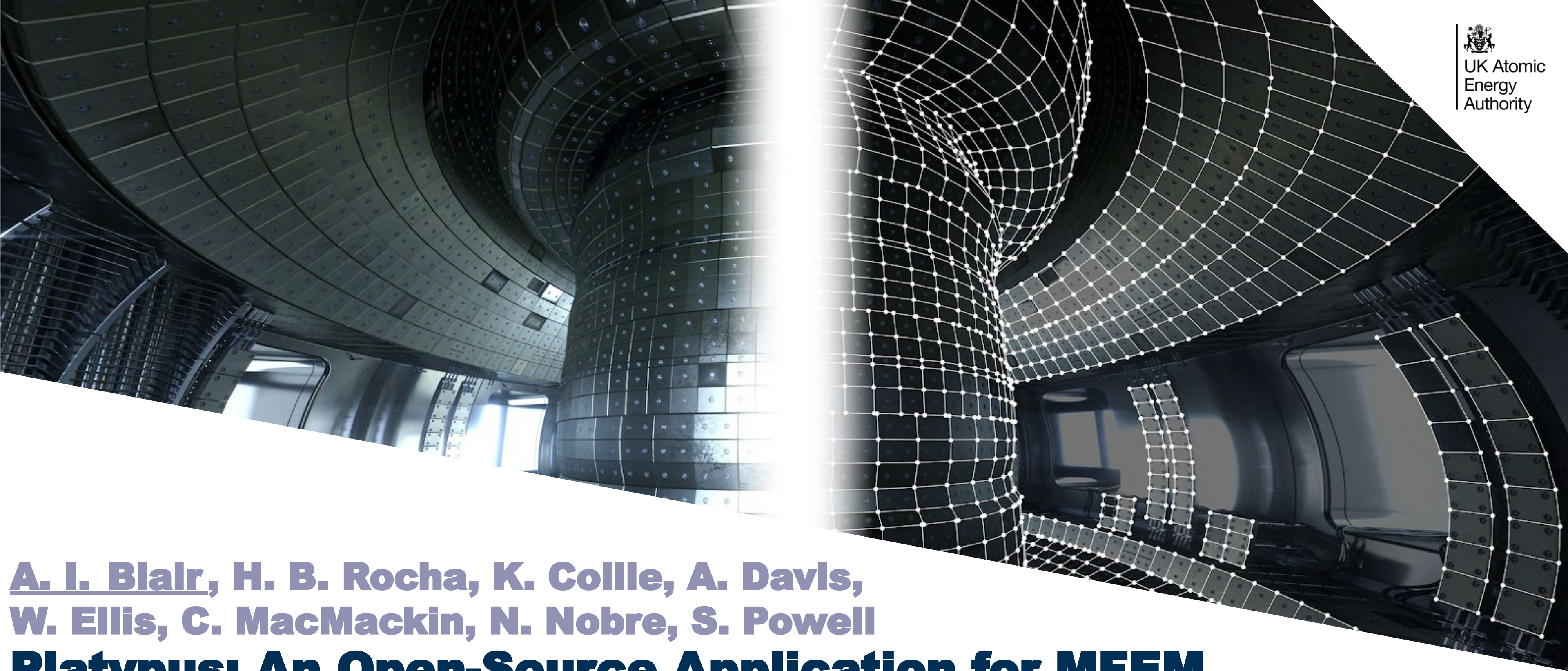


UK Atomic Energy Authority

A. I. Blair, H. B. Rocha, K. Collie, A. Davis,
W. Ellis, C. MacMackin, N. Nobre, S. Powell

Platypus: An Open-Source Application for MFEM Problem Set-Up and Assembly in the MOOSE Framework

MFEM Community Workshop 2024 | 23rd October 2024

Outline

UK Atomic Energy Authority

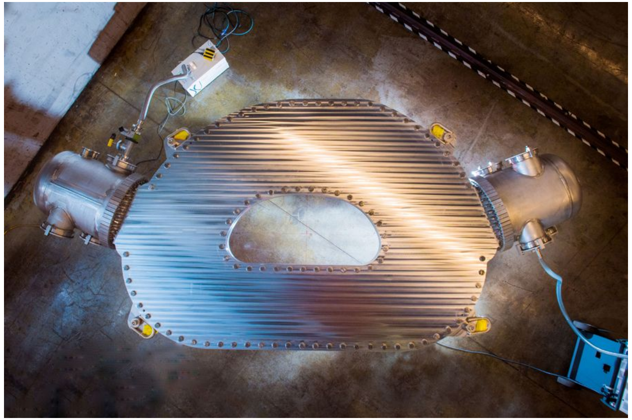
1. **Motivation**
2. **MOOSE at UKAEA**
3. **Platypus Introduction**
4. **Platypus Examples**
5. **Future Work**
6. **Summary**

Motivation

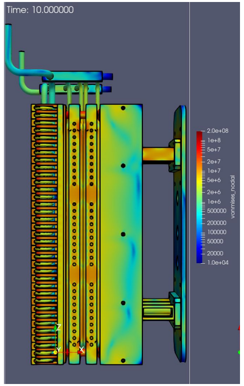
There will not be sufficient experimental data by 2040 to fully qualify components for next-generation fusion reactors!

Engineers must have scalable multiphysics simulation tools to enable them to substantively assess and de-risk complex component and reactor designs *in silico.*

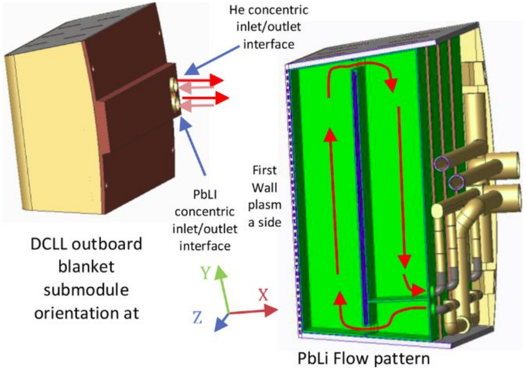
We need multiphysics simulation tools capable of modelling a range of fusion use cases, spanning both time and frequency domains.



https://news.mit.edu/2021/MIT-CFS-major-advance-toward-fusion-energy-0908



CHIMERA CSUT



Khodak A et al. 2018 *Fusion Engineering and Design* **137** 124-129

UKAEA Advanced Engineering Simulation Group

UK Atomic Energy Authority

Part of UKAEA's Computing Division - interdisciplinary group with ~20 people (and growing!)

- Number of open-source applications in the MOOSE ecosystem under development - available at https://github.com/aurora-multiphysics
- Coupling to a range of external tools (OpenFOAM, NekRS, OpenMC…)
- Aim to provide tools to enable engineers to run actionable simulations of fusion-relevant problems on complex geometries

AURORA
Radiation transport + thermo-mechanics

Apollo
Electromagnetism MFEM adaptor

Hephaestus
MFEM EM Library

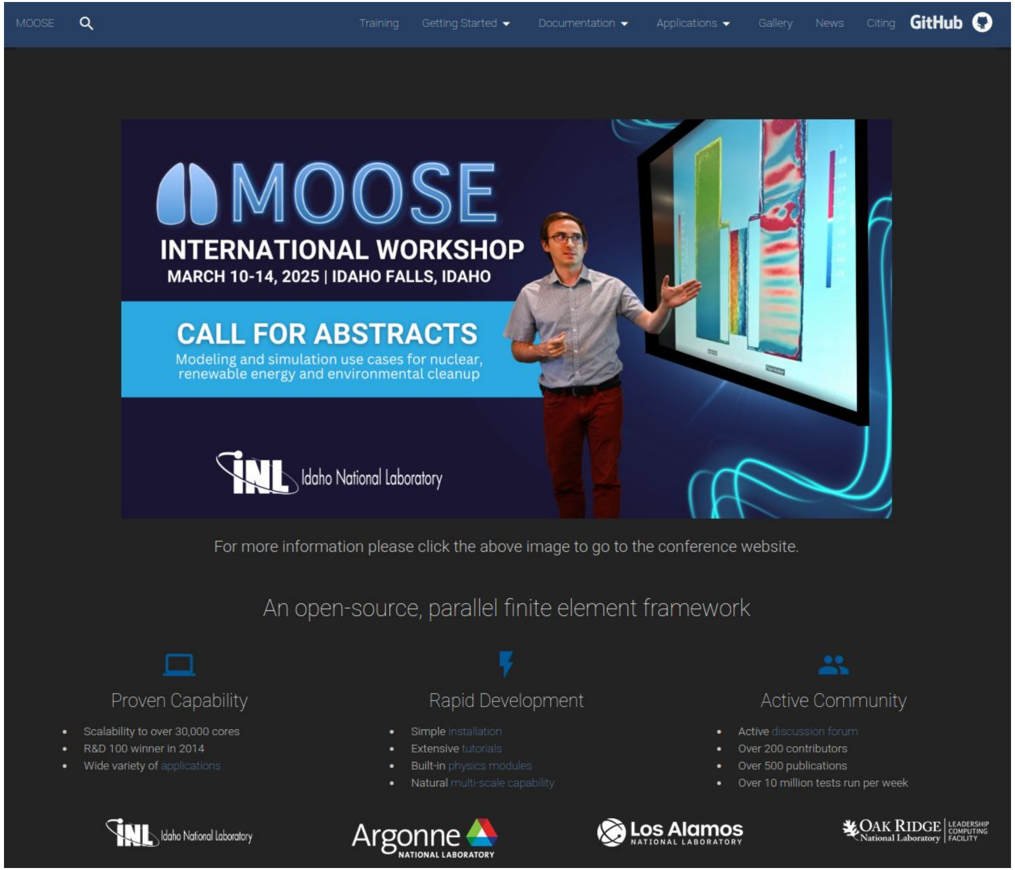
Astrea / Hippo
OpenFOAM adaptor

Proteus
Turbulent Fluids MHD

Achlys
Hydrogen ion transport

What is MOOSE?

UK Atomic Energy Authority

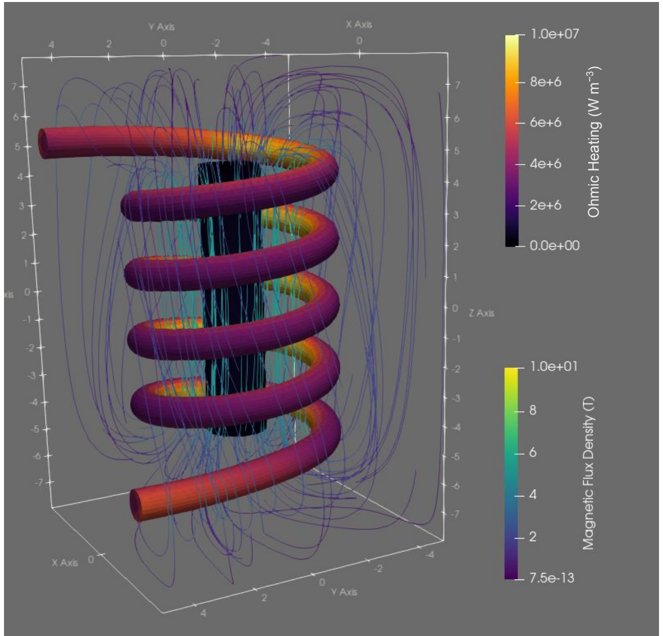


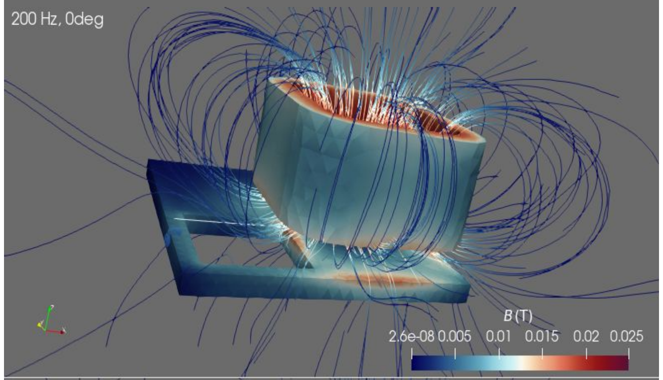
- Open source parallel FE framework from INL
- Active, large user community (repository starred by >1.7k users on GitHub)
- Wide range of existing validated physics modules
- Simple user interface for problem definition
- Demonstrated scalability on tens of thousands of CPU cores

LibMesh Limitations

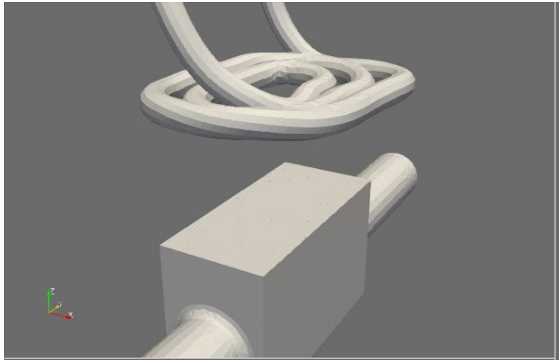
However - **MOOSE**’s underlying FE library libMesh has a number of limitations compared to MFEM:

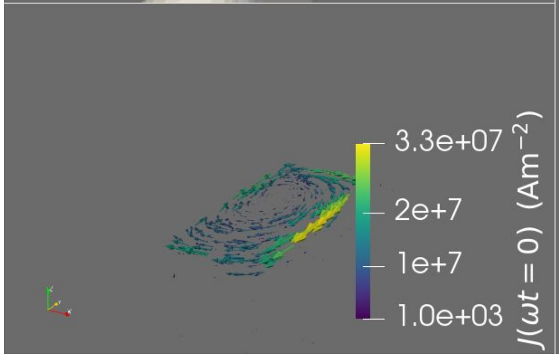
- Limited support for H(curl) and H(div) conforming vector FE types (e.g. Dirichlet BCs for ND1 and RT FEs not supported)
- Performant preconditioners for EM diffusion problems in H(curl) (eg. HypreAMS) require additional fine-grid orientation information at setup not yet available from MOOSE
- No GPU support

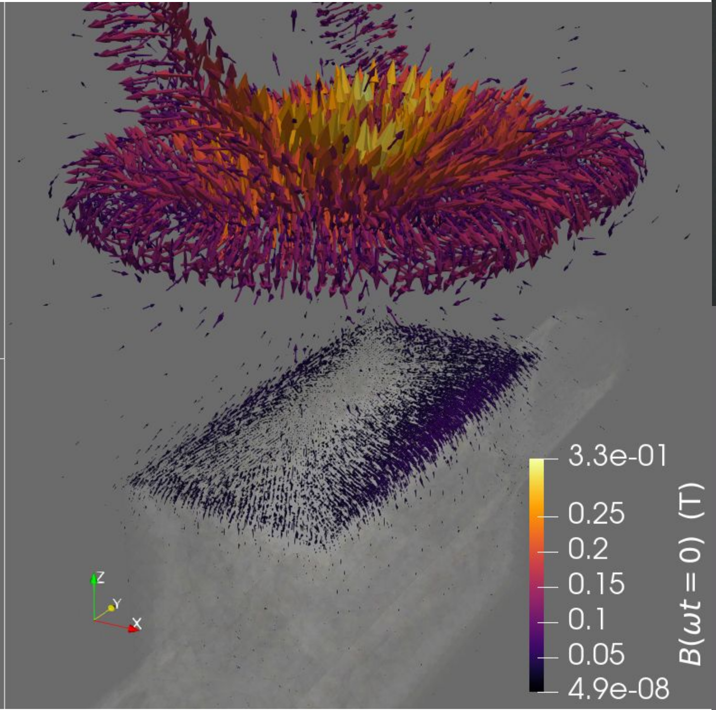


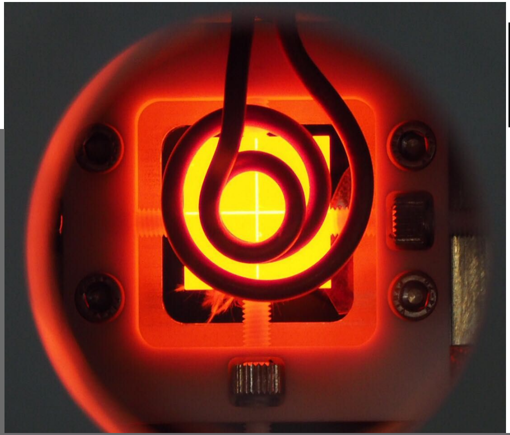


HIVE Digital Twin Development

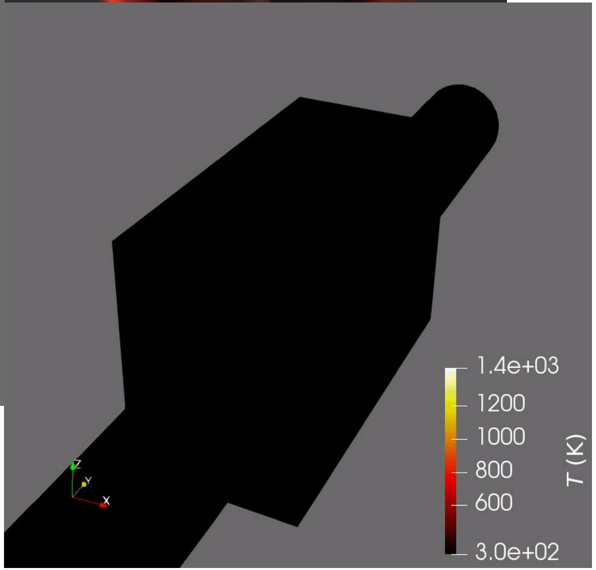








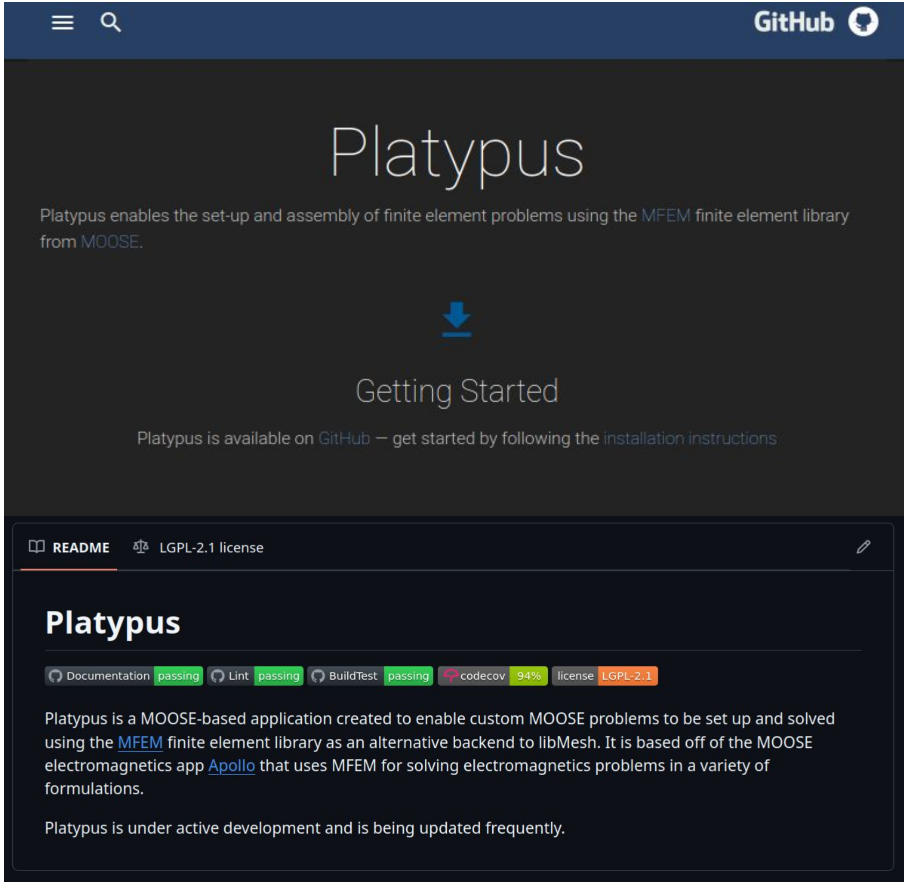
UK Atomic Energy Authority



Some success writing (and wrapping) an MFEM EM solver library for use with MOOSE - **Apollo/Hephaestus**

Platypus

UK Atomic Energy Authority



GitHub

Platypus

Platypus enables the set-up and assembly of finite element problems using the MFEM finite element library from MOOSE.

Getting Started

Platypus is available on GitHub — get started by following the installation instructions

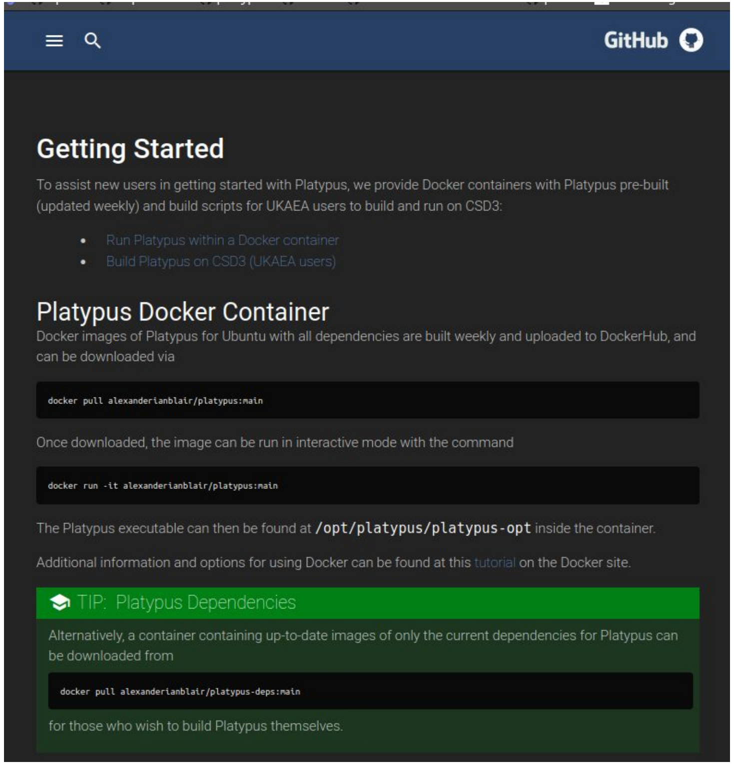
README LGPL-2.1 license

Platypus

Documentation passing | Lint passing | BuildTest passing | codecov 94% | license LGPL-2.1

Platypus is a MOOSE-based application created to enable custom MOOSE problems to be set up and solved using the MFEM finite element library as an alternative backend to libMesh. It is based off of the MOOSE electromagnetics app Apollo that uses MFEM for solving electromagnetics problems in a variety of formulations.

Platypus is under active development and is being updated frequently.



GitHub

Getting Started

To assist new users in getting started with Platypus, we provide Docker containers with Platypus pre-built (updated weekly) and build scripts for UKAEA users to build and run on CSD3:

- Run Platypus within a Docker container
- Build Platypus on CSD3 (UKAEA users)

Platypus Docker Container

Docker images of Platypus for Ubuntu with all dependencies are built weekly and uploaded to DockerHub, and can be downloaded via

```
docker pull alexanderianblair/platypus:main
```

Once downloaded, the image can be run in interactive mode with the command

```
docker run -it alexanderianblair/platypus:main
```

The Platypus executable can then be found at `/opt/platypus/platypus-opt` inside the container.

Additional information and options for using Docker can be found at this tutorial on the Docker site.

TIP: Platypus Dependencies

Alternatively, a container containing up-to-date images of only the current dependencies for Platypus can be downloaded from

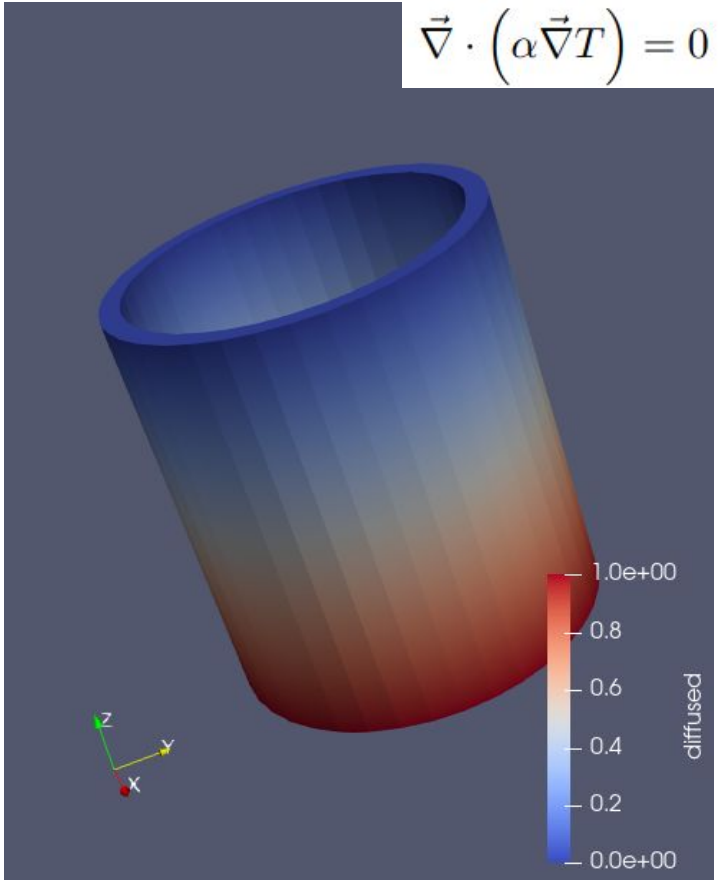
```
docker pull alexanderianblair/platypus-deps:main
```

for those who wish to build Platypus themselves.

https://aurora-multiphysics.github.io/platypus/

Diffusion Example

$$\vec{\nabla} \cdot \left(\alpha \vec{\nabla} T \right) = 0$$




```
[Mesh]
  type = MFEMMesh
  file = gold/mug.e
[]

[Problem]
  type = MFEMProblem
[]

[FESpaces]
  [H1FESpace]
    type = MFEMFESpace
    fec_type = H1
    fec_order = FIRST
  []
[]

[Variables]
  [diffused]
    type = MFEMVariable
    fespace = H1FESpace
  []
[]

[Materials]
  [DiffusiveMaterial]
    type = MFEMGenericConstantMaterial
    prop_names = diffusivity
    prop_values = 1.0
  []
[]

[Kernels]
  [diff]
    type = MFEMDiffusionKernel
    variable = diffused
    coefficient = diffusivity
  []
[]
```

```
[Coefficients]
  [TopValue]
    type = MFEMConstantCoefficient
    value = 0.0
  []
  [BottomValue]
    type = MFEMConstantCoefficient
    value = 1.0
  []
[]

[BCs]
  [MugBottom]
    type = MFEMScalarDirichletBC
    variable = diffused
    boundary = 1
    coefficient = BottomValue
  []
  [MugTop]
    type = MFEMScalarDirichletBC
    variable = diffused
    boundary = 2
    coefficient = TopValue
  []
[]

[Preconditioner]
  [BoomerAMG]
    type = MFEMHypreBoomerAMG
  []
[]

[Solver]
  type = MFEMHypreGMRES
  preconditioner = BoomerAMG
  l_tol = 1e-16
  l_max_its = 1000
[]

[Executioner]
  type = MFEMSteady
  device = cpu
  assembly_level = legacy
[]

[Outputs]
  [ParaViewDataCollection]
    type = MFEMParaViewDataCollection
    file_base = OutputData/MFEM/Diffusion
    vtk_format = ASCII
    execute_on = 'final'
  []
[]
```

UK Atomic Energy Authority

Diffusion Example – Mesh & FE Variables

UK Atomic Energy Authority

Platypus

```
[Mesh]
  type = MFEMMesh
  file = gold/mug.e
[]

[Problem]
  type = MFEMProblem
[]

[FESpaces]
  [H1FESpace]
    type = MFEMFESpace
    fec_type = H1
    fec_order = FIRST
  []
[]

[Variables]
  [diffused]
    type = MFEMVariable
    fespace = H1FESpace
  []
[]
```

Native MOOSE

```
[Mesh]
  type = FileMesh
  file = gold/mug.e
[]

[Problem]
  type = FEProblem
[]

[Variables]
  [diffused]
    order = FIRST
    family = LAGRANGE
  []
[]
```

- **MFEMProblem** Problem type used to set up and solve FE problem using MFEM rather than libMesh
- **MFEMVariable** sets up an `mfem::GridFunction`; allows use of MFEM's FE types and orders.

Diffusion Example – Materials & Kernels

UK Atomic Energy Authority

Platypus

```
[Materials]
  [DiffusiveMaterial]
    type = MFEMGenericConstantMaterial
    prop_names = diffusivity
    prop_values = 1.0
  []
[]

[Kernels]
  [diff]
    type = MFEMDiffusionKernel
    variable = diffused
    coefficient = diffusivity
  []
[]
```

Native MOOSE

```
[Materials]
  [DiffusiveMaterial]
    type = GenericConstantMaterial
    prop_names = diffusivity
    prop_values = 1.0
  []
[]

[Kernels]
  [diff]
    type = MatDiffusion
    variable = diffused
    diffusivity = diffusivity
  []
[]
```

- **MFEMMaterial** types add (piecewise) coefficients to the problem on mesh subdomains, representing material properties.
- **MFEMKernels** add domain integrators to the weak form.

Diffusion Example – Boundary Conditions

UK Atomic Energy Authority

Platypus

```
[Coefficients]
  [TopValue]
    type = MFEMConstantCoefficient
    value = 0.0
  []
  [BottomValue]
    type = MFEMConstantCoefficient
    value = 1.0
  []
[]

[BCs]
  [MugBottom]
    type = MFEMScalarDirichletBC
    variable = diffused
    boundary = 1
    coefficient = BottomValue
  []
  [MugTop]
    type = MFEMScalarDirichletBC
    variable = diffused
    boundary = 2
    coefficient = TopValue
  []
[]
```

Native MOOSE

```
[BCs]
  [MugBottom]
    type = DirichletBC
    variable = diffused
    boundary = 1
    value = 1.0
  []
  [MugTop]
    type = DirichletBC
    variable = diffused
    boundary = 2
    value = 0.0
  []
[]
```

- **MFEMCoefficients** can also be added directly as needed
- **MFEMBoundaryConditions** either add boundary integrators to the weak form (for integrated BCs) or set essential BCs.

Diffusion Example – Solver & Outputs

UK Atomic Energy Authority

Platypus

```
[Preconditioner]
  [BoomerAMG]
    type = MFEMHypreBoomerAMG
  []
[]

[Solver]
  type = MFEMHypreGMRES
  preconditioner = BoomerAMG
  l_tol = 1e-16
  l_max_its = 1000
[]

[Executioner]
  type = MFEMSteady
  device = cpu
  assembly_level = legacy
[]

[Outputs]
  [ParaViewDataCollection]
    type = MFEMParaViewDataCollection
    file_base = OutputData/MFEM/Diffusion
    vtk_format = ASCII
    execute_on = 'final'
  []
[]
```

Native MOOSE

```
[Executioner]
  type = Steady
  solve_type = 'LINEAR'
  petsc_options_iname = '-ksp_type -pc_type -pc_hypre_type'
  petsc_options_value = 'gmres hypre boomeramg'
[]

[Outputs]
  [ExodusOutput]
    type = Exodus
    file_base = OutputData/MOOSE/Diffusion
    execute_on = 'final'
  []
[]
```

- Available MFEM solvers – including from Hypre – can be set up and used directly
- Execution on CPU/GPU and MFEM assembly level controlled by the **Executioner**
- File export of results to `mfem::DataCollections` supported using MOOSE's Outputs system.

Diffusion Example – GPU Execution

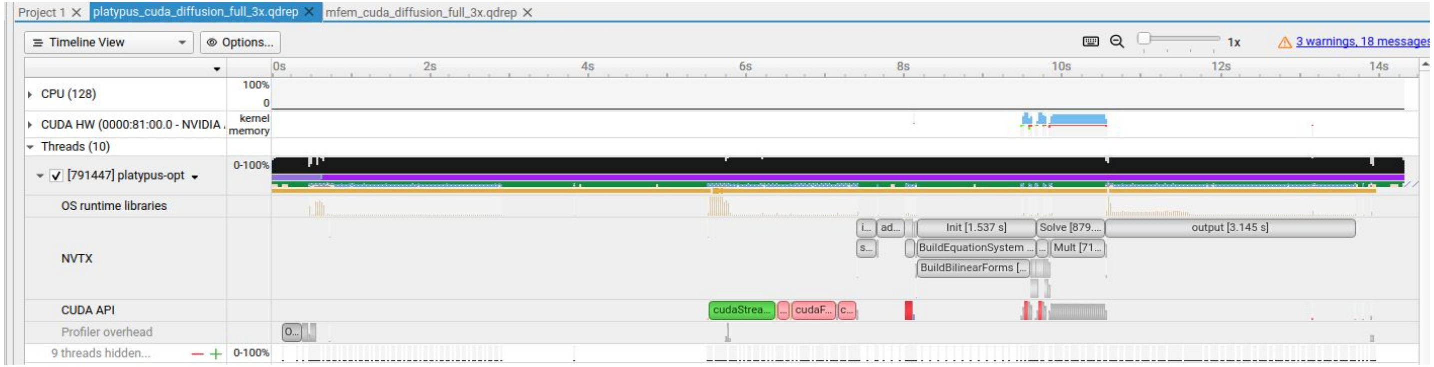
UK Atomic Energy Authority

```
[Executioner]
  type = MFEMSteady
  device = cuda
  assembly_level = full
[]
```

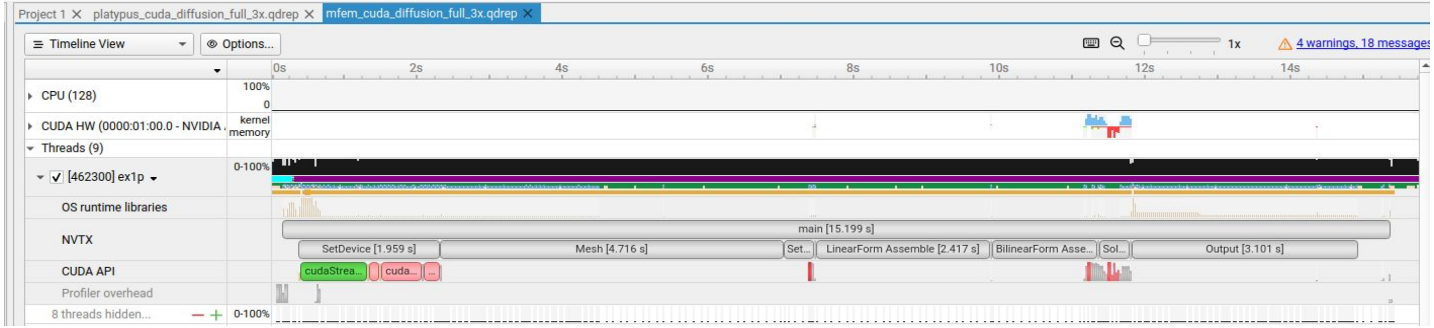
- Support for execution on GPU currently only for single variable steady state problems.
- Minimal overhead compared to native MFEM runs

1,349,545 DoFs

Platypus



MFEM Example 1

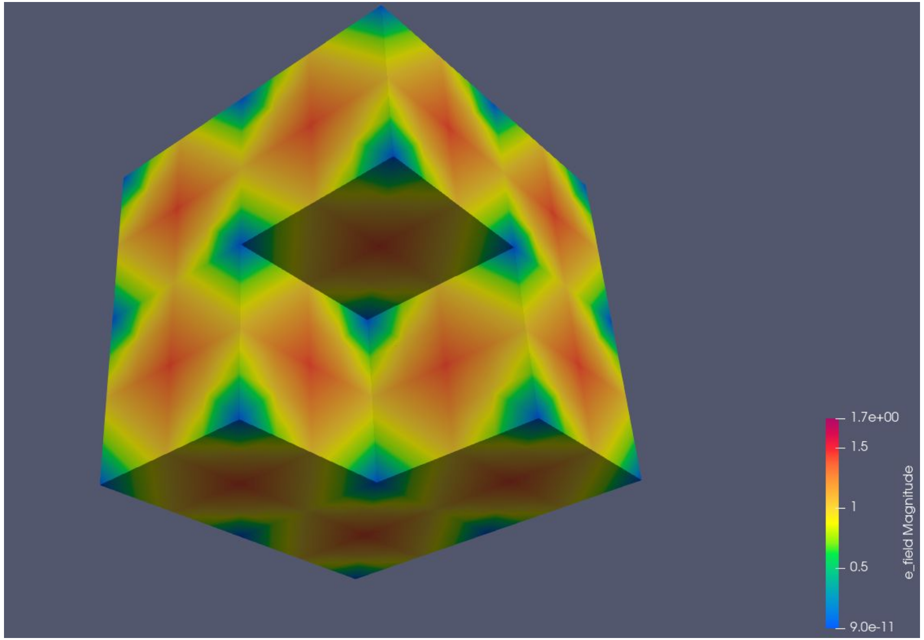


Definite Maxwell Example

UK Atomic Energy Authority

```
# Definite Maxwell problem solved with Nedelec elements of the first kind
# based on MFEM Example 3.

[Mesh]
  type = MFEMMesh
  file = gold/small_fichera.mesh
[]

[Problem]
  type = MFEMProblem
[]

[FESpaces]
  [HCurlFESpace]
    type = MFEMFESpace
    fec_type = ND
    fec_order = FIRST
  []
[]

[Variables]
  [e_field]
    type = MFEMVariable
    fespace = HCurlFESpace
  []
[]

[Preconditioner]
  [ams]
    type = MFEMHypreAMS
    fespace = HCurlFESpace
  []
[]

[Solver]
  type = MFEMHypreGMRES
  preconditioner = ams
  l_tol = 1e-6
[]

[Executioner]
  type = MFEMSteady
  device = cpu
[]
```

$$\vec{\nabla} \times \vec{\nabla} \times \vec{E} + \vec{E} = \vec{f}$$



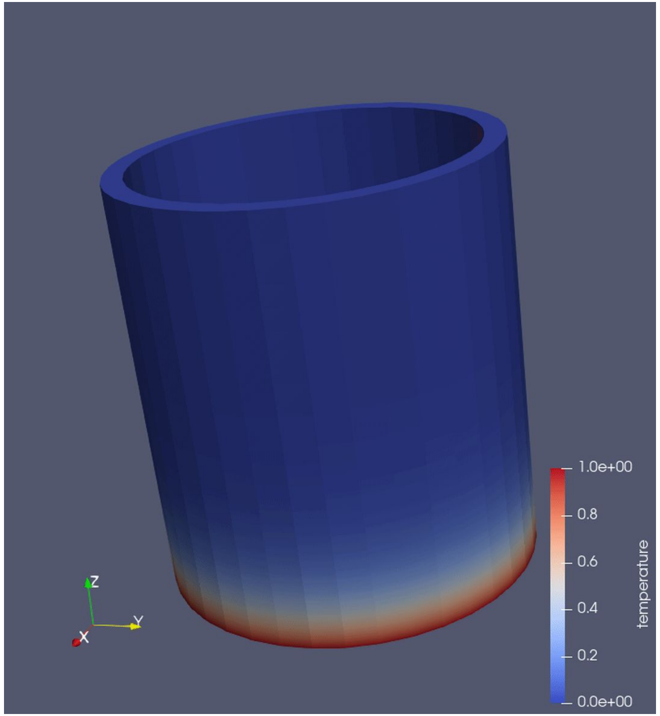
Transient Example – Heat Conduction

UK Atomic Energy Authority

$$\rho c_p \frac{\partial T}{\partial t} - \vec{\nabla} \cdot \left(k \vec{\nabla} T \right) = 0$$

```
[Materials]
  [Substance]
    type = MFEMGenericConstantMaterial
    prop_names = 'thermal_conductivity volumetric_heat_capacity'
    prop_values = '1.0 1.0'
  []
[]

[Coefficients]
  [TopValue]
    type = MFEMFunctionCoefficient
    function = value_top
  []
  [BottomValue]
    type = MFEMFunctionCoefficient
    function = value_bottom
  []
[]

[Kernels]
  [diff]
    type = MFEMDiffusionKernel        You, 2 months ago • Add assem
    variable = temperature
    coefficient = thermal_conductivity
  []
  [dT_dt]
    type = MFEMTimeDerivativeMassKernel
    variable = temperature
    coefficient = volumetric_heat_capacity
  []
[]
```

```
[Executioner]
  type = MFEMTransient
  device = cpu
  dt = 0.25
  start_time = 0.0
  end_time = 1.0
[]
```

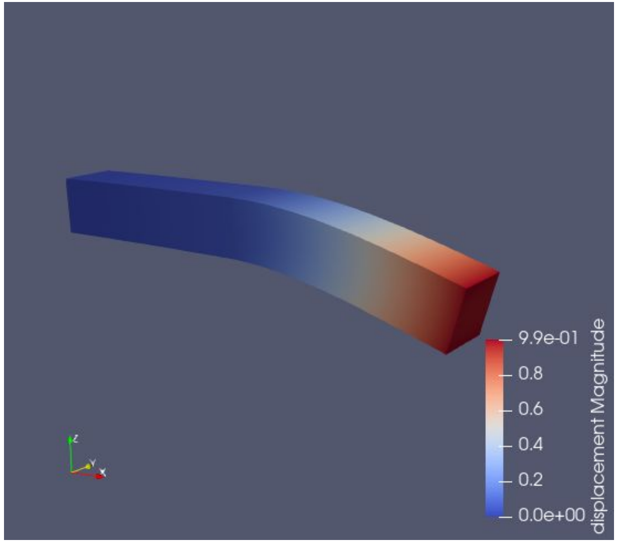


Upcoming – Mesh Updates

UK Atomic Energy Authority

```
[Mesh]
  type = MFEMMesh
  file = gold/beam-tet.mesh
  uniform_refine = 2
  displacement = "displacement"
[]

[Problem]
  type = MFEMProblem
[]

[FESpaces]
  [H1FESpace]
    type = MFEMFESpace
    fec_type = H1
    fec_order = FIRST
    vdim = 3
  []
[]

[Variables]
  [displacement]
    type = MFEMVariable
    fespace = H1FESpace
  []
[]
```

```
[Materials]
  [Rigidium]
    type = MFEMGenericConstantMaterial
    prop_names = 'lambda mu'
    prop_values = '50.0 50.0'
    block = 1
  []
  [Bendium]
    type = MFEMGenericConstantMaterial
    prop_names = 'lambda mu'
    prop_values = '1.0 1.0'
    block = 2
  []
[]

[Kernels]
  [diff]
    type = MFEMLinearElasticityKernel
    variable = displacement
    lambda = lambda
    mu = mu
  []
[]
      You, 6 minutes ago • Uncommitted char
[VectorCoefficients]
  [FixedValue]
    type = MFEMVectorConstantCoefficient
    value_x = 0.0
    value_y = 0.0
    value_z = 0.0
  []
  [PullDownValue]
    type = MFEMVectorConstantCoefficient
    value_x = 0.0
    value_y = 0.0
    value_z = -0.01
  []
[]

[BCs]
  [dirichlet]
    type = MFEMVectorDirichletBC
    variable = displacement
    boundary = '1'
    vector_coefficient = FixedValue
  []
  [pull_down]
    type = MFEMVectorBoundaryIntegratedBC
    variable = displacement
    boundary = '2'
    vector_coefficient = PullDownValue
  []
```

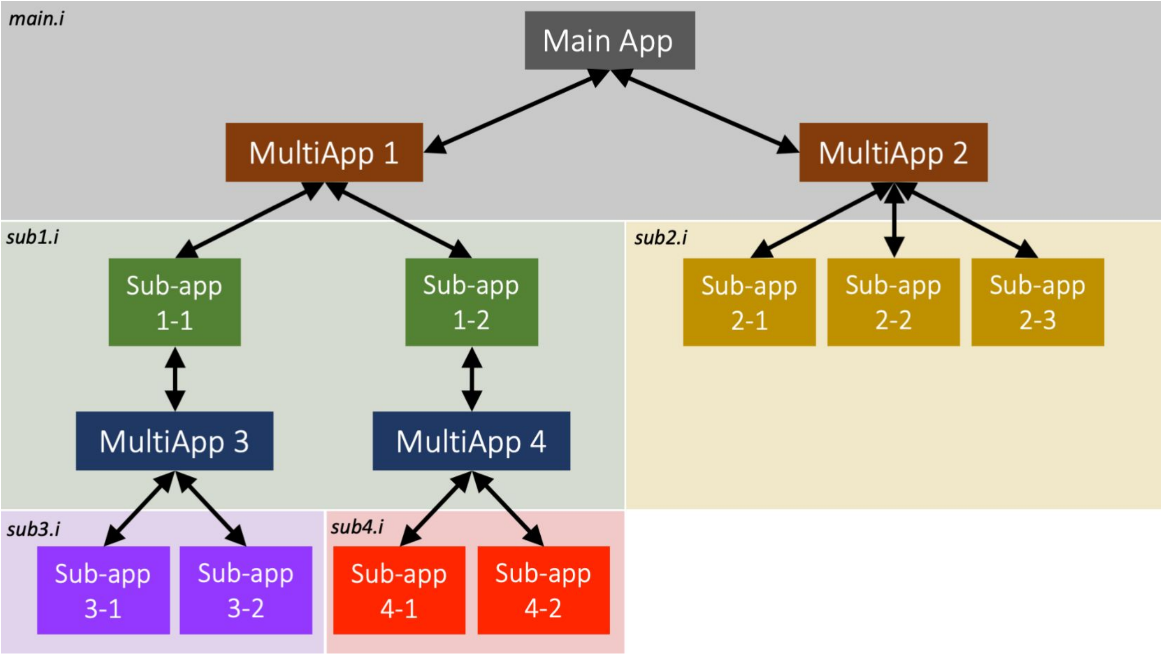


MFEM Example 2 run using Platypus

- Support for mesh updates coming soon!
- Intended for AMR and transient mechanical problems

Upcoming – MOOSE/MFEM Transfers

UK Atomic Energy Authority

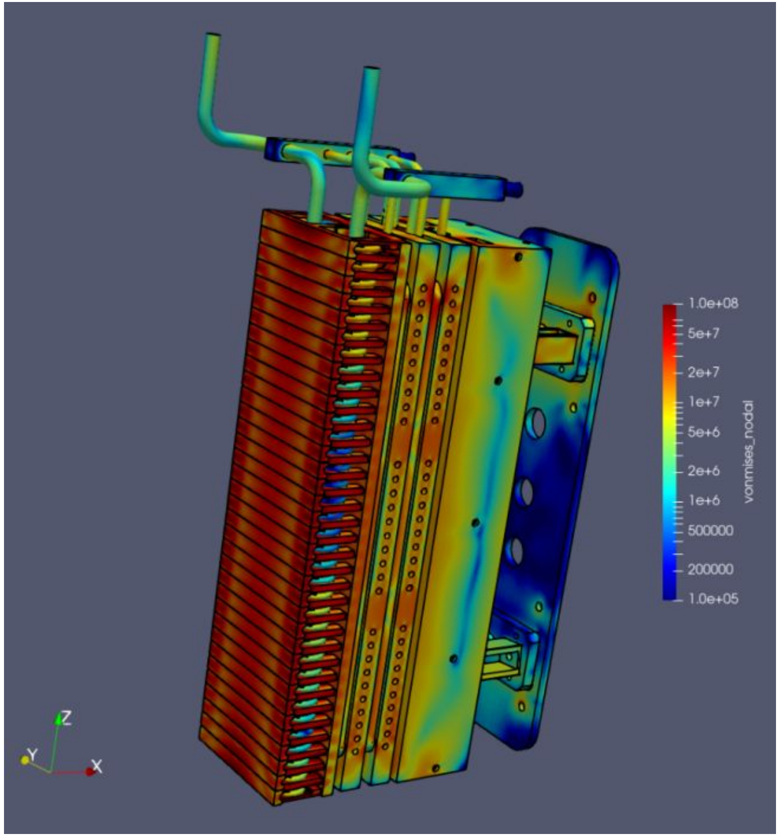


https://mooseframework.inl.gov/syntax/MultiApps/

- MOOSE Transfers with MFEM problems in Platypus needed to facilitate interoperability with existing MOOSE modules.
- Handles the transfer of data from variables in one running MOOSE sub-app to another.
- libMesh to MFEM node orderings for GridFunctions/libMesh variables known from **Apollo/Hephaestus.**

Upcoming – Nonlinear Problems

UK Atomic Energy Authority



Von Mises Stresses on CHIMERA CSUT, A. Davis

- Extension of Platypus's `EquationSystem` to support `mfem::NonlinearForm` contributions to the weak form
- Exploration of using of Enzyme for AD is also planned

Summary

UK Atomic Energy Authority

- We need new scalable tools to enable engineers to carry out actionable multiphysics simulations of reactor components in the fusion environment, to qualify and de-risk designs, where experimental data will be scant.

- We are developing **Platypus** to support this, enabling the solution of large-scale FE problems on CPU or GPU in the MOOSE FE framework using MFEM as the underlying FE backend.

- Initial support for solving single-variable steady or transient linear problems has been added – first formal release coming soon!

This work has been funded by the Fusion Futures Programme. As announced by the UK Government in October 2023, Fusion Futures aims to provide holistic support for the development of the fusion sector.