RAPTOR tutorial

code version 2.0
November 2017

Federico Felici

Eindhoven University of Technology (The Netherlands)
Department of Mechanical Engineering
Control Systems Technology Group

SWISS PLASMA CENTER

Technische Universiteit
Eindhoven
University of Technology

Outline

- **Part I - This talk**
  - **Introduction to RAPTOR**
  - **Equations, physics model and numerical method**
  - **RAPTOR code structure: directories, data objects, functions**
  - **Version management - SVN**
  - **Where to get help**
  - **License agreement**
- **Part II - Try it for yourself**
  - **Work through the RAPTOR tutorials to familiarize yourself with options**
  - **Optionally try your own simulations**

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Basics of RAPTOR

- **RApid Plasma Transport simulatOR**
  - **Fast transport code time evolution of 1D tokamak plasma profiles, designed for *real-time* and *control-oriented* use.**
- **Solve coupled nonlinear PDEs for poloidal flux and Te**
  - **Assume fixed MHD equilibrium, parametrized actuators and transport**
- **Numerics**
  - **using cubic splines**
  - **implicit numerical solver.**
  - **Returns Jacobians w.r.t. state and inputs, and local linearization.**
- **Modular**
  - **Feedback controllers can be used for any actuator.**
  - **Easy to add more physics or more functionality.**
- **Language/versioning**
  - **Matlab (Simulink for real-time C code generation)**
  - **SVN for code version control**

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Equations solved by RAPTOR (1/2)

- **1D, (flux surface averaged) diffusion of poloidal flux**

$$\sigma_{\parallel}\left(\left.\frac{\partial\psi}{\partial t}\right|_{\hat{\rho}}-\frac{\hat{\rho}\dot{\Phi}_b}{2\Phi_b}\frac{\partial\psi}{\partial\hat{\rho}}\right)=\frac{F^2}{16\pi^2\mu_0\Phi_b^2\hat{\rho}}\frac{\partial}{\partial\hat{\rho}}\left[\frac{g_2g_3}{\hat{\rho}}\frac{\partial\psi}{\partial\hat{\rho}}\right]-\frac{B_0}{2\Phi_b\hat{\rho}}V'_{\hat{\rho}}j_{ni}$$

  - **Neoclassical conductivity, bootstrap current.**
  - **Current drive sources as sums of gaussians or manual profiles,**
  - **Boundary condition through total I_p or ψ_{edge},**
- **Flux surface averaged temperature diffusion**

$$\frac{3}{2}(V'_{\hat{\rho}})^{-5/3}\left(\left.\frac{\partial}{\partial t}\right|_{\hat{\rho}}-\frac{\dot{\Phi}_b}{2\Phi_b}\frac{\partial}{\partial\hat{\rho}}\hat{\rho}\right)[(V'_{\hat{\rho}})^{5/3}n_eT_e]+\frac{1}{V'_{\hat{\rho}}}\frac{\partial}{\partial\hat{\rho}}\left(-\frac{g_1}{V'_{\hat{\rho}}}n_e\chi_e\frac{\partial T_e}{\partial\hat{\rho}}+\frac{5}{2}T_e\Gamma_eg_0\right)=P_e$$

  - **Solve T_e and optionally T_i equation.**
  - **Heat sources as sums of gaussians or manually prescribed**
  - **Simple analytical sink models, + alpha power**
  - **Ad-hoc model for thermal diffusivity with various transport models.**
    - **BgB [Erba 1998], Simple analytical [Felici PPCF 2012], qlkANN [Citrin NF 2015], Critical-gradient model [Teplukhina PPCF2017]**

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Equations solved by RAPTOR (2/2)

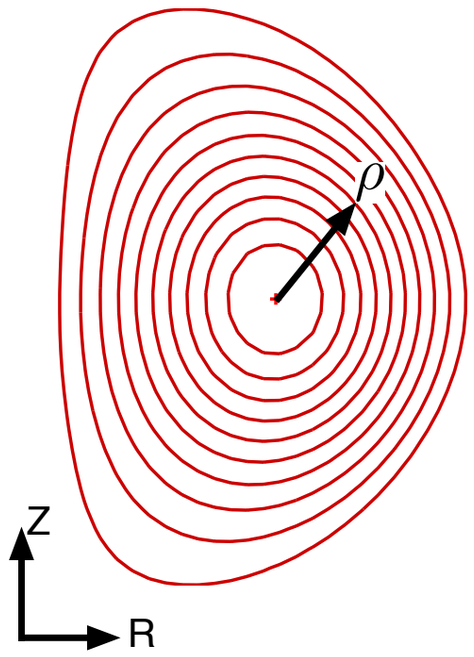
- **Flux surface averaged particle diffusion**

$$\frac{1}{V'_{\hat{\rho}}}\left(\frac{\partial}{\partial t}\right|_{\hat{\rho}} - \frac{\dot{\Phi}_b}{2\Phi_b}\frac{\partial}{\partial\hat{\rho}}\hat{\rho})[(V'_{\hat{\rho}})n_s] + \frac{1}{V'_{\hat{\rho}}}\frac{\partial}{\partial\hat{\rho}}(-\frac{g_1}{V'_{\hat{\rho}}}D_s\frac{\partial n_s}{\partial\hat{\rho}} + g_0 V_s n_s) = S_s$$

  - **For choice of species, electrons, ions, up to 3 impurities**
  - **Non-simulated species can be set as fractions of other species, directly prescribed, or constrained by Z_{eff} and quasineutrality.**

- **Magnetic equilibrium terms (V',g_0,g_1,g_2,g_3,Φ_b,..) taken from user-supplied equilibrium.**
  - **GS equilibrium not evolved self-consistently**

- **Other physics (optional)**
  - **H-mode pedestal**
  - **MHD: Sawteeth and NTM**

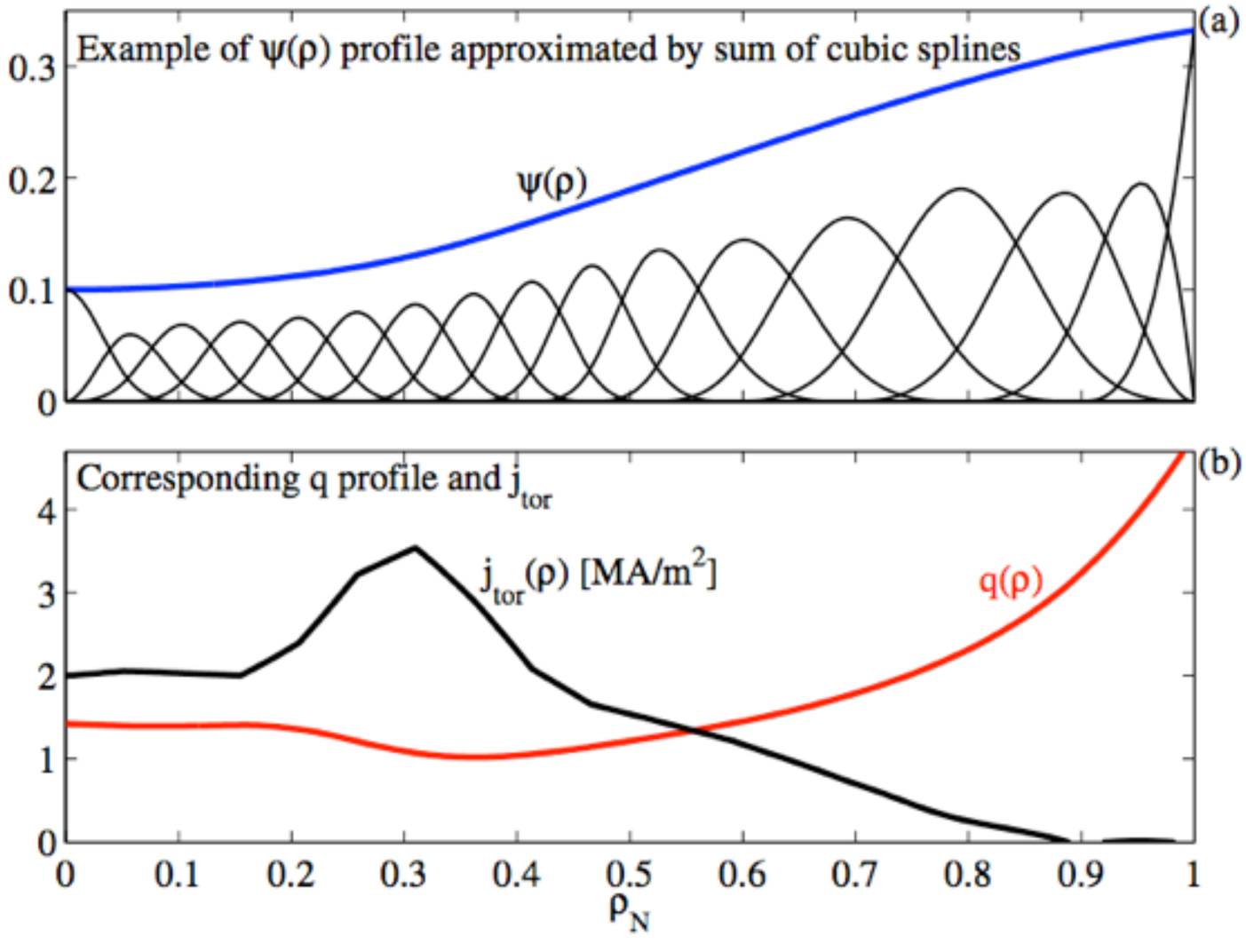


EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Some background: spatial discretization of profiles

- **Express profile as sum of basis functions**

$$\psi(\rho,t)=\sum_{\alpha=1}^{n_{sp}}\Lambda_\alpha(\rho)\hat{\psi}_\alpha(t) \quad \text{and} \quad T_e(\rho,t)=\sum_{\alpha=1}^{n_{sp}}\Lambda_\alpha(\rho)\hat{T}_{e\alpha}(t)$$



EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

From Continuous time PDE to Discrete-time matrix ODE

- **PDE with basis functions**

$$\sum_{\alpha=1}^{n_{sp}} m \frac{d\hat{y}_\alpha(t)}{dt} \Lambda_\alpha(\rho) = \sum_{\alpha=1}^{n_{sp}} \hat{y}_\alpha(t) \frac{\partial}{\partial \rho} \left[g \frac{\partial \Lambda_\alpha(\rho)}{\partial \rho} \right] + kj,$$

- **Multiply left and right by Λ_b and integrate**

$$\sum_{\alpha=1}^{n_{sp}} \frac{d\hat{y}_\alpha(t)}{dt} \underbrace{\int_0^{\rho_e} m \Lambda_\beta \Lambda_\alpha \, \mathrm{d}\rho}_{M_{\beta\alpha}(t)} = -\sum_{\alpha=1}^{n_{sp}} \hat{y}_\alpha(t) \underbrace{\left[\int_0^{\rho_e} g \frac{\partial \Lambda_\beta}{\partial \rho} \frac{\partial \Lambda_\alpha}{\partial \rho} \mathrm{d}\rho \right]}_{=D_{\beta\alpha}}$$

$$+ \underbrace{\left[g \Lambda_\beta \frac{\partial y}{\partial \rho} \right]_0^{\rho_e}}_{=l_\beta} + \underbrace{\left[\int_0^{\rho_e} \Lambda_\beta k j \mathrm{d}\rho \right]}_{=s_\beta}.$$

$$\mathbf{M} \frac{d\hat{\mathbf{y}}}{dt} = -\mathbf{D}\hat{\mathbf{y}} + \mathbf{l} + \mathbf{s},$$

NB: M,D,I,s depend nonlinearly on (y,u)

- **Temporal discretization:**

$$\dot{x}(t_k) = (x_{k+1} - x_k)/\Delta t, \qquad x(t_k) = \theta x_{k+1} + (1-\theta) x_k, \qquad u(t_k) = u_k.$$

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Implicit numerical scheme for time-evolution

- **PDE(ρ,t) -> discretize -> Nonlinear ODE at each time step:**

$$\tilde{f}(x_{k+1}, x_k, u_k) = \tilde{f}_k = 0 \; \forall \; k$$

- **At each time step, given state x_k, inputs u_k at time step k,**
  - **Take steps in Newton descent direction d**

$$\mathcal{J}_{k+1}^k d = \tilde{f}_k,$$

  - **Need Jacobian, obtained from analytical expression for all the derivatives (copious application of chain rule)**

$$\mathcal{J}_{k+1}^k = \frac{\partial \tilde{f}_k}{\partial x_{k+1}}$$

  - **Iterate until residual f_k < tolerance**
  - **Go to next time step**
- **Store Jacobians at each time step**

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Parameter sensitivity of profile evolution

- **Time evolution depends on mode parameters**
  - **One example: a transport model parameter**
  - **Another example: a parameter defining the input trajectory**

$$\tilde{f}(x_{k+1}, x_k, u_k) = \tilde{f}_k = 0 \; \forall \; k$$

- **Differentiating with respect to parameter *p*, we get the *sensitivity equation***

$$0 = \frac{\mathrm{d}\tilde{f}_k}{\mathrm{d}p} = \frac{\partial \tilde{f}_k}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial p} + \frac{\partial \tilde{f}_k}{\partial x_k} \frac{\partial x_k}{\partial p} + \frac{\partial \tilde{f}_k}{\partial u_k} \frac{\partial u_k}{\partial p} + \frac{\partial \tilde{f}_k}{\partial p}$$

  - **Linear ODE for dx_k/dp, solve while evolving nonlinear PDE: *Forward sensitivity analysis***
  - **Jacobians df_k/dx_k, df_k/dx_{k+1} are known from Newton iterations**
  - **Computational cost proportional to *p***
- **dx_k/dp gives the linearization of the state trajectories in the parameter space**

$$T_e(\rho, t)|_{p=p_0+\delta p} \approx T_e(\rho, t)_{p_0} + \frac{\partial T_e}{\partial x} \frac{\partial x}{\partial p} \delta p$$

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Equilibrium definition

- **Presently, RAPTOR uses profile information from a CHEASE run (2D fixed-boundary GS solver)**
  - **$V'(\rho)$, $g_{0,1,2,3}(\rho)$ etc**
- **Pre-calculated equilibria exist for TCV, AUG, JET, ITER**
  - **May scale them with R_0, B_0 as necesasry**
- **Analytical equilibria could be added.**
- **For other equilibria, will need to run CHEASE**
  - **talk to F. Felici or O. Sauter**
- **2-way coupling to general 2D equilibria (from fast GS solver or parametrized profiles) envisioned.**

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

RAPTOR data objects: structures

- `config`: contains configuration info for generating RAPTOR model and parameter structures
  - Physics (transport coefficients, heating&current drive, boundary conditions, ...)
  - Numerics (radial and temporal grid, spline order, solver tolerances, ...)
  - Environment (paths etc), run-time plot and display options
- `model`: contains all pre-calculated data (matrices etc) that should not be changed manually.
  - Spline basis matrices, radial grid, boundary condition type, transport model type
  - Fixed profiles/quantities which do not change in time.
- `params`: contains all parameters that can be tuned between runs.
  - Time grid
  - Parameters for transport models, other physics modules
  - Numerics / debugging flags
- `init`:parameters for generating initial conditions

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

RAPTOR data objects: time-varying vectors

- `x`: state quantities which the code solves for.
- `v`: pre-known but time-varying kinetic profile quantities.
- `g`: geometry-related quantities
- `u`: Actuator inputs

- **In terms of these objects, a time step in the code solves**
  - $x_{k+1} = f(x_k, g_k, v_k, u_k, \text{model}, \text{params})$

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

RAPTOR workflow

get defaults

```
config = RAPTOR_config
```

user : optional modification by user

user

Generate RAPTOR model, g,v,

```
[model,params,init,g,v,U] = build_RAPTOR_model(config)
```

user

user

initial conditions

```
[x0] = ...
RAPTOR_initial_conditions(...
model,init,g,v)
```

user

```
simres = RAPTOR_predictive(x0,g,v,U,model,params);
```

run

```
out=RAPTOR_out(simres,model,params);
```

post-processing

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Directory structure

- **RAPTOR/**
  - **`code`: core code files.**
  - **`tests`: tests to check that code runs correctly**
  - **`demos`: demonstration files and tutorials.**
  - **`doc`: documentation**
  - **`equils`: equilibrium files defining equilibrium profiles.**
  - **`projects`: projects that are built "on top of" RAPTOR**
  - **`optimization`: Tools for nonlinear optimization**
  - **`data`: store temporary data here (not versioned)**
  - **`personal`: store personal scripts here (not versioned)**
  - **`license`: license agreement**
  - **`RT`: Real-time implementations (Simulink)**
  - **`tokamakdensitymodel`: a related code for density evolution**
    - **[Blanken FusEngDes 2017]**

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Version control - SVN (Subverison)

- Hosted on SPCSVN at SPC-EPFL Lausanne (T.M. Tran)
- Code is developed in trunk, regular “tags” at stable releases
  - This presentation applies to tags/release-2.0
- Many tutorials exist for svn. Important commands (UNIX):
  - Check out the RAPTOR trunk from server to local directory
    - `svn co https://spcsvn.epfl.ch/repos/RAPTOR/tags/release-2.0 ./RAPTOR`
  - Check differences between local and current version
    - `svn stat -uv`
  - Revert to version on server (beware! undoes local changes!)
    - `svn revert my_file.m`
  - Commit local changes to server
    - `svn ci my_file.m -m“Comment”`
  - `Get latest updates`
    - `svn update`
- Contact F. Felici to get read/write access to SVN repository

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

Where to get help

- **Getting started: tutorials and demo files**
  - **in the /demos directory, type ‘help demos’ and do the tutorials**
  - **Run some standard scenarios in /demos/standard_scenarios.**
- **Explanations on the meaning various parameters:**
  - **As comments next to default value definition.**
    - **RAPTOR_config (for general parameters)**
    - **Inside the module to which the parameter applies (for the rest)**
- **Physics and numerics**
  - **F. Felici thesis, online http://dx.doi.org/10.5075/epfl-thesis-5203**
  - **F. Felici NF 2011 http://dx.doi.org/10.1088/0029-5515/51/8/083052**
  - **F. Felici PPCF 2012 http://stacks.iop.org/0741-3335/54/i=2/a=025002**
- **Equation details**
  - **RAPTOR equation document in /doc directory**

- **E-mail: f.felici@tue.nl**

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology

User's license and conditions

The undersigned has received a copy of RAPTOR under the conditions that:

1.- The code does not change its name even if modified.

2.- Modifications of the code that are developed are made available to the CRPP.

3.- Results produced with the original or the modified versions of RAPTOR should appropriately reference the original publications:

For use of RAPTOR as a real-time interpretative code:
F. Felici et al. Nuclear Fusion **51**(8), p.083052 (2011)
For the predictive version of RAPTOR or its use in nonlinear optimization routines:
F. Felici et.al. Plasma Physics and Controlled Fusion **54**(2), p.025002 (2012)

4.- RAPTOR nor its progeny may be transferred or made available to other research groups without the written authorisation from the CRPP.

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER
TU/e Technische Universiteit Eindhoven University of Technology