Development of PyMFEM python wrapper for MFEM and scalable RF wave simulation for nuclear fusion

S. Shiraiwa (PPPL)

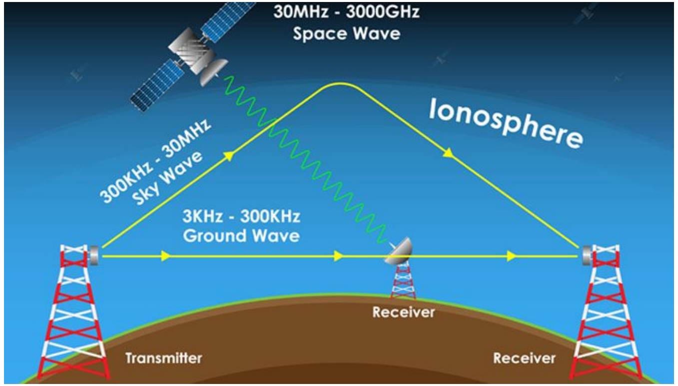
Acknowledgements: N. Bertelli (PPPL), E. H-. Kim (PPPL), J. C. Wright (MIT),
P.T.Bonoli (MIT), T. Kolev, M. Stowell (LLNL), J. Hillairet (CEA),
J. Myra (Lodestar), M. Ono (PPPL), R. Ragona (LPP), and RF SciDAC

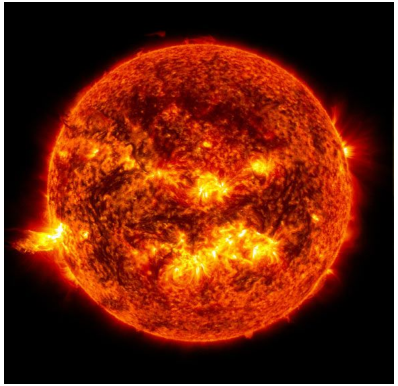
MFEM community workshop 20 Oct. 2021

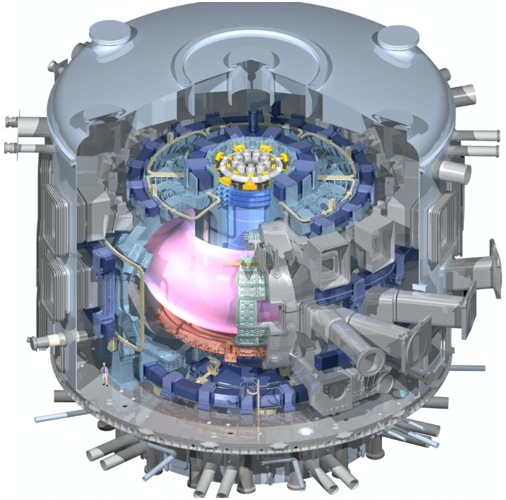
RF-SciDAC

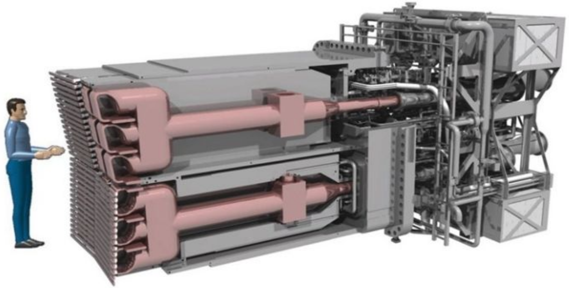
Introduction: RF waves in plasmas and magnetic fusion

- Waves in plasmas:
  - many different waves in the universe
  - described by Maxwell-Vlasov equations, or simply by Maxwell eq. at high frequency.
- Waves ares used to heat a fusion plasma (RF heating)
  - Simulation of the RF heating requires to solve the frequency domain Maxwell in an strongly anisotropic, spatially non-uniform dielectric.









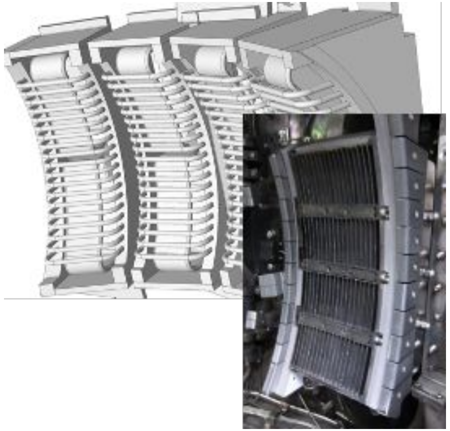
[source: NASA, Solar Dynamic Observatory, ITER]

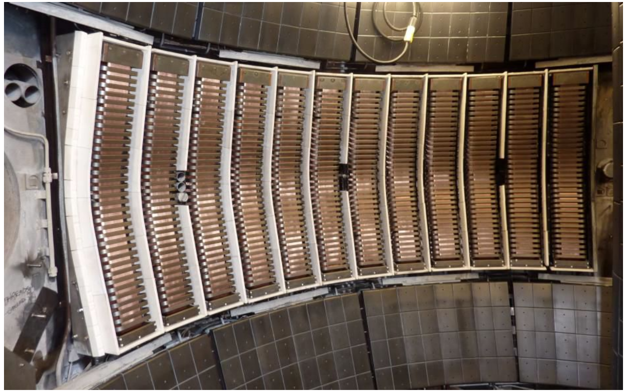
RF fullwave simulation presents multiple challenges, leading us to develop versatile FEM analysis framework

Complicated variety of antenna structures for different frequencies (50MHz - 100GHz)

Waves with very different wave lengths exists even in the same place (and spatially dispersive)

Background plasma is not uniform and turbulent and RF waves can change its characteristics.





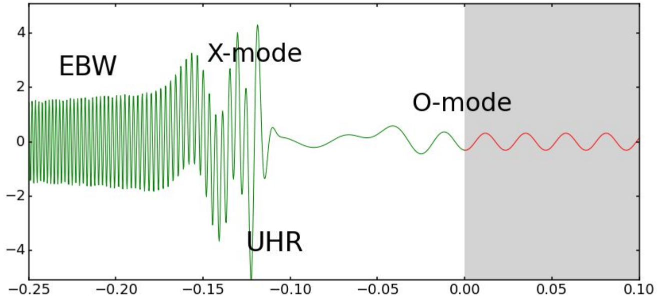
[source: Alcator C-Mod, NSTX]

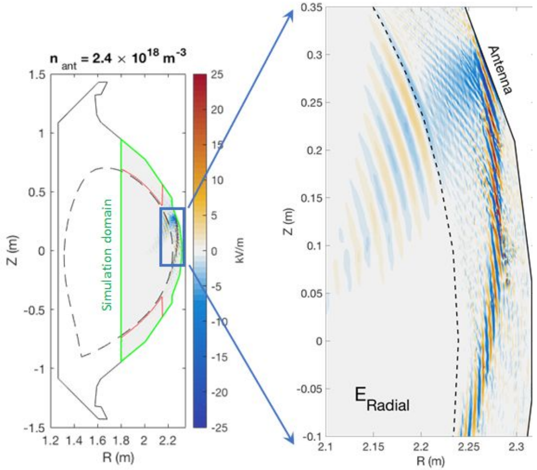
RF fullwave simulation presents multiple challenges, leading us to develop versatile FEM analysis framework

Complicated variety of antenna structures for different frequencies (50MHz - 100GHz)

Waves with very different wave lengths exists even in the same place (and spatially dispersive)

Background plasma is not uniform and turbulent and RF waves can change its characteristics.





RF fullwave simulation presents multiple challenges, leading us to develop versatile FEM analysis framework

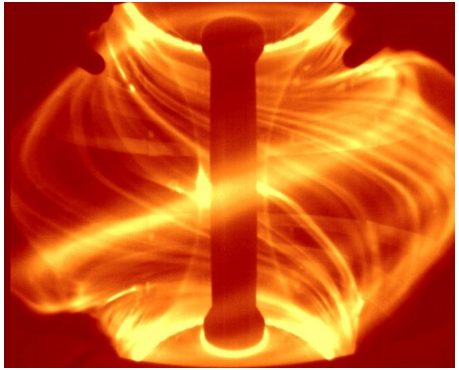
Complicated variety of antenna structures for different frequencies (50MHz - 100GHz)

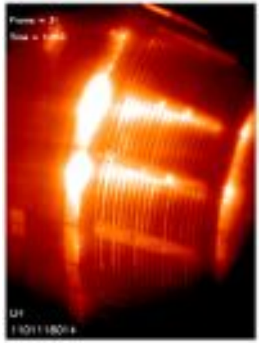
Waves with very different wave lengths exists even in the same place (and spatially dispersive)

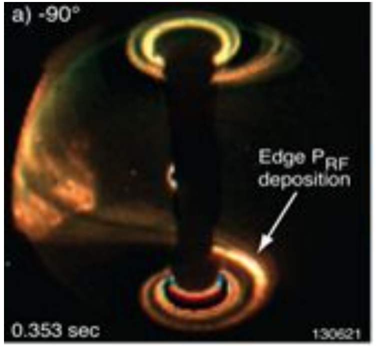
Background plasma is not uniform and turbulent and RF waves can change its characteristics.

Two track approach

- Physics model block development
- Integration for real world simulation



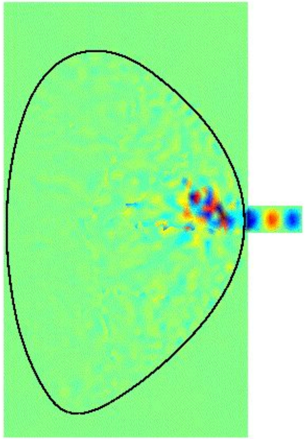




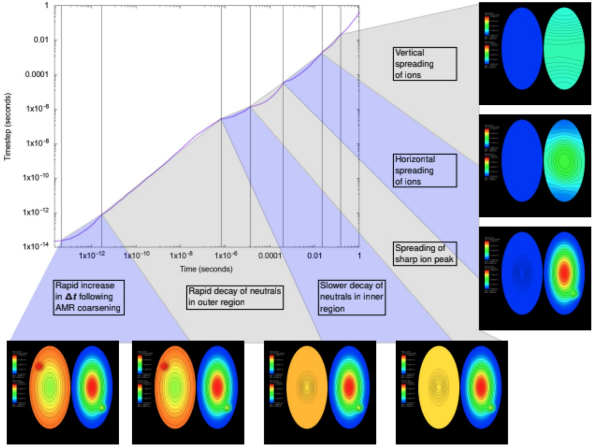
[source: MAST, Alcator C-Mod, NSTX]

RF SciDAC center adopted the MFEM library for developing the new generation of RF simulations

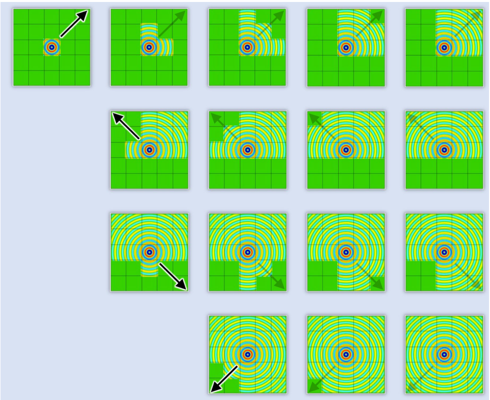
- 2D SOL transport simulation (Braginskii)
    - Time dependent solver of energy/momentum/continuity coupled equations in DG FEM
- Adaptive mesh refinement for resolving slow wave propagation
- DD-based preconditioner
- Fokker-Planck solver on GPUs



AMR for LH waves M. Hakim et. al, (2021)



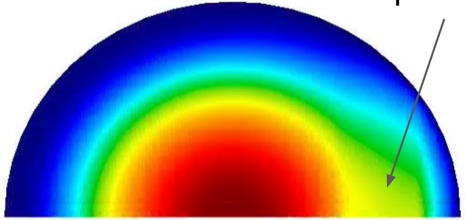
Time stepping for transport solver



2D domain sweep in DD solver

FP solver on GPU

LH plateau



$$\left(\frac{\partial f_a}{\partial t}\right)_c^b = c f_a + \vec{d}\cdot\nabla f_a + \nabla^T\cdot(E\nabla f_a) + h(v_x)\frac{\partial^2 f_a}{\partial v_x^2}$$

Petra-M: towards whole device scale RF fullwave simulation

Petra-M : Physics equation translator for MFEM

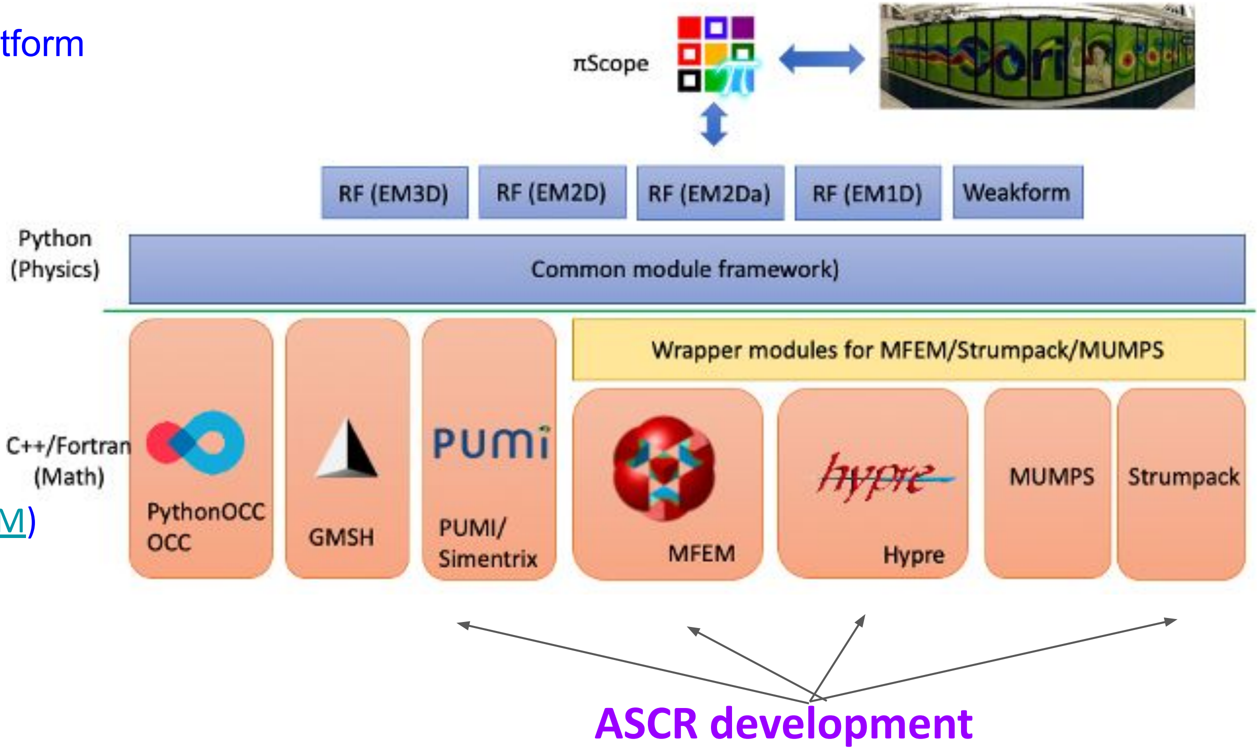
Integrated multi-physics FEM platform

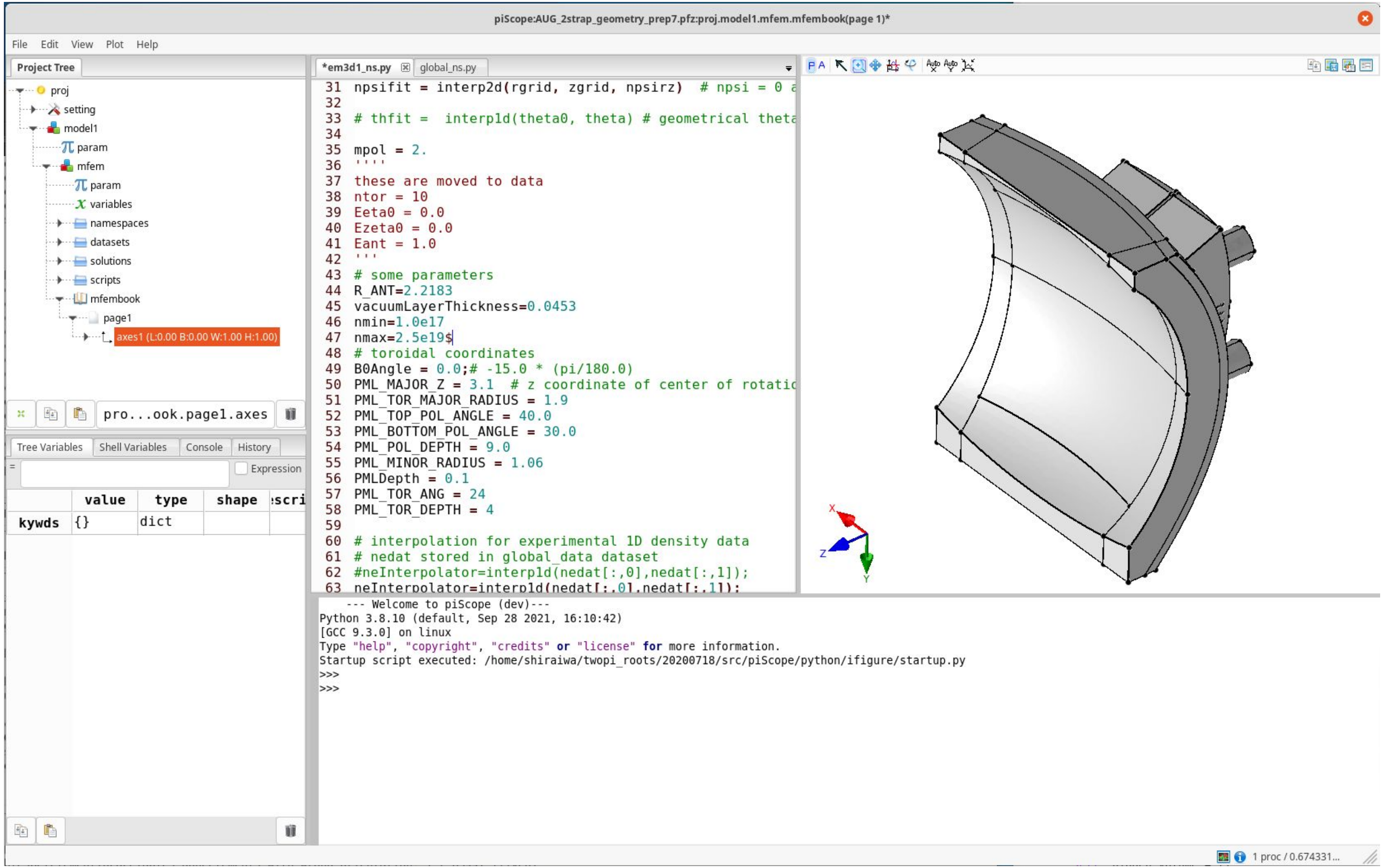
- Geometry creation
- Mesh generation
- FEM assembly and solve
- Visualization

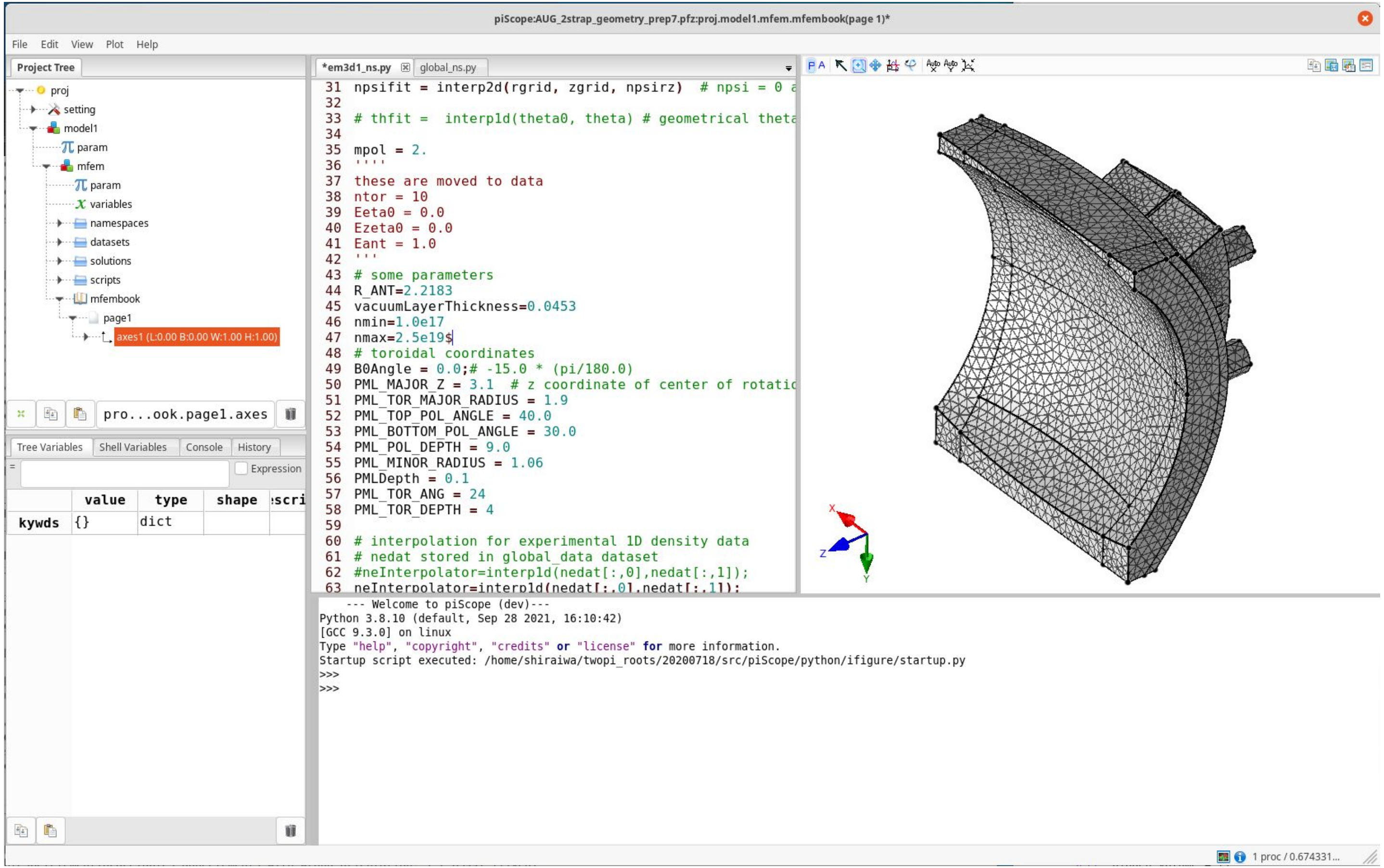
Solves user defined PDEs

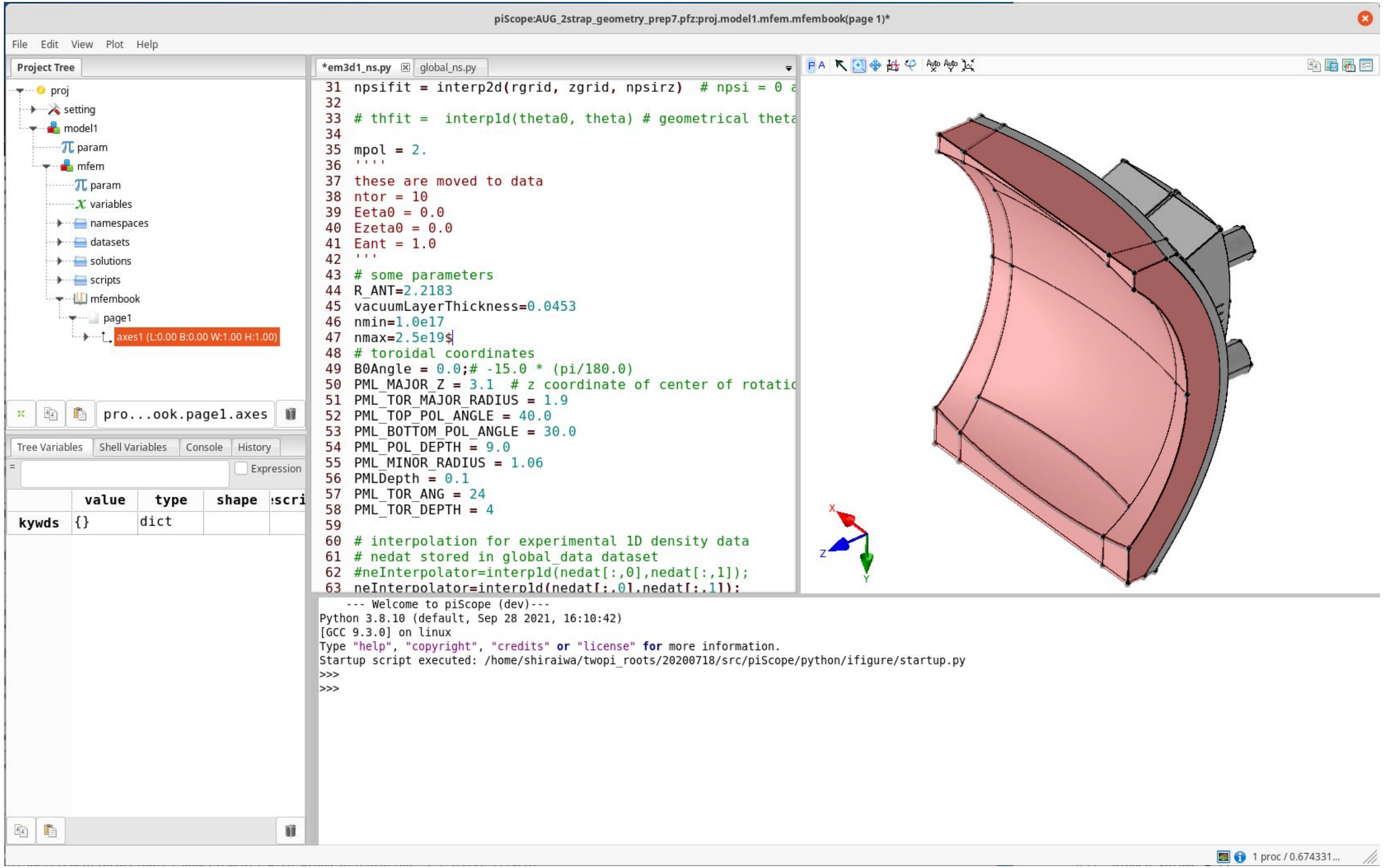
Scales from laptop to cluster
(https://github.com/piScope/Petra-M)

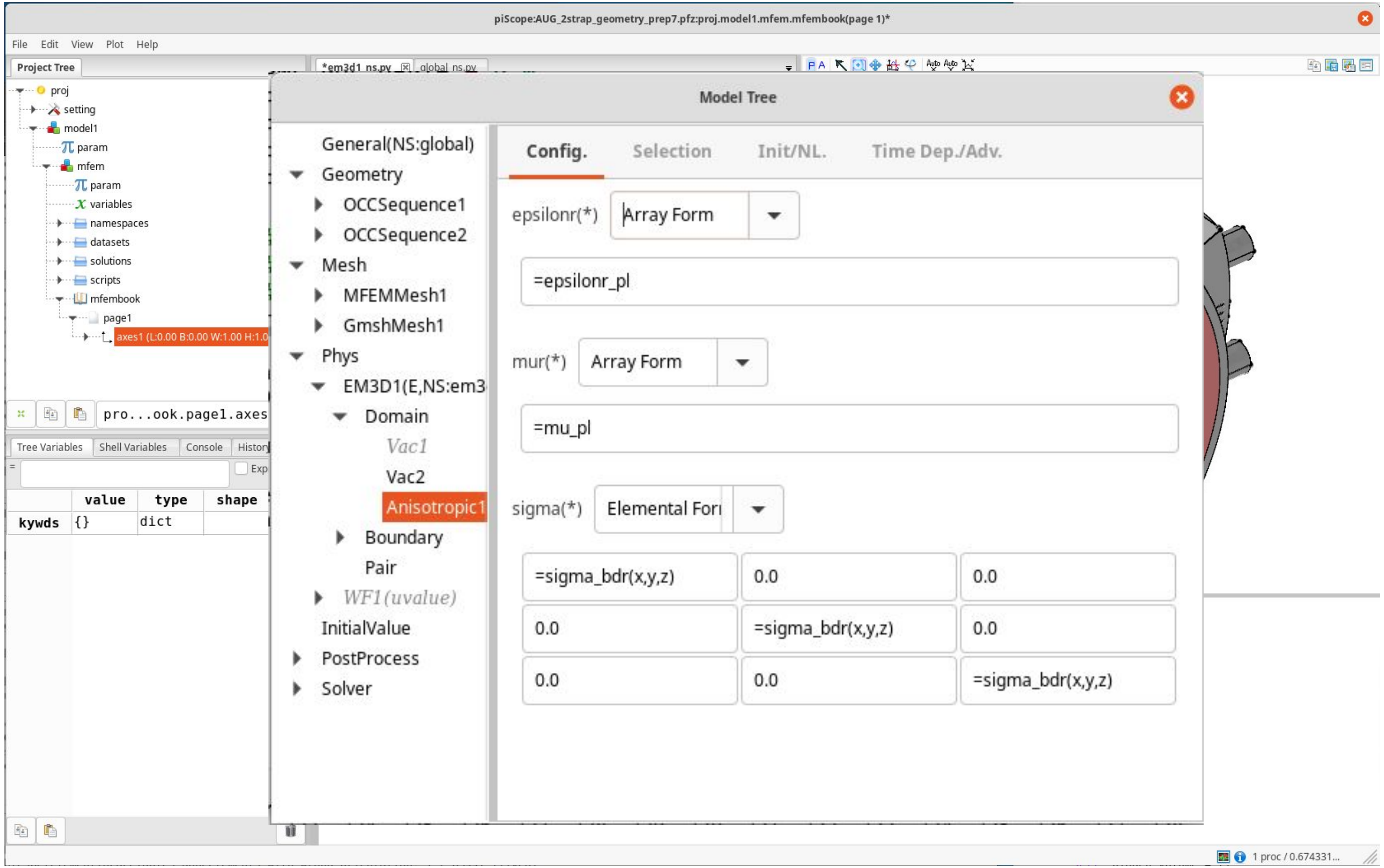
Built on OOS and ASCR developed
high-performance libraries

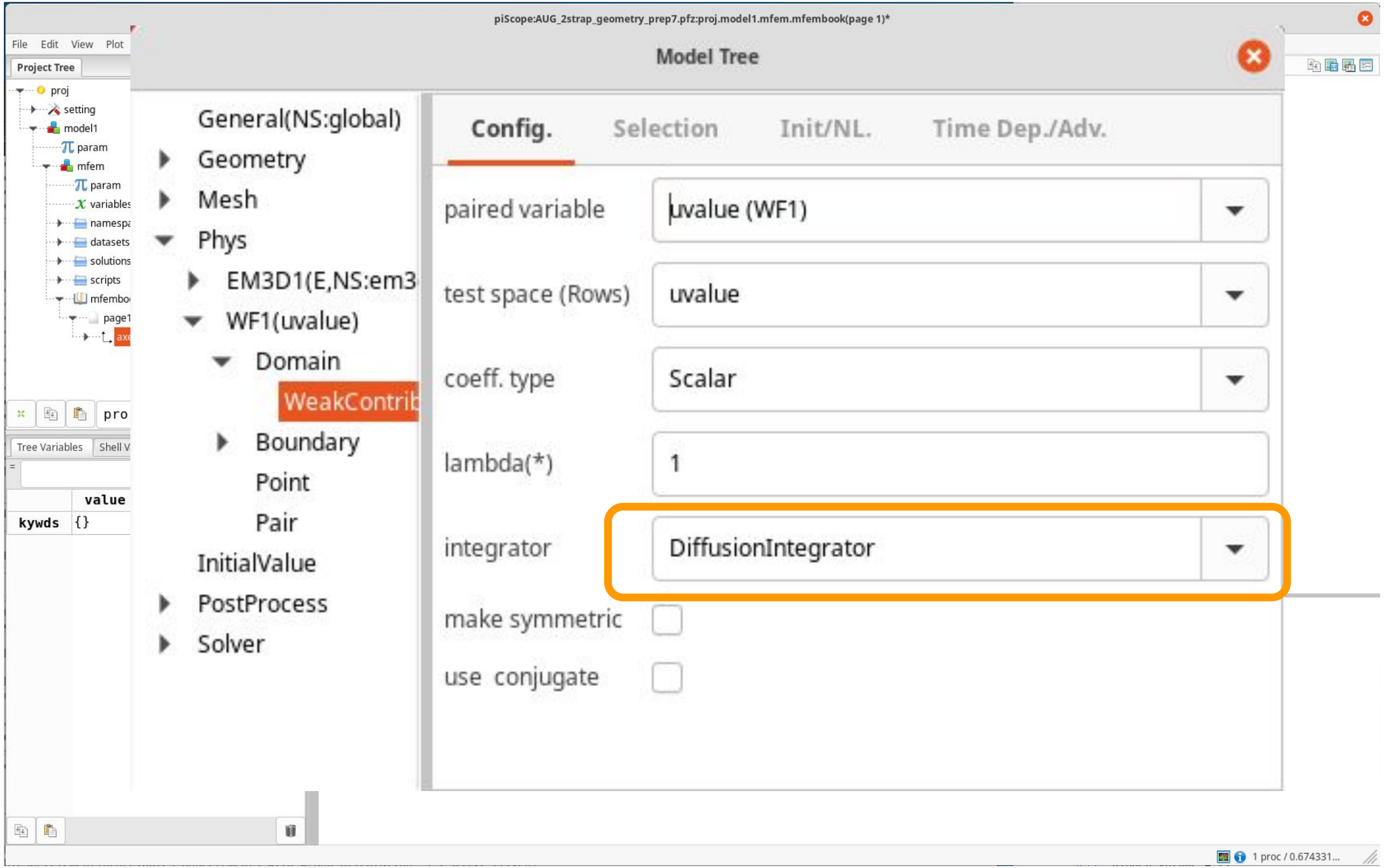


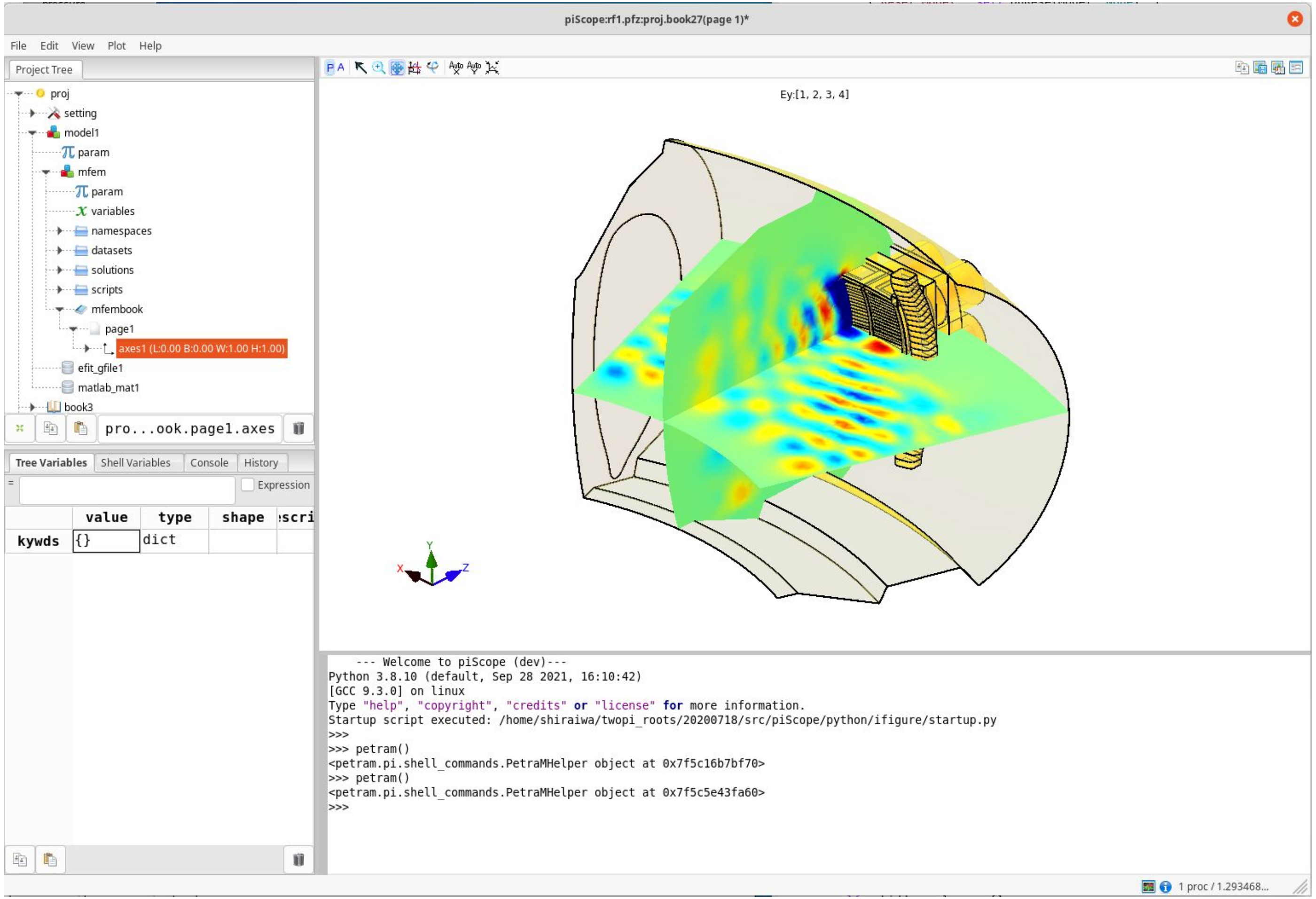

piScope:AUG_2strap_geometry_prep7.pfz:proj.model1.mfem.mfembook(page 1)*
File Edit View Plot Help
Project Tree
proj
setting
model1
param
mfem
param
variables
namespaces
datasets
solutions
scripts
mfembook
page1
axes1 (L:0.00 B:0.00 W:1.00 H:1.00)
pro...ook.page1.axes
Tree Variables Shell Variables Console History
Expression
value type shape
kywds {} dict
*em3d1_ns.py global_ns.py
31 npsifit = interp2d(rgrid, zgrid, npsirz) # npsi = 0 a
32
33 # thfit = interp1d(theta0, theta) # geometrical theta
34
35 mpol = 2.
36 ''''
37 these are moved to data
38 ntor = 10
39 Eeta0 = 0.0
40 Ezeta0 = 0.0
41 Eant = 1.0
42 '''
43 # some parameters
44 R_ANT=2.2183
45 vacuumLayerThickness=0.0453
46 nmin=1.0e17
47 nmax=2.5e19$
48 # toroidal coordinates
49 B0Angle = 0.0;# -15.0 * (pi/180.0)
50 PML_MAJOR_Z = 3.1 # z coordinate of center of rotatio
51 PML_TOR_MAJOR_RADIUS = 1.9
52 PML_TOP_POL_ANGLE = 40.0
53 PML_BOTTOM_POL_ANGLE = 30.0
54 PML_POL_DEPTH = 9.0
55 PML_MINOR_RADIUS = 1.06
56 PMLDepth = 0.1
57 PML_TOR_ANG = 24
58 PML_TOR_DEPTH = 4
59
60 # interpolation for experimental 1D density data
61 # nedat stored in global_data dataset
62 #neInterpolator=interp1d(nedat[:,0],nedat[:,1]);
63 neInterpolator=interp1d(nedat[:,0],nedat[:,1]);
X
Z
Y
--- Welcome to piScope (dev)---
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /home/shiraiwa/twopi_roots/20200718/src/piScope/python/ifigure/startup.py
>>>
>>>
1 proc / 0.674331...


piScope:AUG_2strap_geometry_prep7.pfz:proj.model1.mfem.mfembook(page 1)*
File Edit View Plot Help
Project Tree
proj
setting
model1
param
mfem
param
variables
namespaces
datasets
solutions
scripts
mfembook
page1
axes1 (L:0.00 B:0.00 W:1.00 H:1.00)
pro...ook.page1.axes
Tree Variables Shell Variables Console History
Expression
value type shape
kywds {} dict
*em3d1_ns.py global_ns.py
npsifit = interp2d(rgrid, zgrid, npsirz) # npsi = 0 a
# thfit = interp1d(theta0, theta) # geometrical theta
mpol = 2.
''''
these are moved to data
ntor = 10
Eeta0 = 0.0
Ezeta0 = 0.0
Eant = 1.0
'''
# some parameters
R_ANT=2.2183
vacuumLayerThickness=0.0453
nmin=1.0e17
nmax=2.5e19$
# toroidal coordinates
B0Angle = 0.0;# -15.0 * (pi/180.0)
PML_MAJOR_Z = 3.1 # z coordinate of center of rotatic
PML_TOR_MAJOR_RADIUS = 1.9
PML_TOP_POL_ANGLE = 40.0
PML_BOTTOM_POL_ANGLE = 30.0
PML_POL_DEPTH = 9.0
PML_MINOR_RADIUS = 1.06
PMLDepth = 0.1
PML_TOR_ANG = 24
PML_TOR_DEPTH = 4
# interpolation for experimental 1D density data
# nedat stored in global_data dataset
#neInterpolator=interp1d(nedat[:,0],nedat[:,1]);
neInterpolator=interp1d(nedat[:,0],nedat[:,1]);
X
Z
Y
--- Welcome to piScope (dev)---
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /home/shiraiwa/twopi_roots/20200718/src/piScope/python/ifigure/startup.py
>>>
>>>
1 proc / 0.674331...

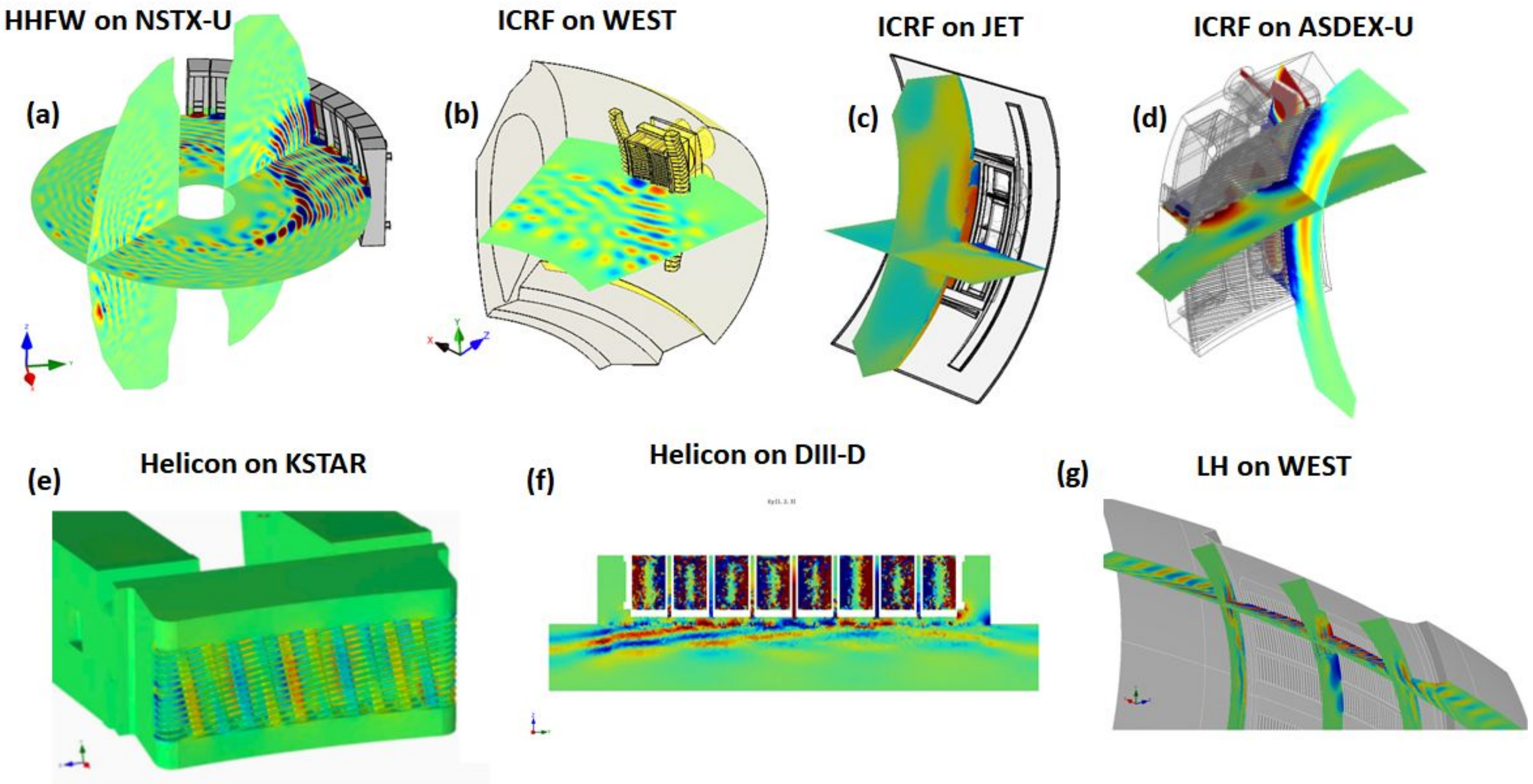

piScope:AUG_2strap_geometry_prep7.pfz:proj.model1.mfem.mfembook(page 1)*
File Edit View Plot Help
Project Tree
proj
setting
model1
param
mfem
param
variables
namespaces
datasets
solutions
scripts
mfembook
page1
axes1 (L:0.00 B:0.00 W:1.00 H:1.00)
pro...ook.page1.axes
Tree Variables Shell Variables Console History
Expression
value type shape
kywds {} dict
*em3d1_ns.py global_ns.py
31 npsifit = interp2d(rgrid, zgrid, npsirz) # npsi = 0 a
32
33 # thfit = interp1d(theta0, theta) # geometrical theta
34
35 mpol = 2.
36 ''''
37 these are moved to data
38 ntor = 10
39 Eeta0 = 0.0
40 Ezeta0 = 0.0
41 Eant = 1.0
42 '''
43 # some parameters
44 R_ANT=2.2183
45 vacuumLayerThickness=0.0453
46 nmin=1.0e17
47 nmax=2.5e19$
48 # toroidal coordinates
49 B0Angle = 0.0;# -15.0 * (pi/180.0)
50 PML_MAJOR_Z = 3.1 # z coordinate of center of rotatic
51 PML_TOR_MAJOR_RADIUS = 1.9
52 PML_TOP_POL_ANGLE = 40.0
53 PML_BOTTOM_POL_ANGLE = 30.0
54 PML_POL_DEPTH = 9.0
55 PML_MINOR_RADIUS = 1.06
56 PMLDepth = 0.1
57 PML_TOR_ANG = 24
58 PML_TOR_DEPTH = 4
59
60 # interpolation for experimental 1D density data
61 # nedat stored in global_data dataset
62 #neInterpolator=interp1d(nedat[:,0],nedat[:,1]);
63 neInterpolator=interp1d(nedat[:,0],nedat[:,1]);
X
Z
Y
--- Welcome to piScope (dev)---
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /home/shiraiwa/twopi_roots/20200718/src/piScope/python/ifigure/startup.py
>>>
>>>
1 proc / 0.674331...


piScope:AUG_2strap_geometry_prep7.pfz:proj.model1.mfem.mfembook(page 1)*
File
Edit
View
Plot
Help
Project Tree
proj
setting
model1
param
mfem
param
variables
namespaces
datasets
solutions
scripts
mfembook
page1
axes1 (L:0.00 B:0.00 W:1.00 H:1.0
pro...ook.page1.axes
Tree Variables
Shell Variables
Console
value
type
shape
kywds
{}
dict
Model Tree
General(NS:global)
Geometry
OCCSequence1
OCCSequence2
Mesh
MFEMMesh1
GmshMesh1
Phys
EM3D1(E,NS:em3
Domain
Vac1
Vac2
Anisotropic1
Boundary
Pair
WF1(uvalue)
InitialValue
PostProcess
Solver
Config.
Selection
Init/NL.
Time Dep./Adv.
epsilonr(*)
Array Form
=epsilonr_pl
mur(*)
Array Form
=mu_pl
sigma(*)
=sigma_bdr(x,y,z)
0.0
0.0
0.0
=sigma_bdr(x,y,z)
0.0
0.0
0.0
=sigma_bdr(x,y,z)
1 proc / 0.674331...


piScope:AUG_2strap_geometry_prep7.pfz:proj.model1.mfem.mfembook(page 1)*
File Edit View Plot
Project Tree
proj
setting
model1
param
mfem
param
variables
datasets
scripts
Tree Variables
value
kywds {}
Model Tree
General(NS:global)
Geometry
Mesh
Phys
WF1(uvalue)
Domain
Boundary
Point
Pair
InitialValue
PostProcess
Solver
Config.
Selection
Init/NL.
Time Dep./Adv.
paired variable
uvalue (WF1)
test space (Rows)
uvalue
coeff. type
Scalar
lambda(*)
1
integrator
DiffusionIntegrator
make symmetric
use conjugate
1 proc / 0.674331...

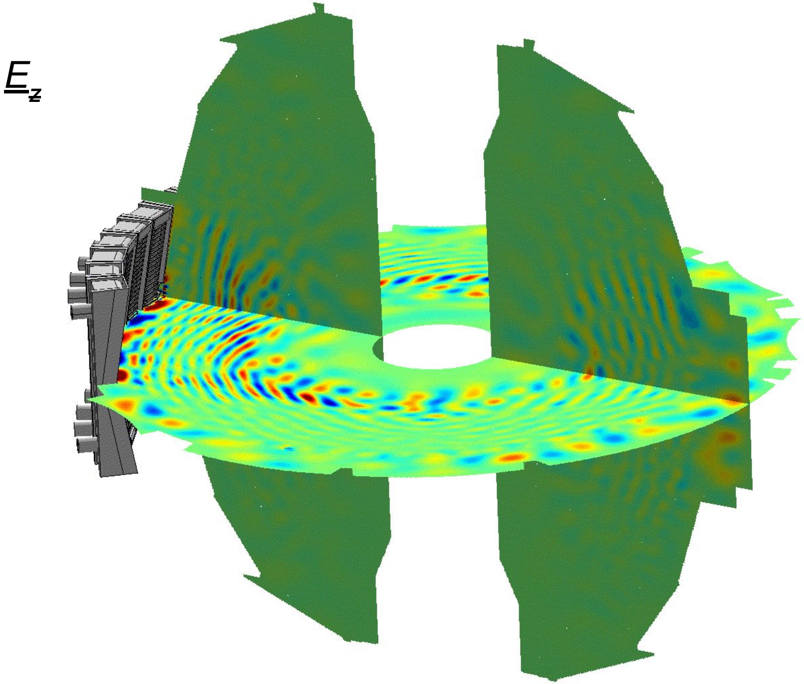

piScope:rf1.pfz:proj.book27(page 1)*
File
Edit
View
Plot
Help
Project Tree
proj
setting
model1
param
mfem
param
variables
namespaces
datasets
solutions
scripts
mfembook
page1
axes1 (L:0.00 B:0.00 W:1.00 H:1.00)
efit_gfile1
matlab_mat1
book3
pro...ook.page1.axes
Tree Variables
Shell Variables
Console
History
Expression
value
type
shape
kywds
{}
dict
Ey:[1, 2, 3, 4]
Y
X
Z
--- Welcome to piScope (dev)---
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /home/shiraiwa/twopi_roots/20200718/src/piScope/python/ifigure/startup.py
>>>
>>> petram()
<petram.pi.shell_commands.PetraMHelper object at 0x7f5c16b7bf70>
>>> petram()
<petram.pi.shell_commands.PetraMHelper object at 0x7f5c5e43fa60>
>>>
1 proc / 1.293468...

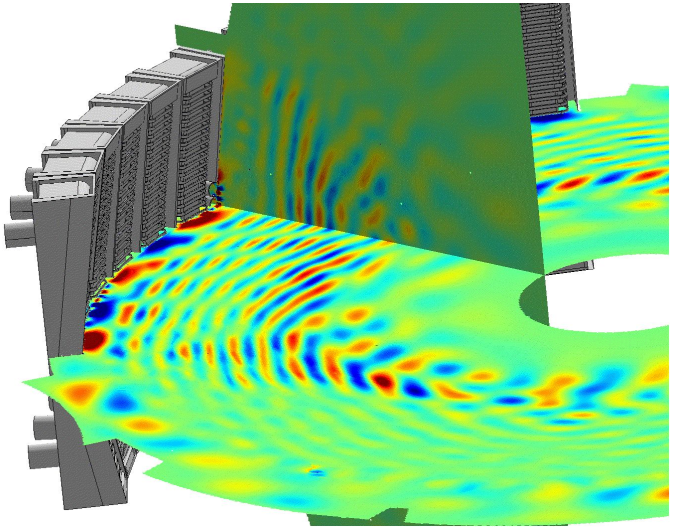
Petra-M has applied on variety of fusion devices worldwide



First fully resolved 3D HHFW field on NSTX-U

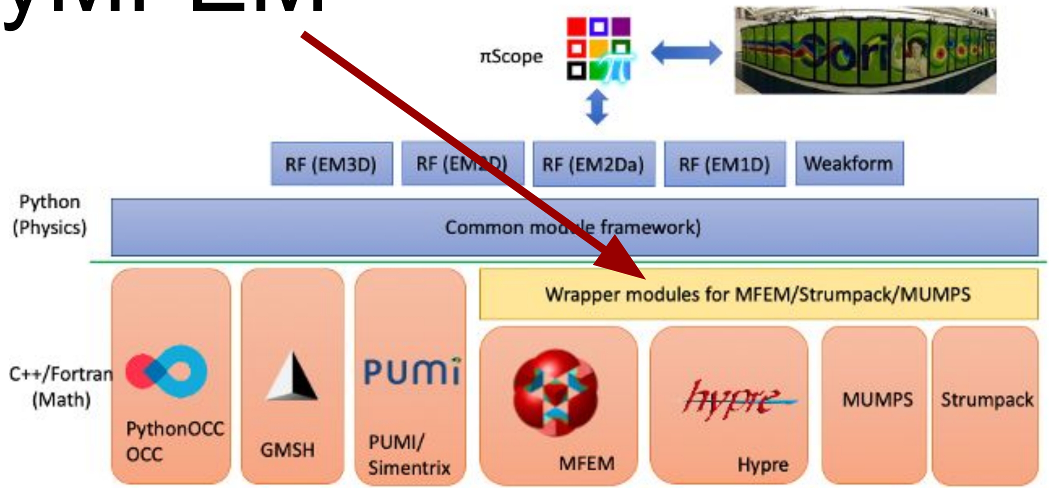
Bt = 1T, n_{e0} = $5x10^{19}m^{-3}$
150 º phasing
Cold plasma dielectric





- Obtained using 4th order basis functions.
- 50M DoFs at 4th order basis. λ/L ~ 15 is close to what is required for resolving ICRF wave field on ITER.

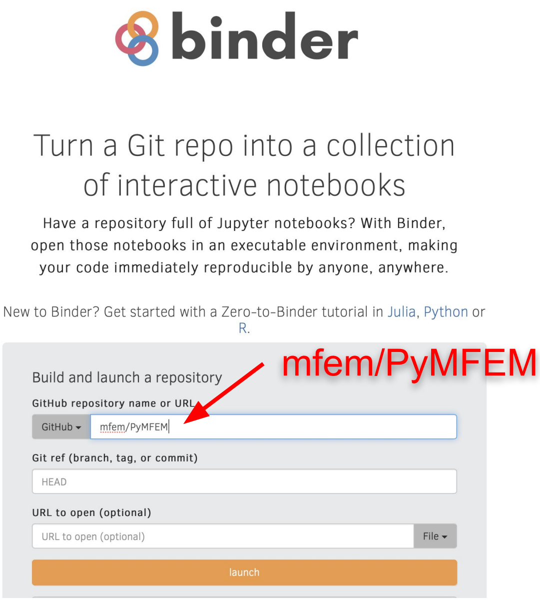
PyMFEM


πScope
RF (EM3D)
RF (EM2D)
RF (EM2Da)
RF (EM1D)
Weakform
Python
(Physics)
Common module framework)
Wrapper modules for MFEM/Strumpack/MUMPS
C++/Fortran
(Math)
PythonOCC
OCC
GMSH
PUMI
PUMI/
Simentrix
MFEM
hypre
Hypre
MUMPS
Strumpack

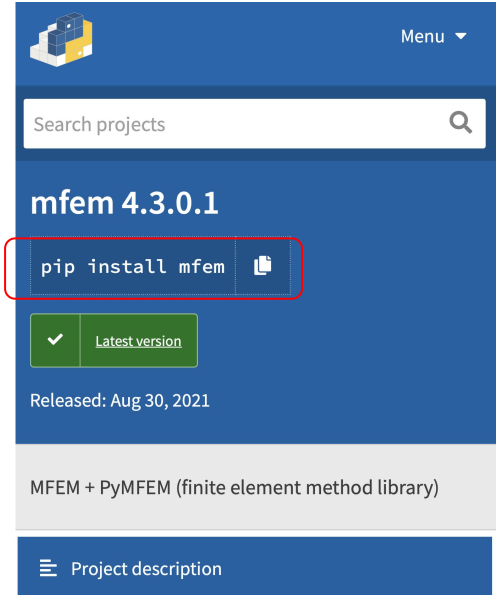
PyMFEM: Python binding for MFEM

Python binding is good for many things including

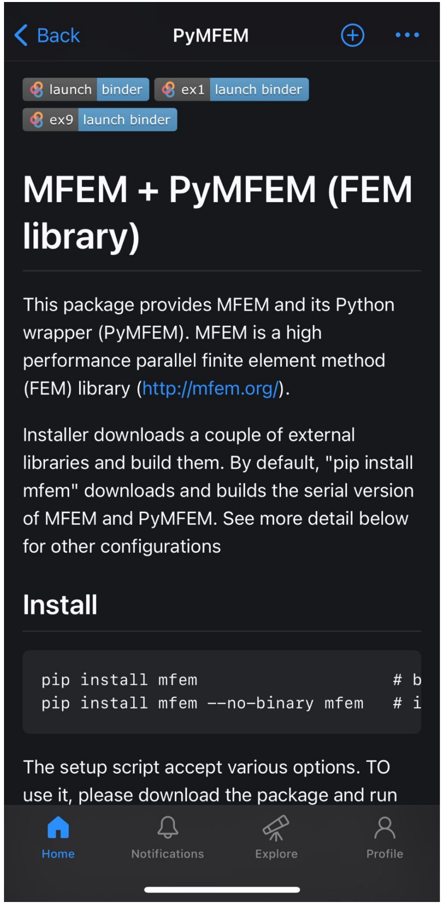
- Prototyping/rapid app development
- Education/demonstration
- Analysis using MFEM simulation output
- Still can perform a large scales computing on leading class clusters



http://mybinder.org



http://pypi.org



http://github.com

PyMFEM allows for writing an MFEM application w/o C++

- Provide convenient access to MFEM class objects from Python
- Extend MFEM to support additional features for Python user
  - Object construction/data access using numpy array
  - Numba based coefficient
  - Distributed hypre Matrix construction from scipy.sparse
- Attempt to support ever expanding list of MFEM features as much as possible (using setup.py options)
  - MPI parallelism
  - GPU
  - libCEED
- Mainly developed by one person at PPPL + robots (Github actions)
  - Help is more than welcome!

```cpp
Device device(device_config);
device.Print();

Mesh mesh(mesh_file, 1, 1);
int dim = mesh.Dimension();
```

ex1.cpp

```python
device = mfem.Device(device)
device.Print()

mesh = mfem.Mesh(meshfile, 1, 1)
dim = mesh.Dimension()
```

ex1.py

In many cases, C++ and Python codes are nearly identical except for language syntax.

PyMFEM uses SWIG to generate most of wrapper codes.

SWIG creates a Python module for each C++ header

- Input: C/C++ header file (*.hpp) + wrapper recipe file (*.i)
- Output: C extension for Python (*_wrap.cxx) and Python proxy module (*.py)

Almost automatic, but the recipe file needs to be prepared well so that generated wrapper becomes Python friendly.

- C array to numpy.array
- Pointer argument as a return value

MyModule.hpp

```
class MyClass {
  private:
     int _hidden;
  Public:
     void function(int x);
};
```

MyModule.i

```
%module(package=”xxx”)
MyClass

%include “MyModulue.hpp”
```

MyModule_wrap.cxx

MyModule.py

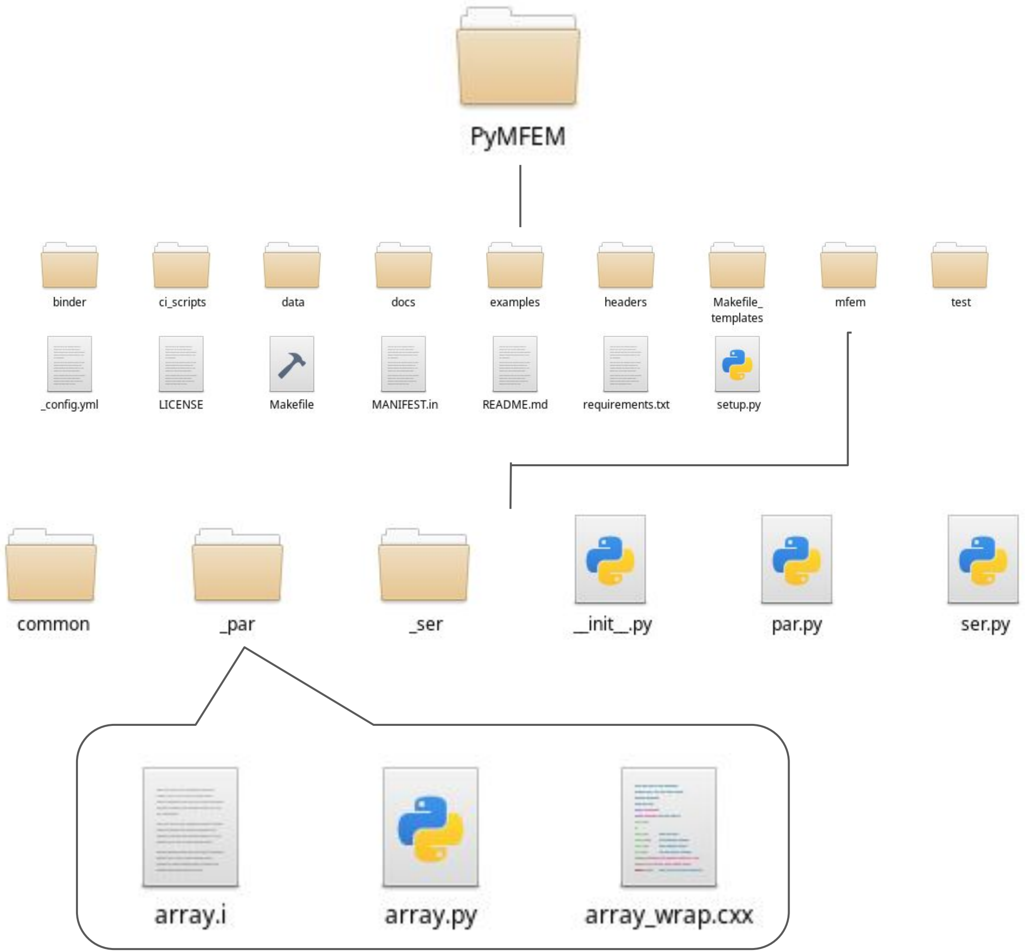
PyMFEM module structure

Serial (mfem._ser)

- Standalone (no external dependency)
- MyBinder uses mfem.ser
- Loaded to mfem.ser namespace
  - import mfem.ser as mfem

Parallel (mfem._par)

- Built with Hypre and METIS
- Loaded to mfem.par namespace
  - Import mfem.par as mfem



Updating PyMFEM wrapper codes takes a few steps

Case (1) API changes in MFEM. ex) a new overloaded class method

- Regenerating a wrapper
  - python setup.py install --swig
  - python setup.py install --swig --with-parallel
- Note, the recipe file (*.i) may need an update.
- Inform it in GitHub!

Case (2) New header file is added

- Add an interface recipe file (*.i) for each *.hpp
  - Copy a short *.i as template and modify it
  - SWIG Doc. http://www.swig.org/Doc4.0/index.html
- Add a new module to setup.py
  - mfem/_ser/setup.py
  - mfem/_par/setup.py
- Edit mfem/ser.py and mfem/par.py to load a new module

26 lines (21 sloc) | 599 Bytes

```
%module(package="mfem._ser") triangle
%{
#include  "mfem.hpp"
#include "mesh/triangle.hpp"
#include "numpy/arrayobject.h"
%}

%init %{
import_array();
%}
%include "exception.i"
%import "fe.i"
%import "element.i"
%include "../common/typemap_macros.i"
%include "../common/exception.i"

LIST_TO_INTARRAY_IN(const int *ind, 2)
INTARRAY_OUT_TO_TUPLE(int *GetVertices, 2)

%include "../common/deprecation.i"
DEPRECATED_OVERLOADED_METHOD(mfem::Triangle::GetNFaces,
                             Triangle::GetNFaces(int & nFaceVertices) is deprecated,
                             len(args) == 1)


%include "mesh/triangle.hpp"
```

triangle.i

Updating PyMFEM wrapper codes takes a few steps

Case (1) API changes in MFEM. ex) a new overloaded class method

- Regenerate a wrapper
  - python setup.py install --swig
  - python setup.py install --swig --with-parallel
- Note, the recipe file (*.i) may need an update.
- Inform it in GitHub!

Case (2) New header file is added

- Add an interface recipe file (*.i) for each *.hpp
  - Copy a short *.i as template and modify it
  - SWIG Doc. http://www.swig.org/Doc4.0/index.html
- Add a new module to setup.py
  - mfem/_ser/setup.py
  - mfem/_par/setup.py
- Edit mfem/ser.py and mfem/par.py to load a new module

Module name

26 lines (21 sloc) | 599 Bytes

```
%module(package="mfem._ser") triangle
%{
#include  "mfem.hpp"
#include "mesh/triangle.hpp"
#include "numpy/arrayobject.h"
%}

%init %{
import_array();
%}
%include "exception.i"
%import "fe.i"
%import "element.i"
%include "../common/typemap_macros.i"
%include "../common/exception.i"

LIST_TO_INTARRAY_IN(const int *ind, 2)
INTARRAY_OUT_TO_TUPLE(int *GetVertices, 2)

%include "../common/deprecation.i"
DEPRECATED_OVERLOADED_METHOD(mfem::Triangle::GetNFaces,
                             Triangle::GetNFaces(int & nFaceVertices) is deprecated,
                             len(args) == 1)


%include "mesh/triangle.hpp"
```

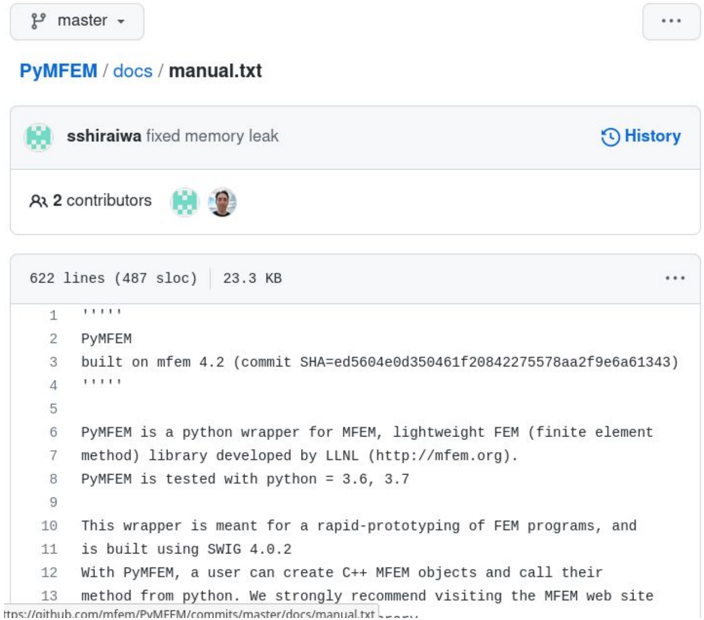
dependency

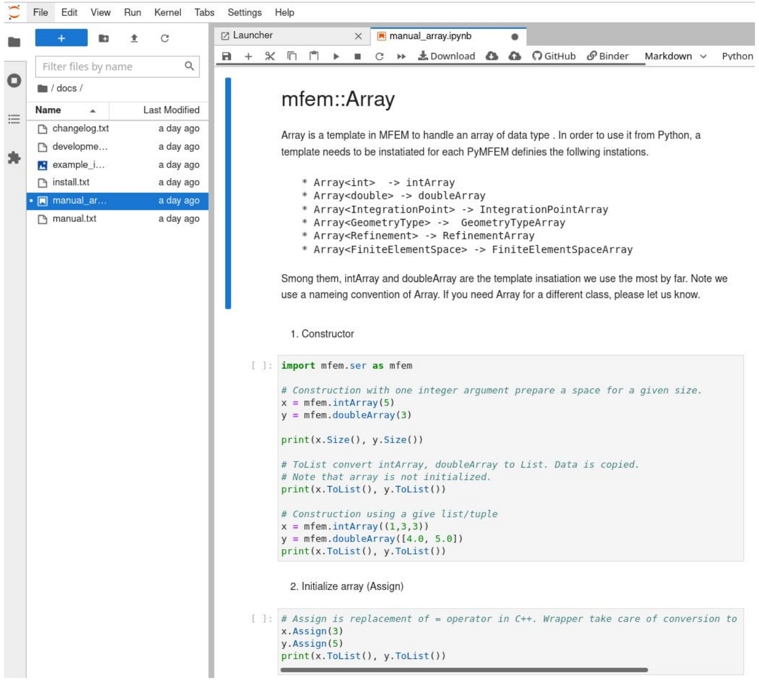
custom config.

Header file

triangle.i

Documentation still needs improvement quite a bit





Started migration to Notebook

Waiting for contribution/suggestions !