

Integrated Modeling: Evolution of Wave and
Fokker Planck Solvers Coupling
The SWIM Plasma State

**US-Japan Workshop on Integrated Simulation of
Fusion Plasmas**

Lee A. Berry

January 30, 2007

Contributors

- **Bob Harvey and CompX colleagues; Doug McCune, Fred Jaeger, Don Batchelor, Wael Elwasif, Paul Bonoli, John Wright, and Cynthia Phillips.**

Topics

- **Need for a useable and well documented “plasma state.”**
- **Manual and automated coupling of AORSA and CQL3D.**
- **The SWIM plasma state.**
- **Plans.**

The need

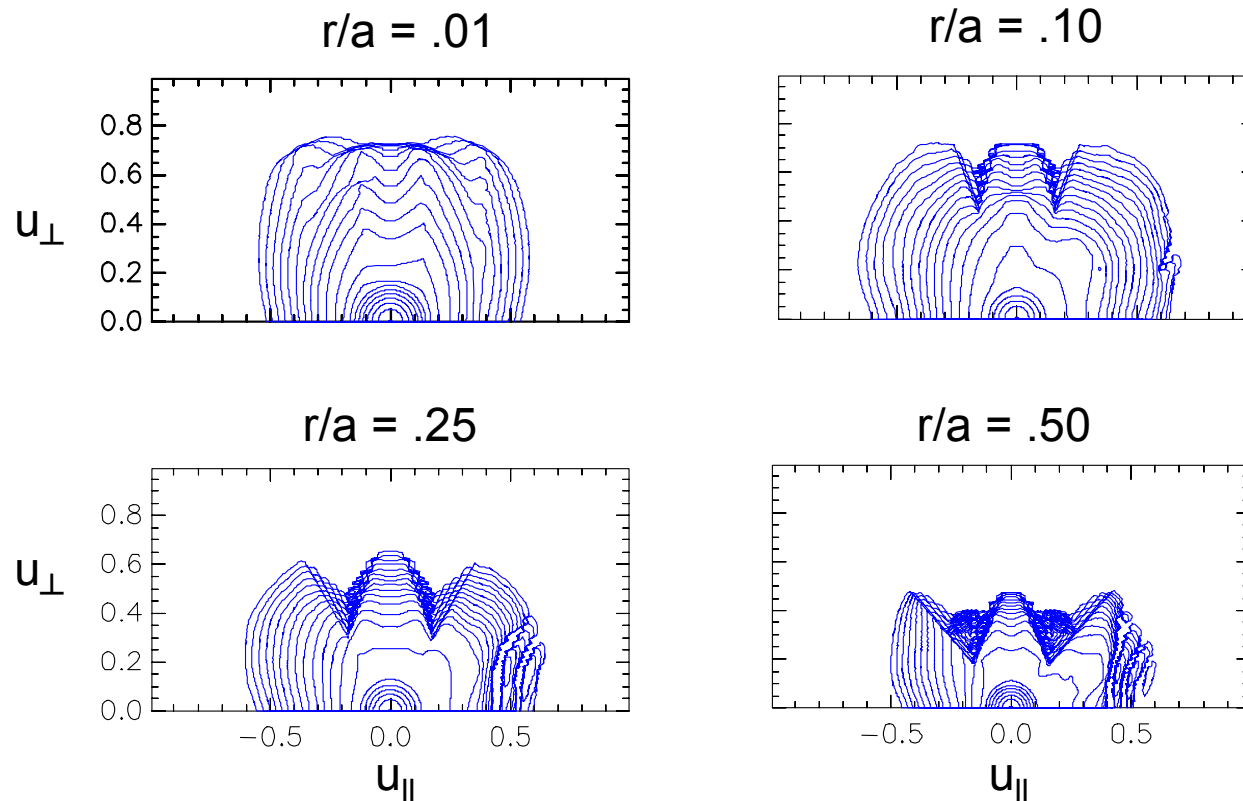
- **Simulations:**
 - **File-based communication of plasma data for computationally intensive, calculations—RF or distribution functions (for longer timescales).**
 - **In-core communication for closely-coupled modules (transport and equilibrium).**
- **“offline”:**
 - **Benchmarking of similar codes (e.g. AORSA and TORIC) using the same input data.**
 - **Comparison of reduced models with more complete models.**

Phase I. Do it by hand

- **Coupling of AORSA2D with CQL3D to study high harmonic fast wave heating of NSTX discharges with neutral beam heating.**
- **AORSA requires numerical distribution function (3D file) for fast ions from CQL3D.**
- **CQL3D requires the quasilinear diffusion tensor from AORSA2D (4D file).**
- **Procedure:**
 - **Modify input files for each code and so that same equilibrium and plasma profiles are used.**
 - **Iterate by moving files by hand and sequentially submitting an AORSA run and a CQL3D run for each iteration.**

NSTX: shot 108251 (A. Rosenberg, 2003) with beams and HHFW

Distribution function calculated with CQL3D + AORSA – B. Harvey
(finite difference Fokker-Planck code with orbits tied to a flux surface)



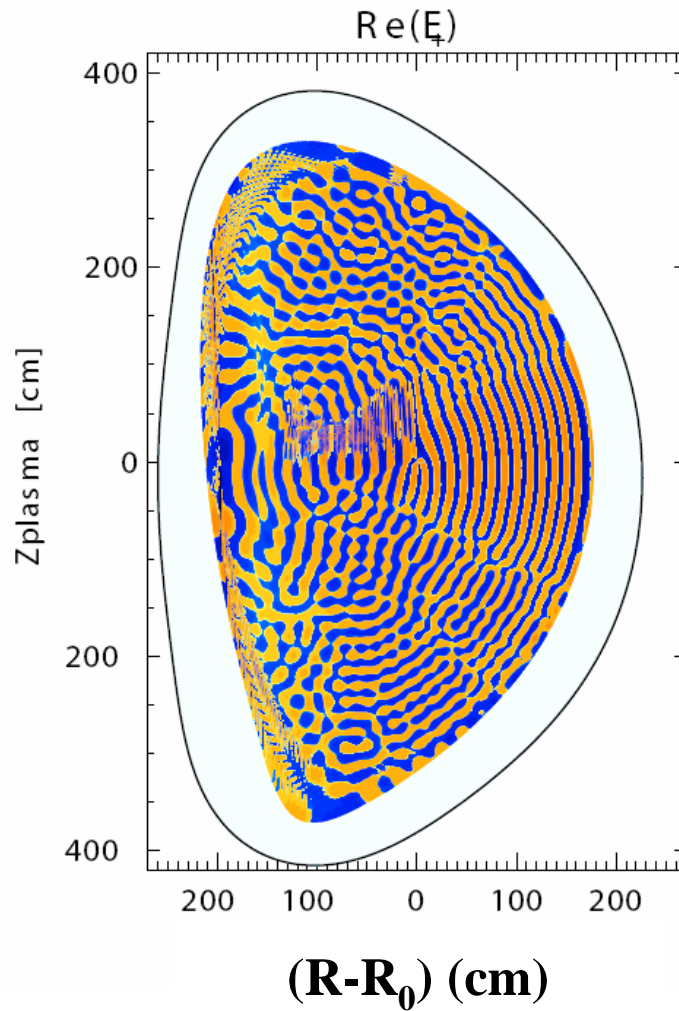
Plasma and ICRF parameters of TORIC-AORSA benchmark

Q~10 Case with Second-Harmonic Tritium heating

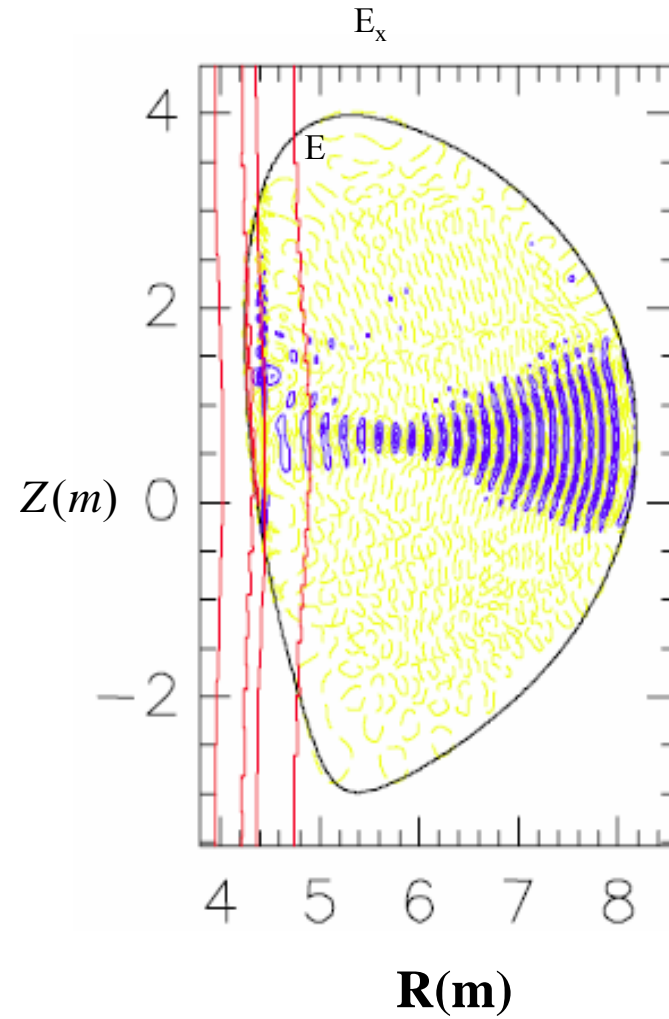
Plasma Parameters	ICRF Parameters
$n_e(0) = 1.02 \cdot 10^{20} \text{ m}^{-3}$	$f = 53 \text{ MHz}$
$T_e(0) = 24.8 \text{ keV}$	$N_\phi = 27$
$T_i(0) = 21.2 \text{ keV}$	$P_{\text{rf}} = 20 \text{ MW}$
$n_D : n_T = 31.6 : 50$	$H_{\text{ant}} = 1.76 \text{ m}$
$n_{\alpha_{\text{fast}}} / n_e = 0.76\% \text{ (on axis)}$	$R_{\text{ant}} = 8.165 \text{ m}$
$n_{\alpha_{\text{thermal}}} / n_e = 4.4\%$	$J_{\text{ant}}(\theta) = \text{Gaussian}$
$n_{\text{Be}} / n_e = 2\%$	

Comparison of perpendicular fields

TORIC



AORSA2D



Species ICRH Absorption Comparison: AORSA-TORIC

Absorption by species	TORIC Maxwellian α's	AORSA Maxwellian α's	AORSA Slowing down α's
P($2\Omega_T$)	43.1 %	38.4 %	39.1 %
P(ELD)	50.4 %	45.6 %	47.0 %
P(D)	4.2 %	4.3 %	4.5 %
P(Be)	0.6 %	3.7 %	4.05 %
P(He-4)	0.41 %	2.8 %	3.0 %
P(fast- α)	0.90 %	5.1 %	2.3 %

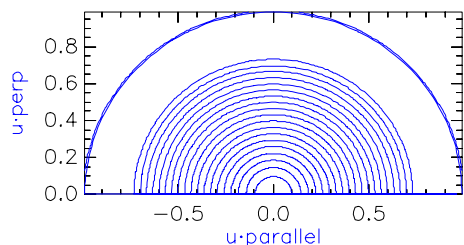
Phase II: Automate iteration with a Python program

- **Run CQL3D—`os.popen('yod -sz 4 cql3d')`.**
- **Move `cql3d` output distribution function to AORSA i/o area.**
- **Run AORSA—`os.popen('yod -sz 256 xaorsa2d.jaguar')`**
- **Move AORSA output quasi linear operator to CQL3D i/o area.**
- **Loop.**

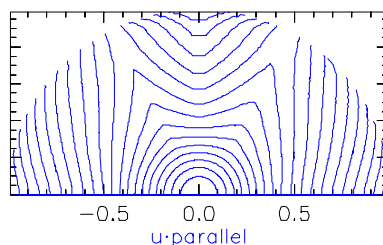
ITER second harmonic T interaction with ICRF

ITER-AT (Snowmass) with a 50-50 DT mixture and 2nd harmonic T

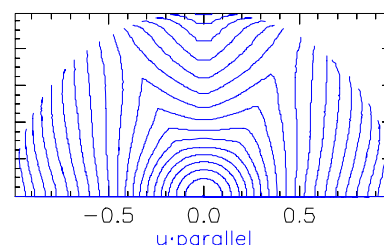
0th iteration
 $P_{RF} = 0$



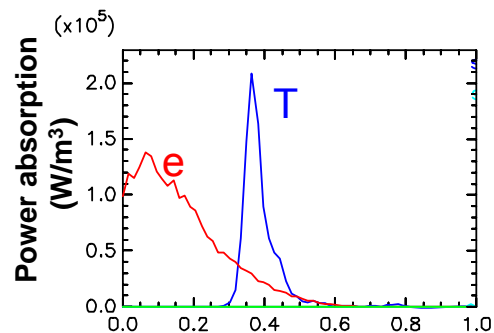
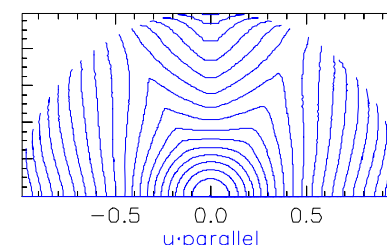
1st iteration
 $P_{RF} = 20$ MW



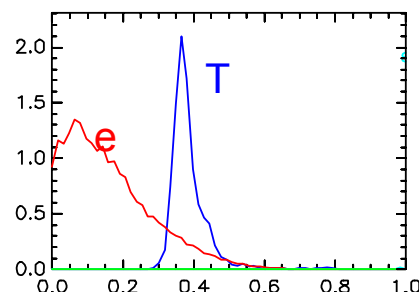
2nd iteration
 $P_{RF} = 20$ MW



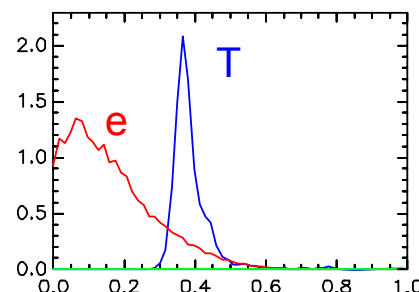
3rd iteration
 $P_{RF} = 20$ MW



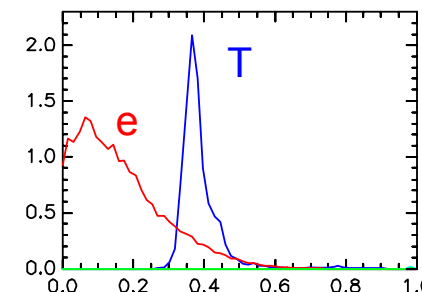
$P(T) = 8.12$ MW
= 39.6 %



$P(T) = 8.13$ MW
= 40.4 %



$P(T) = 8.12$ MW
= 40.3 %



$P(T) = 8.12$ MW
= 40.3 %

Phase III: SWIM—(Simulation of Wave Interactions with MHD)

- **A time-dependent, component-based tokamak simulation, with an emphasis on interaction of RF and MHD (and extensions).**
- **Target problems are minority ICRF + internal disruptions and ECCD + NTMs.**
- **Components include RF, Fokker-Planck solvers, linear stability, profile advance (transport and equilibrium), and neutral beam and pellet sources.**
- **Two special components are the driver and state.**

[Documents on cswim.org](http://cswim.org)

The fundamental time loop (macroscopic time steps) Is controlled by the driver

- **The driver is a script (Python) that calls a sequence of executables (for loosely coupled components). Communication via the state for these components (e.g. RF + Fokker-Planck) is via the state file.**
 - **These executables must implement three functions:**
 1. **prepare_input**—extract data from state and write to code specific input files.
 2. **Call the actual component code (e.g. xaorsa).**
 3. **process_output**—read output from code and write to state.
- **For microscopic time steps, required for closely coupled components (transport + equilibrium), the called-executable will couple components via an in-core state plasma state.**

Communication between components is via the “state”

- **Accessible via a “use plasma_state_mod” statement:**
 - e.g., `ps%ns(nspec, nrho)` contains density profiles.
- **A minimal implementation has methods to instantiate the state, load it with data, extract data, and save to file while maintaining original data.**
- **Both file and in-core communications are required.**
- **Present implementation is built on xplasma2, but direct use of xplasma calls is not necessary.**

Specifying and compiling a state

- **Specify data in a data file.**
- **Process data file with a Python code generator.**
- **Compile, link state with utilities, components (wrappers).**
- **Instantiate (specify dimensions, name species, allocate data).**

Example of swim_state_spec.dat, corresponding module definition

- **Swim_state_spec.dat:**

author: PLASMA

G rho(nrho) ! rho grid (PLASMA)

R|units=m^-3|alias=n|step ns(~nrho,0:nspec_th) ! thermal specie density

- **Plasma_state_def_mod:**

REAL(KIND=rspec), DIMENSION(:,:), ALLOCATABLE :: ns

! thermal specie density

! R|units=m^-3|alias=n|step ns(~nrho,0:nspec_th)

RF component status

- **A state initialization based on namelist and eqdsk file inputs has been implemented for testing (and regression testing).**
- **Routines to create AORSA2D data files from state and write AORSA2D output to state are running.**
- **Python scripts have been developed and tested for:**
 - **setting up file structure;**
 - **looping over AORSA2D and model transport ($T=1.1*T$);**
 - **creating directories to hold inputs and outputs for each macro timestep.**

Keeping i/o files allows for testing by code developers when things break.

Profile advance component

- **Requires in-core state interaction between transport and equilibrium because of close coupling (file-based ~10x slowdown).**
- **Talk by Chuck Kessel will present results.**

Future plans

- **Extend and refine make/install structure to “automate” state build. (.dat -> .f90 -> plasma_state_mod -> linked code).**
- **“Componetize” CQL3D, TORIC.**
- **Develop software tools to produce plasma state files from, e.g., TRANSP/pTRANSP.**