reason, the most efficient way to search for the new host elements is via the vectorized neighbour-to-neighbour algorithm described in.[17]

E.   FEFLO

The design criteria and numerical techniques described above were implemented in FEFLO,[21] a general-purpose computational fluid dynamics (CFD) code. The code has had a long history of relevant applications in incompressible flows[15, 16, 25, 27, 32], free-surface hydrodynamics[33] and dispersion[3], and has been ported to vector, shared memory[30], distributed memory[25, 26] and GPU-based[4] machines.

American Institute of Aeronautics and Astronautics

VII.   Appendix 2: Modeling of Heat Propagation in the Motor

A.   Equation Describing the Temperature in the Solid

The equation describing the conservation energy for solids may be written as

$$\rho c_p T_{,t} = \nabla \cdot k \nabla T + s_m \quad . \tag{B.1.3}$$

Here ρ, T, c_p, k denote the density, temperature, specific heat coefficient and conductivity respectively, and $\mathbf{s}_m$ the source terms (e.g. due to magnetic hysteresis).

B.   Numerical Integration of the Heat Equation

For the numerical integration of the heat equation the following design criteria were implemented:

- Spatial discretization using **unstructured grids** (in order to allow for arbitrary geometries and adaptive refinement);

- Spatial approximation of unknowns with **simple linear finite elements** (in order to have a simple input/output and code structure);

- Edge-based data structures (for reduced access to memory and indirect addressing);

- Temporal approximation using **implicit integration**;

- **Low-storage, iterative solvers** with **deflation-based preconditioning**[1, 18] for the resulting systems of equations (in order to solve large 3-D problems).

The resulting discretization in time is given by the usual Θ scheme:

$$\left[\frac{\rho c_p}{\Delta t} - \Theta \nabla \cdot k \nabla \right] \Delta T = \nabla \cdot k \nabla T + \mathbf{s} \quad . \tag{B.2.1}$$

C.   FEHEAT

The design criteria and numerical techniques described above were implemented in FEHEAT,[22] a general-purpose computational thermo dynamics (CTD) code. The code has had a long history of relevant applications in heat conduction and aerothermodynamics[12, 14], and has been ported to vector, shared memory, distributed memory and GPU-based machines.

American Institute of Aeronautics and Astronautics