

EDRIVE

Wave-To-Wire Modelling Update

Richard Crozier
Institute for Energy Systems, The University of Edinburgh



THE UNIVERSITY of EDINBURGH
School of Engineering

Institute for Energy
Systems



Universidad Nacional
Autónoma de México



Contents

1. Introduction
2. Electrical machine simulation tools
3. Hydrodynamic simulation tools
4. Multi-body dynamics simulation tools
4. Multi-rate simulation
5. Putting it all together



A collection of tools for the design and simulation of renewable energy systems.

Mostly Matlab based (but also works with Octave) does not need simulink (but some simulink tools are available)

A suite of tools:

- Electrical machine simulation

- Wave energy related hydrodynamic simulation

- Multi-body dynamics

- Structural calculation tools

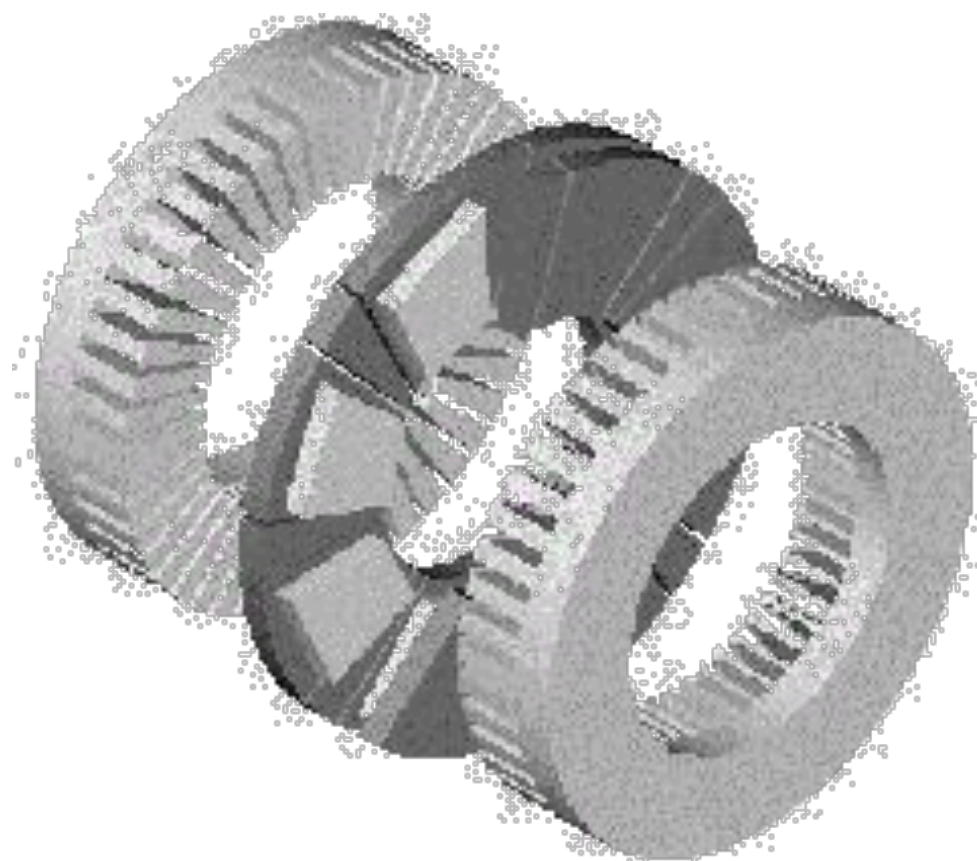
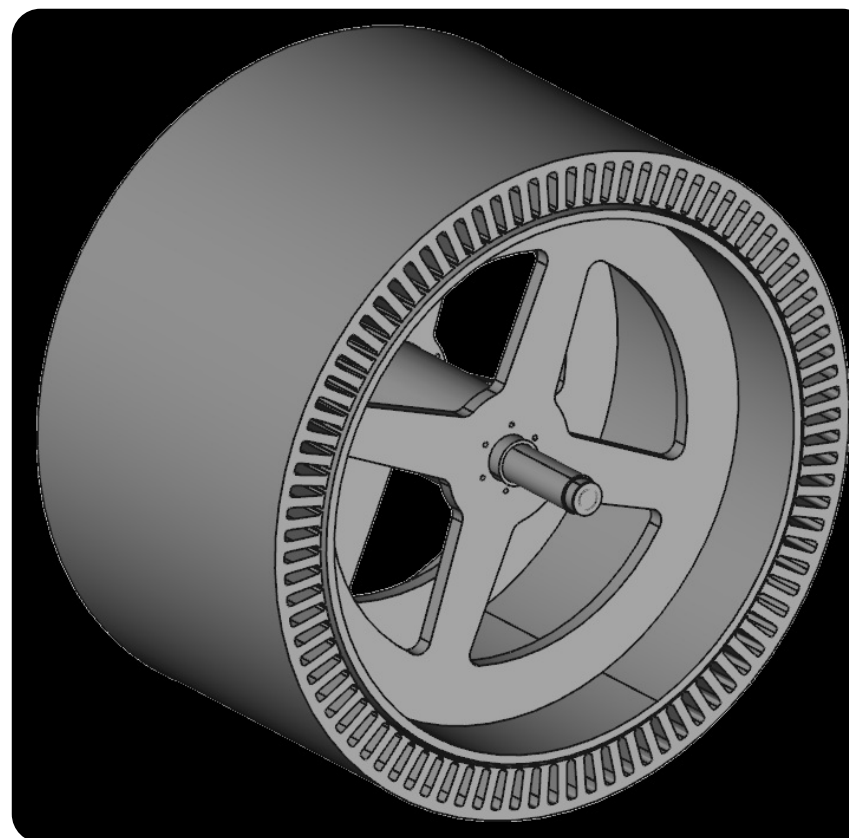
- Optimisation framework (supporting distributed computation, without Matlab's parallel computing toolbox)

Code based, no fancy gui tools really



Repository of code for the simulation and evaluation of a range of electrical machine types. Currently all complete models are synchronous PM machines.

Rotary Machines



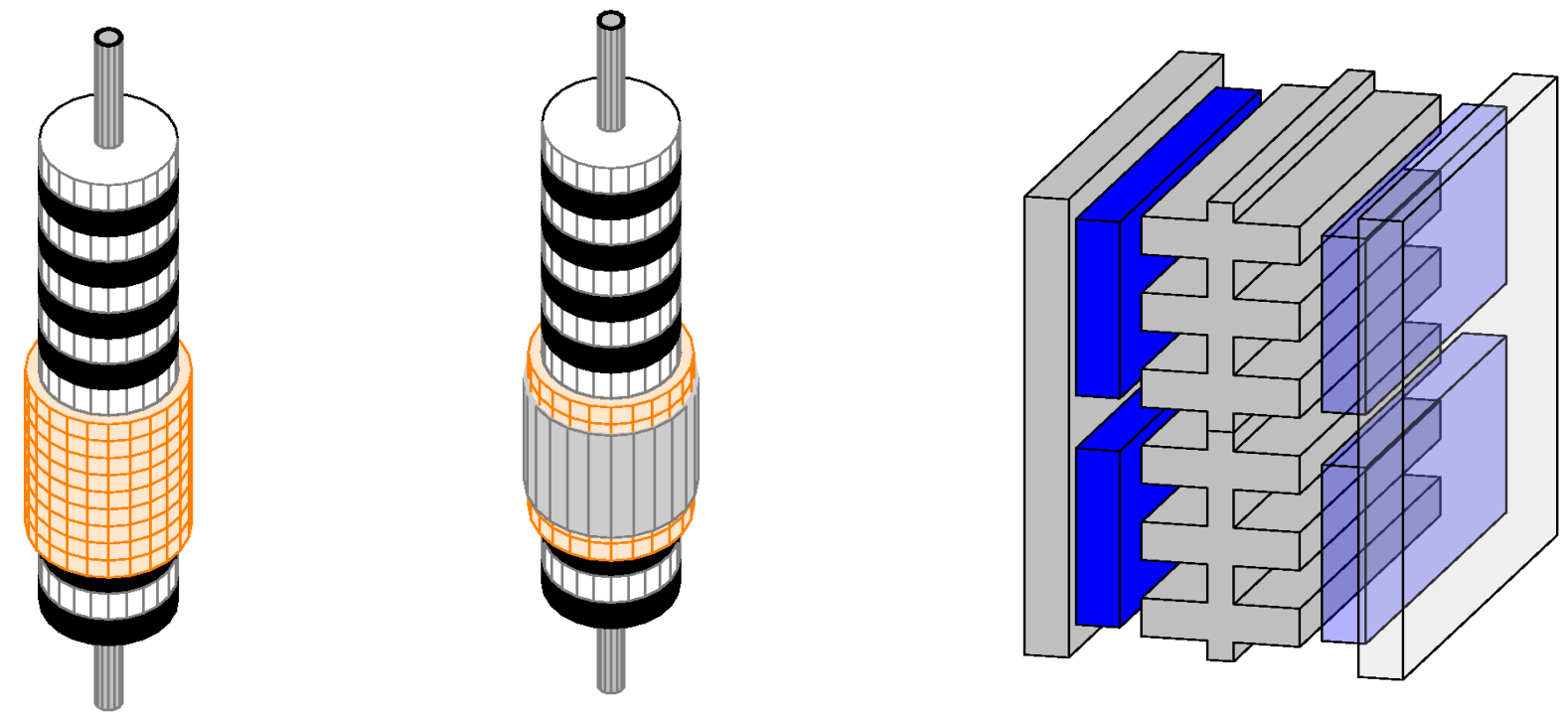
Radial Flux

- Slotted iron cored
- Slotless air gap
- Air Cored

Axial Flux

- TORUS (slotless)
- Slotted Iron cored
- Multi-stage stacked

Linear Machines

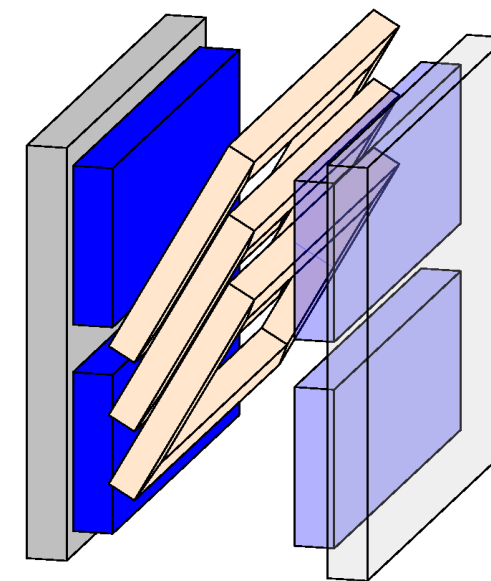


Tubular

- Slotted iron cored
- Slotless air gap
- Air Cored

Flat

- Slotted iron cored
- Slotless air gap



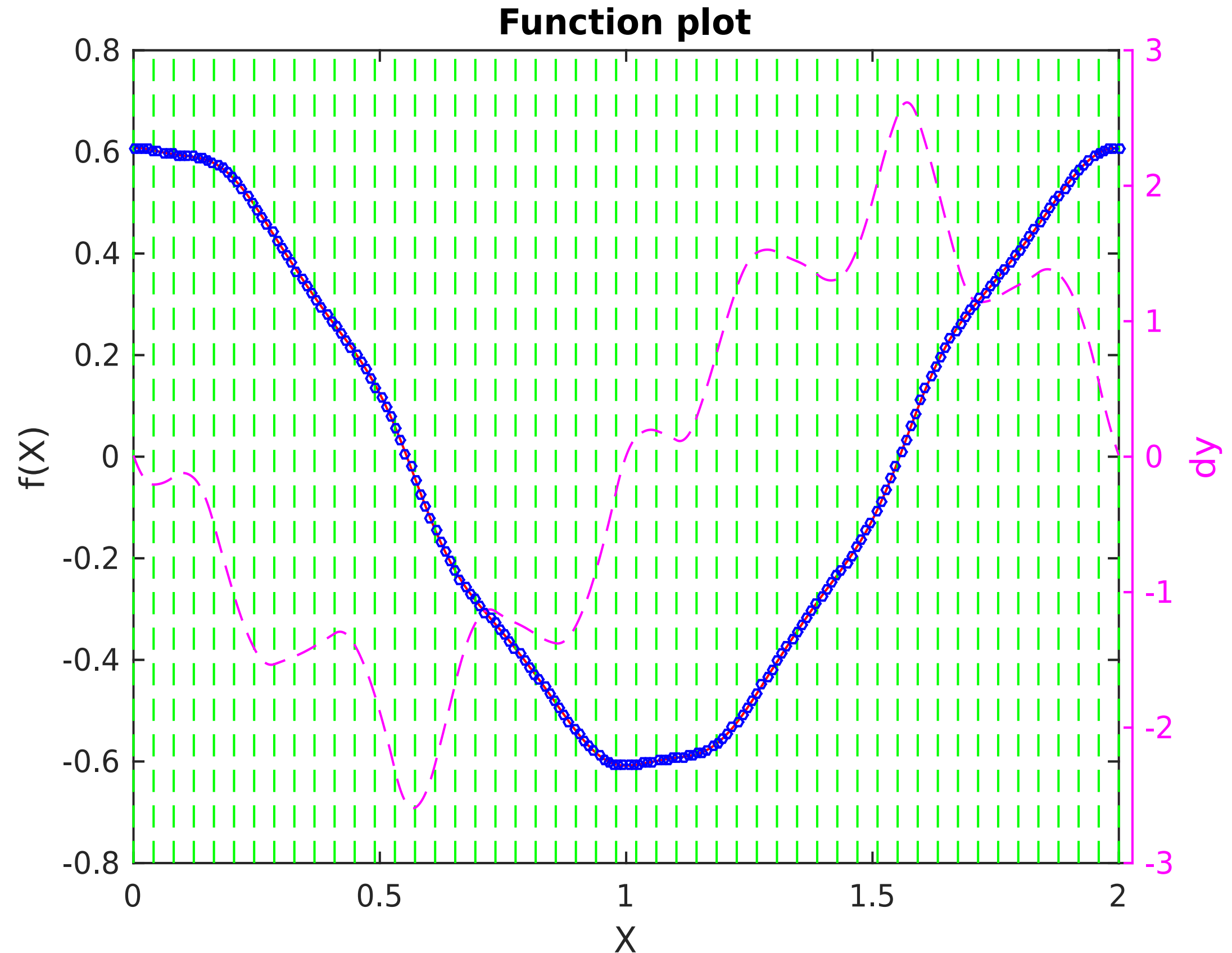


A series of FEA simulations are performed

Using the FEA data, other quantities are calculated, e.g:

- Mean turn length in coil
- Coil resistance
- Phase resistance
- Coil inductance
- Phase inductance
- Mutual inductance
- Total harmonic distortion of EMF
- Peak flux density in armature teeth
- Material volumes, copper, iron etc.
- Cogging forces/torques

Differs for different machine types





Dynamic Transient Sim

Using the precalculated data, a dynamic sim can then be performed. Some predefined simulation types are available

You can also use the models as part of a larger, system simulation, as we have done for the WEC models

The results of a dynamic simulation are post processed to give results like:

- Power loss in windings

- Iron losses

- Power exported to load

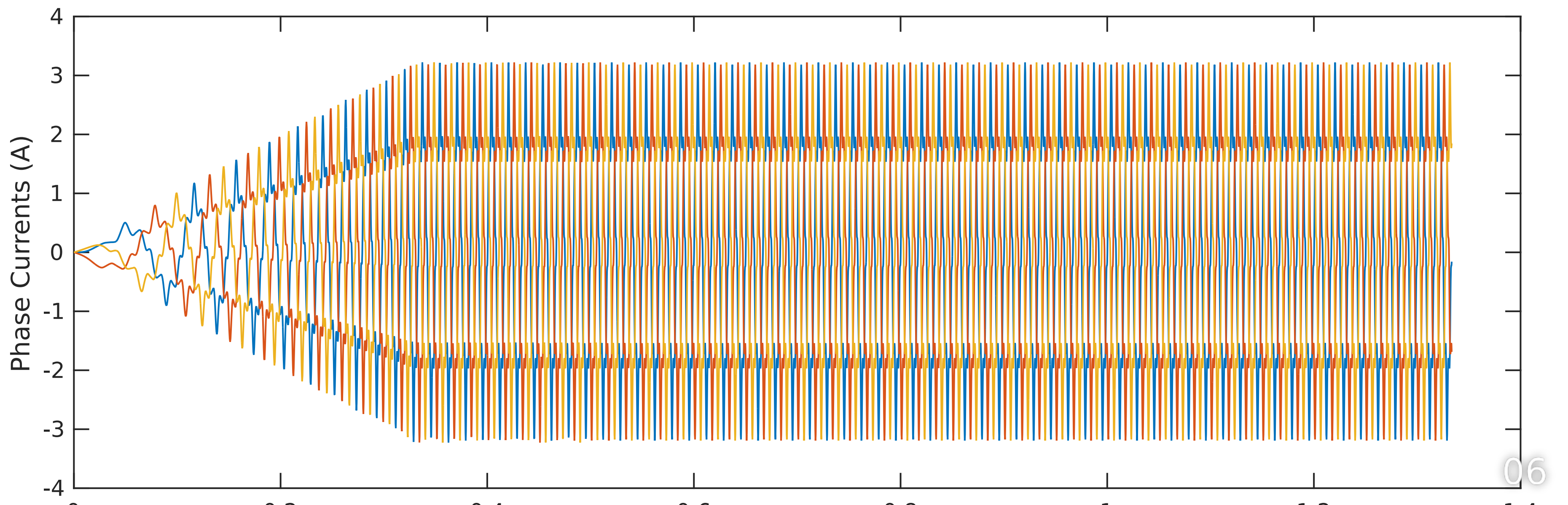
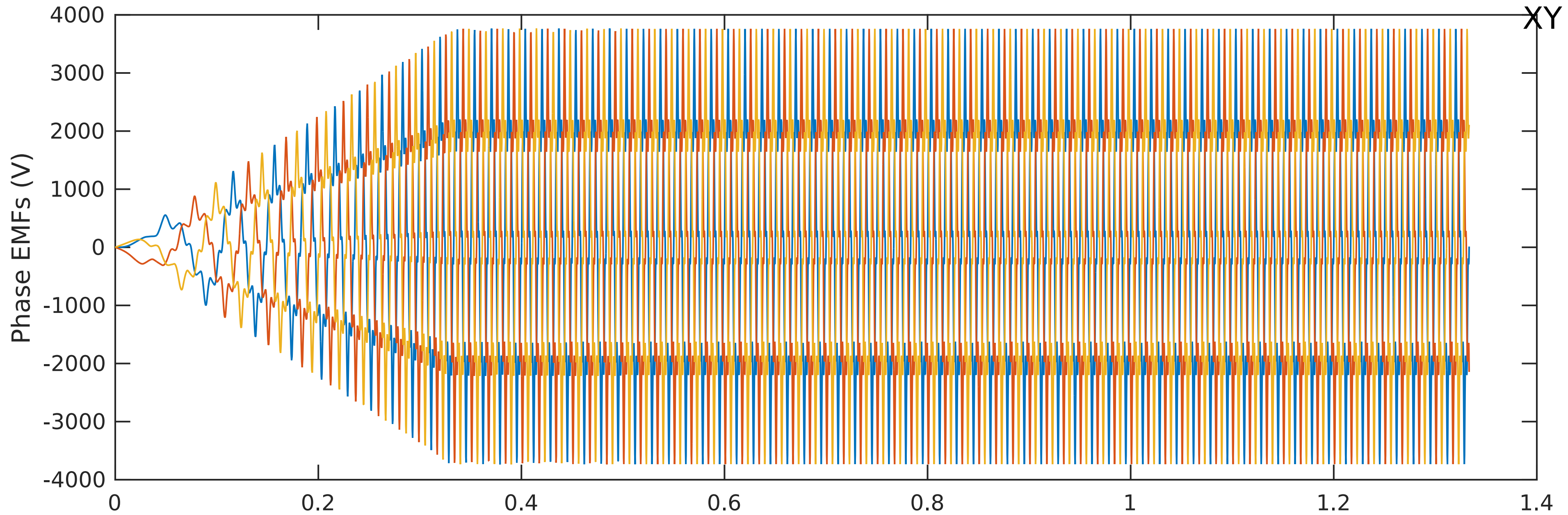
- RMS EMF over whole simulation

- RMS Current over whole simulation

- Peak EMF and Current

- Efficiency

- Peak electrical frequency

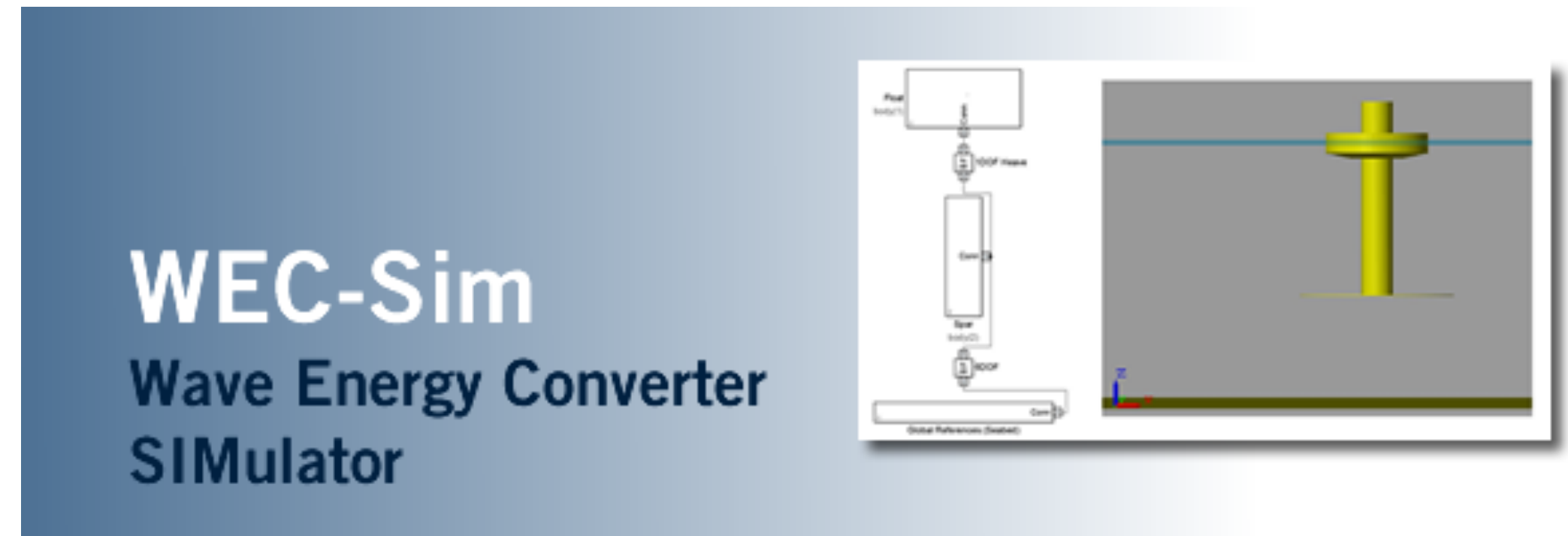


XY

06

Hydrodynamic Modelling

Two Levels



Developed by Steve McDonald

Simplified model to use as part of a power converter rough sizing tool

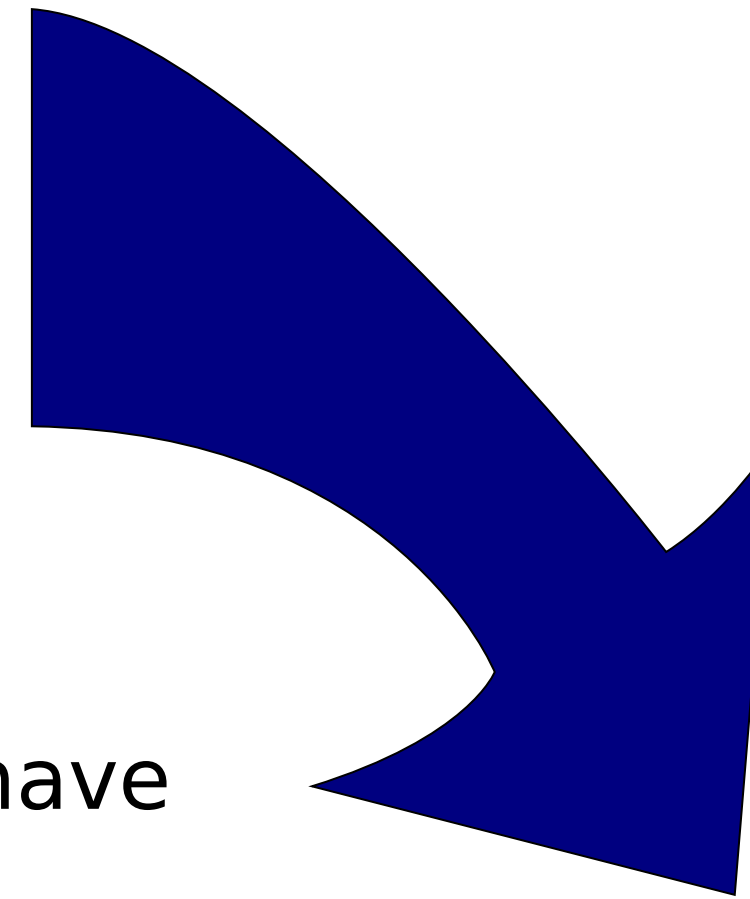
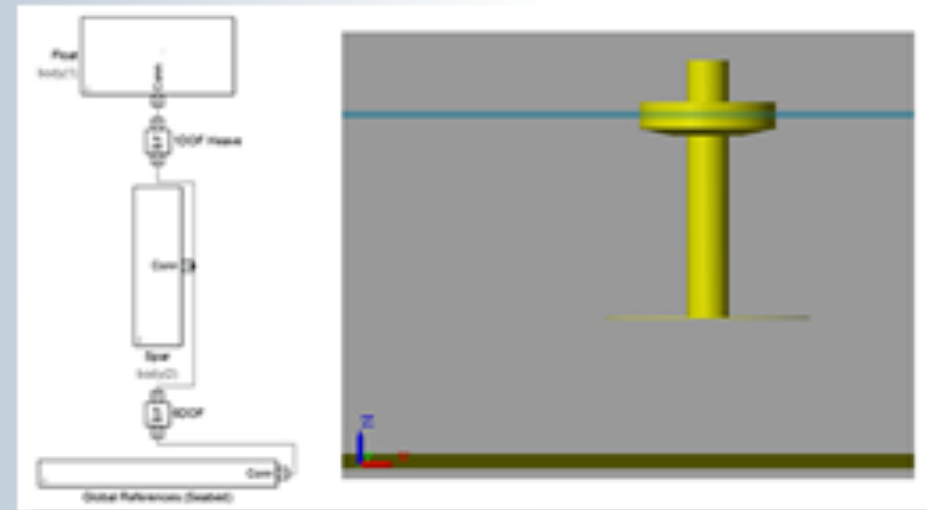
Will be discussed in more detail later

Advanced Model Derived from WEC-Sim, an open-source wave energy converter (WEC) simulation tool.

Developed by the US National Renewable Energy Laboratory (NREL) and Sandia National Laboratories (Sandia)

WEC-Sim

Wave Energy Converter
SIMulator



Developed by Experts and tested by users who have provided bug fixes

Lots of levels of fidelity and simulation method choices

Requires Simulink and SimMechanics (now called Simscape) \$\$\$!

Unusual interface/method of setting up models

Difficult to maintain and extend

Replaced Simscape multibody modeling with another free code library (MBDyn)

Created version purely in Matlab/Octave code, so more maintainable, portable and extendable

Multibody dynamics is the simulation of motion of the bodies in the WEC system, constrained by joints and linkages



Free MultiBody Dynamics Simulation

MBDyn is free and open source simulation engine written in C++.

- Focussed on scientific accuracy (rather than speed)
- Huge range of joint and other element types including hydraulics etc.
- No graphical interface, systems are described using special input files

We have

- Created a matlab wrapper/interface for the library, so Matlab can communicate with MBDyn during a simulation
- Created a matlab-based preprocessor (to generate input files), post-processing and visualisation tools

Double pendulum example: about 20 lines of code to set up 2 body, 2 joint system

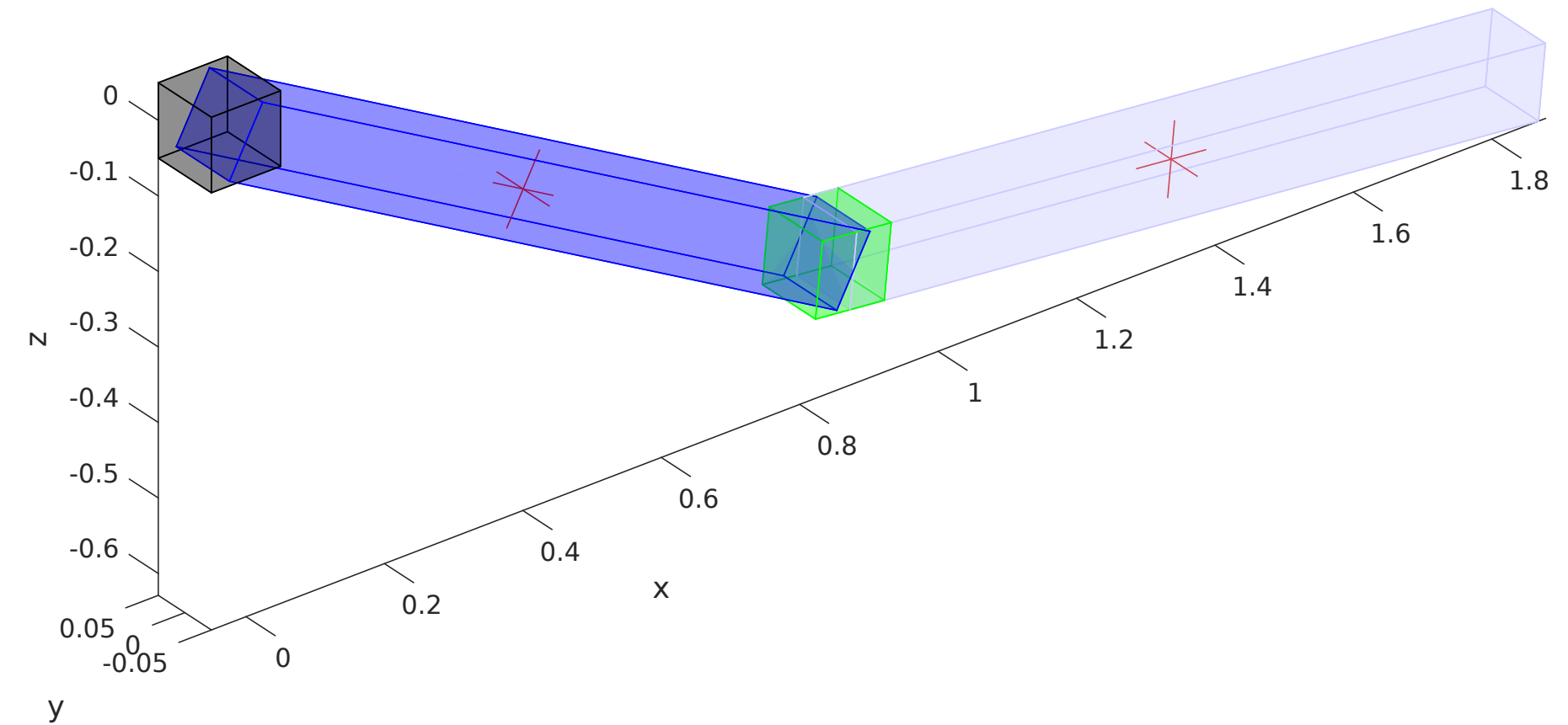
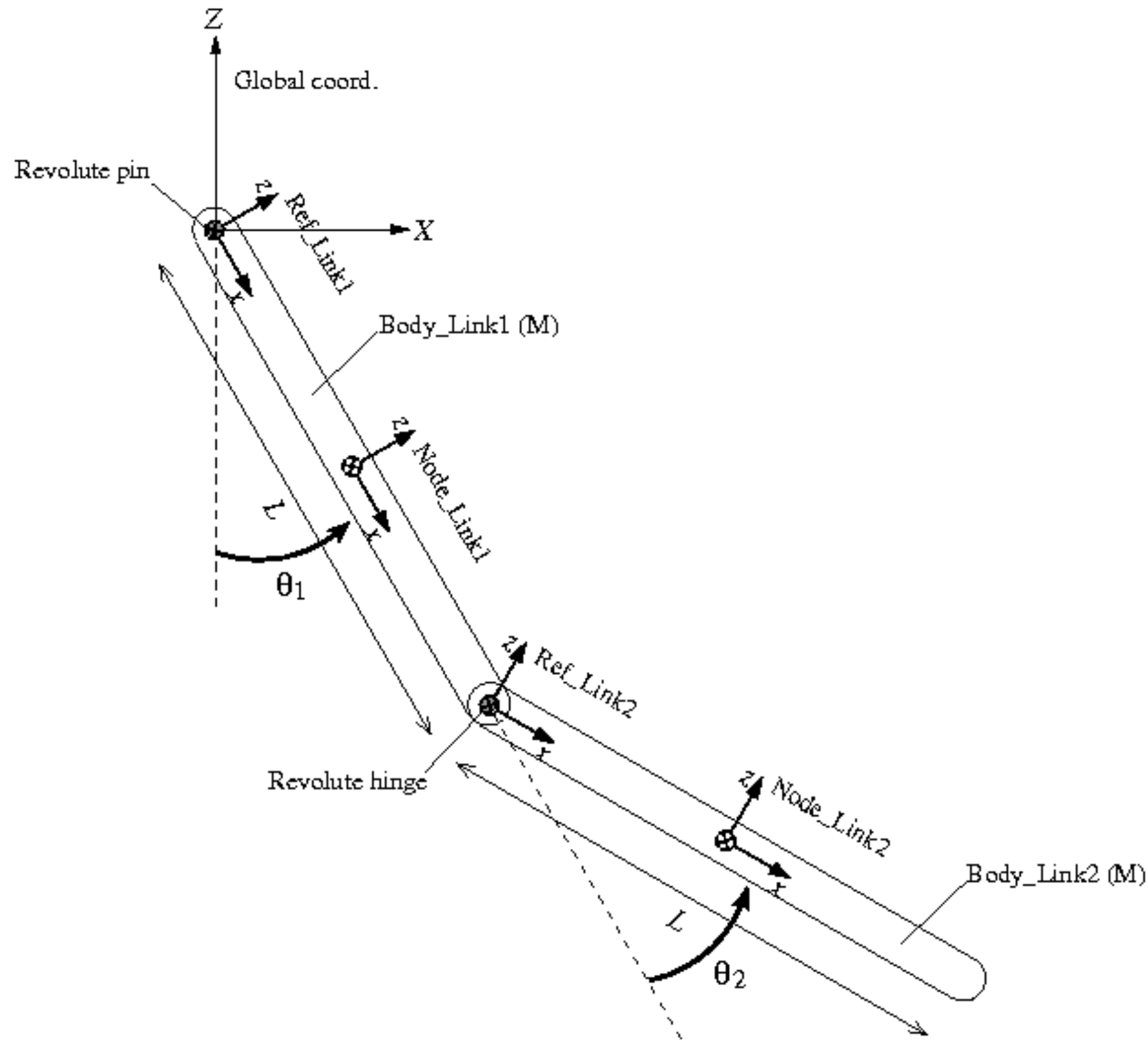


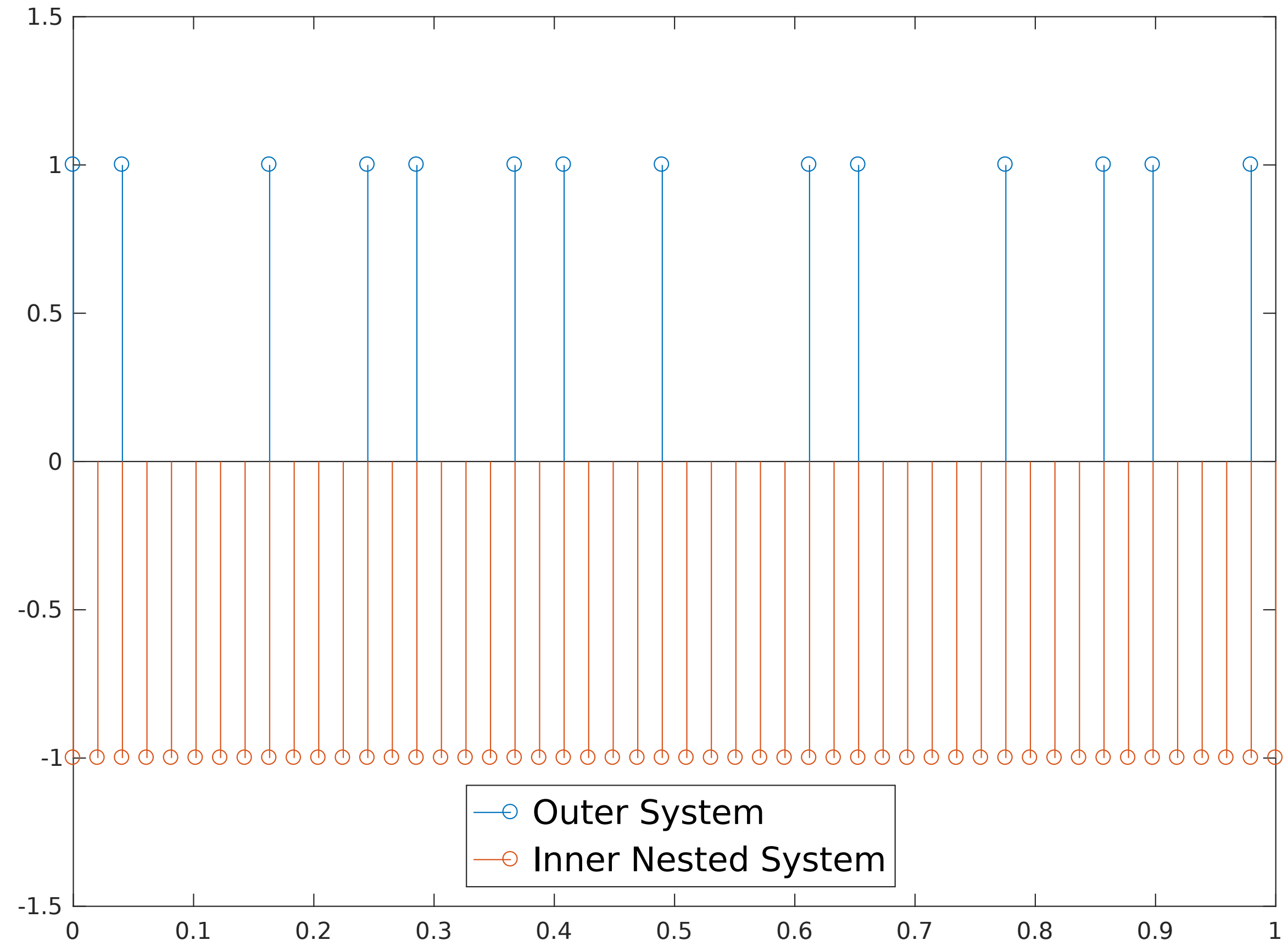
Figure above shows Matlab visualisation of the system using default shapes. Default shapes can be replaced with more complex ones, including STL files.

<https://youtu.be/DdoBRinQrEk>

Wave energy system time constant is very different to generator time constant.

Wave energy system is quite loosely coupled to generator dynamics (via forces, position and velocity)

We can retain a high level of fidelity in the electrical system simulation by simulating multiple electrical system time steps between each larger system simulation time step.



Start with initial values of outer and nested system inputs

advance time step

Inner and outer solutions based on standard ODE solver algorithms (e.g. Matlab's ode45)

advance time step

Generate Interpolation function for nested solver

$$y = mx + c$$

spline model another possible alternative, after a few time steps

Solve outer system and either advance or try different step

Solve nested system and pass final values to outer sim

Other advantages:

Can choose appropriate ODE solution algorithm for different simulation components

e.g

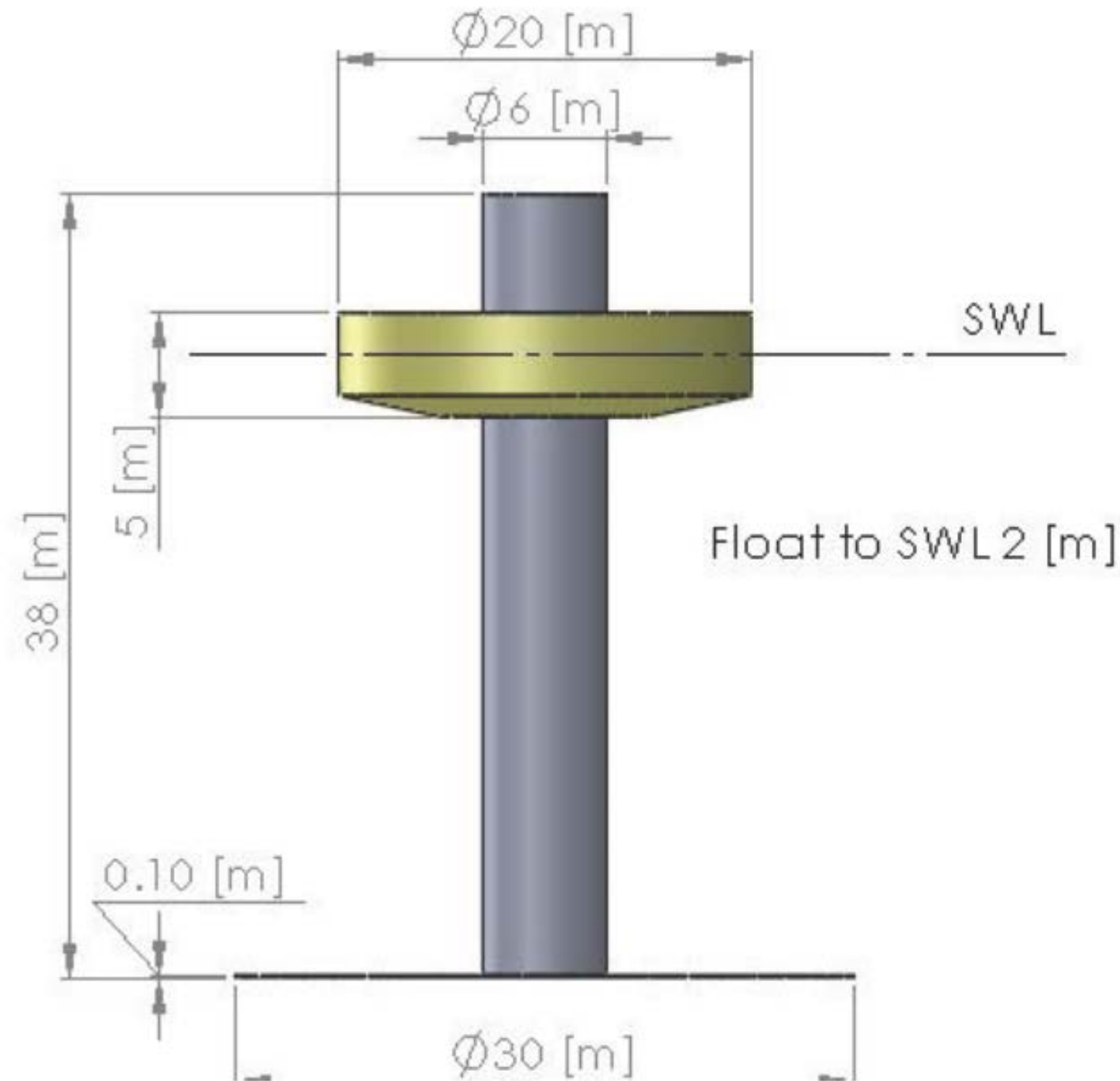
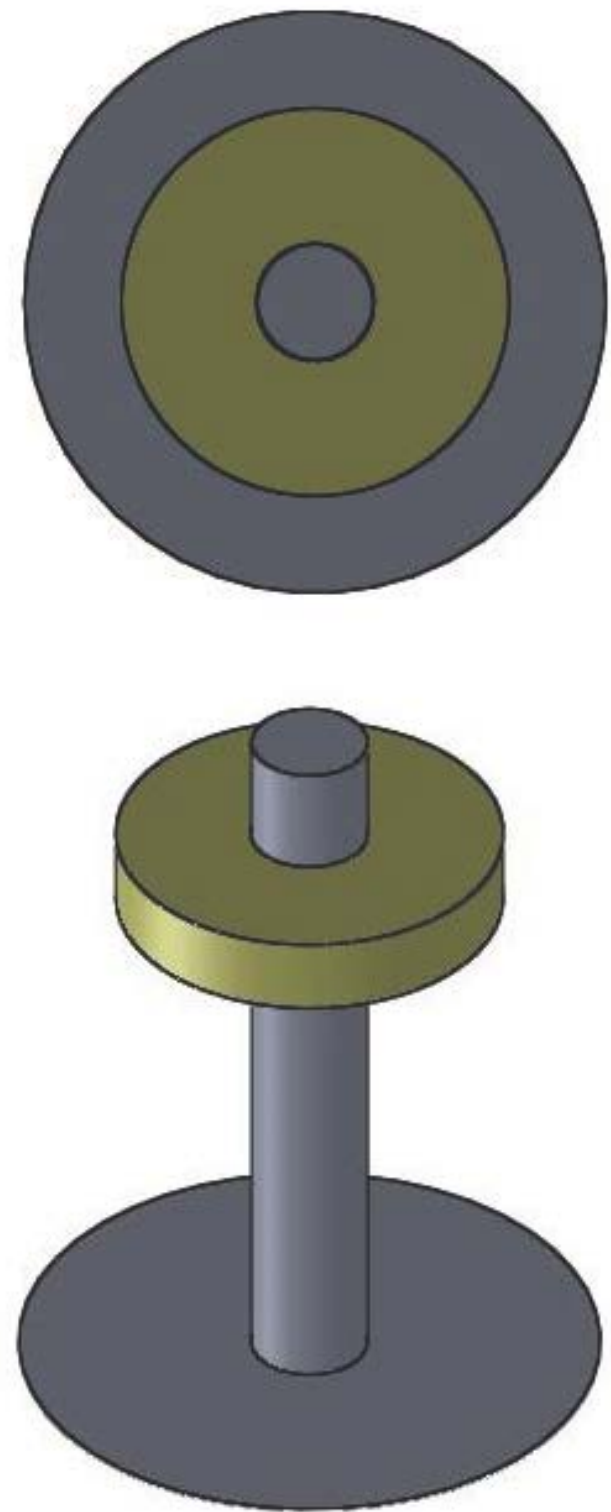
ode45 might be good for hydrodynamics and motion but will not work well for generator variables

ode15s required for generator variables but would result in many more time steps than necessary for hydrodynamics

This can result in significantly reduced simulation times

e.g. forty variable hydrodynamic system takes 1/10 the time steps with ode45 than ode15s as this is an inefficient solver for this problem type, so even with restricted max step there are significant reductions in computation time

Putting It All Together



US DOE Reference Model 3

Used as proof of concept/
demonstration of WEC-Sim model

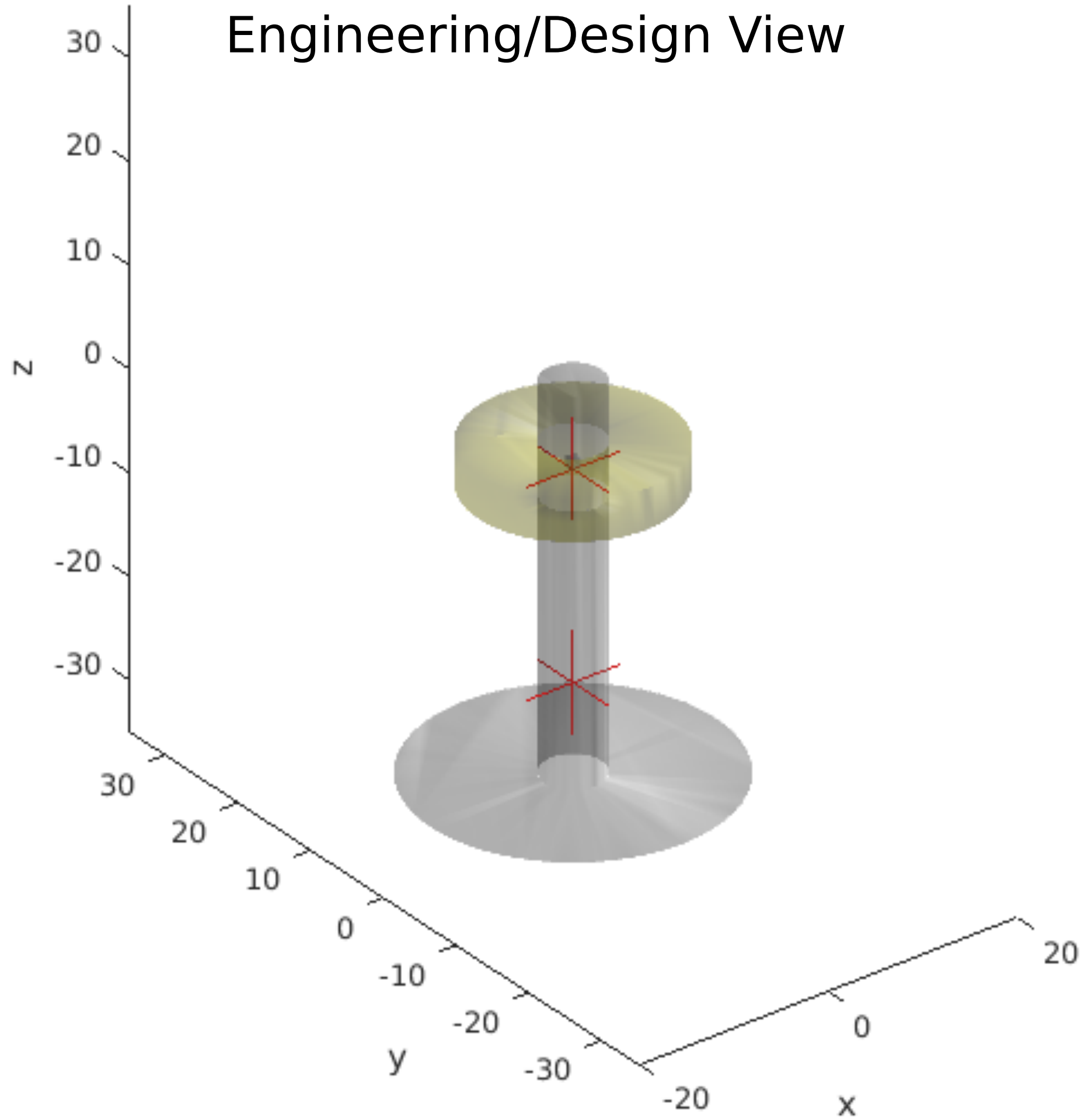
Simple Point Absorber with float
and spar with reaction plate.

Float is constrained to move along
spar riser (prismatic joint).

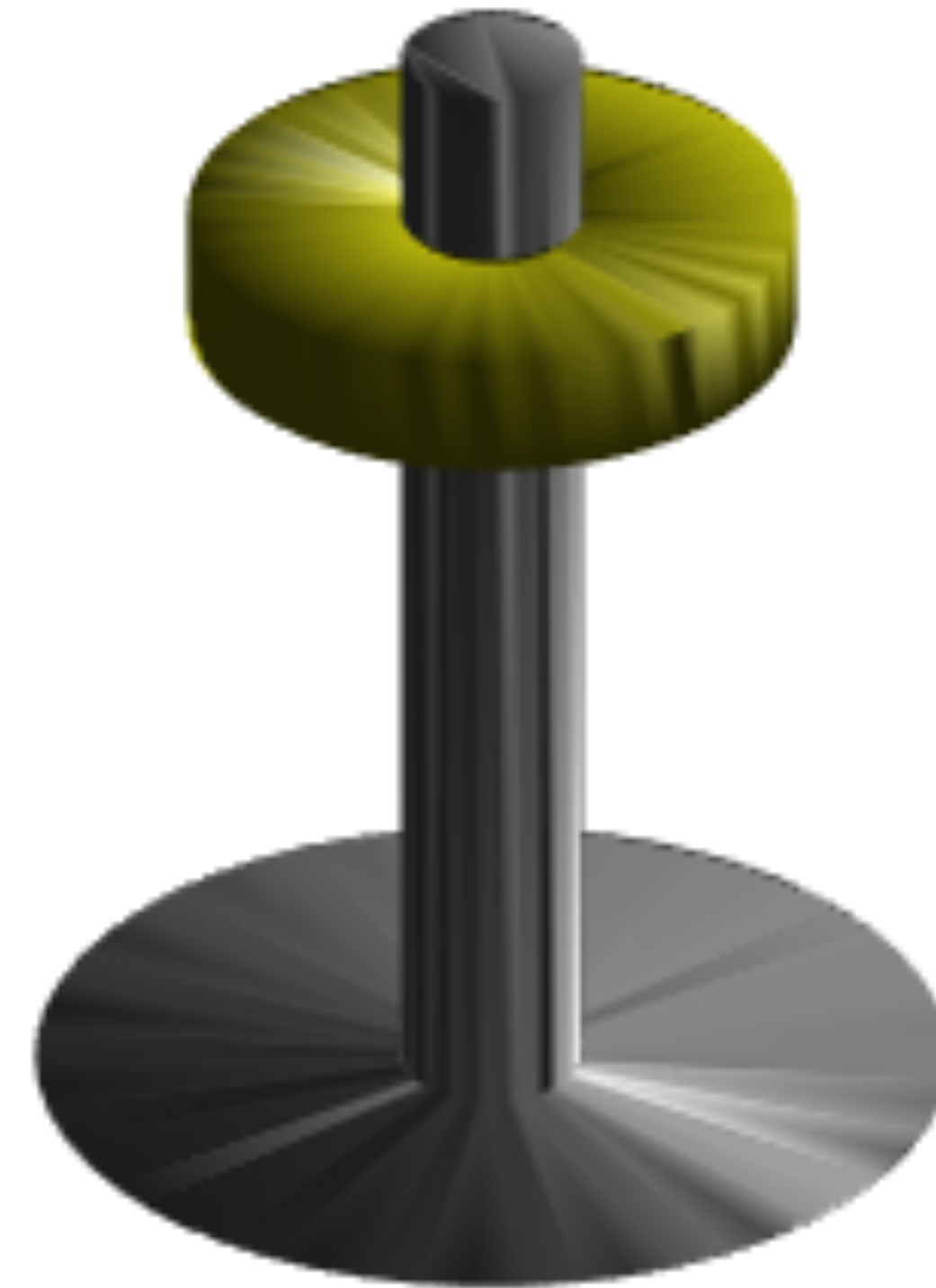
Replicated this as validation step
before developing case studies.

Our modified/ported
hydrodynamic model produces
identical outputs (forces)

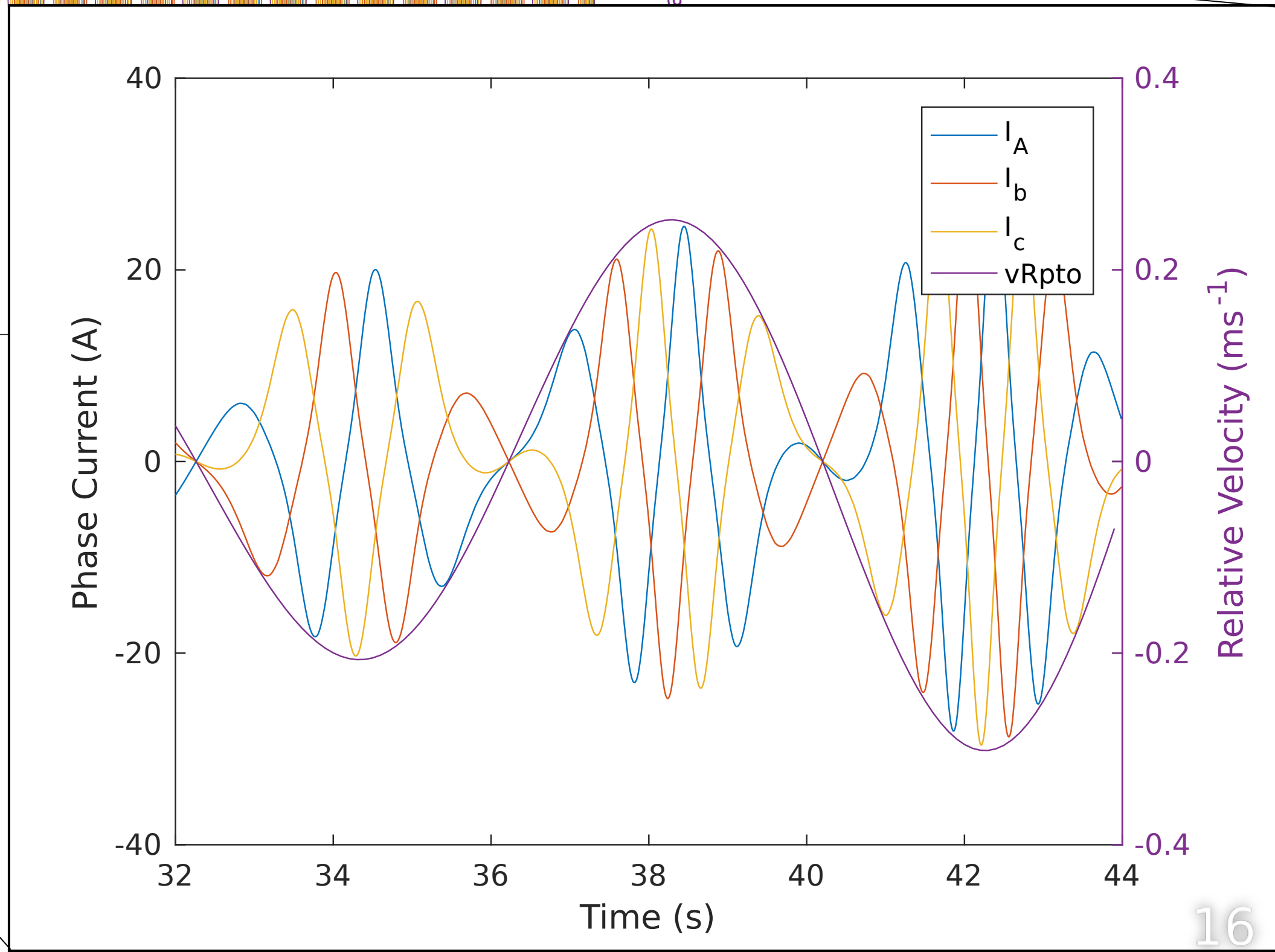
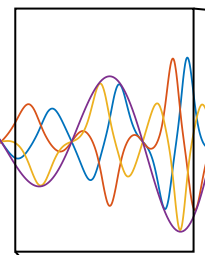
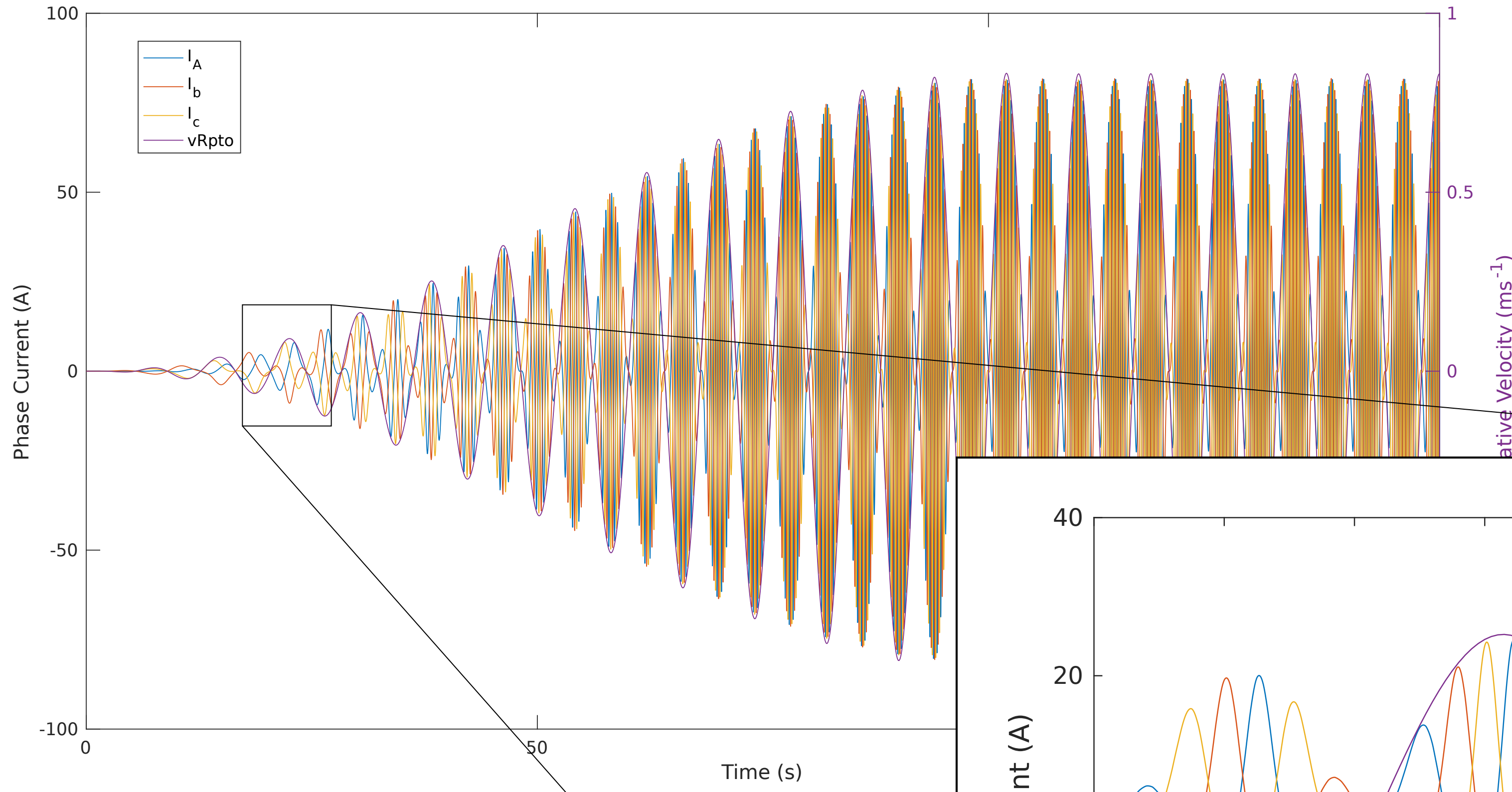
Engineering/Design View



Investor/Brochure View



Results



Video of system response:
<https://youtu.be/ZHil5qh4rGA>

Next Steps

Refine simulation setup process

Perform further testing and validation (discover and iron out bugs)

Use advanced method of generating state-space representation coefficients developed at Edinburgh

Integrate with control algorithms and 'next level down' the power electronics operating at khz

Integrate with new generator designs, and other work done by Newcastle University

Develop Case Studies

- Preferably useful ones for partners
- NEED DATA!!

QUESTIONS?



RenewNet Foundry - Optimisation



There is a well developed optimisation framework for electrical machines based on genetic algorithms

Optimisation is made more rapid using a variety of techniques:

- Fully exploiting symmetry in generators
- Only doing full detail simulation if initial test indicates design is in the right ball-park
- Optimisation can be split over multiple processors (or multiple computers in a network) using Condor, also works on Google Compute Cloud

Scoring functions based on multiple objectives, e.g.

- desired voltage
- desired power
- max THD
- desired physical size
- max temperature
- max structural deflections



Characterised by:

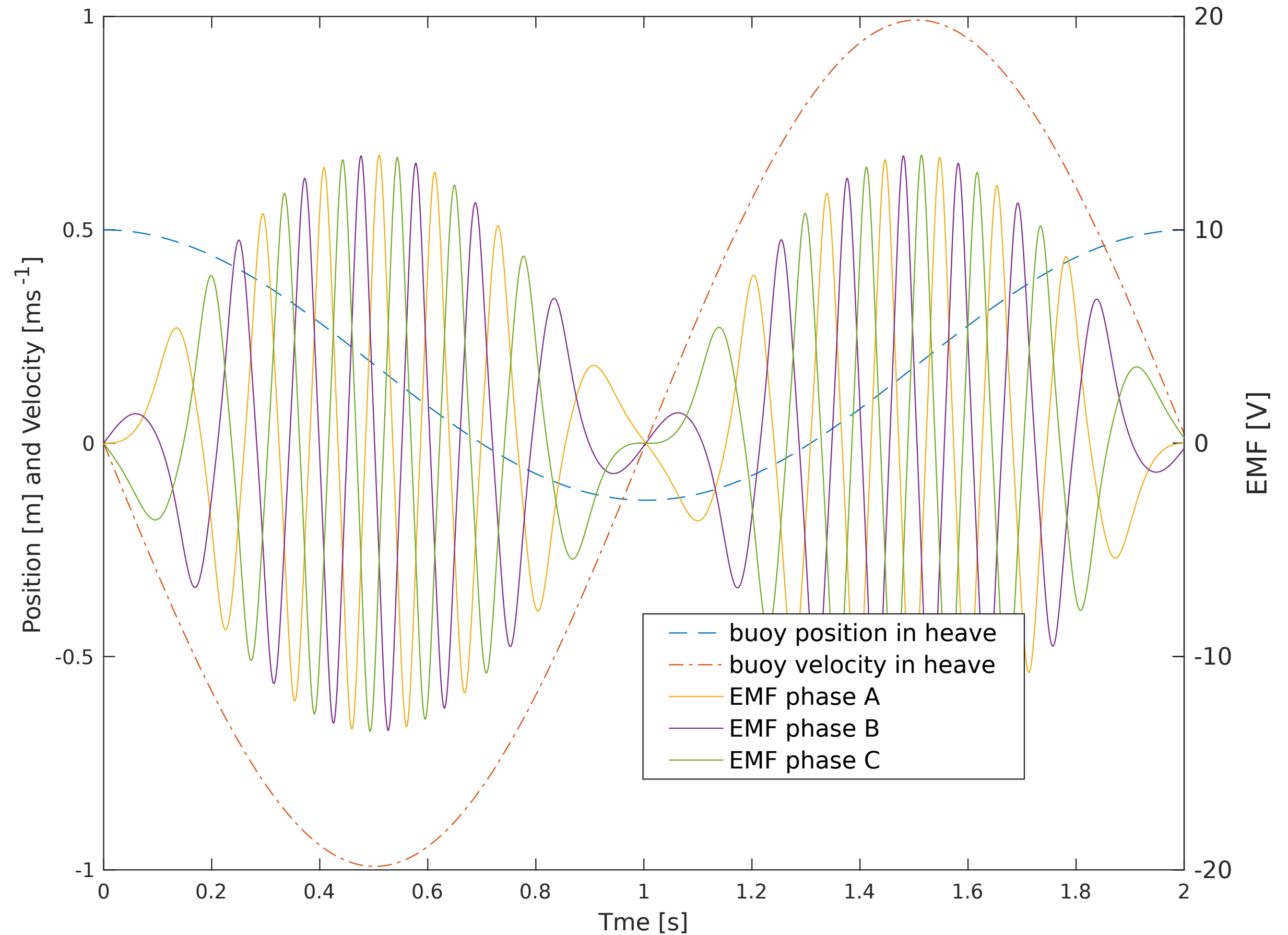
Low velocities

Large Forces

Direction of Motion suits linear generators

May require some kind of magnetic gearing, e.g. Vernier hybrid machine, or transverse flux machine (or actual magnetic gear)

Simulation can be very computationally expensive



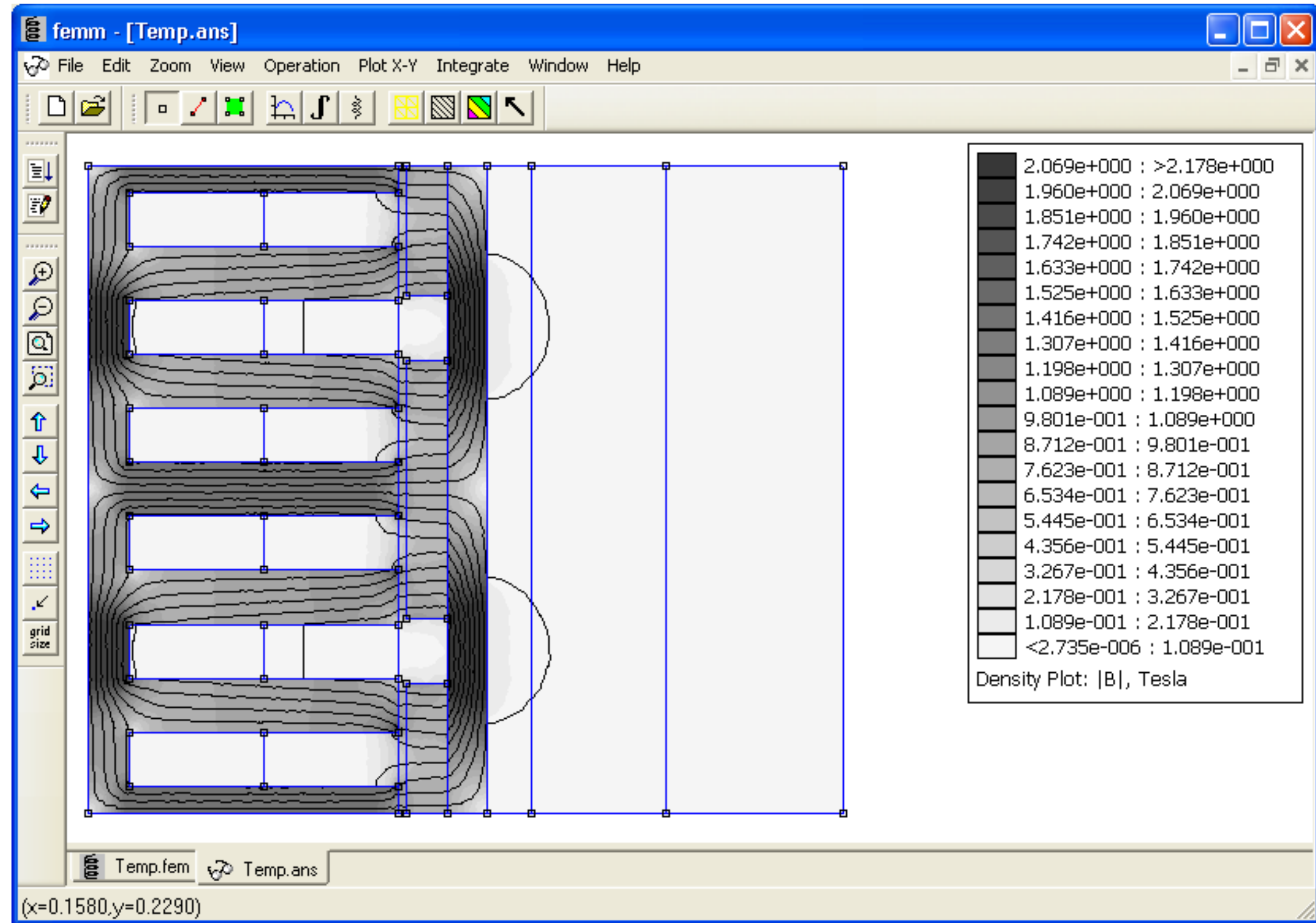
Open source tool for finite element analysis, particularly magnetics.

Very widely used (about 50% of FEA on conf posters done using FEMM¹)

Also does thermal, electrostatics, and (electric) current flow

Has built in scripting language (lua)

Has interface to Matlab, so you can create problems in m script



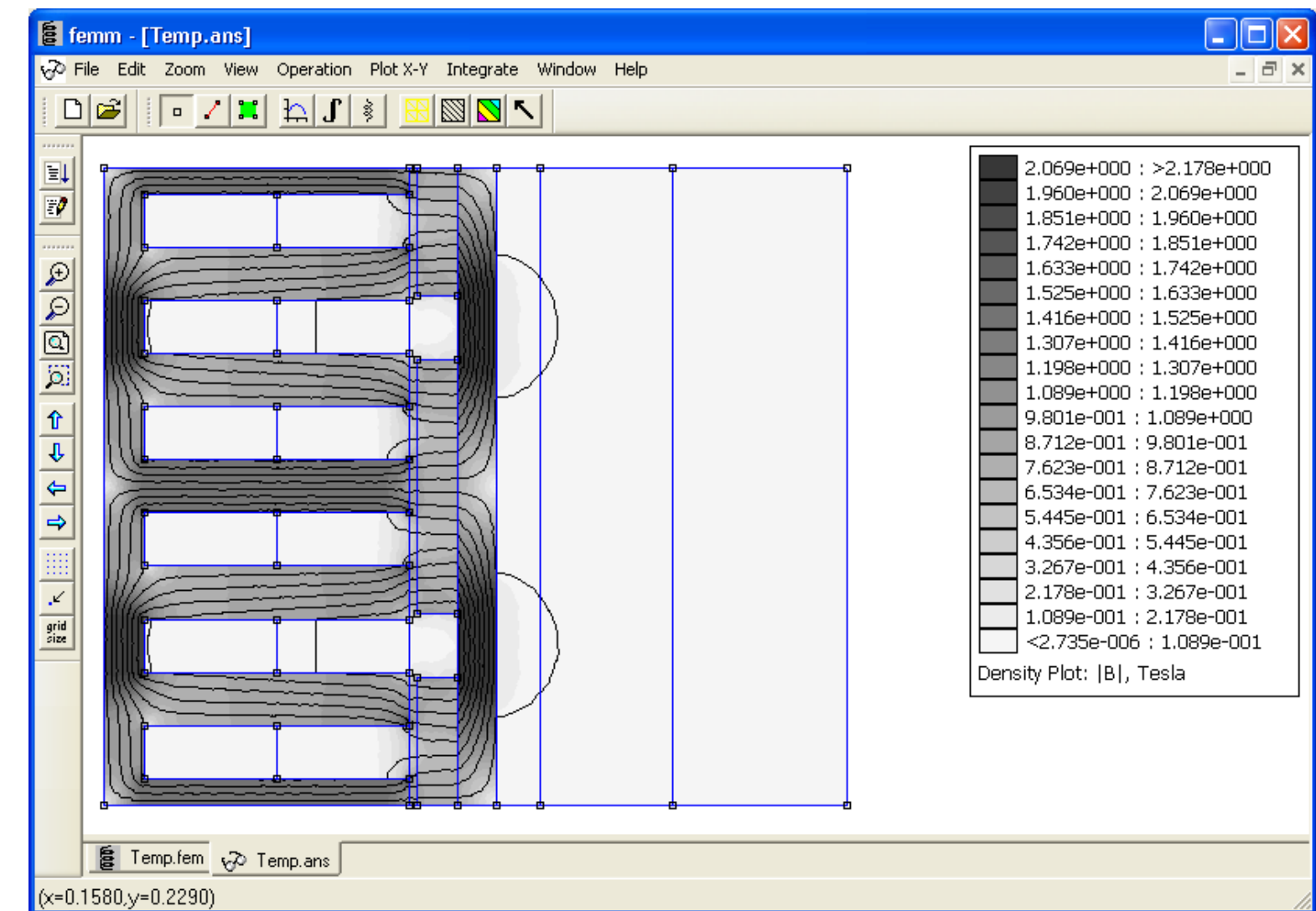
Problem creation from Matlab (or whatever) is *slow* due to interface method, ActiveX (problems built up by sending many individual lua commands)

FEMM is really windows only¹, and usually we want to run batch runs on linux servers

A FEMM instance must be launched to communicate with it and create/solve the problem

Post processing results also sent as as text (slow)

Problem creation scripts tend to be long and cumbersome (and therefore difficult to read, maintain, or make modular)



xfemm is a new interface to the finite element codes in FEMM

The underlying C++ code for the meshing, FEA and post-processing has been extracted and turned into a cross-platform library

An interface to this library has then been created in Matlab (and Octave)

Problems created in a normal Matlab data structure and written to a .fem file for processing

Post-processing is done by direct memory access, so much faster than old method

Parallel execution of problems much easier

```
% Create a new FemmProblem structure for the problem.
% The first argument is either a string or scalar
% value. Here we use `axi` to choose an axisymmetric
% simulation type. Note the length units of 'inches'.
% Other length units are available.
FemmProblem = ...
    newproblem_mfemm('axi', ...
                    'Frequency', 0, ...
                    'LengthUnits', 'inches');

% Add a wire material to the problem for use later
% a large database of materials is present, and new
% materials are easy to add
FemmProblem = ...
    addmaterials_mfemm( FemmProblem, '18 AWG' );

% Add a circuit for later use with a 1A current
FemmProblem = ...
    addcircuit_mfemm( FemmProblem, 'Coil', ...
                     'TotalAmps_re', 1 );

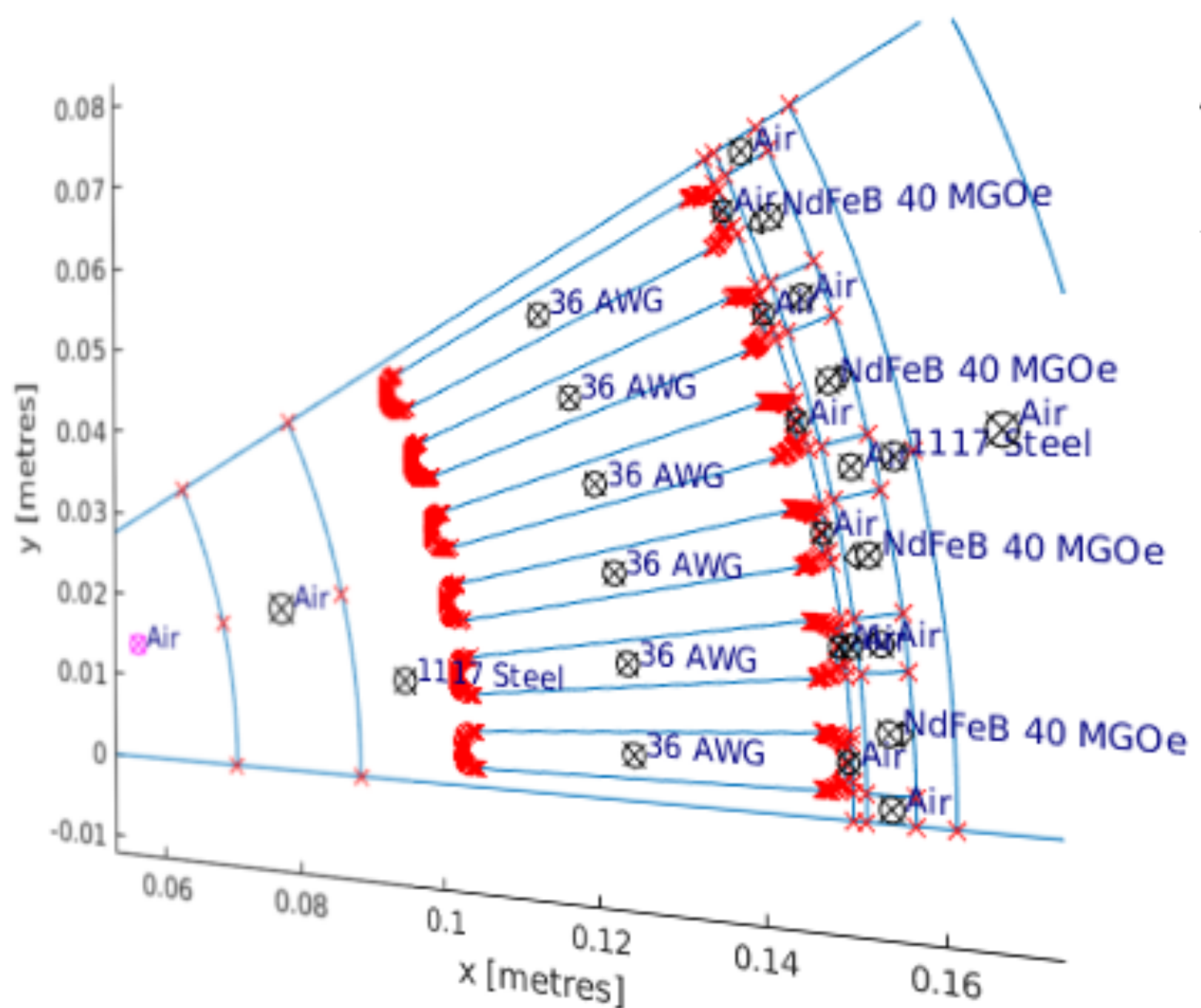
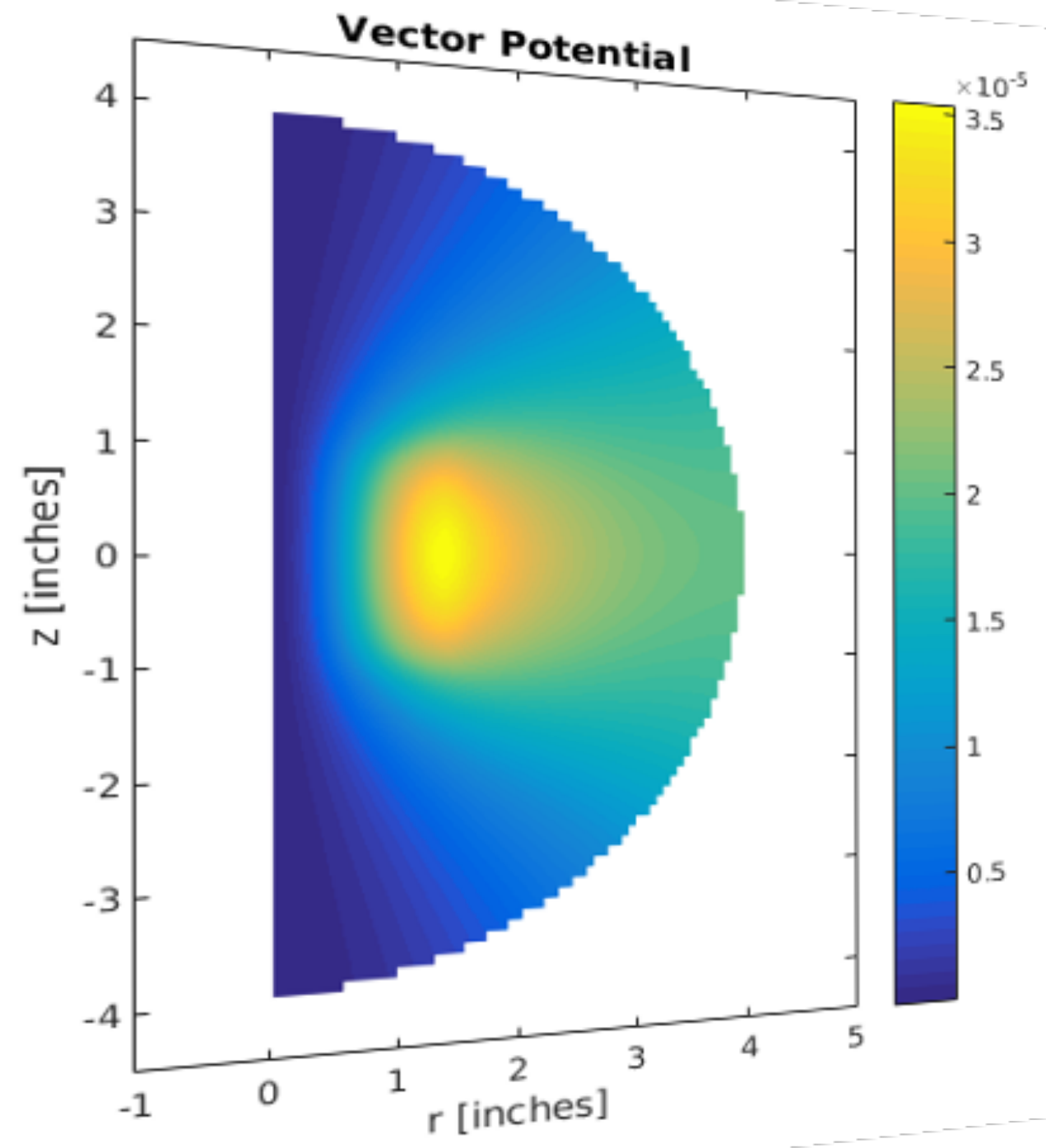
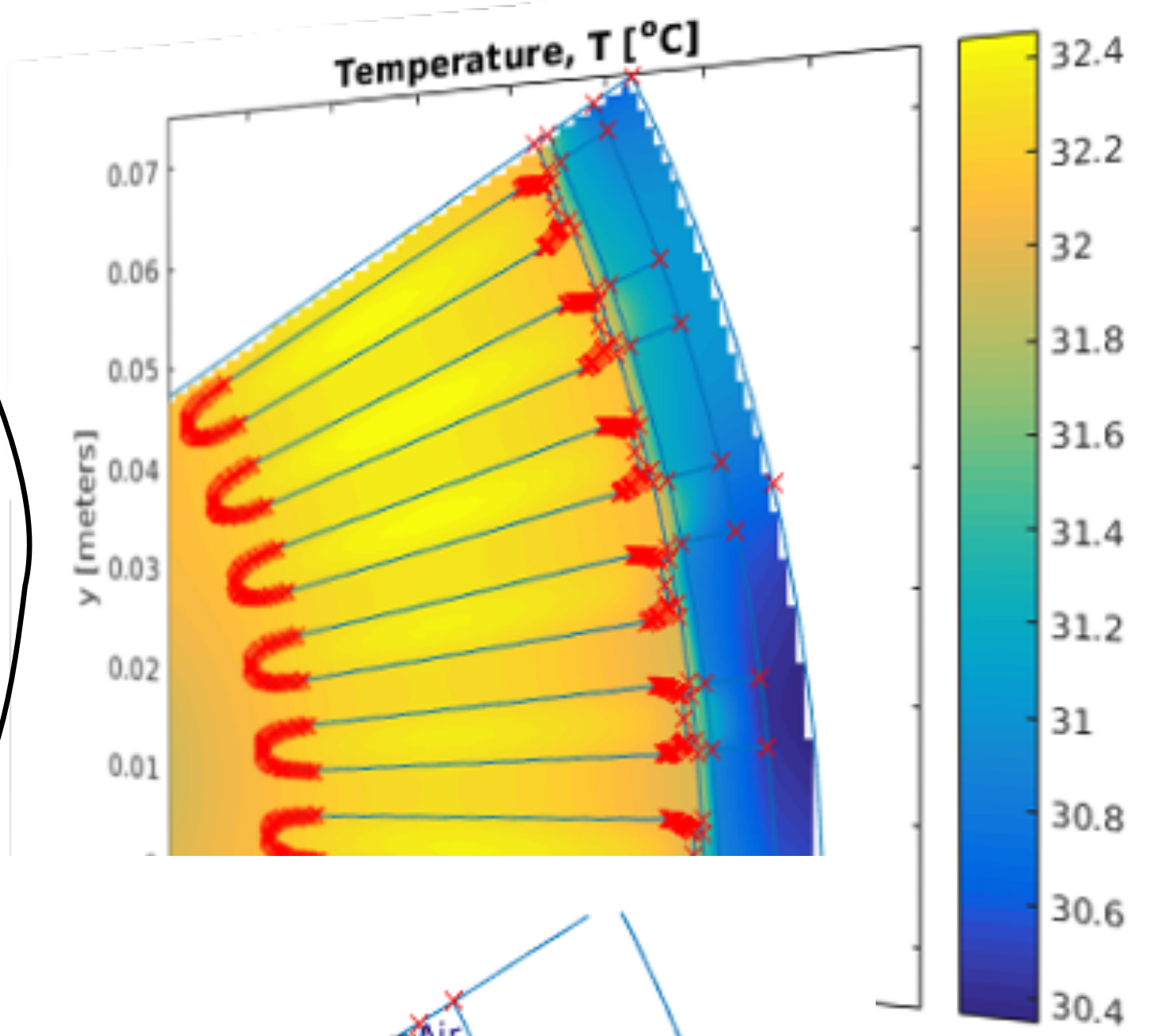
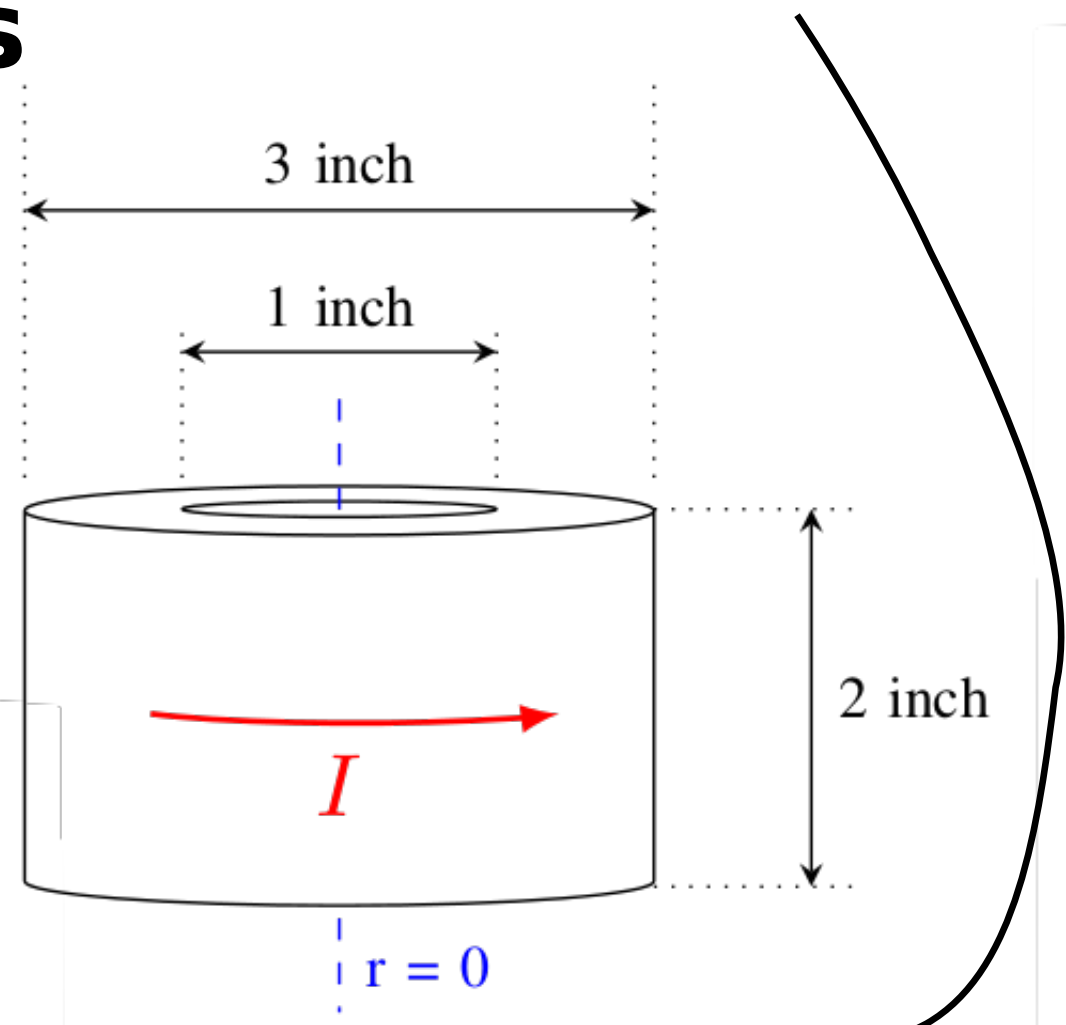
% Now start creating the problem geometry
outernodes = [ 0, -4;
               0,  4 ];

[FemmProblem, nodeinds, nodeids] = ...
    addnodes_mfemm( FemmProblem, ...
                   outernodes(:,1), ...
                   outernodes(:,2) );

[FemmProblem, arcind] = ...
    addarcsegments_mfemm( FemmProblem, ...
                          nodeids(1), ...
                          nodeids(2), ...
                          180, ...
                          'Coil' );
```


xfem - Examples

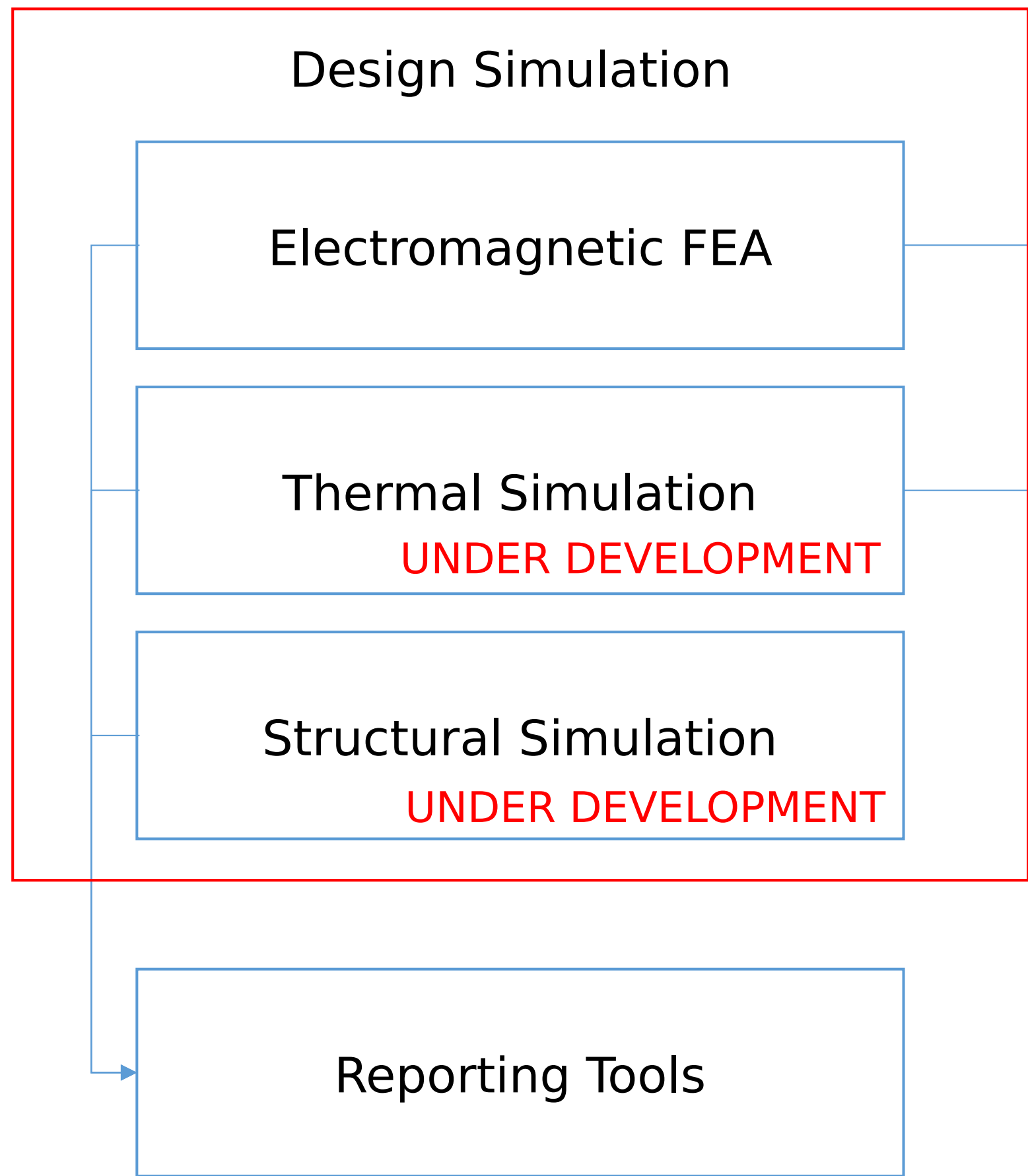
Simple Coil, harmonic problem



Radial Flux PM Machine



Architecture Overview



The toolbox is entirely implemented in Matlab, and most functionality is also compatible with Octave.

The core of the software is the design simulation code. This code can be called by an optimisation procedure which generates new designs based on set of desired parameters and constraints.

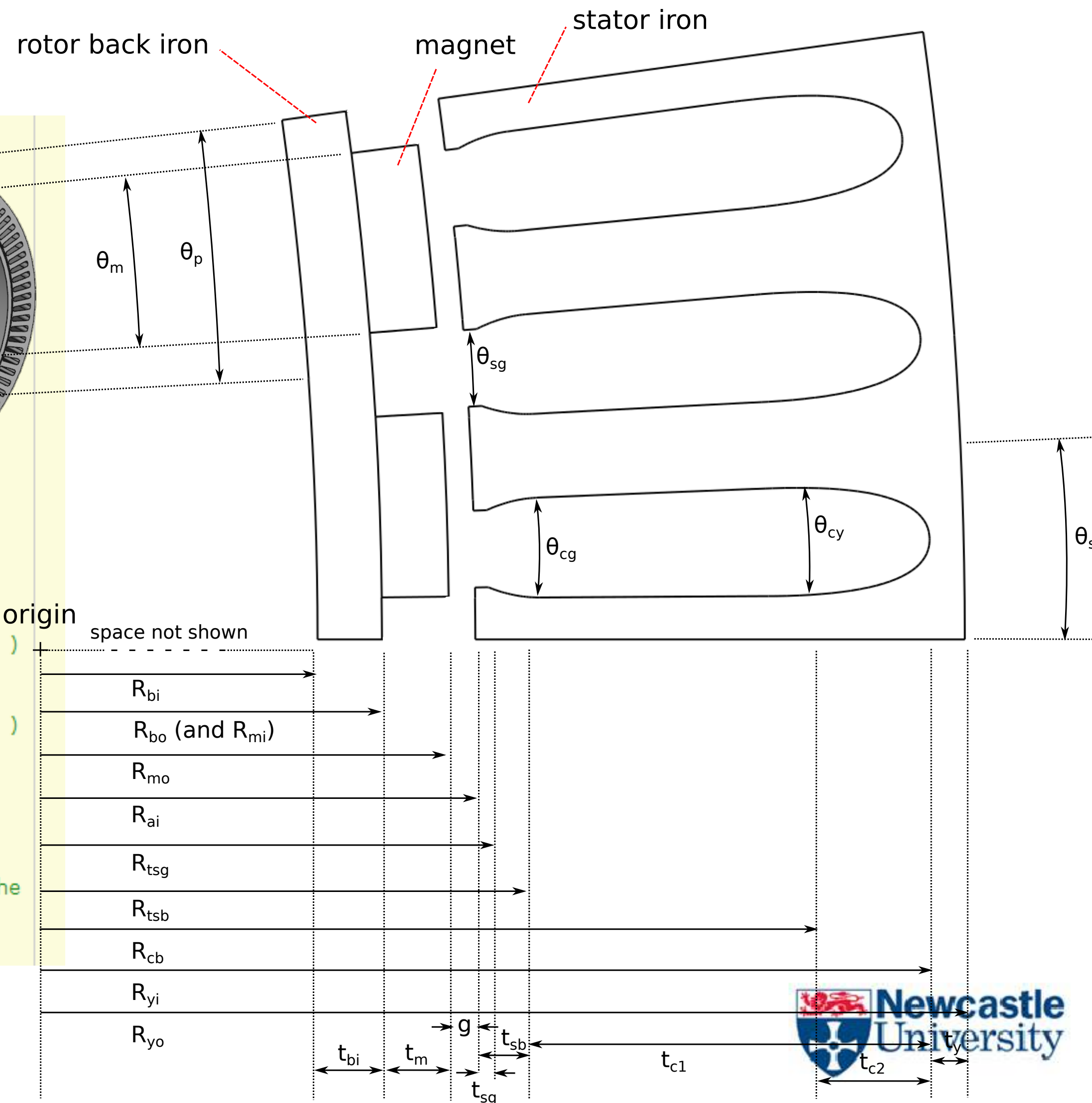
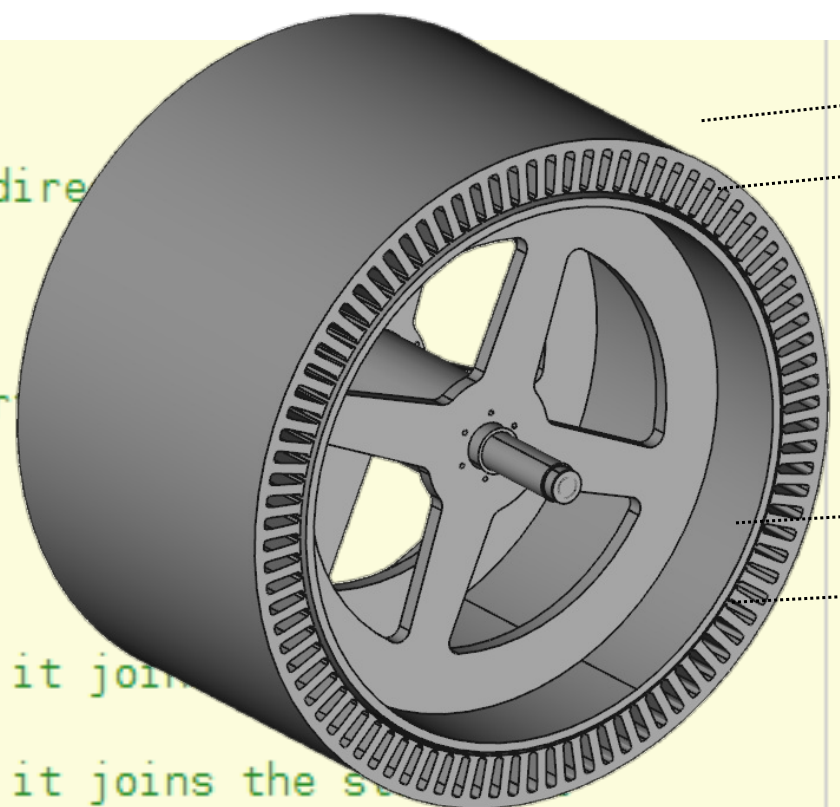
In addition to the core simulation code, utility functions for reporting on the design and exporting information are also provided.

The primary documentation for the toolbox is in the form of example scripts, tutorials and function help text. The main entry points to these will be introduced in this guide.

Example: Radial flux slotted machine

```

% The outer radius of the rotor back iron
design.Ryo = 95e-3;
% The height of the magnets in the radial direction
design.tm = 6.3e-3;
% The thickness of the rotor back iron
design.tbi = 29.9523e-3;
% The thickness of the stator yoke (the part of the stator that is not a pole)
design.ty = 17.4e-3;
% The radial height of a slot opening
design.tc = 16.4960e-003;
design.tc(2) = 2.1995e-003;
% The height of the slot shoe at the point it joins the stator yoke
design.tsb = 2.604e-3;
% The height of the slot shoe at the point it joins the rotor back iron
design.tsg = 1.7364e-3;
% The radial air gap length between rotor and stator
design.g = 2e-3;
% magnet pitch in radians
design.thetam = (tau / design.Poles) * 0.667;
% coil slot opening pitch in radians ( space a coil has to fit in a slot )
% at the slot end closest to the air gap
design.thetacg = 84.7661e-003;
% coil slot opening pitch in radians ( space a coil has to fit in a slot )
% at the slot end closest to the armature yoke
design.thetacy = 93.0181e-003;
% the pitch in radians of gap between the shoes that partially cover the
% slot openings
design.thetasg = 38.6622e-003;
% stack length of the machine, the depth along the cylindrical axis of the
% machine
design.ls = 88.9e-3;
    
```

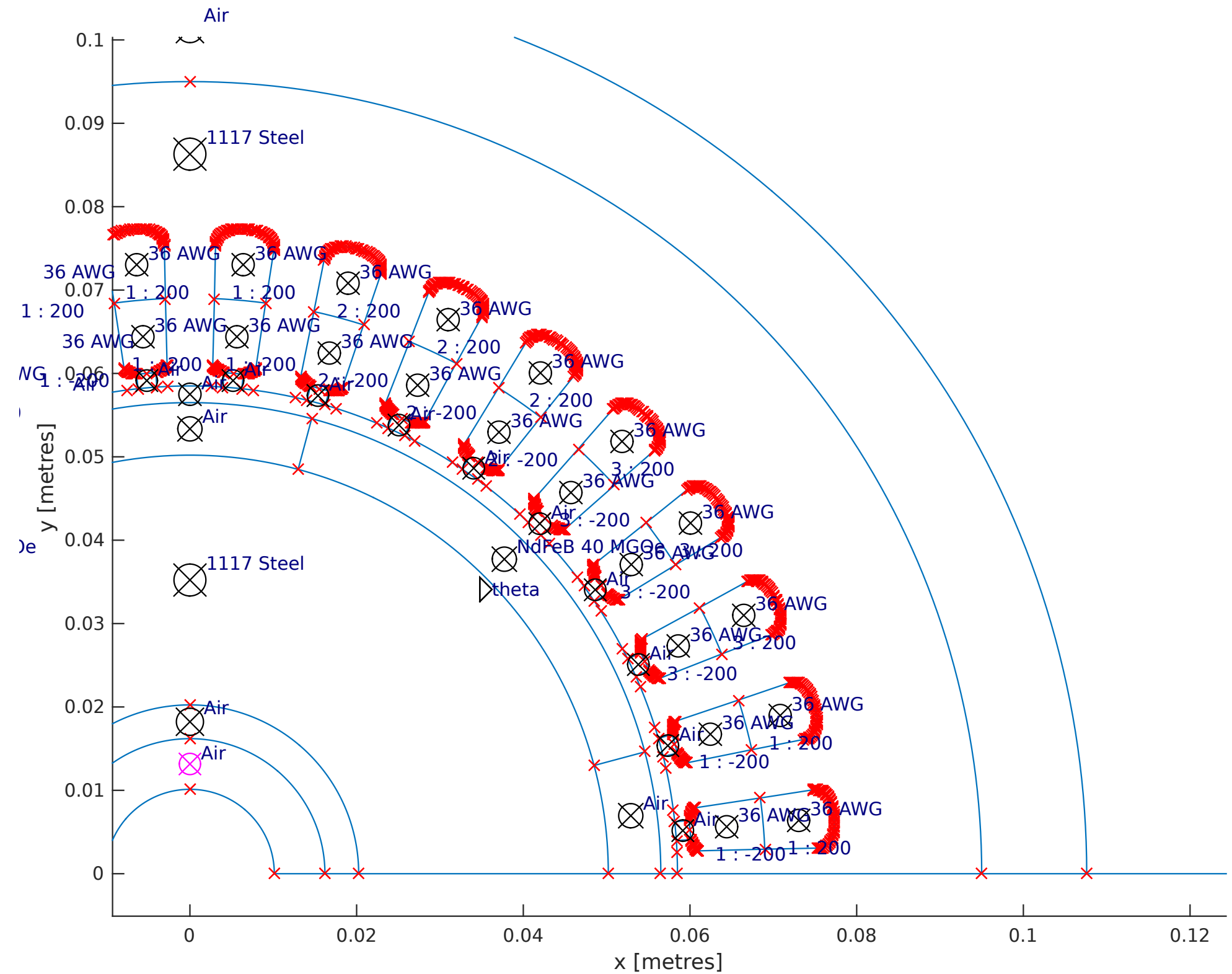


Example: Radial flux slotted machine

Static Sim

For iron cored machine, multiple static finite element simulations are performed with the rotor in slightly different positions to extract information such as forces and flux linkage (for other types only one FEA sim may be required)

Default number of positions is 10, but this can be changed. There are many options like this with sensible defaults, but available for fine-grained control of the sim process.





RenewNet Foundry - Electrical Tools

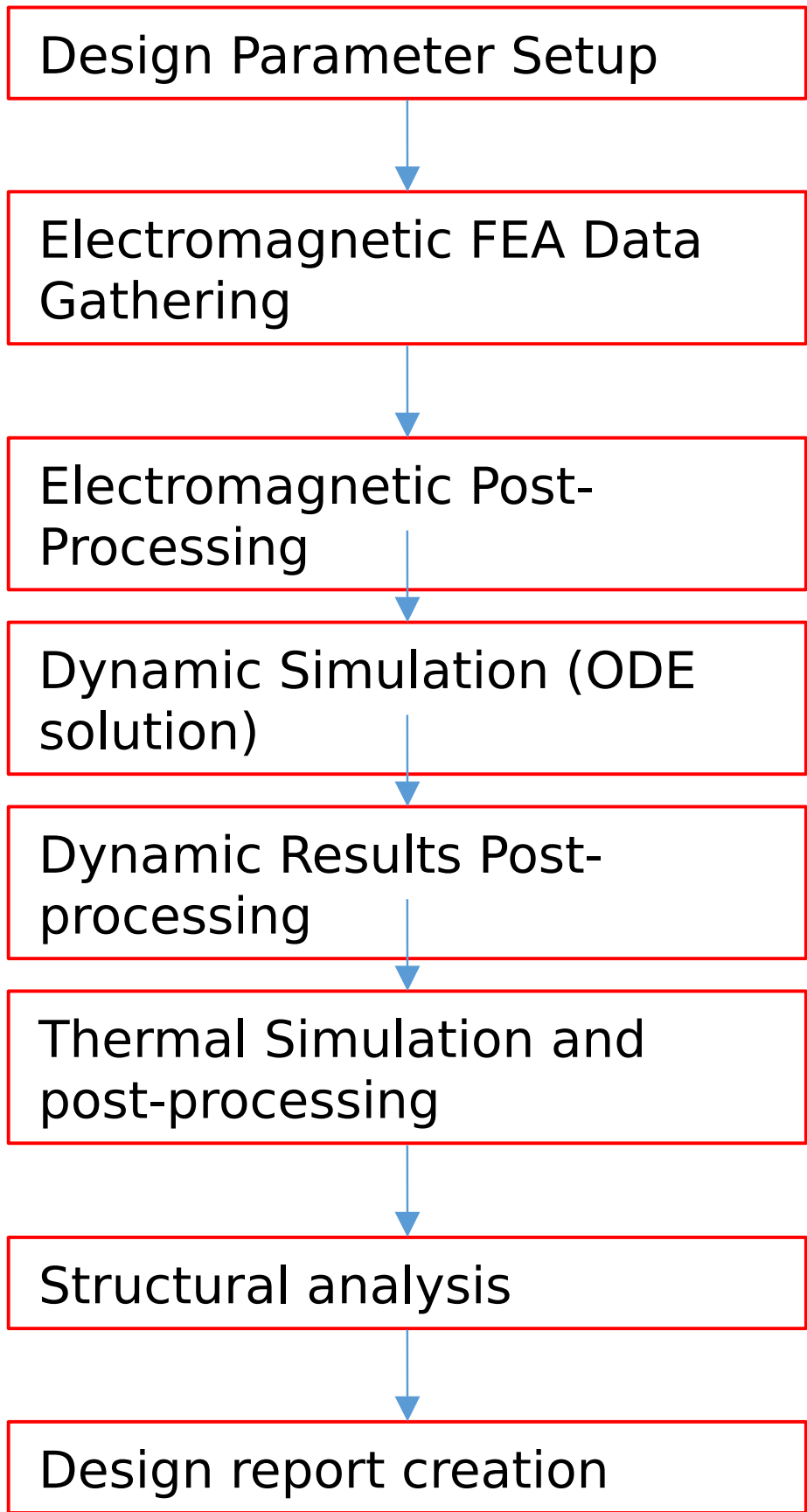


Workflow and Modularity

- The code is quite modular, e.g.
- separate functions for drawing rotors (or translators) and armatures
 - standalone functions for calculating winding layouts
 - common post-processing functions (power from currents etc.)

Therefore even if you are not interested in PM machines, there will likely be interesting code you could use to develop your own simulation.

Simulation Workflow



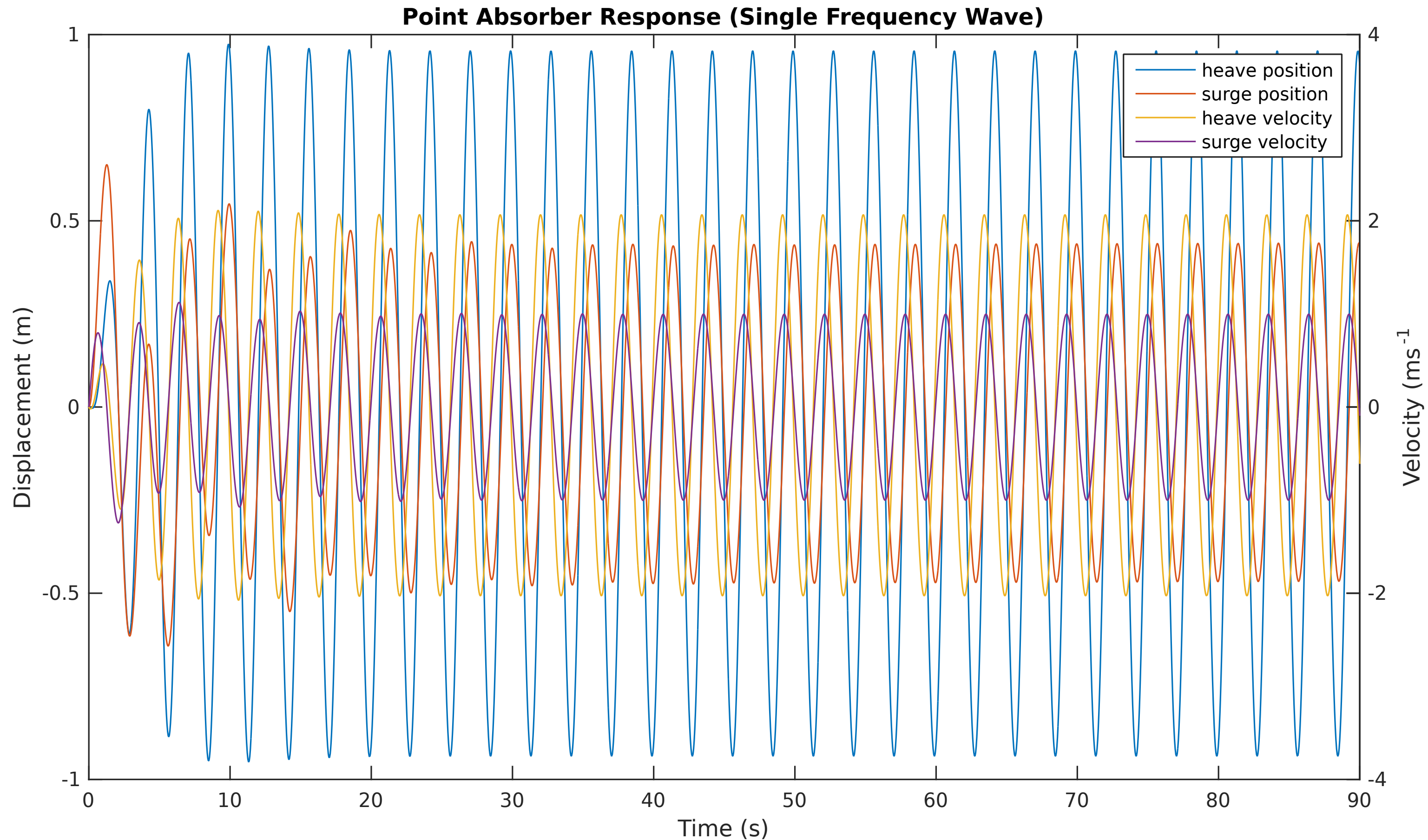
Main Related Functions

- completedesign_RADIAL_SLOTTED
- completedesign_RADIAL_SLOTTED
simfun_RADIAL_SLOTTED.m
simulatemachine_AM
- finfun_RADIAL_SLOTTED.m
prescribedmotfinfun_RADIAL_SLOTTED.m
- prescribedmotodetorquefcn_ROTARY.m
- prescribedmotresfun_RADIAL_SLOTTED_NOVA.m
- heatflow_RADIAL_SLOTTED_NOVA.m
- structureeval_RADIAL_SLOTTED_NOVA.m
- designreport_RADIAL_SLOTTED_NOVA.m





RenewNet Foundry - Buoy Simulation



The foundry includes code to simulate a point absorber in ocean waves based on prony's method.





RenewNet Foundry - Turbine Simulation



A basic wind/tidal turbine simulation exists (although less developed and integrated than buoy sim)

Developed by master's student and given some benchmarking against NREL test case.

Could probably be easily be integrated with generator simulations

