RAPTOR tutorial

code version 1.3
November 2013

Federico Felici

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

Technische Universiteit
Eindhoven
University of Technology

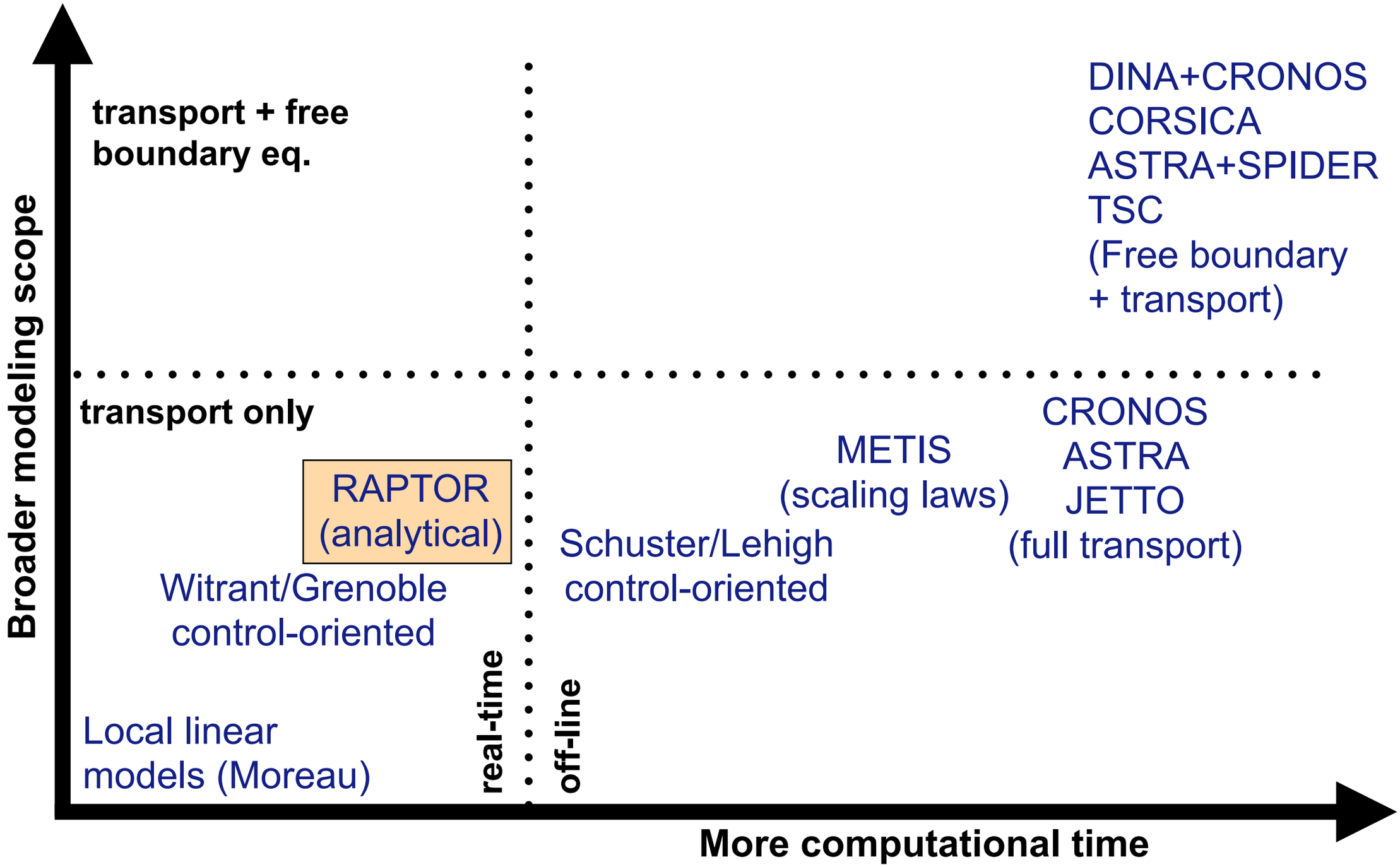
Outline

- **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**

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**

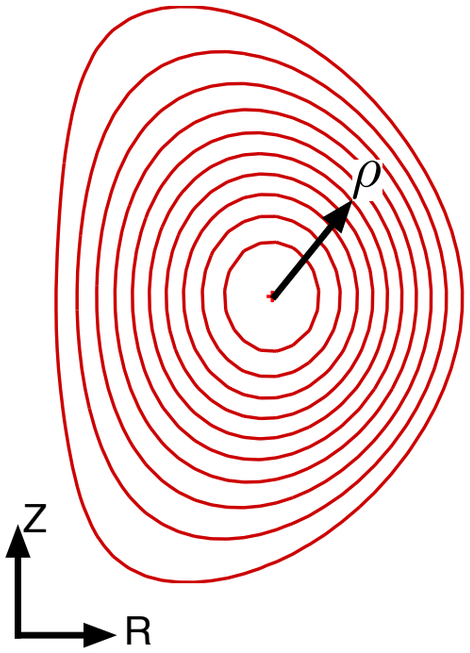
The tokamak transport simulator zoo



Equations solved by RAPTOR

- **Noncircular, axisymmetric, fixed toroidal flux surface shape**
- **1D, (flux surface averaged) diffusion of poloidal flux**

$$\sigma_{||}\frac{\partial\psi}{\partial t} = \frac{R_0 J^2}{\mu_0 \rho}\frac{\partial}{\partial\rho}\left(\frac{G_2}{J}\frac{\partial\psi}{\partial\rho}\right) - \frac{V'}{2\pi\rho}(j_{BS}+j_{ext})$$



  - **Neoclassical conductivity ~$T_e^{3/2}$**
  - **Bootstrap current ~ ∇p_e**
  - **Current drive sources as sums of gaussians**
  - **Boundary condition through total I_p or ψ_{edge}**
- **Flux surface averaged electron temperature diffusion**

$$V'\frac{\partial}{\partial t}[n_e T_e] = \frac{\partial}{\partial\rho} n_e \chi_e \frac{\partial T_e}{\partial\rho} + V' P_e$$

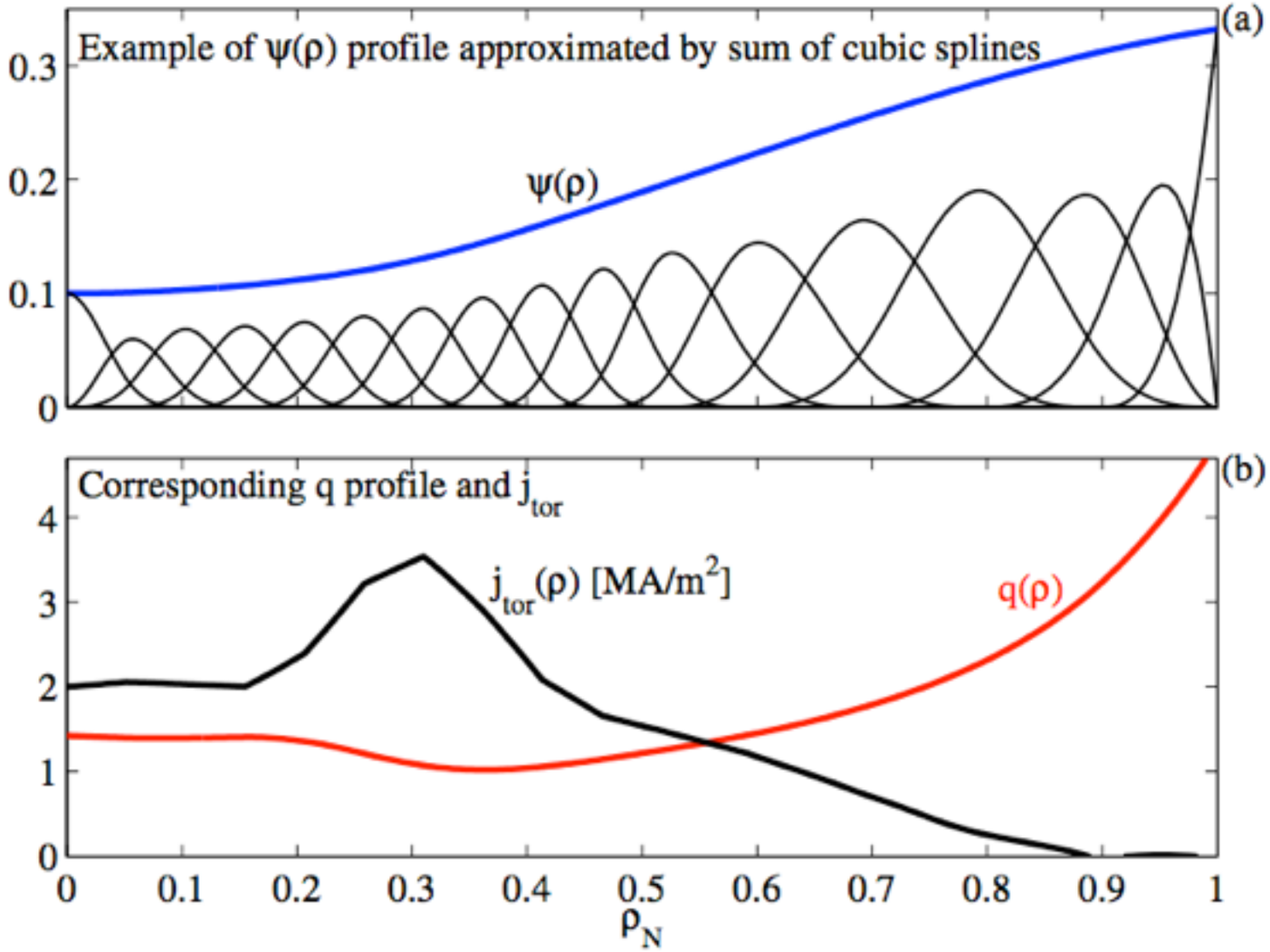
  - **Prescribed ion and density profiles.**
  - **Heat sources as sums of gaussians**
  - **Ad-hoc model for thermal diffusivity, of the form:**

$$\chi_e = \chi_{neo} + c_{ano}\rho q F(s) + \chi_{central} e^{-\rho^2/0.1^2}$$

Spatial discretization of profiles

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

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



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:**

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

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 xk, inputs uk 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**

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**

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

Equilibrium definition

- Presently, RAPTOR uses profile information from a CHEASE run (2D fixed-boundary GS solver)
  - V'(ρ), G_2(ρ) etc
- Pre-calculated equilibria exist for TCV and ITER
- Analytical equilibria could be added.
- For other equilibria, will need to run CHEASE
  - talk to F. Felici
- Coupling to general 2D equilibria (from fast GS solver or parametrized profiles) envisioned.

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
  - Numerics / debugging flags
- `init:`parameters for generating initial conditions

RAPTOR data objects: vectors

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

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

RAPTOR workflow

get defaults

```
config = RAPTOR_config
```

user : optional modification by user

user

Generate RAPTOR model

```
[~,model,params,init] = RAPTOR_config(config)
```

user

user

initial conditions

```
[x0,g0,v0] = ...
RAPTOR_initial_conditions(...
Ip0,model,init)
```

user

actuator inputs

User

```
simres = RAPTOR_predictive(x0,g0,v0,U,model,params);
```

run

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

post-processing

Directory structure

- **RAPTOR/**
  - **`RAPTOR_code`: core code files.**
  - **`RAPTOR_demos`: demonstration files and tutorials.**
  - **`chease_equils`: CHEASE equilibrium files defining equil. profiles.**
  - **`projects`: built "on top of" RAPTOR, e.g. analyses of a scenario**
  - **`optimization`: Tools for nonlinear optimization**
  - **`data`: store temporary data here (not versioned)**
  - **`personal`: store personal scripts here (not versioned)**
  - **`test`: some preliminary tests**
  - **`license`: license agreement**

Version control - SVN (Subverison)

- Hosted on CRPPSVN at CRPP-EPFL Lausanne (T.M. Tran)
- Code is developed in trunk, regular “tags” at stable releases
  - This presentation applies to tag/1.3
- Many tutorials exist for svn. Important commands (UNIX):
  - Check out the RAPTOR trunk from server to local directory
    - `svn co https://crppsvn.epfl.ch/repos/RAPTOR/tags/release-1.3 ./RAPTOR`
  - Check differences between local and current version
    - `svn stat -uv`
  - Revert to version on server (undo 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

Where to get help

- **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 (2012 draft) in /docs directory**
- **Tutorials and demo files**
  - **in the /RAPTOR_demos directory**
- **E-mail: f.felici@tue.nl**

New features and work in progress
- to be done / already done

- **Physics**
  - **New NBI module with NBCD and 2D geometry**
  - **Alpha particle heating module**
  - **Radiation losses, equipartition power modules**
  - **H-mode module**
  - **Bohm-Gyrobohm transport module**
  - **Time-averaged sawtooth effects on q profile and Te**
  - **Sawtooth module (D. Kim (CRPP))**
  - **Time-varying (but prescribed) shape evolution (F. Felici)**
  - **Solve self-consistently for shape using parametrization or full 2D Grad-Shafranov (F. Felici)**
- **Code**
  - **Optimization of transport coefficients to match experiments (Geelen)**
  - **Benchmark vs. CRONOS for ITER plasmas (van Dongen)**
  - **Real-time version on ASDEX-upgrade and TCV (Felici, others)**

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.