

FACETS: Framework Application for Core-Edge Transport Simulations

J. R. Cary (Tech-X, CU)

for the

FACETS TEAM

US-Japan Workshop on the Integrated Simulation
of Fusion Plasmas

January 29, 2007

framework application: an application designed to allow a series of computations with ever increasing fidelity and, therefore, to include successively more sophisticated models, in particular of each of the aspects of a fusion confinement device.



Thanks for getting me out of Colorado



5 feet of snow in 4 weeks

US-Japan Workshop on Integrated Simulation of Fusion Plasmas, January 29-31, 2007





FACETS Background

- Part of SciDAC portfolio of the Office of Fusion Energy Sciences
- Proposed in April, 2006
- Funded January 1, 2007
- Multi-institutional main project: Tech-X (Physics, CS/AM); LLNL (Physics, CS/AM); PPPL (Physics); ANL (CS/AM); UCSD (Physics); CSU (AM); ORNL (CS, perf); ParaTools (CS, perf)
- Appended SAP: GA, ORNL
- Advisory: Columbia; LBNL; IU; MIT; NYU; Lodestar





FACETS Goals

- Provide coupled core-edge-wall computational capability to the fusion community
 - at various levels of detail
 - serial and **parallel**
- Make impact on ITER
 - Improvement
 - Device selection (heating)
 - Scenario development
 - Operation
 - Analysis
- Take advantage of petascale computing facilities: **a priori parallel**
- Have FACETS broadly installed and in use (move beyond “users = developers”)



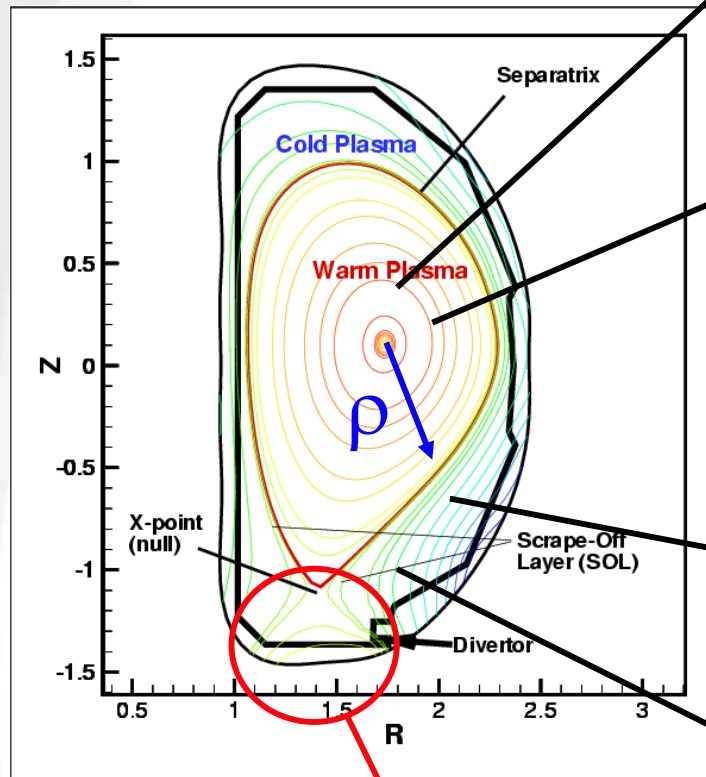


FACETS has modeling needs in each of Physics, Applied Math, and Computer science

- Physics:
 - Better models (core, edge, wall)
 - Understanding of coupling
- Applied math
 - Understand best strategies for nonlinear solves
- Computer science
 - Parallel components and strategies



Core-edge-wall integration involves multiple dimensionalities



Closed field lines: slow perpendicular + fast parallel transport

⇒ **Quantities 1D**

Hot plasma

⇒ **Collisionless, no significant atomic physics (except beams)**

Open field lines: so parallel transport must balance perpendicular

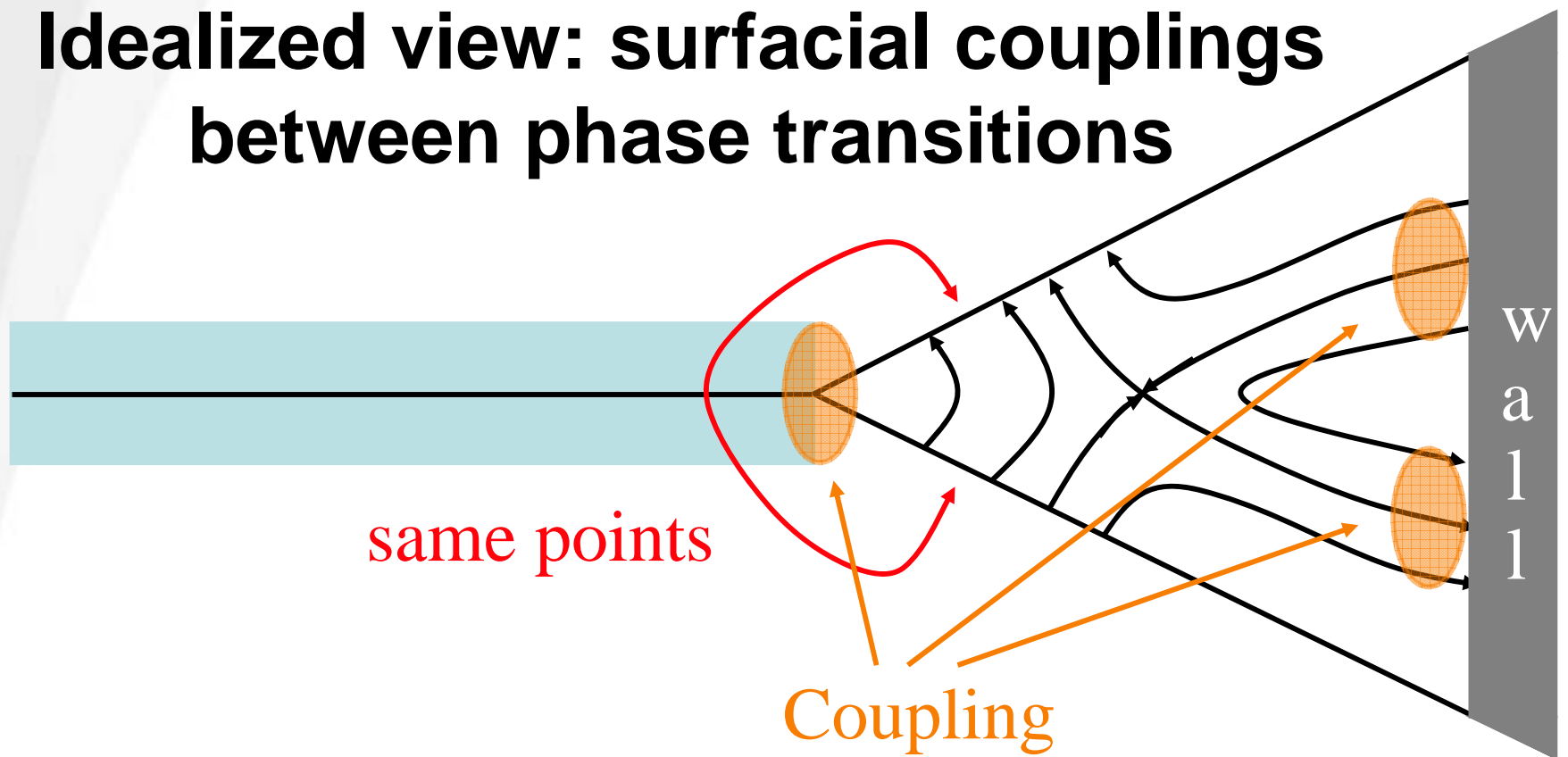
⇒ **Quantities are 2D**

Cool plasma

⇒ **Collisional, atomic physics is important**

Plasma-wall interaction is 2D

Idealized view: surfacial couplings between phase transitions

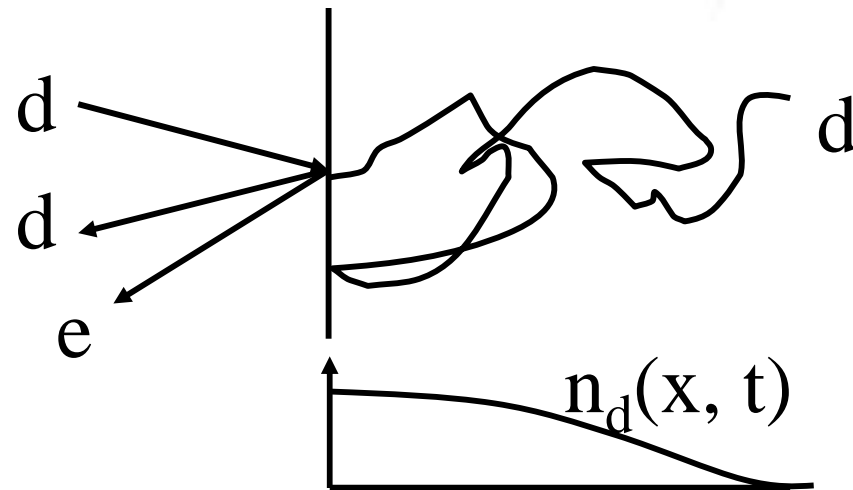


- Core is a collisionless, 1D transport system with local, only-cross-surface fluxes
- Edge is a collisional, 2D transport system
- Wall: beginning of a particle trapping matrix

Surfacial couplings



Idealized view likely okay for edge-wall interaction

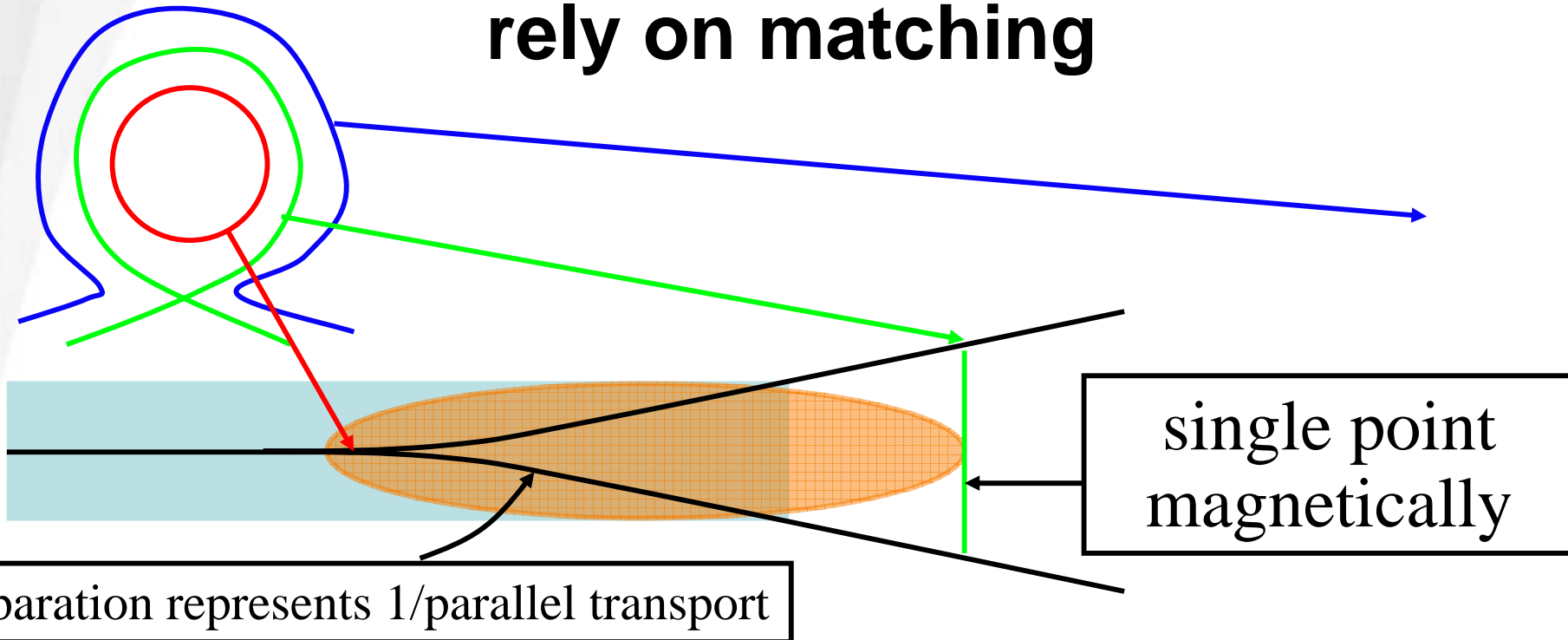


- Edge-plasma/wall analogous to atmosphere/ocean
- Wall acts as a boundary condition for edge plasma
 - Sputtering
 - Secondary electron/ion emission
- Refinement needs wall model to account for internal state
 - Wall has embedded H/D density
 - H/D diffuses in metal, both in and out
 - Impact of electrons, ions, and neutrals can cause release of embedded H/D

Valid basis for independent components



Justification for core-edge coupling might rely on matching



- Sufficiently inside the last closed flux surface, 2D effects are small
- Moving out, plasma has become collisional
- Both approximations exist - allows matching

Challenge to analytic theorists: provide matching theory

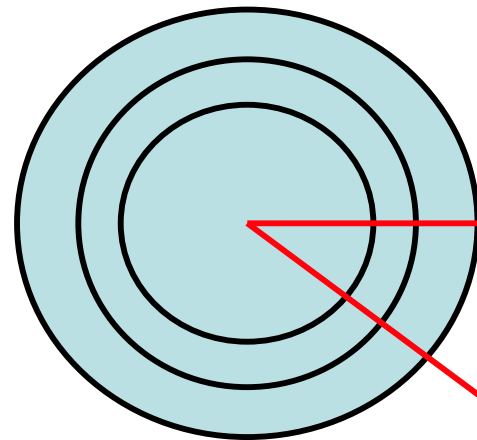


Early target: simplified core-edge coupling

- pTRANSP core transport code or at least sources
- UEDGE fluid edge code
- Coupling in a parallel environment



Zooming in identifies core needs



Core: 1D embedded in 2D

- Need modules for computing cross-surface fluxes of energy, number, momentum, ... (GLF23, ...)
- Need modules for computing deposition of energy, particles, momentum ... (McCune)
- Modules depend on shape of surfaces
- Shapes of surfaces depend on profiles
- Need to advance the profiles

Core (in part) is set of 1D PDEs

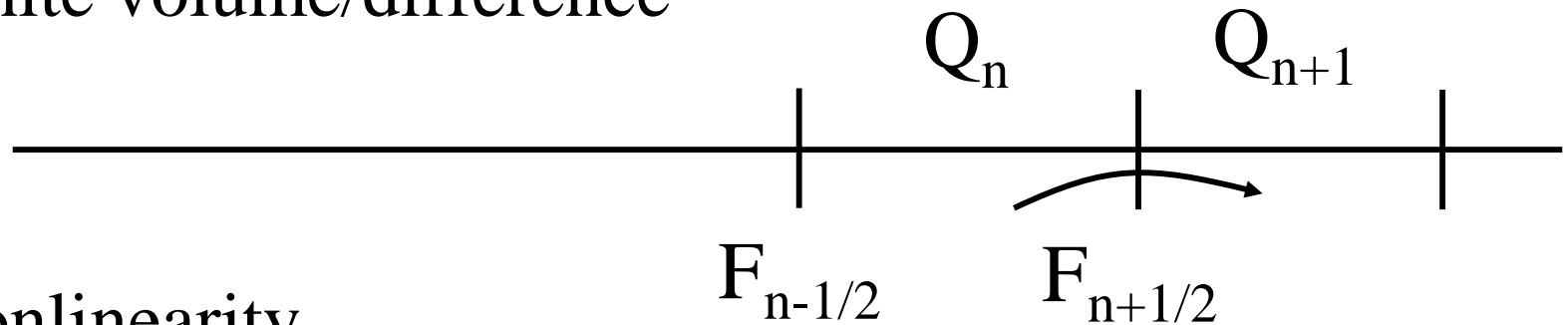
- Advance profiles in 1D

$$\frac{\partial Q(\rho, t)}{\partial t} = \frac{1}{V(\rho, t)} \frac{\partial}{\partial \rho} (VF(\rho, t)) + S(\rho, t)$$

From shape

Transport models

- Finite volume/difference



- Nonlinearity
- Locality
- Diffusion-like

$$F_{n+1/2}(Q_n, Q_{n+1}) = \text{nonlinearfunc}(Q_{n+1} - Q_n)$$

Not diffusion

Will need fast, parallel 1D block quasi-tridiagonal, nonlinear solver

- Nonlinear solve

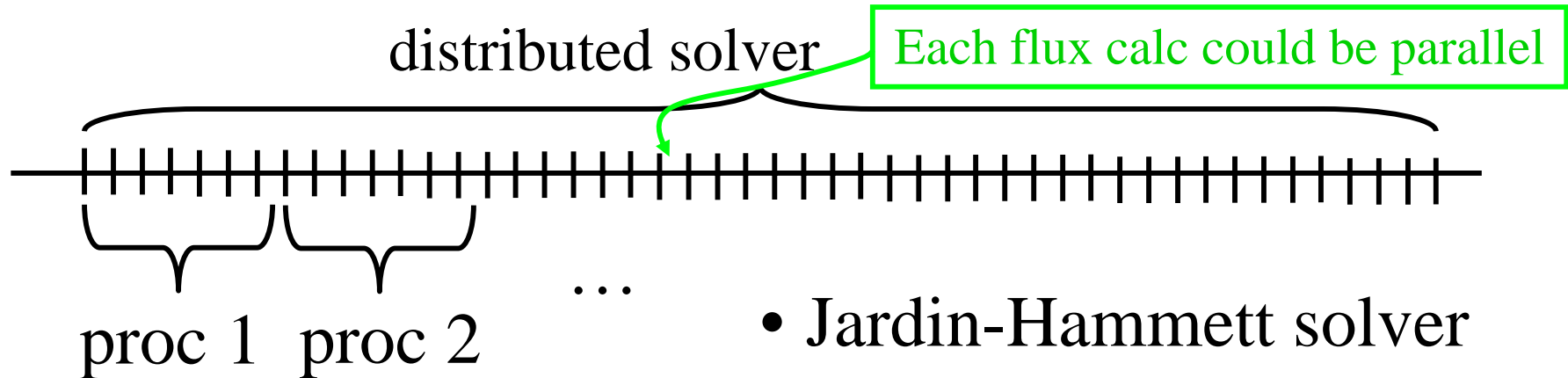
$$\frac{Q_n^{m+1} - Q_n^m}{\Delta t} = \frac{1}{V(\bar{Q})} (F_{n-1/2}(\bar{Q}_{n-1}, \bar{Q}_n) - \bar{F}_{n+1/2}) + \bar{S}(\bar{Q})$$

$$\bar{Q}_n \equiv (Q_n^m + Q_n^{m+1})$$

slow

fast

- Natural to decomp along ρ , as subcalcs of fluxes separate



- Jardin-Hammett solver
- Carlsson



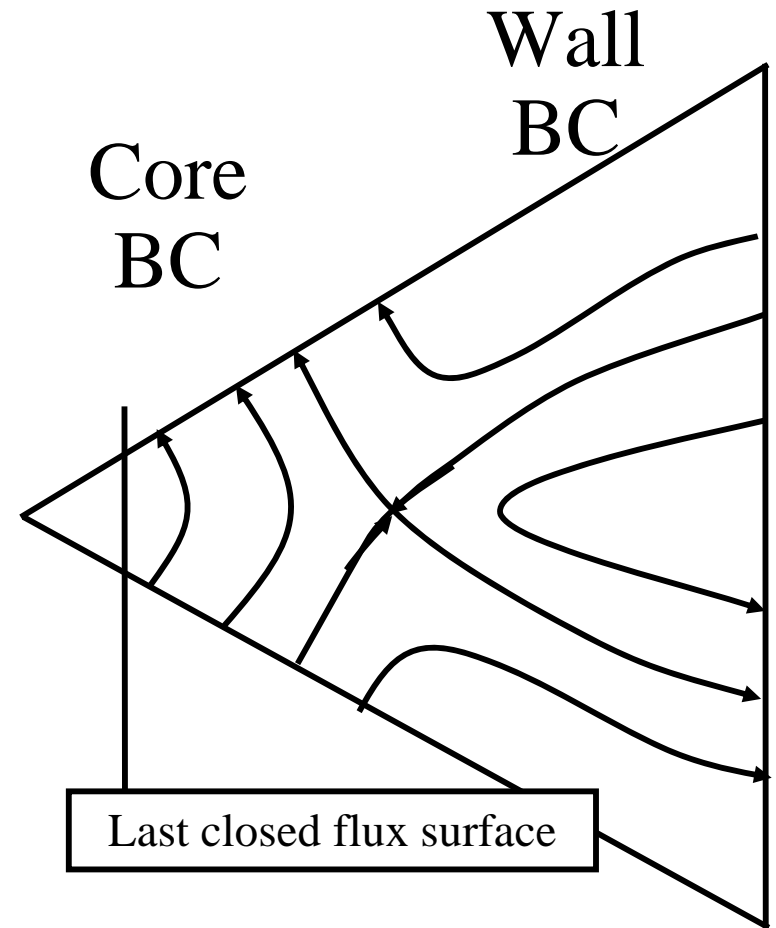
The Core 1D domain-decomp challenge

- Applied mathematicians:
 - How can we get past the (blocking) Thomas and Divide/Conquer algorithms?
- Computer scientists:
 - How do we describe the domain decomposition?
 - How do we redo it?
 - How does one work polymorphically with modules that might be parallel or serial?
- Computational scientists (domain scientists):
 - How do we get transport and source modules that work well with the magnetic surface decomposition?



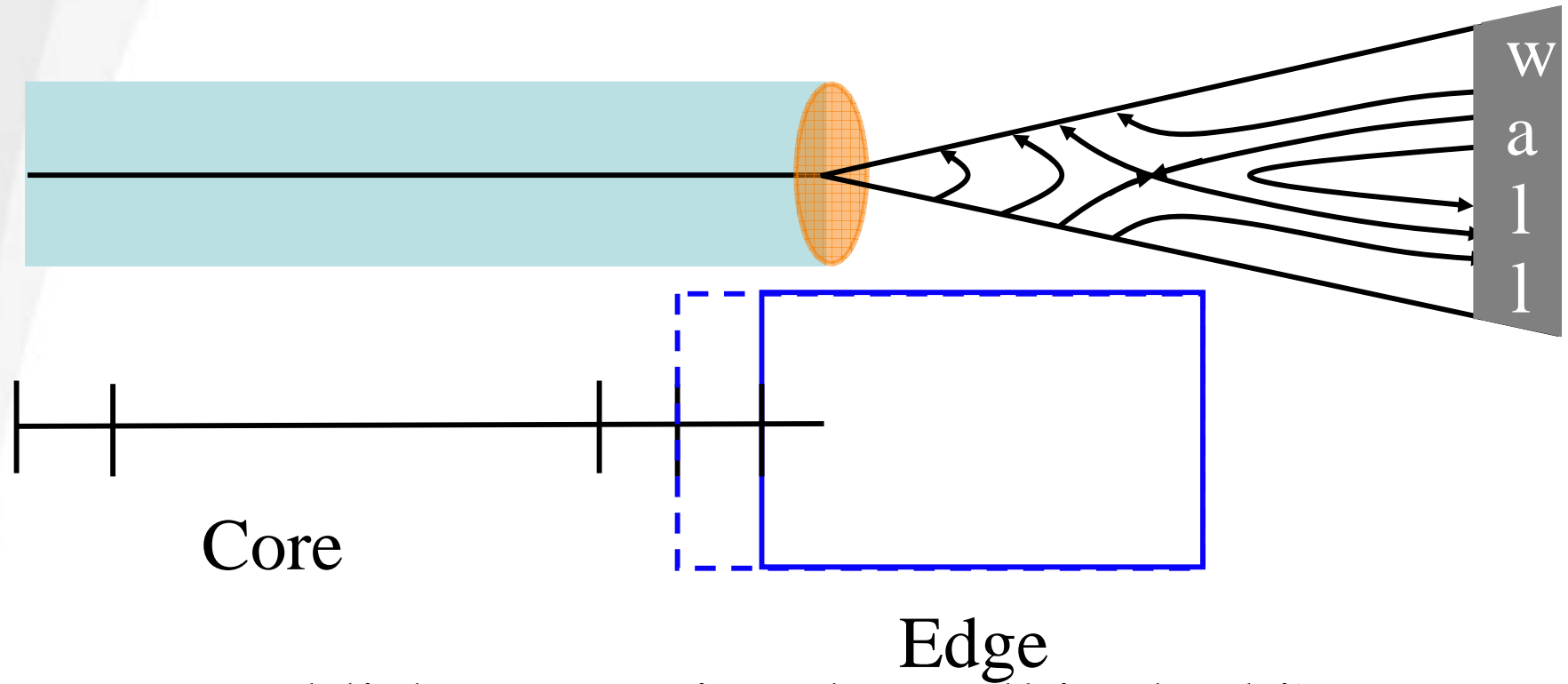
Edge can be isolated through BCs

- UEDGE
 - Electrostatic, potential determined by $\nabla \cdot \mathbf{J}$
 - Transport given by Braginskii + flux limits + anomalous transport coefficients
- Core BC: Flux or field
- Wall BC: Flux (impact/eject, wall state)
- Implicit Newton-Krylov solver to advance



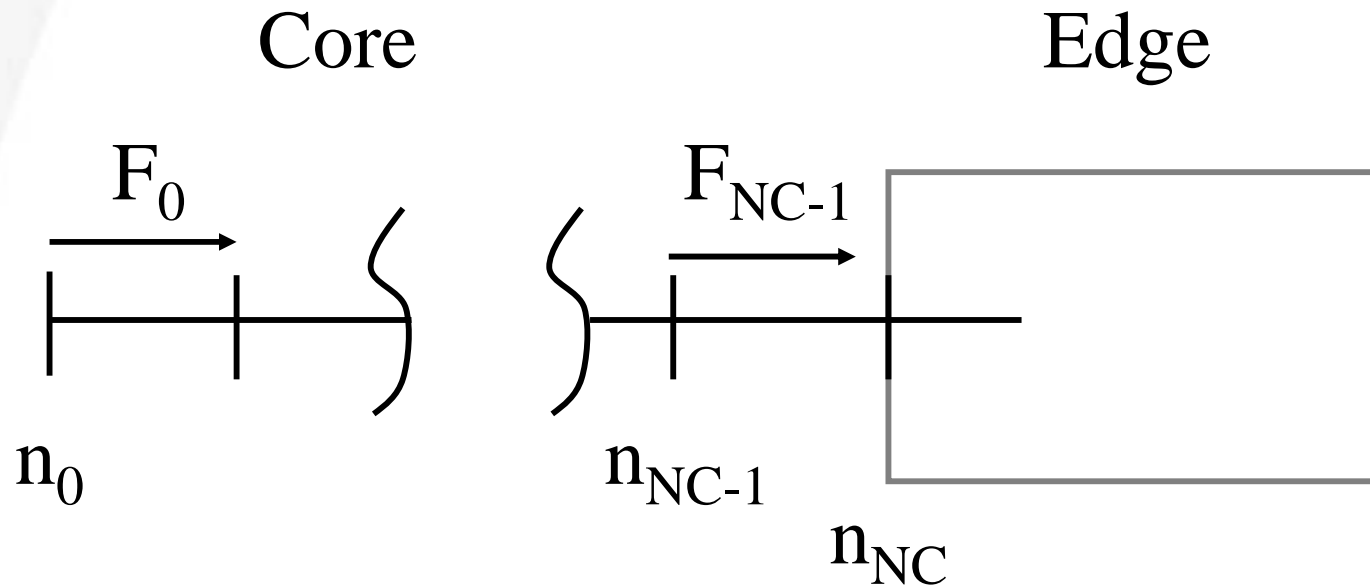
Similar aspects for wall

Communication needed for core-edge



- Establish convention that cell is $[l_0, h_i)$
 - Edge owns last cell of 1D system
 - Edge is given flux of last cell of core
 - Core is given density of first cell of edge
- (Variation to test coupling math)

Communication needed for core-edge



- Explicit
 - F_{NC-1} computed from n_{NC-1} and n_{NC} by core
 - n_{NC} computed from F_{NC-1} and internal dynamics by edge



Relaxation times very different

- Core energy relaxation time (energy confinement time)
 - $\sim s$
 - Gradient relaxation means local relaxation very fast
- Edge relaxation times very short
 - Collide into scrape-off layer
 - Stream to walls
- Implicit solver needed for coupling the two systems





Applied math: solve coupled systems

- Core gives $\mathbf{F}_{\text{NC-1}}(\mathbf{n}_{\text{NC}}, t+\Delta t)$
- Edge gives $\mathbf{n}_{\text{NC}}(\mathbf{F}_{\text{NC-1}}, t+\Delta t)$
- Which? (Parametric dependence on $t+\Delta t$ dropped)
 - $\mathbf{n}_{\text{NC}}(\mathbf{F}_{\text{NC-1}}(\mathbf{n}_{\text{NC}})) - \mathbf{n}_{\text{NC}} = 0$
 - $\mathbf{F}_{\text{NC-1}}(\mathbf{n}_{\text{NC}}(\mathbf{F}_{\text{NC-1}})) - \mathbf{F}_{\text{NC-1}} = 0$
 - Coupled
 - $\mathbf{F}_{\text{NC-1}}(\mathbf{n}_{\text{NC}}) = \mathbf{F}_{\text{NC-1}}$
 - $\mathbf{n}_{\text{NC}}(\mathbf{F}_{\text{NC-1}}) = \mathbf{n}_{\text{NC}}$
 - Expose all residuals to global solver

What determines which way to solve?





Core-edge coupling will need embedded coupling as well

- Equilibrium depends on core currents, potentially halo currents
- Equilibrium provides geometry for core and edge
- In a parallel code, how do these collocated components communicate?
 - Single equilibrium solver answering requests from 1000 other ranks?
- If parallel, to minimize communication, how?
 - Solution natural in R-Z
 - Core wants per flux surface





Summary of core-edge physics and math

- Standard core transport solver
- Edge solvers
- Coupling region where physics overlaps
- Solver to couple systems and advance implicitly
- Coupling to equilibrium





Multiple computer science needs

- Parallel component model
 - Description
 - Containers
 - Registration
 - Set up
 - Advance
- Quasipolymorphism





Multiple levels of parallelism present

- Parallel solve of coupled core-edge-wall
- Within core
 - Parallel solve over flux surfaces of transport equations coupled with equilibrium
 - Parallel computation of transport at individual surfaces
 - Coupling with equilibrium code
- Within edge
 - Parallel solve of 2D flows
- Within wall/sheath





Challenges

- Computer science: how to describe and build a hierarchy of parallel containers, each holding solvers, data, and systems, with each system being a parallel container?
- Applied math: How can one advance this system most efficiently, perhaps even allowing solves at multiple levels to proceed simultaneously?
 - and when time scales are very different....
- Domain scientists: How can one grab your systems and abstract and modify them per the needs of the project?





Hierarchies of models - but variables change

- Kinetic effects do exist in edge. To match to core, need more information
 - Always: density, temperatures
 - Sometimes: Description of velocity distribution
- Core transport models differ in detail
 - Always: density, temperatures
 - Sometimes: impurity densities, rotation

Similar for edge: neutrals model, ...

Challenge to Computer Scientists

How to build a code using modern techniques, such as polymorphism, when required parameters change “a little”? Containers? Must be fast...





One architecture choice: framework sets up connections, then steps out of way

- Example, domain decomposition
 - All ranks work independently
 - Know what to send
 - Know what to receive
 - Wait for data, begin computing
 - Send as soon as possible
- Same principles now except that instead of domains one has parallel components





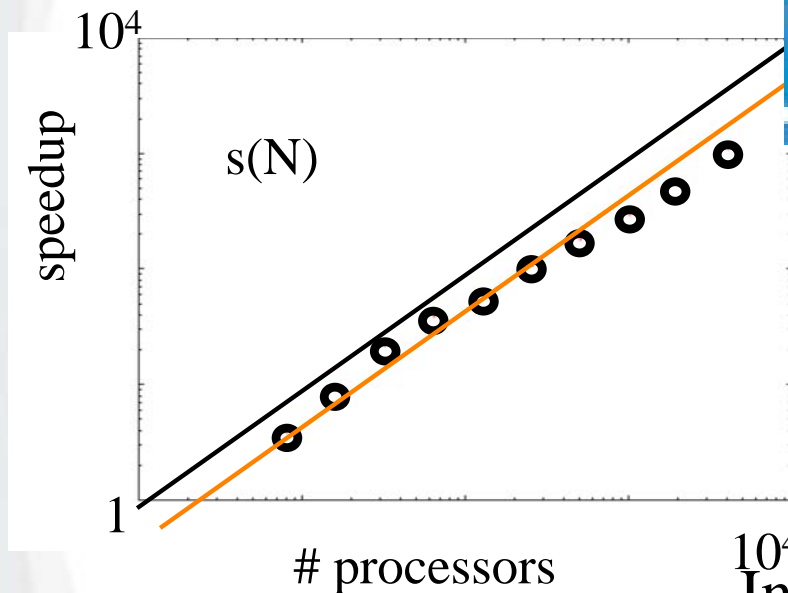
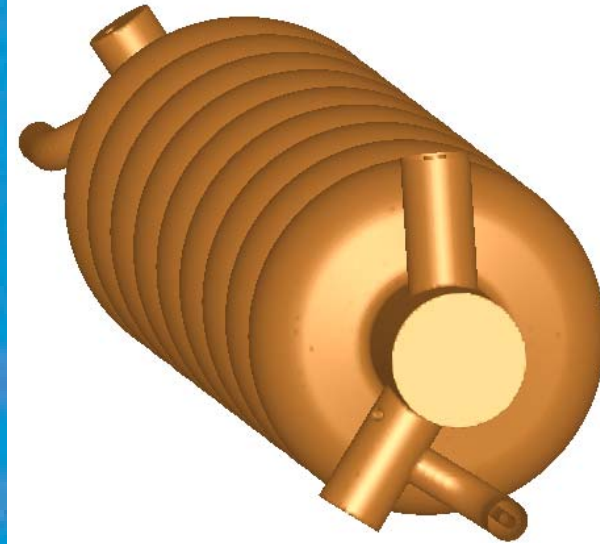
Parallel components will require protocols for setup

- Two items: core, edge
- Create core, edge, stepper
- Stepper lives across all processors
- Core lives on some processor group
- Edge lives on complement
- Assume that each component is registered somehow along with processor group. That information is global.
- In edge:
 - `firstcorerank = lookupfirstrank("core");`
 - `sendMsg(firstcorerank, "On what rank is the last flux surface");`
 - `lastfluxrank = recvMsg("On what rank is the last flux surface");`
- In core:
 - `firstedgerank = lookupfirstrank("edge");`
 - `sendMsg(firstedgerank, "Who needs info from last flux surface");`
 - `lastfluxdepranks = recvMsg("Who needs info from last flux surface");`



How do we build such an application?

- On past success
- Layering
- Data and algorithm separation
- Hierarchy

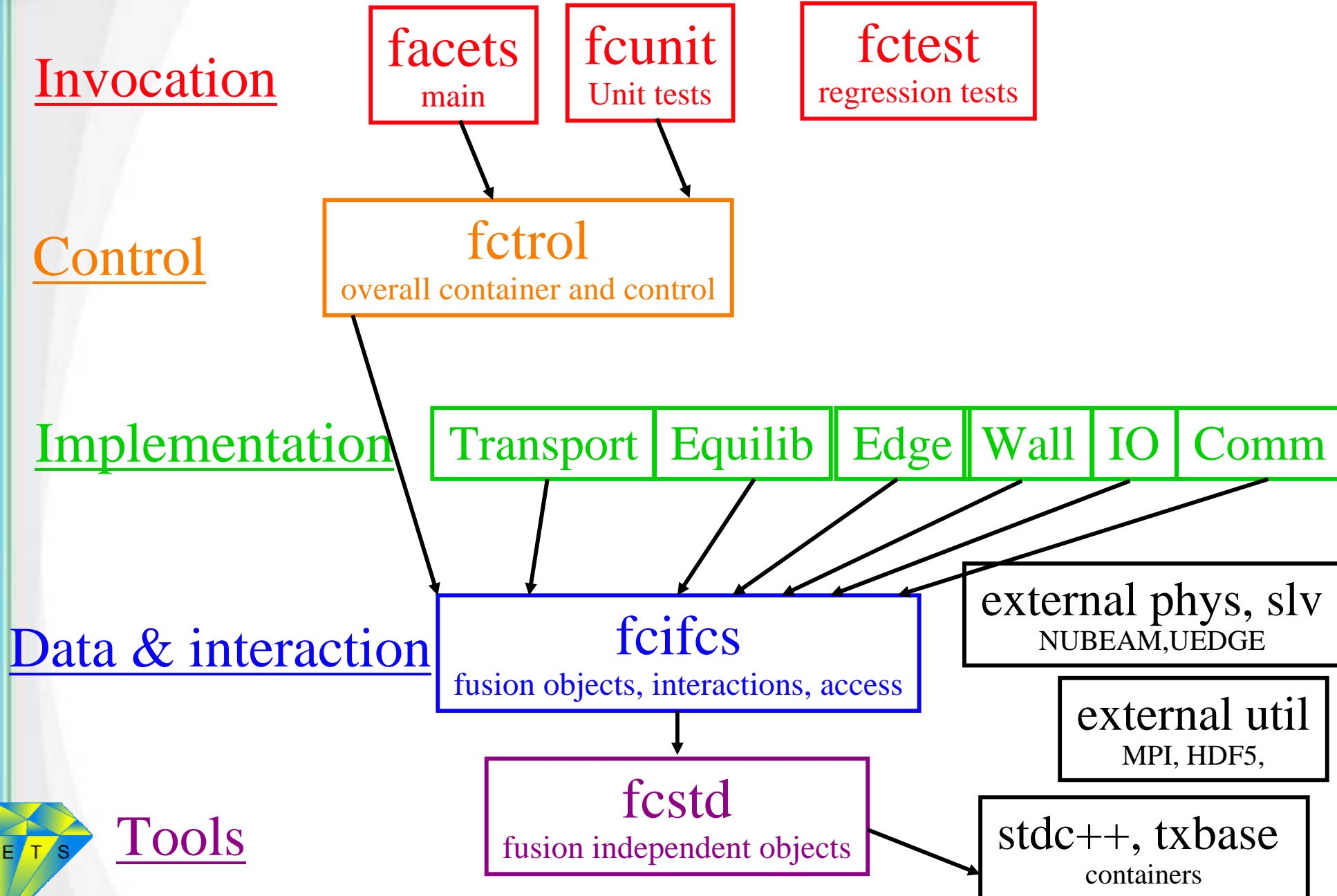


200k, highly templated C++ LOC
 Bootleg funding (SBIR)
 Seaborg scaling
 More than 25 developers: ONE code
 Installed at ANL, BNL, FNAL, Jlab, LBNL
 One of major codes of Accelerator SciDAC
 Commercial sales



Layering to improve maintenance

packages not dependent on their layer or higher





FACETS - software architecture considers three main types of relationships

- Inheritance
 - basic capabilities
 - extensions in multiple directions
 - combinations (multiple inheritance or inclusion)
- Containment or ownership
 - EM field has boundary values, which are set by functions, which have parameters
- Connections
 - who needs access to what?





TxHierAttribSet: XML-like input file allows for all three relationships

```
<Species electrons>
  kind = relBorisDF # inheritance
  emField = theComboEmField # connection
  bgEmField = externalMagField # connection
  <SVTFunc equilibDist> # containment
    kind = maxwellian # inheritance
    velocityDim = 3
    <STFunc density> # containment
      function = expression
      expression = DENSITY_MIN + DENSGRAD*(x-STARTRAMP)
    </STFunc> # closing
  ...
</SVTFunc> # closing
...
</Species> # closing
```





FACETS just starting, approach problem from multiple directions

- Top level: Have begun defining the basic components and how they interact.
- Infrastructure software: have started communication, I/O, distributed arrays, ...
- Other infrastructure: version control, release process, cross-platform build system, code integrity checks, unit test system, regression tests
- V&V: connections to data done in NTCC
- Proposal outlined project breakdown, breakdown continuing to more detailed level
 - Framework: (Tech-X, LLNL)
 - Edge (update, componentization): LLNL
 - Core (parallel sources, pTRANSP): PPPL
 - Wall: UCSD
 - Coupling: CSU, ANL
 - Performance: ORNL, ParaTools
- As project progresses, efficient coupling understood, move components to more fundamental level





Significant research component to this project, but moving towards a goal

Core, edge, and wall components, embedded in an equilibrium, updated by a solver component



I cannot forecast to you the action of **Russia** FACETS. It is a riddle, wrapped in a mystery, inside an enigma; but perhaps there is a key. That key is **Russian national interest** ITER Modeling Needs.





End