Comparison of Computational Frameworks

A.Y. Pankin

Tech-X Corporation, CO, USA

Acknowledge discussions with

***S.E. Kruger* (*Tech-X Corp.*), *A.H. Kritz, T. Rafiq* (Lehigh U.)**

Computational Frameworks

Framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful

[IT standards and organizations glossary]

A software framework, in computer programming, is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality. Frameworks are similar to software libraries in that they are reusable abstractions of code wrapped in a well-defined API

[Wikipedia]

In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate. Some computer system frameworks also include actual programs, specify programming interfaces, or offer programming tools for using the frameworks. A framework may be for a set of functions within a system and how they interrelate; the layers of an operating system; the layers of an application subsystem; how communication should be standardized at some level of a network; and so forth. A framework is generally more comprehensive than a protocol and more prescriptive than a structure

Computational Frameworks

Software frameworks have these distinguishing features that separates them from libraries or normal user applications:

1) inversion of control - Unlike libraries or normal user applications, in a framework the overall program's flow of control is not dictated by the caller, but by the framework

2) default behavior - A framework has a default behavior. This default behavior must actually be some useful behavior and not a series of no-ops

3) extensibility - A framework can be extended by the user usually by selective overriding or specialized by user code providing specific functionality

4) non-modifiable framework code - The framework code, in general, is not allowed to be modified. Users can extended the framework, but not modify its code

Definitions of frameworks were not helpful to FSP framework discussions

- ***Lots of arguments devolved into what is a framework, and what is not a framework***
- ***Try simplifying definitions:***
  - Strong Frameworks: Frameworks with which you would want to rewrite TRANSP
  - Loose Frameworks: Frameworks with which you would not want to rewrite TRANSP

Recent Integrated Modeling Efforts in US

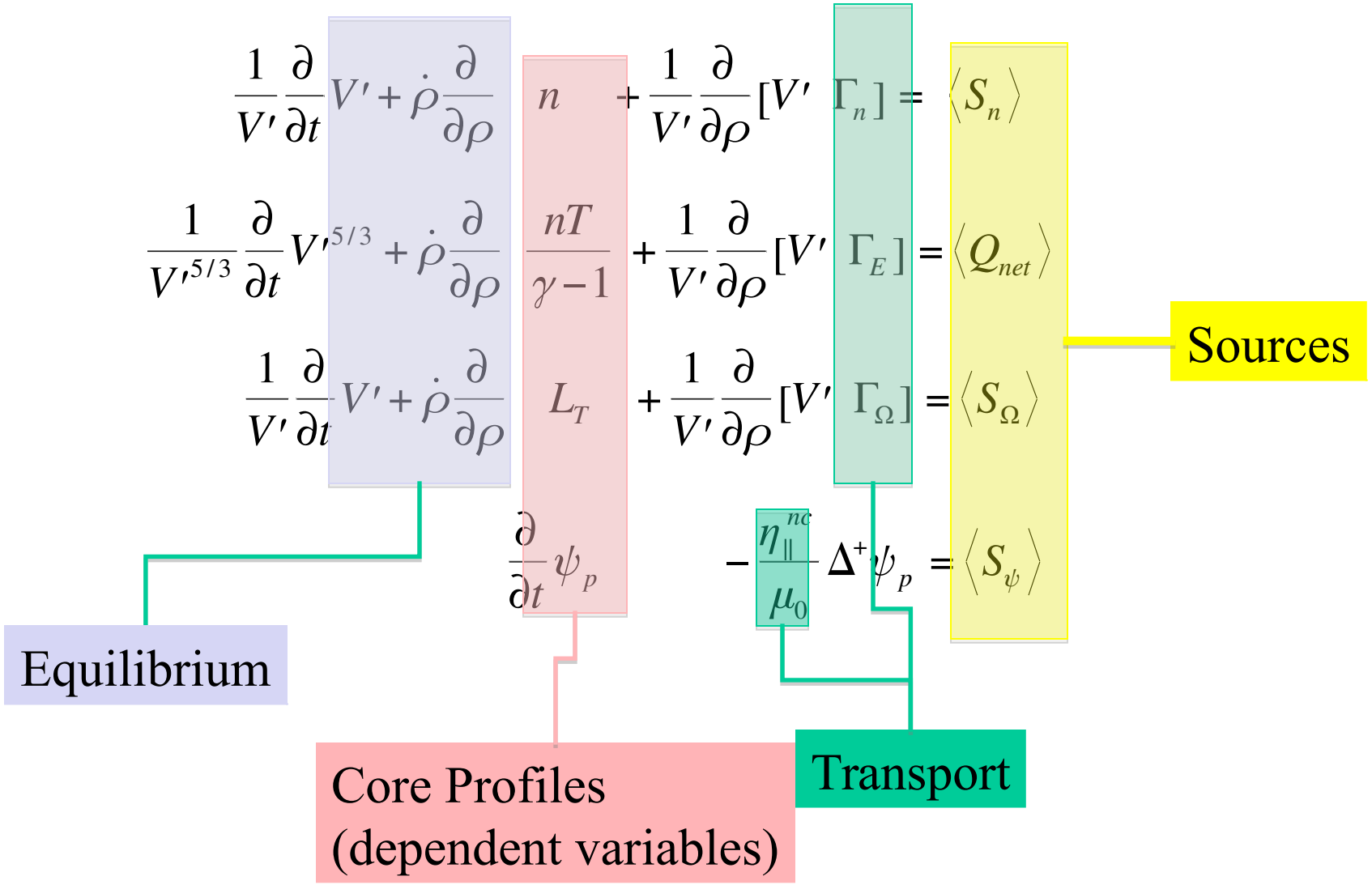
- Three computational frameworks are developed
  - FACETS Framework in the FACETS project is used to couple the core-edge-wall region in the plasma
  - EFFIS (End-to-end Framework for Fusion Integrated Simulation) is used to couple the neoclassical kinetic, turbulence gyro-kinetic and extended MHD codes in the CPES/EPSI project
  - The Integrated Plasma Simulator (IPS) Framework has been developed in the CSWIM project
- Two different coupling schemes (tight in FACETS and loose in EFFIS and IPS) are investigated
- New visualization and front-end interfaces such as FACETS Composer
- New standards for IO and coupling interfaces are developed
  - PlasmaStates (NETCDF based) in IPS and FACETS
  - XML based input in FACETS
  - HDF5 based in FACETS
  - ADIOS IO in EFFIS

Computational Frameworks

Framework defines

- **Mechanisms to include tasks**
  - Physics modules
  - Data modules
  - Visualization modules
- **Interfaces**
  - Access to experimental data
  - Storage of simulation results (Restart capability)
  - Physics model verification and regression analysis
  - Physics model validation (Synthetic diagnostic)
- **Interrelation between tasks – Workflows**
  - Communication between tasks
    - Coupling schemes
    - Data flows

1D Core Transport Equations



Component often classified by how they fit into WDM paradigm

Component list at:
http://fspcomp.web.lehigh.edu/index.php/Existing_components
Lists 67 components

PLASMA TRANSPORT	
Turbulent:	Neoclassical
Coppi-Tang,…	Chang-Hinton,
GLF23,	NCLASS
TGLF, MMM81	NEO
GYRO, GEM, …	

Increasing computational cost

SOURCES	
NEUTRAL BEAM INJECTION,…	RF HEATING AND CURRENT
NBEAM	GENRAY, TORAY
NUBEAM	TORIC
	AORSA

FUSION HEATING

2D EQUILIBRIA (Includes coils)

TEQ, VMEC, EFIT, ..

PELLET INJECTION

GLAQUELC,…

MHD Linear Stability	MHD Transport Models
DCON, ELITE, MARS, …	ISLAND, ELM, Sawtooth, …

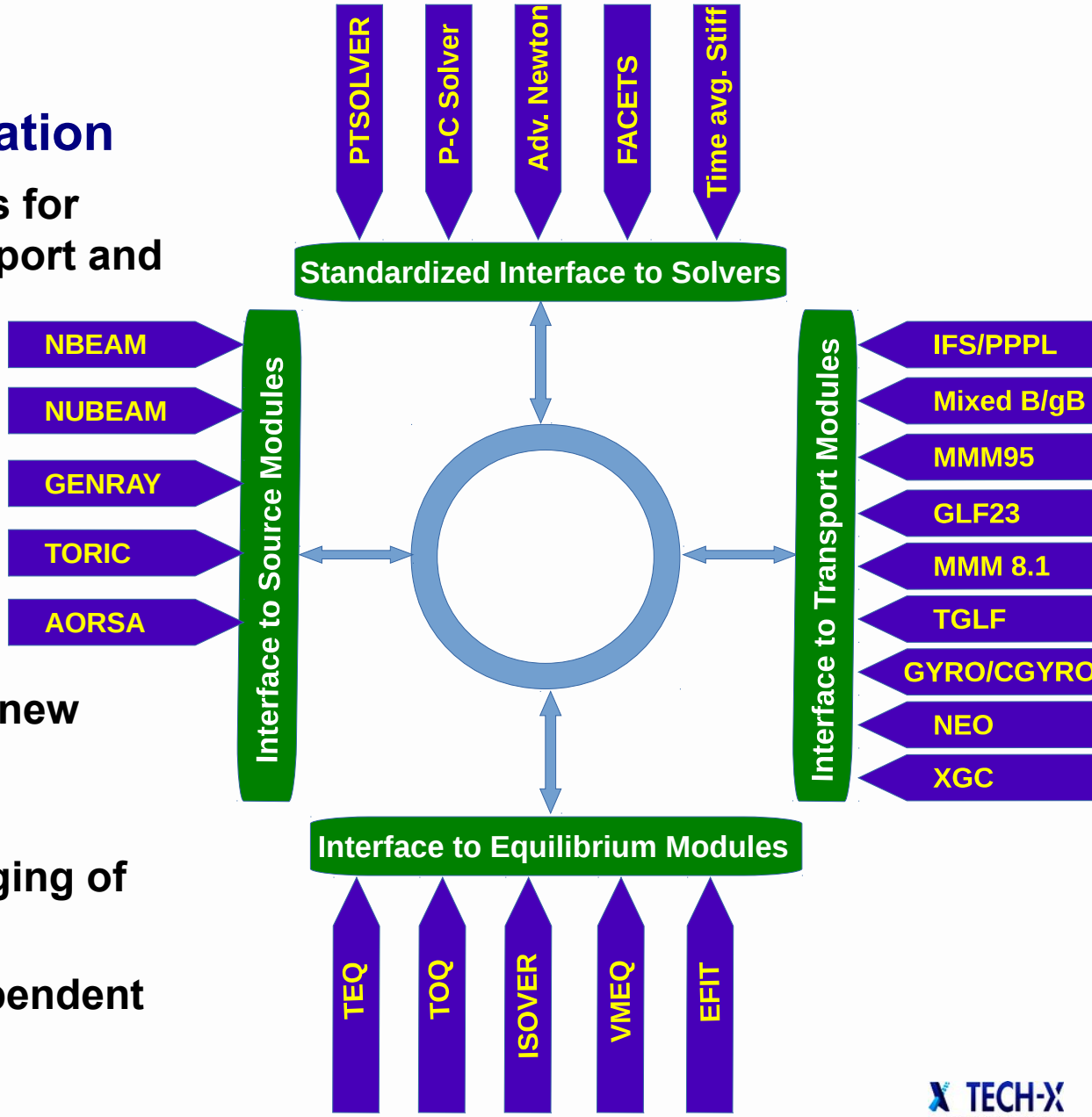
Kinetic Stability

GS2, …

TECH-X

Simplified Framework for Modeling of Plasma Core

- **Layered code organization**
  - Standardized interfaces for solvers, sources, transport and equilibrium modules
    - Specify dependences, inputs and output, units
  - Set standards for team of developers
  - Simplify development and implementation of new models
  - Simplify maintenance, verification and debugging of modules
  - Make the code less dependent on a particular module developer

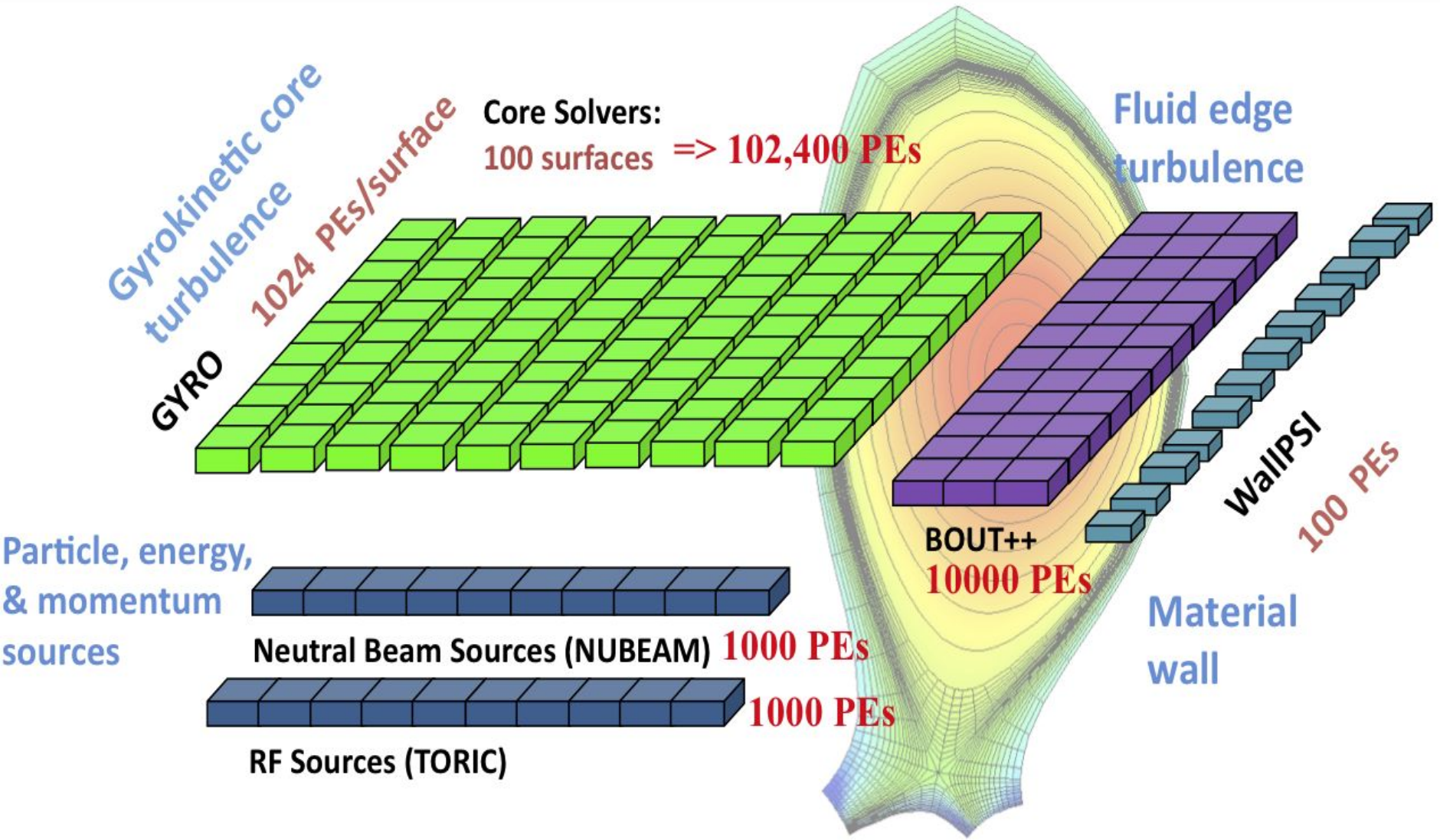


TECH-X

FACETS experience shows issue

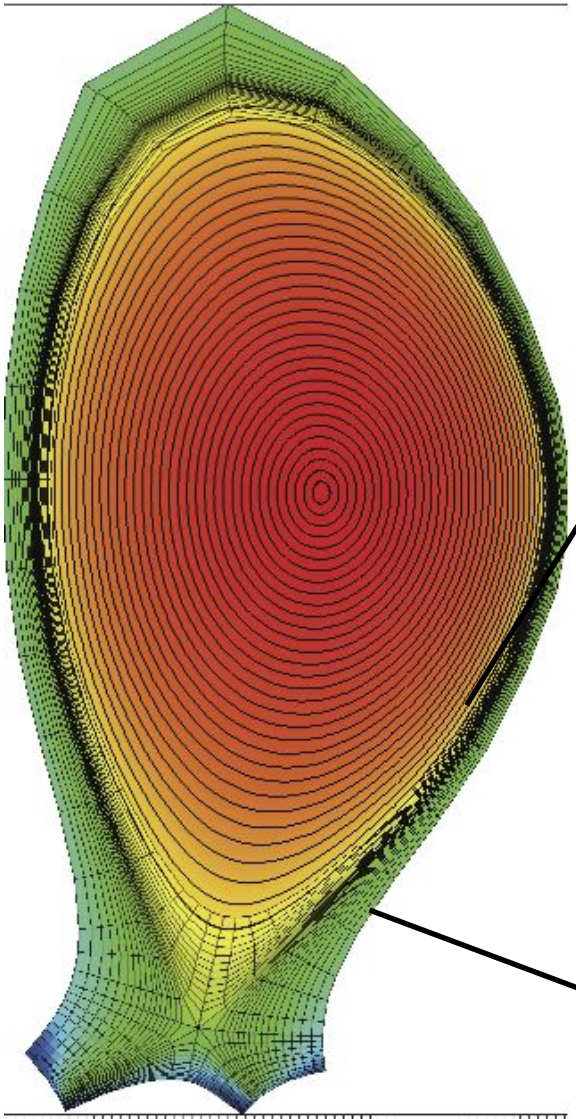
- ***Answers***
  - UEDGE: could be compiled under Linux by a LLNL employee who was not a UEDGE developer. No regression tests in repo
  - NUBEAM: McCune could compile on laptop, but some other FACETS/PPPL team members could not
  - GYRO: All developers can build on their laptops. Set of standard tests with accepted results.
- ***Fastest time to incorporation within FACETS: GYRO, NUBEAM, UEDGE in order***
  - Exact opposite from most expectations
  - Similar experience for CSWIM: NIMROD was quickly incorporation, CQL3D was longer time scale

Coupling of Core and Edge Regions in FACETS



TECH-X
SIMULATIONS EMPOWERING YOUR INNOVATIONS

“Edge region” acts as boundary condition for core region



2D TRANSPORT

UEDGE, TEMPEST,XGC0

NEUTRAL MODEL

UEDGE, DEGAS, …

RADIATION

CRETIN, …

WALL MODELING

WALLPSI, REDEP, …

3D TRANSPORT

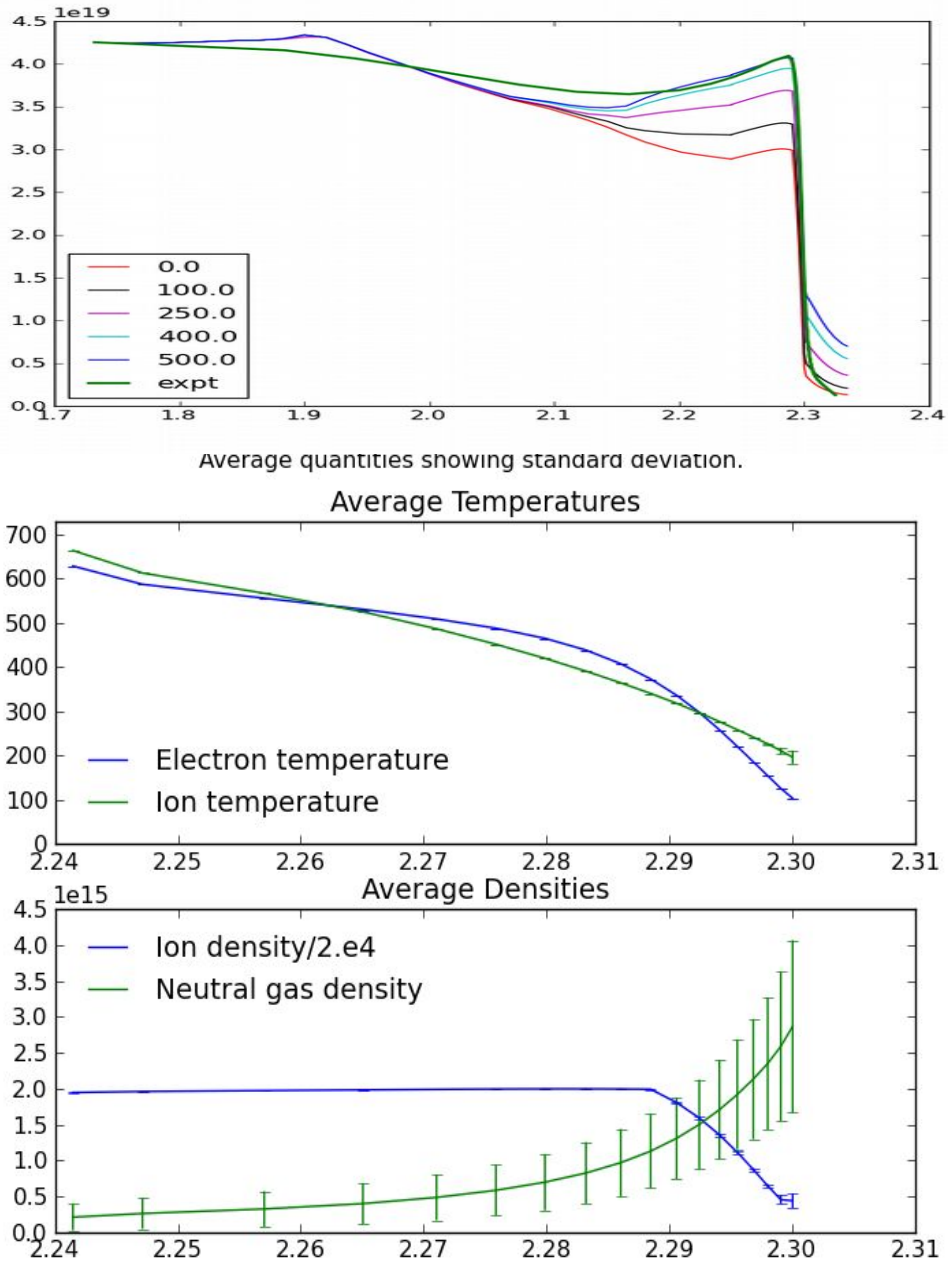
BOUT++, XGC1, …

Many ways harder than core plasmas

TECH-X

Coupling of Core and Edge Regions in FACETS

- In coupled core-edge simulations of DIII-D discharges, neutral was varied to investigate its effect on plasma profiles
  - Neutral influx has been varied up and down from this analysis value, 250 MA
  - Plasma density profile shown in purple crosses correspond to the neutral influx found from the UEDGE analysis simulations
  - Experimental plasma density profile is shown as solid gree curve.
- Plasma quantities rapidly become one dimensional, validating the coupling algorithm



Workflow: EPSI Computational Framework

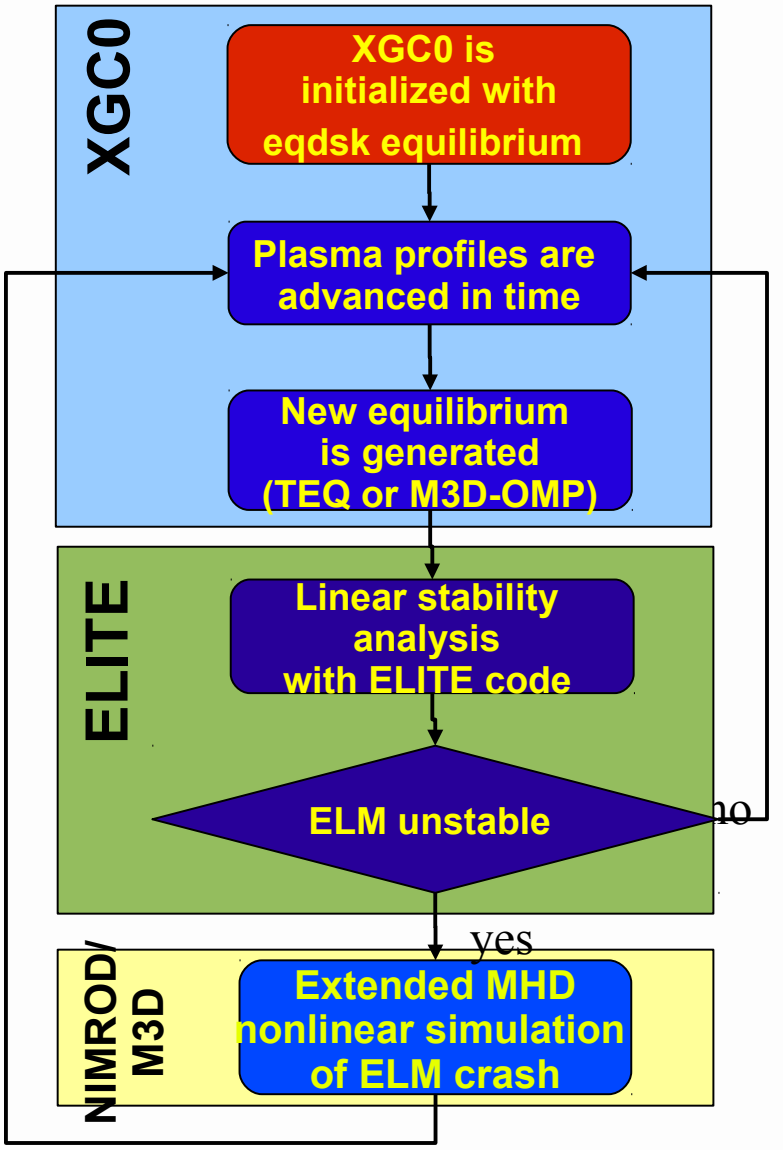
Coupled XGC0-ELITE-NIMROD framework is used in simulation of H-mode pedestal formation and ELM cycle dynamics

- Extended MHD M3D code is also available for ELM study within EPSI framework

The End-to-End Fusion Framework for Integrated Simulation (EFFIS) is used to couple the CPES codes together in one framework

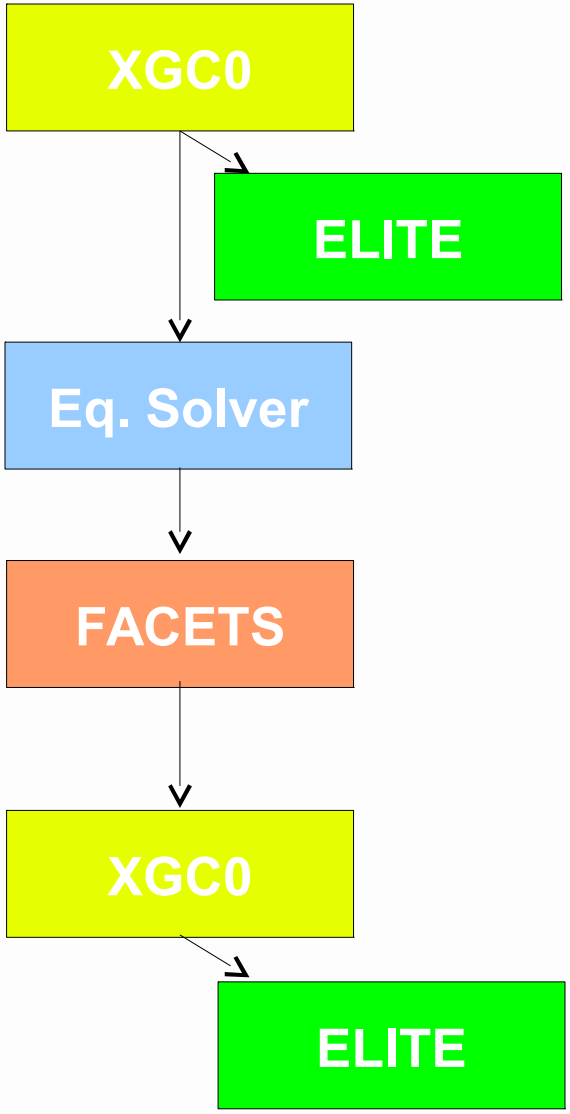
Equilibrium that is based on XGC0 plasma profiles is passed to the ELITE and NIMROD/M3D codes

The XGC0, ELITE and NIMROD codes are described below



Workflow: Coupling of XGC0, FACETS and ELITE Codes to Study the Transient Effects Triggered by Sawteeth

- Simulation starts with DIII-D equilibrium
- Pedestal buildup is modeled with XGC0
- ELM stability is monitored with ELITE
- XGC0 simulation is interrupted when pedestal reaches quasi-stable state below peeling-ballooning stability boundary
- New equilibrium is generated using M3D-OMP
- New equilibrium is sent to FACETS
- Plasma core profiles are advanced using FACETS with boundary conditions at the top of the pedestal
- The effect of sawteeth on the pedestal structure are studied by short increase (200 μs) of fluxes from plasma core in XGC0
- Stability of modified pedestal is studied with ELITE



Loose frameworks

- ***There are 3 loose frameworks under consideration in the fusion community:***
  - KEPLER: EPSI, EPS
    - Java based
  - IPS: Used by SWIM and others
    - Python
  - OMFIT: GA, AToM
    - Python
- ***Without trying to start a flamewar, these statements can generally be made:***
  - Python is the most widely used scripting within the scientific community
  - Scientific users have been able to adapt to the python-based frameworks for this reason

OMFIT

- ***OMFIT excels at:***
  - Local setup with remote execution and local analysis
  - Excellent file management
  - Simple extensibility
  - Integrating within an experimental environment

IPS

- ***IPS excels at:***
  - Time-dependent simulations with time loop controlled by python
  - Using coarse-grained components where the components are written in fortran (handling multiple runs with lack of output file namespacing)
  - Parallel load balancing of coarse-grained simulations
  - Queue structure that allows task-farming type simulations such as those needed for UQ analysis

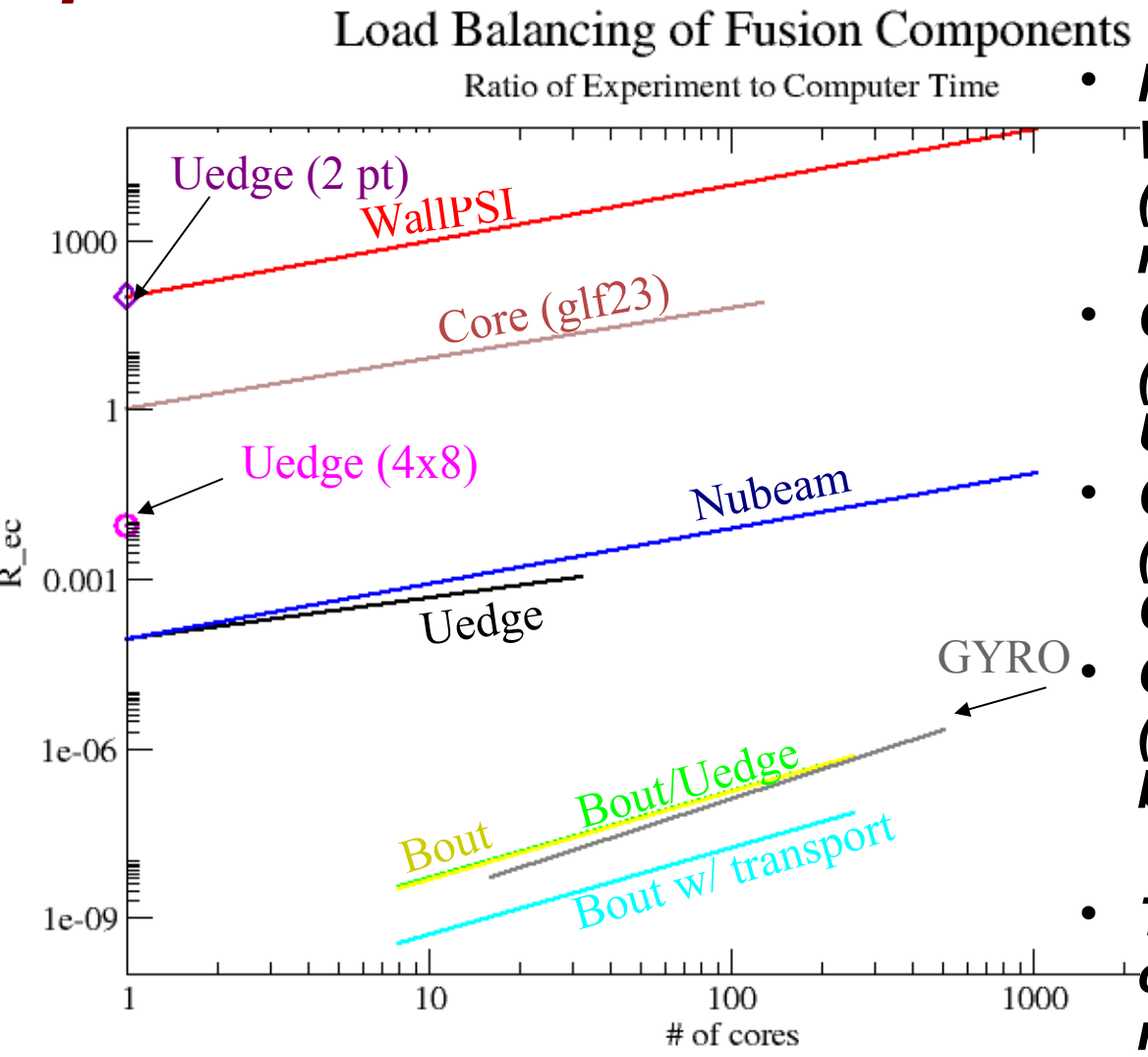
Strong Framework Desired Properties

- ***Provides infrastructure (for writing quick codes) and super-structure (for coupling codes)***
- ***Units of code (components or even sub-components) be allowed to:***
  - Compile independently of framework
  - Be tested independently of framework
  - Use common version control system

FACETS Framework

- ***Superstructure (code coupling): 703 lines***
- ***Infrastructure: 40k lines***
- ***Interfaces for codes + internal components: 31k lines***
- ***Infrastructure was robust enough to write first parallel core transport code***

Load balancing is well-known obvious problem



- ***Nubeam Core UEDGE WallPSI (Nubeam and UEDGE are relatively balanced)***
- ***Core UEDGE (lowres) WallPSI (Adjust the resolution of UEDGE to match core)***
- ***Core Two point WallPSI (faster edge model to match Core's speed)***
- ***GYRO out (various) WallPSI (GYRO and Bout are roughly balanced)***
- ***This assumes that components themselves are robust***

Example: FACETS versus TRANSP

- ***TRANSP included many codes over the years and had a larger code base than FACETS***
- ***Generally:***
  - TRANSP: Test at the TRANSP level
  - FACETS: Test individually, and then at FACETS level as well
  - TRANSP: Build all components
  - FACETS: Flexibility in component building/composition

FACETS goal: WDM for core-edge-wall using HPC resources

	Profile advance	Dynamic Eq.	Parall el NB	Parall el RF	Parallel reduced flux calcs	Embed Turb.	2D Edge Transprt	Wall modeling
TRANSP	YES	YES	YES	YES	NO	NO	NO	NO
TSC	YES	YES	YES	NO	YES	NO	NO	NO
XPTOR	YES	YES	NO	NO	NO	NO	NO	NO
CORSICA	YES	YES	NO	NO	NO	NO	NO	NO
TRINITY	NO	NO	NO	NO	NO	YES	NO	NO
TGYRO	NO	NO	NO	NO	YES	YES	NO	NO
IPS/TSC	YES	YES	YES	YES	YES	NO	NO	NO
FACETS	YES	~6 months	YES	~6 mo. w/ IPS	YES	YES	YES	~6 months

Some Lessons from Recent SciDAC projects

- **Modern integrated modeling code should**
  - **Take an advantages from the developments in other science communities such as applied math and computational scientists**
    - **UQ and new tools for design optimization**
    - **Porting to GPU platforms**
    - **Advances in frameworks and workflow developments**
    - **New techniques to store simulation data**
  - **Have a predictive capabilities for discharge optimization studies and for the design of future fusion reactors**
  - **Be flexible with respect to the discharge plasma regions that are simulated (core, pedestal, and SOL)**
  - **Computational requirements**
    - **Portable**
    - **Modular**
    - **Enhanced options to build and debug**
    - **Regression analysis**
    - **Optimized for speed (load balancing)**

Summary

- ***Integration of fusion codes offers a way of computing at the extreme scale …***
  … while offering an extreme scale of physics fidelity.
- ***Challenges to code integration is often more about the sociological issues that determine code quality of legacy components than any math/computer science/physics issue***
  - Challenges tend to be ugly stuff no one wants to here about or work on
- ***Loose coupling frameworks excel at more rapid integration and offer additional benefits for collaboration***
- ***Strong coupling frameworks introduce additional challenges but the payoff is to handle a wider disparity of code chunks at higher performance***

Summary

- ***Integration of fusion codes offers a way of computing at the extreme scale …***
  … while offering an extreme scale of physics fidelity.
- ***Challenges to code integration is often more about the sociological issues that determine code quality of legacy components than any math/computer science/physics issue***
  - Challenges tend to be ugly stuff no one wants to here about or work on
- ***Loose coupling frameworks excel at more rapid integration and offer additional benefits for collaboration***
- ***Strong coupling frameworks introduce additional challenges but the payoff is to handle a wider disparity of code chunks at higher performance***

Frameworks are plumbing. They are important, but it's still just plumbing.

[David Keyes, speaking at SIAM CSE]

Definitions of frameworks were not helpful to FSP frameworks discussion

- ***Still not entirely helpful as there are always people that say that they CAN do anything***
- *Emphasizing on what you would WANT to do*

  Kernighan (one of the originators of Unix and C):
  In college, before video games, we would amuse ourselves by posing programming exercises. One of the favorites was to write the shortest self-reproducing program. Since this is an exercise divorced from reality, the usual vehicle was FORTRAN. Actually, FORTRAN was the language of choice for the same reason that three-legged races are popular.

- *In other words: No "three-legged races"*
- *So consider definitions sufficient*

There are many issues for all code-coupling efforts

- ***All code coupling efforts face generic coupling issues***
  - "Librarification" or "componentization"
  - Generation of "glue code" for componentization
  - Interface definitions
  - Mathematical formulation
  - Human communication challenges
  - Dependency minimization
- ***In-memory coupling (strong framework) has additional challenges beyond file-based coupling***
  - Interlanguage interoperability
  - Compiler consistency for Fortran
  - Build complexity (link lines, build provenance, …)
  - Symbol collisions

Other common issues for all code coupling efforts that impacts framework

- ***Is source code access allowed or are just provided binaries sufficient?***
  - FACETS, TGYRO: Source code only
  - CSWIM, EPSI/CPES: Source code preferred
- ***Units of code (components or even sub-components) be allowed to:***
  - Compile independently of framework?
  - Be tested independently of framework?
  - Use common version control system?
- ***These issues are often ignored in discussion, but have impact on productivity***

Common issues for using legacy components

- **Useful questions to ask:**
  - Are the software developers able to compile the code on their laptops without help from another team member?
  - Are there software users from an outside institution? If yes, how do they build the code: build themselves, or use pre-built versions at select machines?
  - Are the test cases that are routinely run to ensure correct answers?