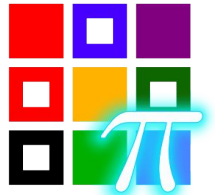


Development of PyMFEM python wrapper for MFEM and scalable RF wave simulation for nuclear fusion

S. Shiraiwa (PPPL)

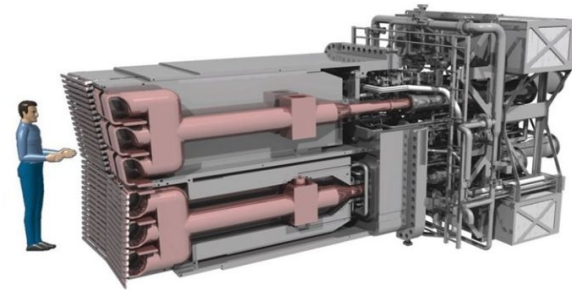
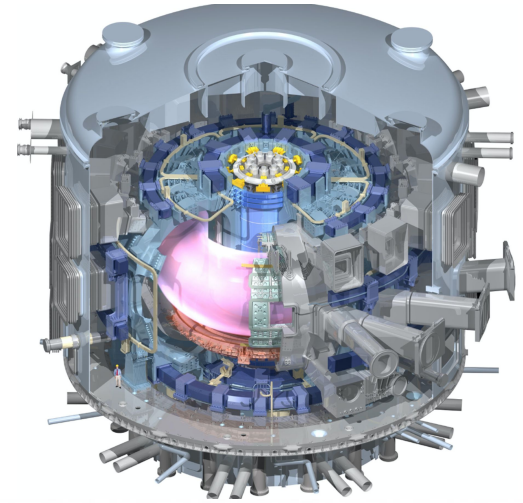
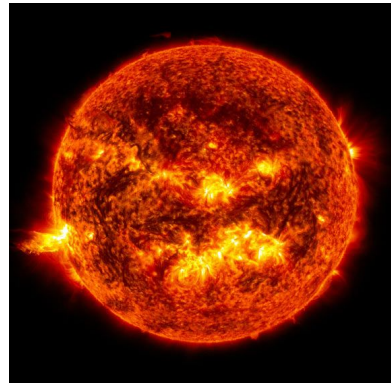
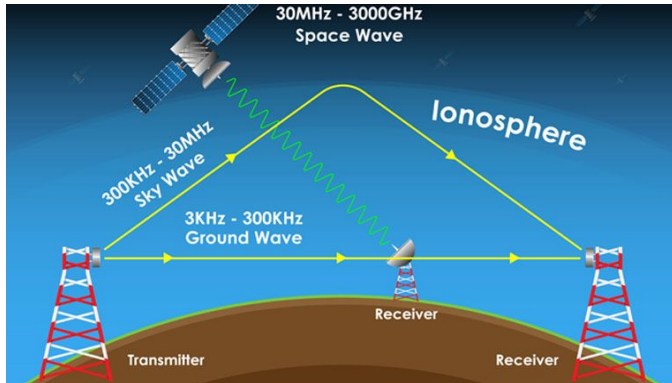
Acknowledgements: N. Bertelli (PPPL), E. H.- Kim (PPPL), J. C. Wright (MIT),
P.T.Bonoli (MIT), T. Kolev, M. Stowell (LLNL), J. Hillairet (CEA),
J. Myra (Lodestar), M. Ono (PPPL), R. Ragona (LPP), and RF SciDAC

MFEM community workshop 20 Oct. 2021



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 are 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.



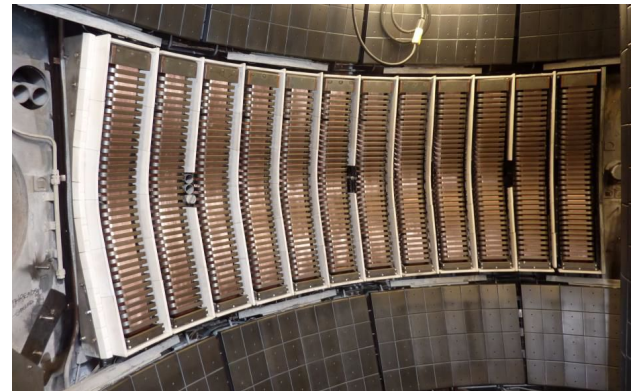
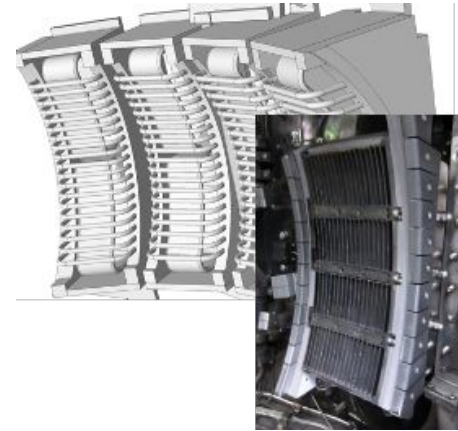
[source: NASA, Solar Dynamic Observatory, ITER]

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.

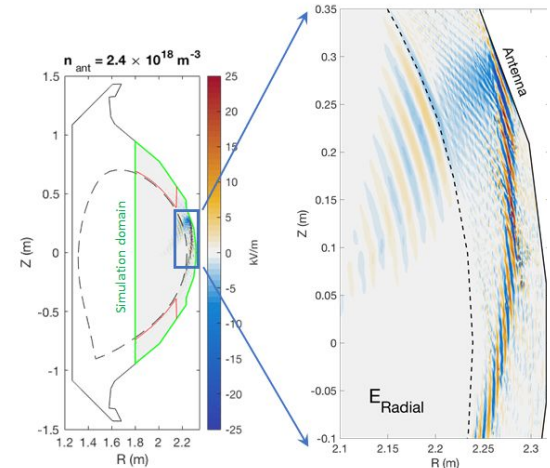
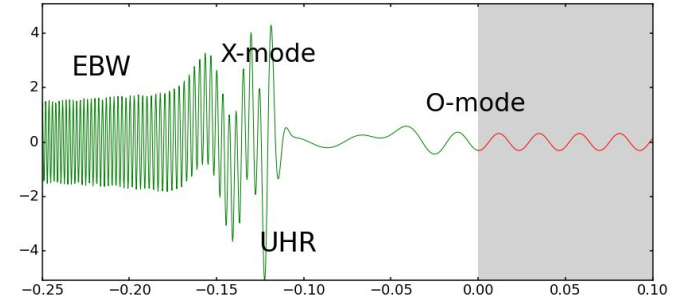


RF fullwave simulation presents multiple challenges, leading us to develop versatile FEM analysis framework

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

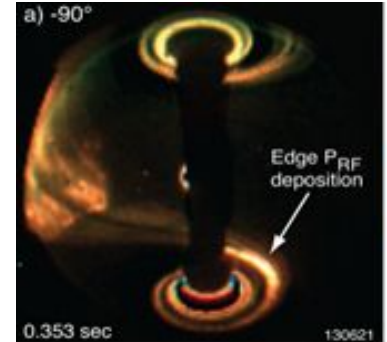
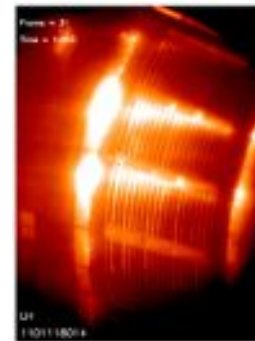
Complicated variety of antenna structures for different frequencies (50MHz - 100GHz)

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

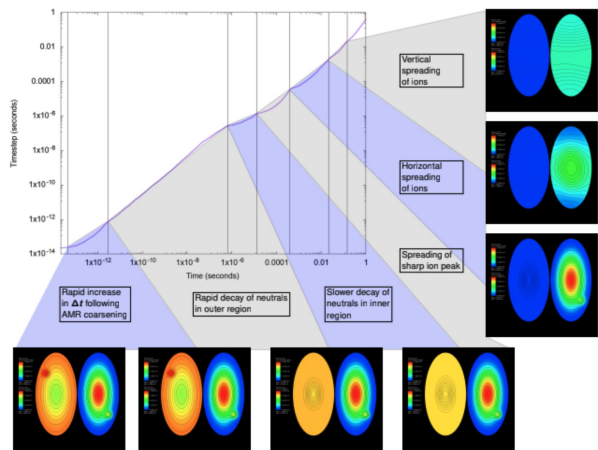
- Physics model block development
- Integration for real world simulation



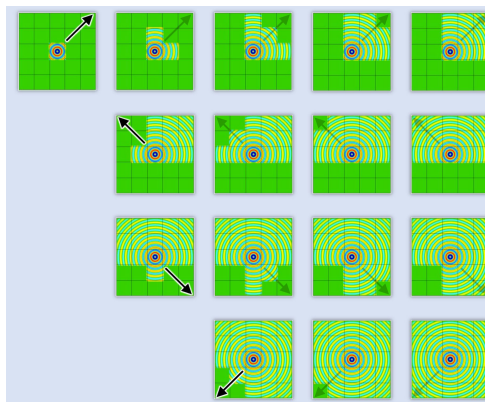
[source: MAST, Alcator C-Mod, NSTX]

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

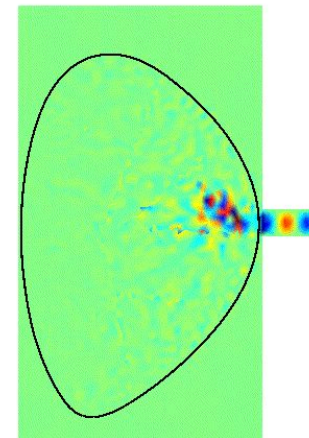
- 2D SOL transport simulation (Braginskii)
 - Time dependent solver of energy/momentum/continuity coupled equations in DG FEM
- Adaptive mesh refinement for resolving slow wave propagation
- DD-based preconditioner
- Fokker-Planck solver on GPUs



Time stepping for transport solver

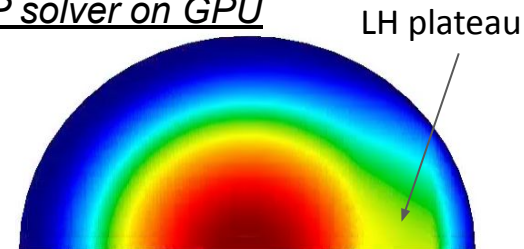


2D domain sweep in DD solver



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

FP solver on GPU



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

Petra-M: towards whole device scale RF
fullwave simulation

Petra-M : Physics equation translator for MFEM

Integrated multi-physics FEM platform

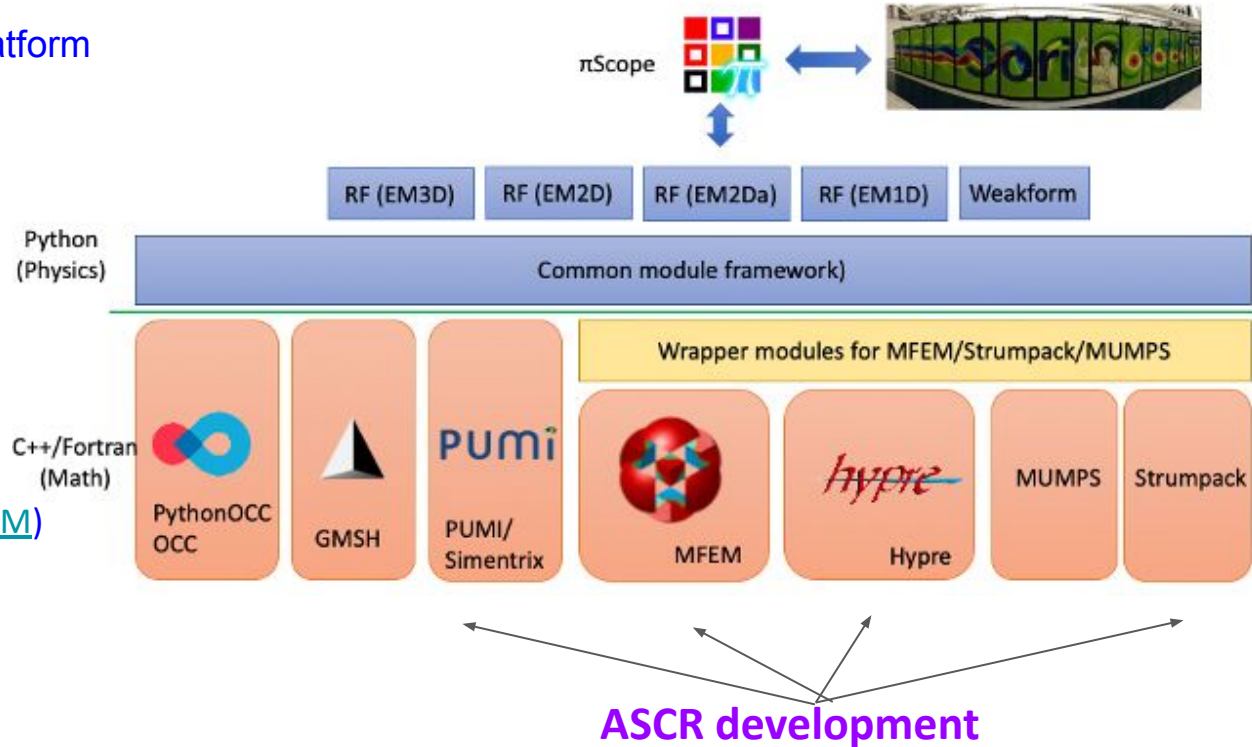
- Geometry creation
- Mesh generation
- FEM assembly and solve
- Visualization

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

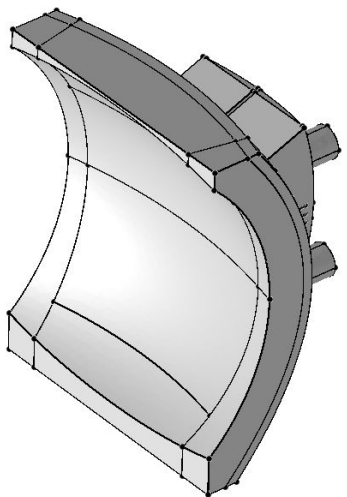
Tree Variables Shell Variables Console History

kwyds	value	type	shape	scri
{}		dict		

```

31 npsifit = interp2d(rgrid, zgrid, npsirz) # npsi = 0
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 nmmin=1.0e17
47 nmax=2.5e19s
48 # toroidal coordinates
49 B0Angle = 0.0;# -15.0 * (pi/180.0)
50 PML_MAJOR Z = 3.1 # z coordinate of center of rotation
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]);

```



```

--- 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/iframe/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

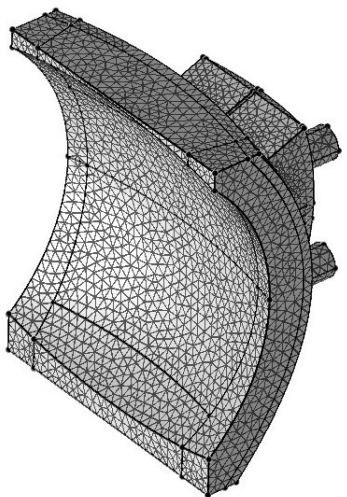
Tree Variables Shell Variables Console History

kwyds	value	type	shape	scri
	{}	dict		

```

31 npsifit = interp2d(rgrid, zgrid, npsirz) # npsi = 0
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=2.5e19s
48 # toroidal coordinates
49 B0Angle = 0.0; # -15.0 * (pi/180.0)
50 PML_MAJOR Z = 3.1 # z coordinate of center of rotation
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]);

```



```

--- 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

kwyds	value	type	shape	scri
{}		dict		

```

*em3d1.ns.py global.ns.py
31 npsifit = interp2d(rgrid, zgrid, npsirz) # npsi = 0
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 nmmin=1.0e17
47 nmax=2.5e19s
48 # toroidal coordinates
49 B0Angle = 0.0;# -15.0 * (pi/180.0)
50 PML_MAJOR Z = 3.1 # z coordinate of center of rotation
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]);

--- 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 History

	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.

epsilon(*) Array Form

=epsilon_pl

mur(*) Array Form

=mu_pl

sigma(*) Elemental Form

=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
 - namespc
 - datasets
 - solutions
 - scripts
 - mfembo
 - page1
 - axi

Tree Variables Shell V

value
kywds {}

Model Tree

General(NS:global)

- ▶ Geometry
- ▶ Mesh
- ▼ Phys
 - ▶ EM3D1(E,NS:em3
 - ▼ WF1(uvalue)
 - ▼ Domain
 - WeakContrib
 - ▶ 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	<input type="checkbox"/>	
use conjugate	<input type="checkbox"/>	

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_file1
 - matlab_mat1

book3

pro...ook.page1.axes

Tree Variables Shell Variables Console History

= Expression

	value	type	shape	scri
kywds	{}	dict		

Eye[1, 2, 3, 4]

X Y Z

```

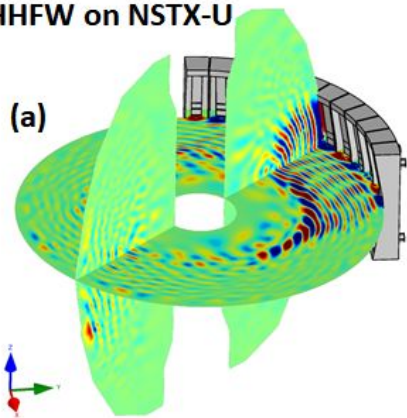
--- Welcome to piScope (dev)---
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Startup script executed: /home/shiraiwa/twopi_roots/20200718/src/piScope/python/ifigure/startup.py
>>>
>>> petram()
<petram.pi.shell_commands.PetramHelper object at 0x7f5c16b7bf70>
>>> petram()
<petram.pi.shell_commands.PetramHelper object at 0x7f5c5e43fa60>
>>>

```

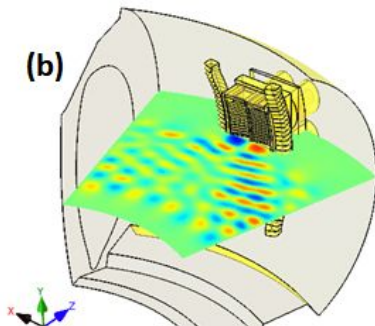
1 proc / 1.293468...

Petra-M has applied on variety of fusion devices worldwide

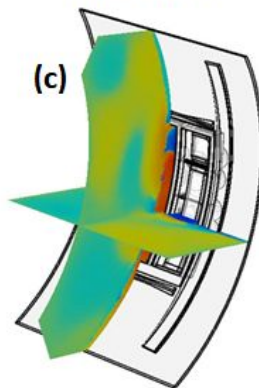
HHFW on NSTX-U



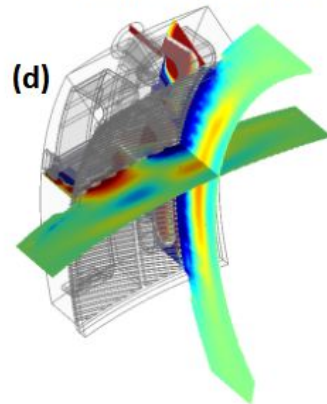
ICRF on WEST



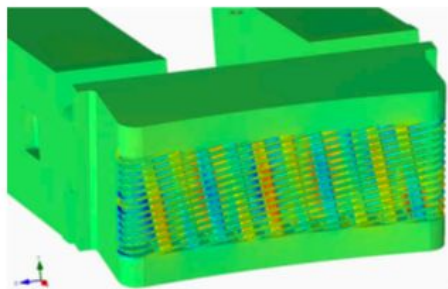
ICRF on JET



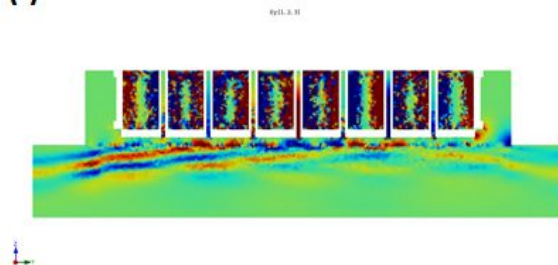
ICRF on ASDEX-U



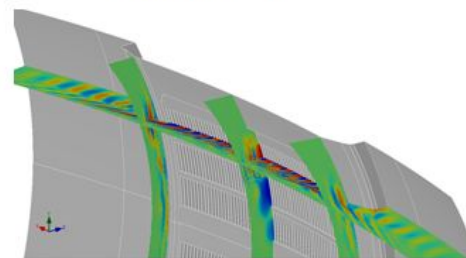
(e) Helicon on KSTAR



(f) Helicon on DIII-D

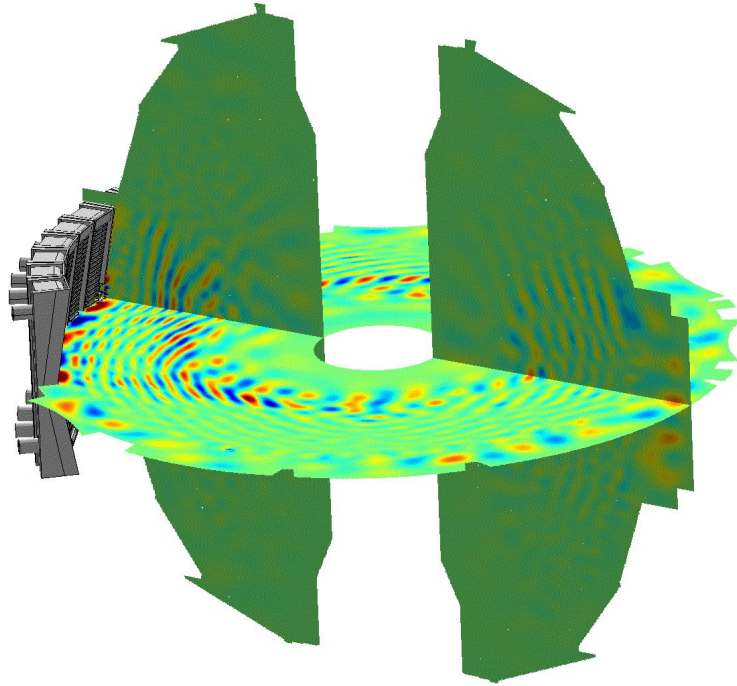


(g) LH on WEST

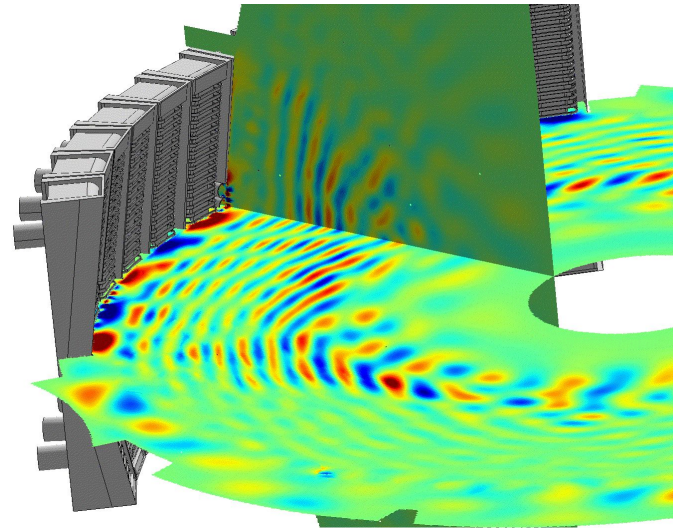


First fully resolved 3D HHFW field on NSTX-U

E_z

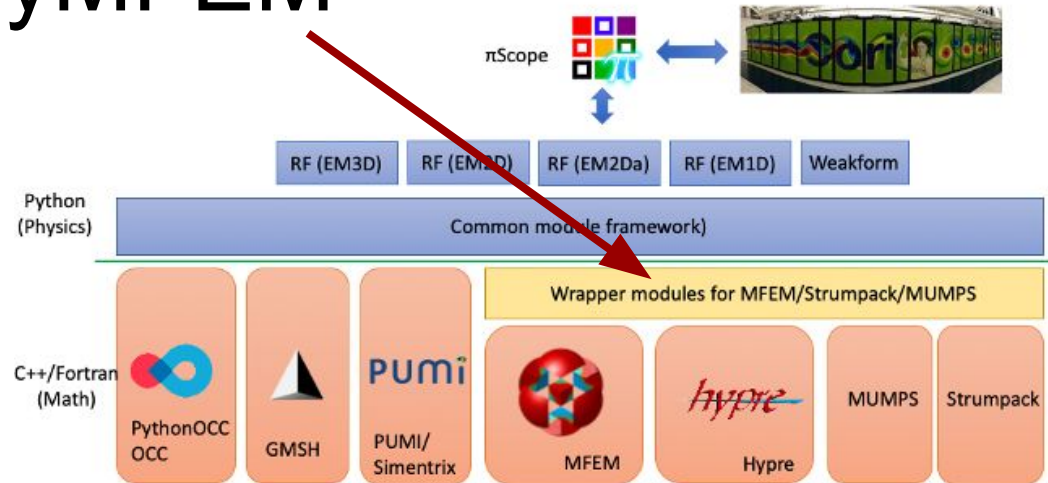


$B_t = 1\text{T}$, $n_{e0} = 5 \times 10^{19} \text{m}^{-3}$
150° phasing
Cold plasma dielectric



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

PyMFEM



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



Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

New to Binder? Get started with a Zero-to-Binder tutorial in Julia, Python or R.

Build and launch a repository

GitHub repository name or URL

GitHub **mfem/PyMFEM**

Git ref (branch, tag, or commit)

HEAD

URL to open (optional)

URL to open (optional) File

launch

<http://mybinder.org>

mfem 4.3.0.1

`pip install mfem`

Latest version

Released: Aug 30, 2021

MFEM + PyMFEM (finite element method library)

Project description

<http://pypi.org>

PyMFEM

MFEM + PyMFEM (FEM library)

This package provides MFEM and its Python wrapper (PyMFEM). MFEM is a high performance parallel finite element method (FEM) library (<http://mfem.org>).

Installer downloads a couple of external libraries and build them. By default, "pip install mfem" downloads and builds the serial version of MFEM and PyMFEM. See more detail below for other configurations

Install

```
pip install mfem # b
pip install mfem --no-binary mfem # i
```

The setup script accept various options. TO use it, please download the package and run

Home Notifications Explore Profile

<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!

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

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

ex1.cpp

```
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.

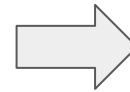
- C array to numpy.array
- Pointer argument as a return value

MyModule.hpp

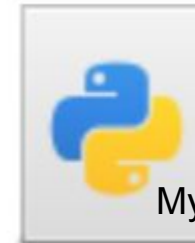
```
class MyClass {  
private:  
    int _hidden;  
Public:  
    void function(int x);  
};
```

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

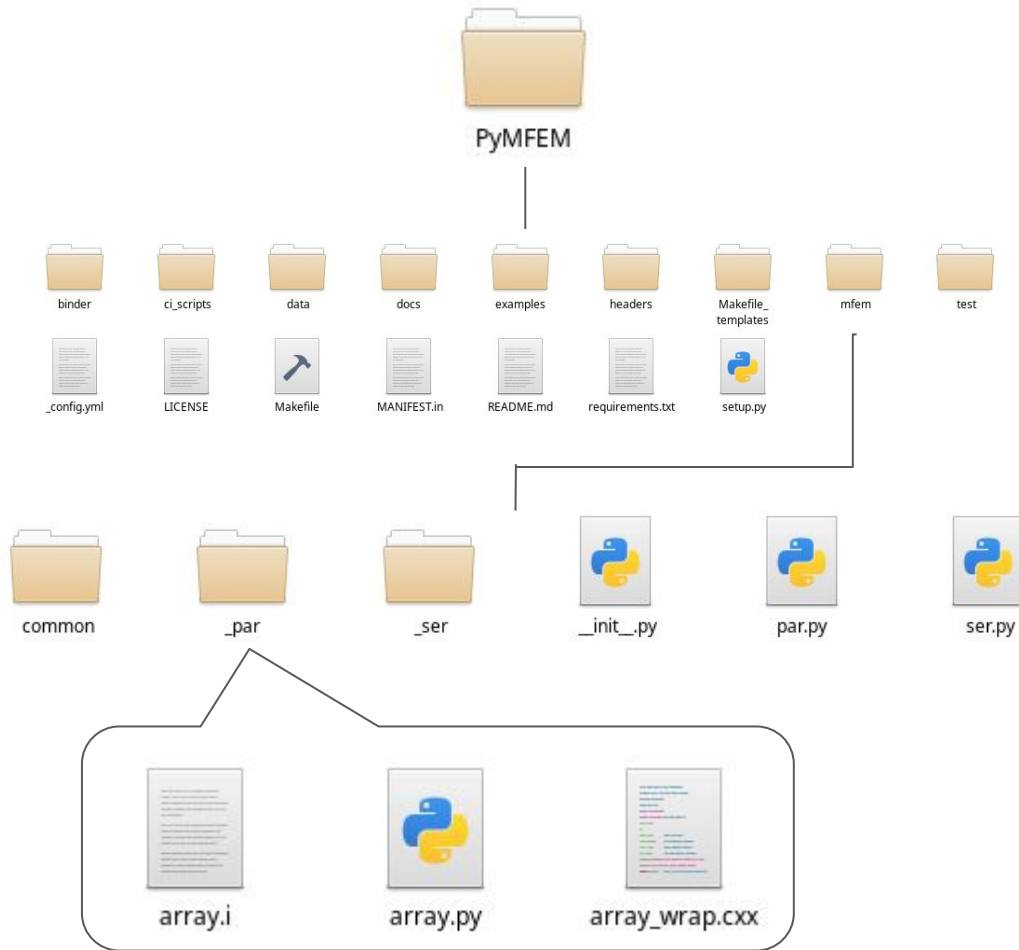
```
import mfem.ser as mfem
```

Parallel (mfem._par)

Built with Hypr 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

```
1 %module(package="mfem_ser") triangle
2 %{
3 #include "mfem.hpp"
4 #include "mesh/triangle.hpp"
5 #include "numpy/arrayobject.h"
6 %}
7
8 %init %{
9 import_array();
10 %}
11 #include "exception.i"
12 #import "fe.i"
13 #import "element.i"
14 #include "../common/typemap_macros.i"
15 #include "../common/exception.i"
16
17 LIST_TO_INTARRAY_IN(const int *ind, 2)
18 INTARRAY_OUT_TO_TUPLE(int *GetVertices, 2)
19
20 #include "../common/deprecation.i"
21 DEPRECATED_OVERLOADED_METHOD(mfem::Triangle::GetNFaces,
22                               Triangle::GetNFaces(int & nFaceVertices) is deprecated,
23                               len(args) == 1)
24
25
26 #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
1 %module(package="mfem_ser") triangle
2 %{
3 #include "mfem.hpp"
4 #include "mesh/triangle.hpp"
5 #include "numpy/arrayobject.h"
6 %}
7
8 %init %{
9 import_array();
10 %}
11 #include "exception.i"
12 %import "fe.i"
13 %import "element.i"
14 #include "../common/typemap_macros.i"
15 #include "../common/exception.i"
16
17 LIST_TO_INTARRAY_IN(const int *ind, 2)
18 INTARRAY_OUT_TO_TUPLE(int *GetVertices, 2)
19
20 #include "../common/deprecation.i"
21 DEPRECATED_OVERLOADED_METHOD(mfem::Triangle::GetNFaces,
22                               Triangle::GetNFaces(int & nFaceVertices) is deprecated,
23                               len(args) == 1)
24
25
26 #include "mesh/triangle.hpp"
```

dependency

custom config.

Header file

triangle.i

Documentation still needs improvement quite a bit

master

PyMFEM / docs / manual.txt

sshiraiwa fixed memory leak History

2 contributors

622 lines (487 sloc) | 23.3 KB

```
1  '''
2  PyMFEM
3  built on mfem 4.2 (commit SHA=ed5604e0d350461f20842275578aa2f9e6a61343)
4  '''
5
6  PyMFEM is a python wrapper for MFEM, lightweight FEM (finite element
7  method) library developed by LLNL (http://mfem.org).
8  PyMFEM is tested with python = 3.6, 3.7
9
10 This wrapper is meant for a rapid-prototyping of FEM programs, and
11 is built using SWIG 4.0.2
12 With PyMFEM, a user can create C++ MFEM objects and call their
13 method from python. We strongly recommend visiting the MFEM web site
```



File Edit View Run Kernel Tabs Settings Help

Filter files by name

Name	Last Modified
changelog.txt	a day ago
developme...	a day ago
example_J...	a day ago
install.txt	a day ago
manual_ar...	a day ago
manual.txt	a day ago

mfem::Array

Array is a template in MFEM to handle an array of data type . In order to use it from Python, a template needs to be instantiated for each PyMFEM defines the following instansions.

- * Array<int> -> intArray
- * Array<double> -> doubleArray
- * Array<IntegrationPoint> -> IntegrationPointArray
- * Array<GeometryType> -> GeometryTypeArray
- * Array<Refinement> -> RefinementArray
- * Array<FiniteElementSpace> -> FiniteElementSpaceArray

Among them, intArray and doubleArray are the template insatiation we use the most by far. Note we use a naming convention of Array. If you need Array for a different class, please let us know.

1. Constructor

```
[ ]: import mfem.ser as mfem
# Construction with one integer argument prepare a space for a given size.
x = mfem.intArray(5)
y = mfem.doubleArray(3)
print(x.Size(), y.Size())
# ToList convert intArray, doubleArray to List. Data is copied.
# Note that array is not initialized.
print(x.ToList(), y.ToList())
# Construction using a give list/tuple
x = mfem.intArray((1,3,3))
y = mfem.doubleArray([4.0, 5.0])
print(x.ToList(), y.ToList())
```

2. Initialize array (Assign)

```
[ ]: # Assign is replacement of = operator in C++. Wrapper take care of conversion to
x.Assign(3)
y.Assign(5)
print(x.ToList(), y.ToList())
```

Started migration to Notebook

Waiting for contribution/suggestions !