Computational Frameworks in Fusion and Plasma Physics

Sterling Smith

1 General Atomics

2020 PPPL Introduction to Fusion Energy and Plasma Physics Course
June 24, 2020
Outline

Personal background

Do not reinvent the wheel

Give back to the community

What is OMFIT?

OMFIT solves common computer usage issues
My mother took me stargazing
In Dominican Republic, I was inspired to find a more steady source of electricity.
Life happens along the way

Birth Stargaze Graduate high school Graduate Utah State Graduate Princeton→GA
Start mission End mission
Married
Child 1
Child 2
Child 3
Child 4
Child 5
Child 6
Today

S.P. Smith - PPPL Intro Course - June 24, 2020
The “best” computer language depends on the application

- I started dabbling with qbasic\(^1\) in high school (late 1990’s)
- I took a C++\(^2\) class as an undergraduate
- The Utah State physics program used MathCAD\(^3\) and Maple\(^4\) for undergraduate instruction
- In graduate school, I first learned about unix\(^5\)/linux\(^6\)
- My 1st year project required IDL\(^7\) for data processing
- My 2nd year project introduced me to python\(^8\)
- The graduate laboratory class exposed me to MatLab\(^9\)
- I wrote my thesis code in Fortran90\(^10\), with python pre- and post- processing
- At DIII-D, I took over and improved some analysis routines written in IDL (ZIPFIT)
- I later took over maintenance of the ONETWO code - a half million lines of Fortran77
- Along the way, I worked to implement the coordination of code running (integrated modeling) via python (IMFIT, then OMFIT\(^11\))
Outline

Personal background

Do not reinvent the wheel

Give back to the community

What is OMFIT?

OMFIT solves common computer usage issues
Computers have been around for a while now

Initial code recipes have evolved into entire libraries and frameworks

- *Numerical Recipes* books\(^{12}\)
- BLAS\(^{13}\) and LAPACK\(^{14}\) - Linear Algebra libraries
- BOUT\(^{15}\) - A framework for solving an arbitrary set of equations including parallelization
- hDF5\(^{16}\)/netCDF4\(^{17}\) - A library for storing self-describing data
- MDS+\(^{18}\) - A library for storing data with a client/server interface (most US tokamaks use this)
- OMFIT\(^{11}\) - One Modeling Framework for Integrated Tasks - More later
- conda\(^{19}\) - Cross-platform package and dependency management
- IMAS\(^{20}\) - ITER Integrated Modeling and Analysis Suite

This list is not exhaustive, but rather representative of the types of libraries that are available.
Do your homework and ask around

If you are going to write some code that is more than a few lines long:

- Do a google search
- Do a search on github.com
- Ask your advisor:
  - Have you done something already?
  - Do you know someone that may have done something similar?
  - How would you do it?
- Ask the community the same questions. Here are some communities:
  - NTCC$^{21}$ - Started down path of code collection for MFE
  - The AToM project$^{22}$ is knowledgeable of existing MFE codes
  - The Plasmapy project$^{23}$ represents the most modern and open collection of efforts in plasma research
- Use existing input/output standards
  - netcdf$^{17}$, hdf5$^{16}$, yaml$^{24}$, IMAS DD$^{20}$, OpenPMD$^{25}$, namelist$^{26}$
Outline

Personal background

Do not reinvent the wheel

Give back to the community

What is OMFIT?

OMFIT solves common computer usage issues
Not reinventing the wheel is not enough; feedback is invaluable

- If the documentation is lacking, report it
- If you need help, ask for it
- If you find a bug, report it
- If you figure out how to fix the bug, report it
Use the http://www.sscce.org pattern when asking for help

Short, Self Contained, Correct (Compilable), Example

**Short** Cut out all of the extra code that is not causing a problem
- Sometimes this step by itself is sufficient to help you see the solution yourself

**Self contained** Include any non-public libraries, files, data, or metadata that are necessary to produce the problem
- If on a cluster be sure file permissions allow access

**Correct** The code provided should work/compile

**Example** The code should actually demonstrate the problem
When you start writing code, use version control

cvs\textsuperscript{28} was one of the initial version control systems

\textit{svn}\textsuperscript{29} came along as a replacement for \textit{cvs} with some improvements for ease of use

\textit{git}\textsuperscript{30} is emerging as the program of choice to track code versions

- All clones of a \textit{git} repository contain the whole repository
- A central canonical version may be hosted at GitHub(.com) or a local GitLab server
- Branching is easy and cheap
- GitHub provides code review and continuous integration testing tools
Random thoughts

• Use a package manager - conda\textsuperscript{19} is my manager of choice
  - The more portable a code is, the more users it will gain
  - The more users it gains, the more contributors it will garner
  - The more contributors it has, the more stable and feature-rich the code will become

• There are no bugs, just unwritten regression tests (see previous talk by Dr. Murphy)

• Get a coding buddy

• Look over someone’s shoulder as they work at a computer - what shortcuts do they have for working?
Homework: Interact with an open source project on GitHub.com

1. Pick a repo
   - https://github.com/MFEFormulary/MFEFormulary
   - https://github.com/PlasmaPy/plasmapy
   - https://github.com/gafusion/OMFIT-source/
   - https://github.com/smithsp/cpsfr

2. Open an issue for a feature request or bug

3. Clone the repo

4. Make some changes on a branch and push it back to GitHub

5. Open a Pull Request

6. Read the Cathedral and the Bazaar\textsuperscript{31}

\textsuperscript{a}Send me your GitHub username for access
Outline

Personal background

Do not reinvent the wheel

Give back to the community

What is OMFIT?

OMFIT solves common computer usage issues
OMFIT is a versatile framework designed to facilitate experimental data analysis and enable integrated simulations.

OMFIT was created to fill a gap in the fusion community

- A unified tool for
  - Accessing experimental data
  - Running existing codes
  - Parsing code inputs and outputs
  - Streaming together outputs of one workflow as the inputs of another workflow
  - Collecting community scripts
  - Creating robust GUls
  - Visualizing results

- A tool that is
  - Portable
  - Based on a modern language
  - Public
  - Documented
  - Tested

OMFIT by the numbers since July 2015

- Cumulative graphical sessions: 120,926
- Cumulative commits: 33048
- Cumulative users: 843
“A versatile framework designed to facilitate experimental data analysis and enable integrated simulations”
Integrated Modeling Framework
- Enables data exchange among different components and coordinates their execution in complex workflows

Lightweight & pure Python
- Remote execution of interactive/batch jobs
- Installs & runs anywhere: public/private, cluster/laptop

Free-form tree data structure
- Support for most fusion-relevant data formats
- Does not exclude use of standard data structures, such as IMAS (ITER Integrated Modeling and Analysis Suite)

Interactive and graphical or scripted
- Accelerate most time consuming tasks: develop, setup, visualize, share

Version control and community
- Grow at scale cheaply and remain focused
OMFIT is installed on many fusion public clusters and accesses data from many tokamaks around the world

- ITER
- GA/DIII-D
- NERSC
- MIT
- PPPL/NSTX
- EPFL/TCV
- CCFE/JET/MAST
- ITM-Gateway
- IGI-CNR/RFX
- IPP-CA/COMPASS
- MPI-IPP/ASDEX-U
- WEST
- ASIPP/EAST
- SWIP/HL2A/HL2M
- NFRI/KSTAR
- CFETR

2019
OMFIT is actively used for current research
At least 54 presentations in the last APS DPP Conference

5 invited, 14 orals, 1 miniconference, 34 posters

<table>
<thead>
<tr>
<th>Name</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kriete</td>
<td>BI2.3</td>
</tr>
<tr>
<td>Austin</td>
<td>BP10.3</td>
</tr>
<tr>
<td>Weisberg</td>
<td>BP10.4</td>
</tr>
<tr>
<td>Haskey</td>
<td>BP10.7</td>
</tr>
<tr>
<td>Zhao</td>
<td>BP10.8</td>
</tr>
<tr>
<td>Morton</td>
<td>BP10.9</td>
</tr>
<tr>
<td>Smith</td>
<td>BP10.10</td>
</tr>
<tr>
<td>Mckee</td>
<td>BP10.11</td>
</tr>
<tr>
<td>Truong</td>
<td>BP10.12</td>
</tr>
<tr>
<td>G. Wang</td>
<td>BP10.16</td>
</tr>
<tr>
<td>Victor</td>
<td>BP10.23</td>
</tr>
<tr>
<td>Park</td>
<td>BP10.24</td>
</tr>
<tr>
<td>Ding</td>
<td>BP10.25</td>
</tr>
<tr>
<td>Park</td>
<td>BP10.30</td>
</tr>
<tr>
<td>Yan</td>
<td>BP10.35</td>
</tr>
<tr>
<td>Collins</td>
<td>BP10.37</td>
</tr>
<tr>
<td>Marini</td>
<td>BP10.39</td>
</tr>
<tr>
<td>Tang</td>
<td>BP10.43</td>
</tr>
<tr>
<td>Crocker</td>
<td>BP10.44</td>
</tr>
<tr>
<td>Xing</td>
<td>BP10.102</td>
</tr>
<tr>
<td>Sciortino</td>
<td>CO5.11</td>
</tr>
<tr>
<td>Ashourvan</td>
<td>CO7.4</td>
</tr>
<tr>
<td>Pankin</td>
<td>CO7.10</td>
</tr>
<tr>
<td>Houshmandyar</td>
<td>GP10.61</td>
</tr>
<tr>
<td>Knolker</td>
<td>JI2.1</td>
</tr>
<tr>
<td>Grierson</td>
<td>JO5.2</td>
</tr>
<tr>
<td>Chen</td>
<td>JO5.5</td>
</tr>
<tr>
<td>Hu</td>
<td>JO5.6</td>
</tr>
<tr>
<td>Jian</td>
<td>JO5.11</td>
</tr>
<tr>
<td>H. Wang</td>
<td>JO5.15</td>
</tr>
<tr>
<td>Munaretto</td>
<td>JP10.23</td>
</tr>
<tr>
<td>Smith</td>
<td>PM8.6</td>
</tr>
<tr>
<td>Logan</td>
<td>PO5.3</td>
</tr>
<tr>
<td>Ernst</td>
<td>PO5.5</td>
</tr>
<tr>
<td>Rea</td>
<td>PO5.7</td>
</tr>
<tr>
<td>Bykov</td>
<td>PO5.11</td>
</tr>
<tr>
<td>Mcclenaghan</td>
<td>PO5.15</td>
</tr>
<tr>
<td>Montes</td>
<td>PP10.67</td>
</tr>
<tr>
<td>Holland</td>
<td>PP10.107</td>
</tr>
<tr>
<td>Xie</td>
<td>PP10.110</td>
</tr>
<tr>
<td>Neiser</td>
<td>PP10.125</td>
</tr>
<tr>
<td>Orlov</td>
<td>PP10.132</td>
</tr>
<tr>
<td>Brookman</td>
<td>TP10.112</td>
</tr>
<tr>
<td>Samuell</td>
<td>TP10.117</td>
</tr>
<tr>
<td>Yu</td>
<td>TP10.118</td>
</tr>
<tr>
<td>Hong</td>
<td>TP10.119</td>
</tr>
<tr>
<td>Nelson</td>
<td>TP10.128</td>
</tr>
<tr>
<td>Zhang</td>
<td>TP10.136</td>
</tr>
<tr>
<td>Abbate</td>
<td>TP10.137</td>
</tr>
<tr>
<td>Morosohk</td>
<td>UO8.3</td>
</tr>
<tr>
<td>Laggner</td>
<td>VI3.4</td>
</tr>
<tr>
<td>Thome</td>
<td>Y13.1</td>
</tr>
<tr>
<td>Casali</td>
<td>YP10.38</td>
</tr>
</tbody>
</table>
Outline

Personal background

Do not reinvent the wheel

Give back to the community

What is OMFIT?

OMFIT solves common computer usage issues
OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
</tbody>
</table>
OMFIT provides access to experimental data and facilitates interpretive analysis.

OMFIT can interface with:
- MDS+
- SQL
- UDA
- s3
- IMAS (ITER)
OMFIT provides access to experimental data and facilitates interpretive analysis.

**TRANSP** is a free-boundary equilibrium and transport solver, used for analysis of tokamak plasma discharges and for experimental planning.

![Experimental electron temperature profile vs time](image)

Comparison of predicted neutron rate and total pressure provides constraint for $Z_{\text{eff}}$

OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
</tbody>
</table>
OMFIT parses and writes a variety of input and output files

- Fortran namelist
- NetCDF 3/4
- HDF 5
- g,k,a,m eqdsk
- SOLPS
- IPS config
- TGLF
- (c)GYRO
- TGYRO
- pfile
- ONETWO
- TRANSP

- BOUT++
- M3D-C1
- NIMROD
- GENRAY
- TORAY
- FASTRAN
- IDL savefile
- IONORB
- GATO
- GPEC
- JSON
- MARS
SOLPS is a fluid code that attempts to solve for the densities and temperatures in the scrape off layer (SOL) given boundary conditions.

Example input file:

```
520  7 : Recycling ~ PFR side  3
521  FF7F
522  2000  10001  6  1
523  1.00000E+00  0.00000E+00  0  0  0  0  0  0  0  0
524  FF7F
525  -1
526  FF7F
527  1
528  7  1  1  1  1  1  53  64  0  0
529  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
530  -1  0  0  0  0  0
531  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
532  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
533  1.00000E+00  9.00000E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
534  1  : Recycling ~ PFR side  4
535  FF7F
536  2000  10001  6  1
537  1.00000E+00  0.00000E+00  0  0  0  0  0  0  0  0
538  FF7F
539  -1
540  FF7F
541  1
542  7  1  1  1  1  1  90  102  0  0
```

OMFIT in-memory parsing provides access to variables in a systematic way.

In-memory variables can be edited, which propagates back to the file on disk.
### OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
<tr>
<td>Technical guide of code is too deep for average user</td>
<td>Provide customizable GUIs and tutorials</td>
</tr>
</tbody>
</table>
OMFIT provides easily customizable graphical user interfaces (GUIs) to expose most common settings and workflows in plain English.

```python
OMFITx.TitleGUI('Basic GUI')

OMFITx.Label('A label, a test image, and a separator')

OMFITx.Label(',,image=OMFITsrc+ '/../samples/test_image.png')

OMFITx.Separator('Separator')

with OMFITx.same_row():
    OMFITx.Entry('OMFIT['entry']', lba='Freeform Entry',
                default=1.0, help='A live text entry')
    OMFITx.Button('Press me!', lambda: None)

with OMFITx.same_tab('Tab 1'):
    OMFITx.Label('Here I am!')

with OMFITx.same_tab('Tab 2'):
    OMFITx.Label('Not here')

OMFITx.ComboBox('OMFIT[combo]', {1: 'This is option 1'},
                 lba='Combo Box', default=1)

OMFITx.ObjectPicker('root[geqdsk]', lba='G-file',
                     objectType=OMFITgeqdsk)
```

A label, a test image, and a separator

```
Freeform Entry = 1.0

Tab 1  Tab 2

Here I am!

Combo Box = '1'

Pick G-file = 
```

S.P. Smith - PPPL Intro Course - June 24, 2020
OMFIT provides easily customizable graphical user interfaces (GUIs) to expose most common settings and workflows in plain English.

**NIMROD** extended MHD code used to follow acceleration and deconfinement of runaway electrons.

**OMFIT NIMROD module:**
- Generates initial conditions
- Generates restart files in hdf5 format for non-standard NIMROD particle orbit routines
- Visualizes the results

72 hours on 12 nodes on cori: $\sim 30k$ CPU-hours

V.A. Izzo and P.B. Parks, Phys. Plasmas, 24:060705
OMFIT provides easily customizable graphical user interfaces (GUIs) to expose most common settings and workflows in plain English.

NIMROD extended MHD code used to follow acceleration and deconfinement of runaway electrons

OMFIT NIMROD module:

- Generates initial conditions
- Generates restart files in hdf5 format for non-standard NIMROD particle orbit routines
- Visualizes the results

72 hours on 12 nodes on cori: $\sim 30k$ CPU-hours

V.A. Izzo and P.B. Parks, Phys. Plasmas, 24:060705
OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
<tr>
<td>Technical guide of code is too deep for average user</td>
<td>Provide customizable GUIs and tutorials</td>
</tr>
<tr>
<td>Code only runs at a few locations</td>
<td>Package OMFIT to be portable, and allow remote execution of arbitrary scripts, even with tunnels or MFA</td>
</tr>
</tbody>
</table>
OMFIT can reach remote servers despite tunnels or multi-factor authentication

If appropriate, OMFIT runs a python library with the same functionality as One Time Password applications such as Google Authenticator.
OMFIT can reach remote servers despite tunnels or multi-factor authentication

**BOUT++** is a framework for solving general differential equations. Our specific use case is Edge Localized Mode stability.

22 nodes for 30 minutes on edison: 240 CPU-hours

---

B.D. Dudson, et al., Comp. Phys. Comm. 180:1467
OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
<tr>
<td>Technical guide of code is too</td>
<td>Provide customizable GUIs and tutorials</td>
</tr>
<tr>
<td>deep for average user</td>
<td></td>
</tr>
<tr>
<td>Code only runs at a few locations</td>
<td>Package OMFIT to be portable, and allow remote execution of arbitrary scripts, even with tunnels or MFA</td>
</tr>
<tr>
<td>Remote visualization of results is</td>
<td>In-situ processing with local rendering</td>
</tr>
<tr>
<td>difficult</td>
<td></td>
</tr>
</tbody>
</table>
OMFIT offers great visualization (via Mayavi and matplotlib)

GPEC is a suite of codes for calculating magnetohydrodynamic (MHD) instabilities

Shown here is the 3D plasma response to the coils, such as would be applied to suppress edge localized modes (ELMs)

Each user does not have to relearn plotting

OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
<tr>
<td>Technical guide of code is too deep for average user</td>
<td>Provide customizable GUIs and tutorials</td>
</tr>
<tr>
<td>Code only runs at a few locations</td>
<td>Package OMFIT to be portable, and allow remote execution of arbitrary scripts, even with tunnels or MFA</td>
</tr>
<tr>
<td>Remote visualization of results is difficult</td>
<td>In-situ processing with local rendering</td>
</tr>
<tr>
<td>Tediouss to run scans</td>
<td>Automate parsing inputs, creating directories, plotting</td>
</tr>
</tbody>
</table>
OMFIT automates the parsing of inputs, creating directories, and plotting across scans

The **GYRO** code solves the gyrokinetic equation for the turbulent linear growth rates or nonlinear fluxes given input local parameters.

Eigenvalue comparison between TGLF and QL-GYRO

Growth rates

![Graph showing growth rates comparison between TGLF and QL-GYRO](image)

Mode frequency

![Graph showing mode frequency comparison between TGLF and QL-GYRO](image)

Flux predictions of QL-GYRO and TGLF for a RLTS scan of GA standard case

![Graph showing flux predictions comparison between QL-GYRO and TGLF](image)


OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
<tr>
<td>Technical guide of code is too deep</td>
<td>Provide customizable GUIs and tutorials</td>
</tr>
<tr>
<td>for average user</td>
<td>Package OMFIT to be portable, and allow remote execution of arbitrary scripts, even with tunnels or MFA</td>
</tr>
<tr>
<td>Code only runs at a few locations</td>
<td>In-situ processing with local rendering</td>
</tr>
<tr>
<td>Remote visualization of results is</td>
<td></td>
</tr>
<tr>
<td>difficult</td>
<td></td>
</tr>
<tr>
<td>Tedious to run scans</td>
<td>Automate parsing inputs, creating directories, plotting</td>
</tr>
<tr>
<td>HPC results cannot be used directly</td>
<td>Harvest results and build NN</td>
</tr>
<tr>
<td>by community</td>
<td></td>
</tr>
</tbody>
</table>
OMFIT builds databases, then trains machine learning models as method for whole community to benefit from HPC usage.

OMFIT has compiled a variety of databases from scans over experimental conditions or simulation parameters and built corresponding neural networks:

- Experimental heat flux
- Pedestal height/width
- Bootstrap current
- Turbulence heat flux
OMFIT builds databases, then trains machine learning models as method for whole community to benefit from HPC usage

**EPED-1** predicts pedestal height and width given 8 inputs

- Stability evaluated for scans of model equilibria
- The IPS (Integrated Plasma Simulator) framework parallelized the workflow to run on $\sim 760$ CPUs, taking $\sim 3$ minutes
- EPED-1 was run for a database of $\sim 20k$ existing and future experimental conditions
- EPED results harvested then used for training neural network

PB. Snyder et al. Phys. Plasmas 16:056118
OMFIT solves common computing issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>OMFIT solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get experimental setup</td>
<td>Access experimental data and facilitate interpretive analysis</td>
</tr>
<tr>
<td>Arcane input format files</td>
<td>Python classes for parsing and writing broad range of files</td>
</tr>
<tr>
<td>Technical guide of code is too deep for average user</td>
<td>Provide customizable GUIs and tutorials</td>
</tr>
<tr>
<td>Code only runs at a few locations</td>
<td>Package OMFIT to be portable, and allow remote execution of arbitrary scripts, even with tunnels or MFA</td>
</tr>
<tr>
<td>Remote visualization of results is difficult</td>
<td>In-situ processing with local rendering</td>
</tr>
<tr>
<td>Tedious to run scans</td>
<td>Automate parsing inputs, creating directories, plotting</td>
</tr>
<tr>
<td>HPC results cannot be used directly by community</td>
<td>Harvest results and build NN</td>
</tr>
<tr>
<td>Code run is part of larger integrated problem</td>
<td>String together tasks regardless of code run type (local serial or queued remote HPC)</td>
</tr>
</tbody>
</table>
OMFIT can string together tasks regardless of code run type (local/remote or serial/batch) to perform integrated modeling with experimental validation.

The STEP module combines many different physics codes to obtain a self-consistent prediction of core kinetic profiles:

- **Serial local**
  - TGLF
  - NEO
  - TGYRO

- **Serial remote**
  - ONETWO
  - EFIT
  - STRAHL

- **Batch remote**
  - EPED

---

W.W. Pfeiffer GA-A16178
K. Behringer JET-R(87)08
PB. Snyder et al. Phys. Plasmas 16:056118
OMFIT can string together tasks regardless of code run type (local/remote or serial/batch) to perform integrated modeling with experimental validation.

- Self-consistent core, edge, pedestal, impurity simulations of DIII-D
- Colors are (0.5, 1.0, 1.5) × experimental \[ \int n_C \, d\rho \]
- The experimental profiles are shown in gray for comparison
OMFIT enables computing usage across fusion topics

- OMFIT streamlines the tedious parts of running existing codes
- OMFIT has a convenient API for accessing remote (even HPC) resources
- OMFIT has a convenient API for creating application specific Graphical User Interfaces
- Code sharing is encouraged
Consult with your advisor to see if OMFIT or plasmapy would be useful for your project. If OMFIT would be useful, do the following:

1. Watch the first OMFIT video tutorial
2. Determine whether you will be using a public installation or installing a personal copy
3. Sign up for GitHub, if not already
4. Fill out the OMFIT User Agreement

If you feel that I missed out on some fusion relevant software efforts, please send me more information, and we can be sure to expand the horizons of this topic in the future.
Weekly community meetings - come join us

**OMFIT**  Mondays at 11 PT/14 ET [https://discord.gg/wJAAFWh](https://discord.gg/wJAAFWh)

**PlasmaPy**  Tuesdays at 11 PT/14 ET [https://www.plasmapy.org/meetings/weekly/](https://www.plasmapy.org/meetings/weekly/)
References I

(4) https://www.maplesoft.com/products/Maple/.
(7) https://www.harrisgeospatial.com/Software-Technology/IDL.
(12) http://numerical.recipes/.
(13) http://www.netlib.org/blas/.
(14) http://www.netlib.org/lapack/.
(15) https://boutproject.github.io/.
(17) https://www.unidata.ucar.edu/software/netcdf/.
(18) https://mdsplus.org/.
(19) https://docs.conda.io/.
(21) https://w3.pppl.gov/ntcc/.
(22) https://atom.scidac.io/.
(23) https://www.plasmapy.org/.
(24) https://yaml.org/.
(25) https://github.com/openPMD.
References III


(31) http://www.catb.org/~esr/writings/cathedral-bazaar/.