Institut für Plasmaforschung
Universität Stuttgart

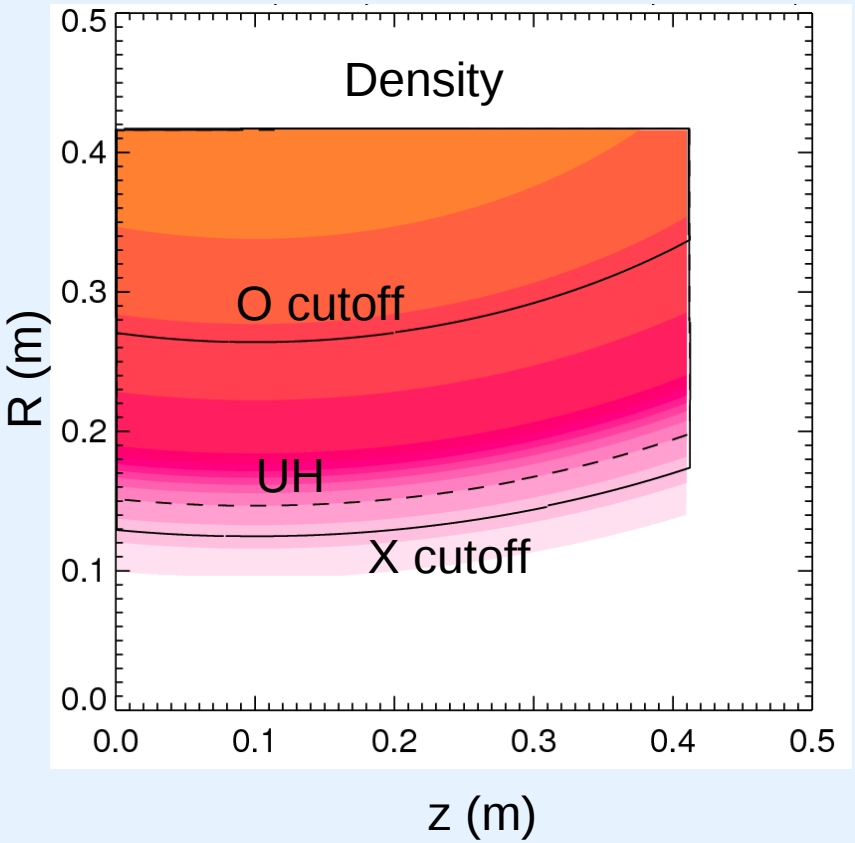
The European 3D Reflectometry Code ERC3D
Overview of structure

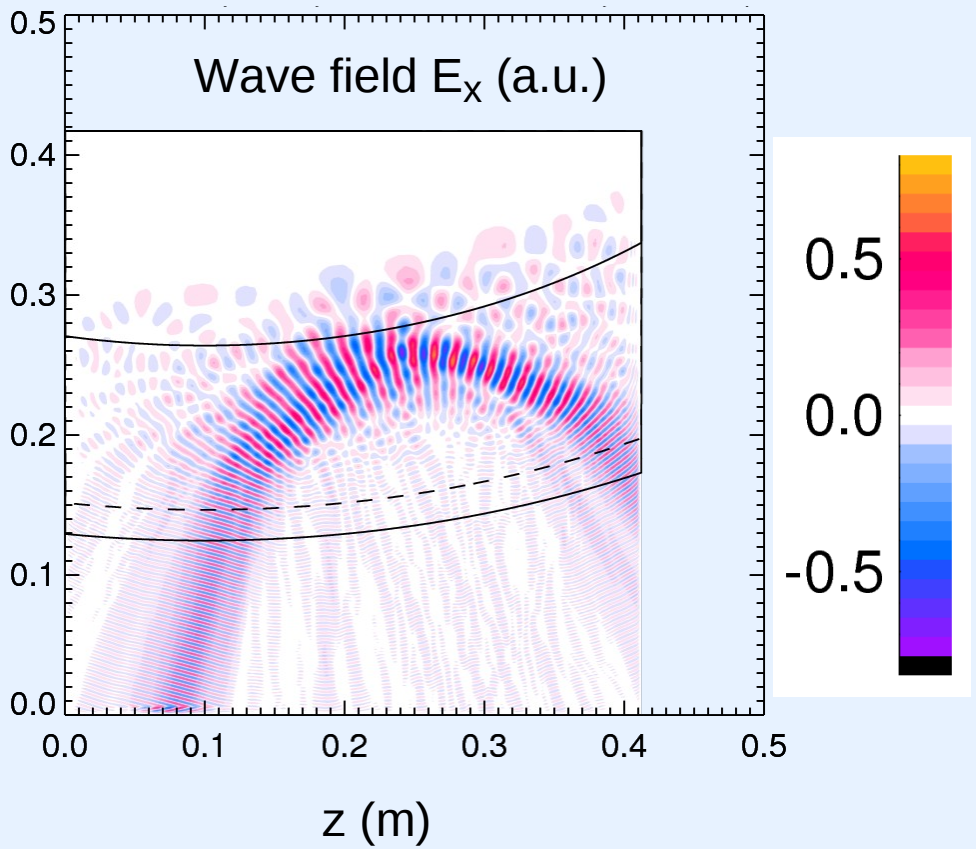
C. Lechte for the ERCC

- Main structure
- Interfaces
- Dependencies
- Challenges with ITM integration

Goal of Code: Simulate Reflectometry Wave Propagation in ITER

Main data structure: 3d arrays with Te, ne, B0, wave fields, plasma currents
Advance this state by 1 timestep dt << 1/f at a time





ERC3D Endeavour

Goal: create a

- versatile optimised reflectometry code
- for full simulation of the diagnostic "from antenna to antenna"
- and apply to turbulent plasmas

Pool expertise of all ERCC members

- Clean implementation of "best practices"
- Current emphasis on FDTD techniques, but multi-algorithm approach planned

Organisation

- Attached to ITM-EDRG, integration into ITM code infrastructure
- Coordination by IPF Stuttgart
- Division into modules, each module "owned" by a different member of ERCC

Code name: ERC3D (European Reflectometry Code 3D)

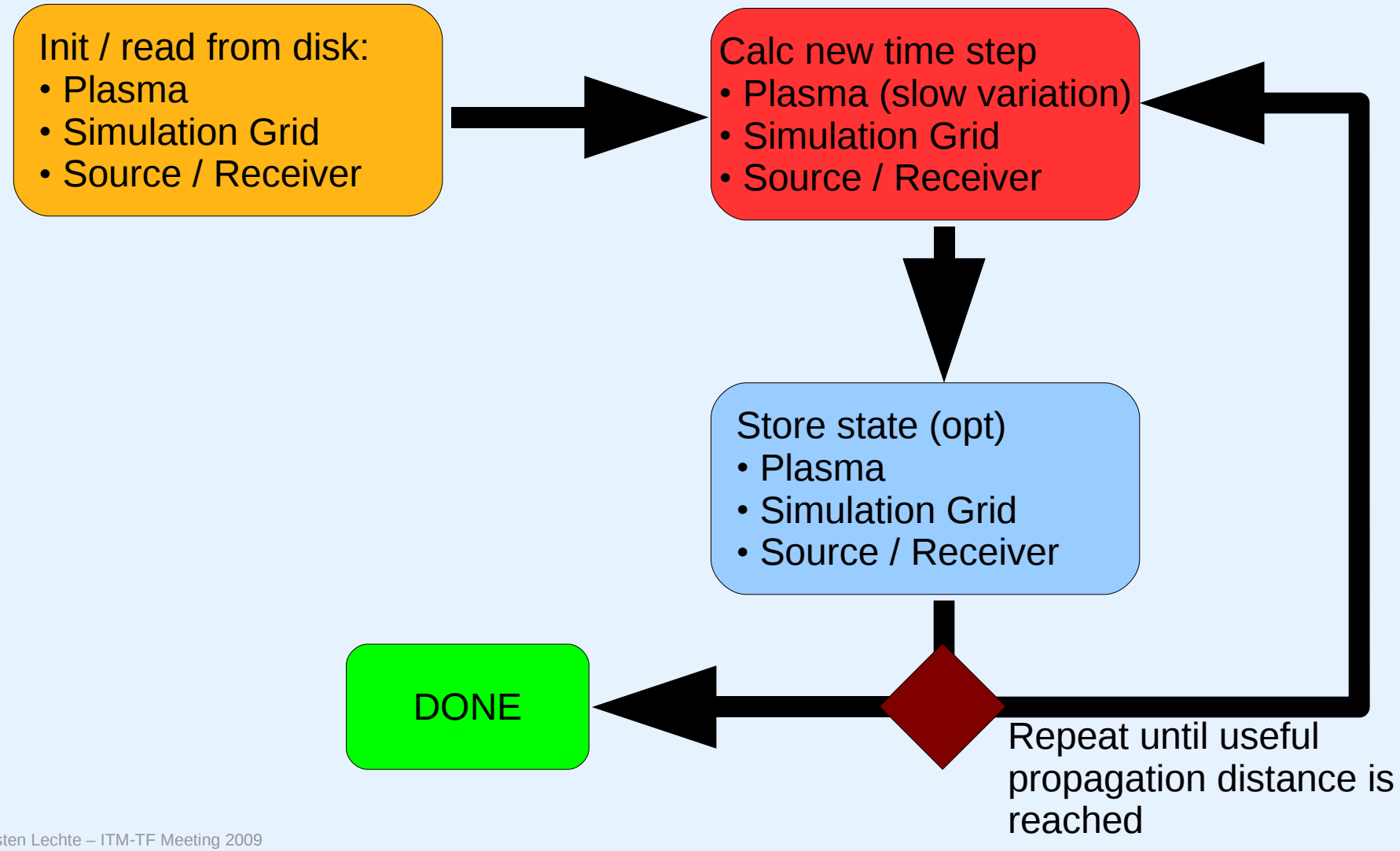
ERC3D Architecture

Group funtionality into modules

- 3D kernel module, mixture of finite difference and faster methods for propagation far from the cutoff
- uni-directional EM-wave source for X and O mode polarisation, coherent amplitude and phase detection
- plasma setup and geometry specification, interfacing to ITM tokamak equilibrium codes and numerical turbulence codes
- interface module to match the EM-wave propagation between the various schemes.
- post-processor tool-box for signal analysis

ERC3D Architecture

Typical user experience: work with 3 modules



ERC3D Architecture

Implementation features (enforced)

- Modularity
- Transparency
- Standard methods (init/save/restore/step/cleanup state of each module)
- Documentation

User writes short programs using high-level functions from the modules

- Module initialisation by various means:
  - Read init parameters from config file, feed to init function(s)
  - Program receives CPOs and uses their parameters

Module: Plasma Geometry

Defines a plasma volume and its properties

- Plasma temperature, density
- External magnetic field

Inputs

- Equilibrium CPO
- Simple analytic geometry from general parameters (from slab with linear density gradient, to toroidal geometry with tabulated profiles)
- Generates/populates a 3D grid data structure for use with the kernel propagation code

Module: Source and Receiver

Implements a send/receive horn antenna inside the simulation region

- Unidirectional wave injection seperates incoming from outgoing wave
- Antenna subroutine needs to be called in lock-step with kernel stepping function

Antenna specification

- From simple parameters
- From CPO
- Output antenna signal into special ERC3D CPO

Module: Kernel

Main simulation state is stored in kernel data structure

- The grid: 3D arrays for all fields and plasma parameters
- Also includes auxiliary fields for absorbing boundary conditions at edge of computation domain

Implements solvers

- One step advances time in grid by dt
- Save into hdf5 file
- Output of whole simulation field into a CPO?

ERC3D Architecture: Language

Implementation language is (plain) C

- Most ERCC members' expertise lies with C
- Unlike C++, standard libraries are settled (albeit less powerful)
- No single standard scientific computing framework, e.g. for array arithmetic

Integration into ITM framework poses some challenges

- Preference in ITM seems to be for Fortran
- Even with C++ interface, need to translate into C (how are dynamic multidimensional arrays represented?)

- Proposal: start work with the C++ interface, write own wrapper in C
- Who could help with this? Someone from ITM who knows C++ and CPOs

Dependencies

Open software, open standards

- Build tools (automake, autoconf, libtool...)
- Standard C library
- OpenMP, MPI for basic parallelisation
- HDF5 file formats for portable, annotated data files
- GLIB keyword/value library for configuration files
- OPEN ISSUE: Scientific computation library?

Integration into ITM frameworks

- Make CPOs available in C

Summary

- ERC3D
- Modules
  - Plasma
  - Kernel
  - Source/receiver
- Language/integration barries CPO ↔ C need to be addressed

The End

Thank you!

Work Assignments

Breakdown of module contributors supported under EFDA ERDG-T5

- C. Lechte (IPF/IPP): Module development (3D kernel) plus overall module coordination
- F. da Silva (IPFN-IST): Module development (EM wave injection techniques), algorithm development (new J-solver)
- E. Blanco (CIEMAT): Code parallelization techniques. Algorithm development (4th order J-solvers)
- S. Heuraux (UHP/CEA): Algorithm prototyping (J-solver), development of interfacing with ITM plasma equilibrium codes/databases
- S. Hacquin (CEA): Module development – with F. Silva (EM wave injection techniques)
- A. Sirinelli (UKAEA): Alternate kernel schemes, including new 3D FDTD J-solver, plasma geometry

Module:

Imp

- Motion

Divs

- Fini