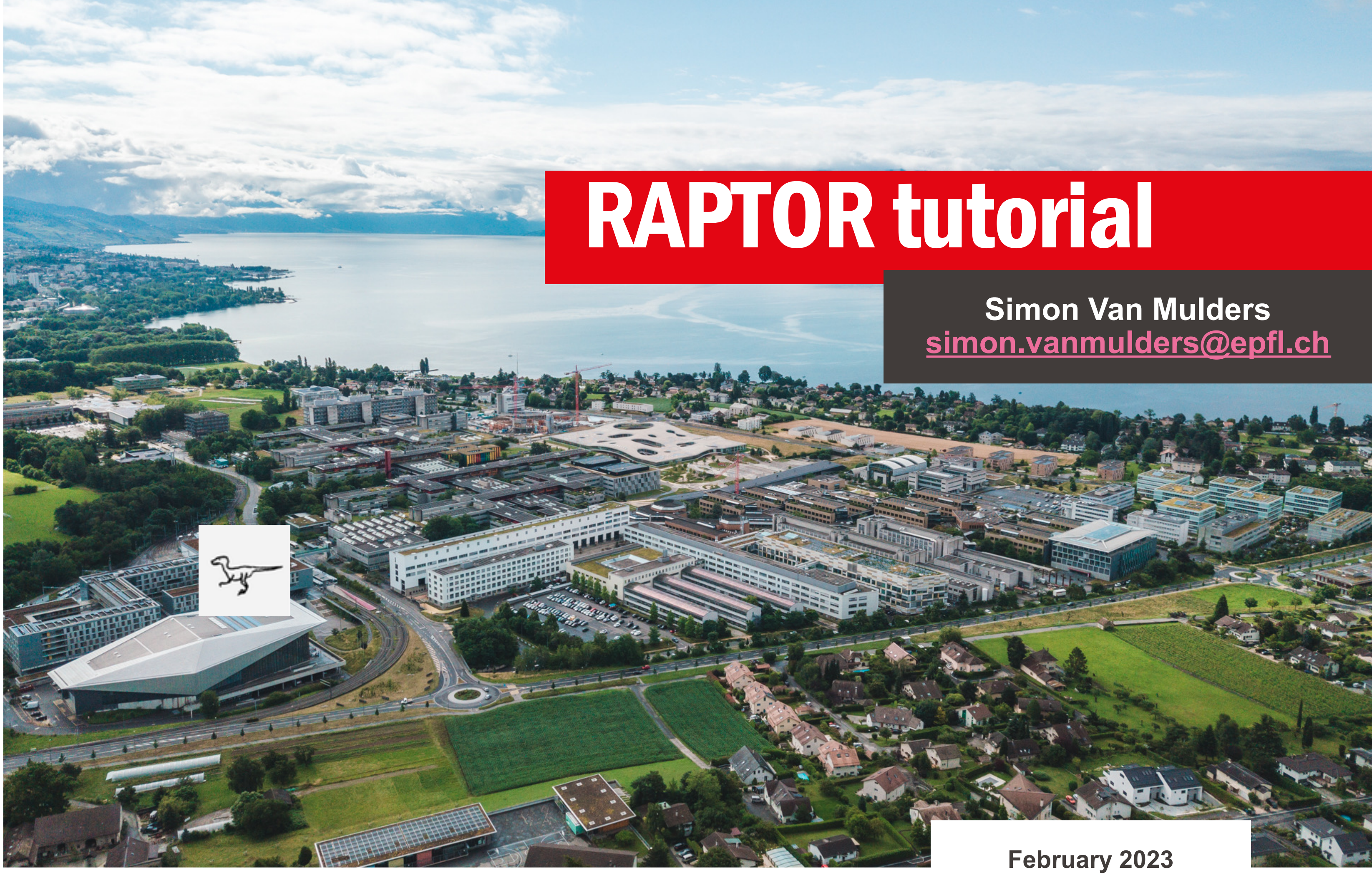
EPFL
Swiss Plasma Center



RAPTOR tutorial

Simon Van Mulders
simon.vanmulders@epfl.ch

February 2023

École polytechnique fédérale de Lausanne

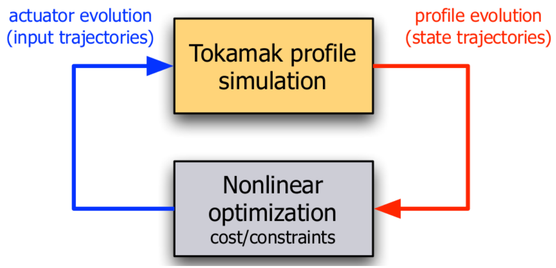
Outline

- Part I - This talk
  - Introduction to RAPTOR
  - Equations, physics model and numerical method
  - RAPTOR code structure: directories, data objects, functions
  - Version management – git (https://gitlab.epfl.ch/spc/raptor)
  - How to get started
  - License agreement
- Part II - Try it for yourself
  - Work through the RAPTOR tutorials to familiarize yourself with options
  - Excercise

Basics of RAPTOR

- **RA**pid **P**lasma **T**ransport simulat**OR** [Felici NF2011, Felici NF2018]
  - Fast transport code for time evolution of **1D tokamak plasma profiles**
  - Designed for ***real-time* and *control-oriented* use**
- Initial conception:
  - Solve coupled, non-linear, 1D diffusion PDEs for poloidal flux ψ and T_e *(now additionally: T_i and n_e)*
  - Fixed MHD equilibrium, parametrized actuators and transport
- Numerics
  - Using cubic splines, boundary condition at any radius
  - Implicit numerical solver
  - Analytic Jacobians w.r.t. state and inputs and local linearization
  - Embedded in non-linear optimizer, w. analytic gradients
- 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)
  - git for code version control (https://gitlab.epfl.ch/spc/raptor)

Non-linear optimizer:



Extensions

- A hierarchy of models to evaluate turbulent transport:
  - Simple formula with tuning parameters [Felici PPCF 2012] and Bohm-gyroBohm model [Erba NF 1998]
  - **Scaling-law-driven gradient-based model** exploiting core profile stiffness [Teplukhina PPCF 2017]
  - **Surrogate neural network** of quasi-linear gyro-kinetic code QuaLiKiz [van de Plassche PoP 2020]
- PDEs for T_i and n_e have been added [Felici NF 2018]
- **Time-varying equilibrium** for transient phases: allows to *model impact of changing geometry during I_p ramps* [Teplukhina PPCF 2017]
- **Stationary state solver**, allowing to *optimize flat-top operating point* [Van Mulders NF 2021]
- *On-line interpretative applications*
  - Real-time state estimation [Piron FED2021] and kinetic equilibrium reconstruction [Carpanese NF2020]
  - Model predictive control [Maljaars NF 2017]
- *Off-line predictive applications*
  - Non-linear scenario optimization [Felici PPCF 2012, van Dongen PPCF 2014, Teplukhina PPCF 2017, Van Mulders NF 2021, Mitchell subm. FED2023]
  - Full-discharge modeling [Teplukhina PPCF 2017],
  - including **1st principles transport** [Felici NF 2018] and **impurity radiation** [Maget PPCF 2022, Ostuni NF 2022]
- Outlook: uncertainty quantification, compatibility with IDS/IMAS, real-time q profile control, couple to fast NBI and ECH/ECCD models, impurity transport …

Swiss Plasma Center

Equations solved by RAPTOR (1/4)

$$\sigma_{\parallel}\left(\left.\frac{\partial\psi}{\partial t}\right|_{\hat{\rho}} - \frac{\hat{\rho}\dot{\Phi}_b}{2\Phi_b}\frac{\partial\psi}{\partial\hat{\rho}}\right) = \frac{F^2}{16\pi^2\mu_0\Phi_b^2\hat{\rho}}\frac{\partial}{\partial\hat{\rho}}\left[\frac{g_2 g_3}{\hat{\rho}}\frac{\partial\psi}{\partial\hat{\rho}}\right] - \frac{B_0}{2\Phi_b\hat{\rho}}V'_{\hat{\rho}}j_{ni}$$

- Flux-surface-averaged diffusion of **poloidal flux**
  - Neoclassical conductivity, bootstrap current
  - Current drive sources as sums of gaussians or manual profiles
  - Neumann boundary condition through total I_p

Swiss Plasma Center

Equations solved by RAPTOR (2/4)

$$\frac{3}{2}(V_{\hat\rho}')^{-5/3}\left(\frac{\partial}{\partial t}\bigg|_{\hat\rho}-\frac{\dot\Phi_b}{2\Phi_b}\frac{\partial}{\partial\hat\rho}\hat\rho\right)[(V_{\hat\rho}')^{5/3}n_eT_e]+\frac{1}{V_{\hat\rho}'}\frac{\partial}{\partial\hat\rho}\left(-\frac{g_1}{V_{\hat\rho}'}n_e\chi_e\frac{\partial T_e}{\partial\hat\rho}+\frac{5}{2}T_e\Gamma_e g_0\right)=P_e$$

- Flux surface averaged **temperature** diffusion
  - Solve T_e (and optionally T_i), with boundary condition at arbitrary ρ
  - Heat sources as sums of gaussians or manually prescribed
  - Simple analytical models for heat sources and sinks
  - ADAS cooling factor data for impurity radiation
  - Ad-hoc model for thermal diffusivity with various transport models (previous slide)

Swiss Plasma Center

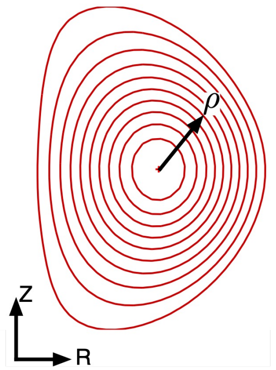
Equations solved by RAPTOR (3/4)

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

- Flux surface averaged **particle** diffusion
  - For choice of species, electrons, ions, up to 3 impurities, with boundary condition at arbitrary ρ
  - Non-simulated species can be set as fractions of other species, directly prescribed, or constrained by Z_{eff} and quasi-neutrality
  - *e.g. set trace W to match experimental radiated power*

Equations solved by RAPTOR (4/4)

- Magnetic equilibrium terms ($V', g_0, g_1, g_2, g_3, \Phi_b$)
  - No coupling to Grad-Shafranov solver
  - Time-dependent sequence can be interfaced
  - Post-processed to correct format from CHEASE output files



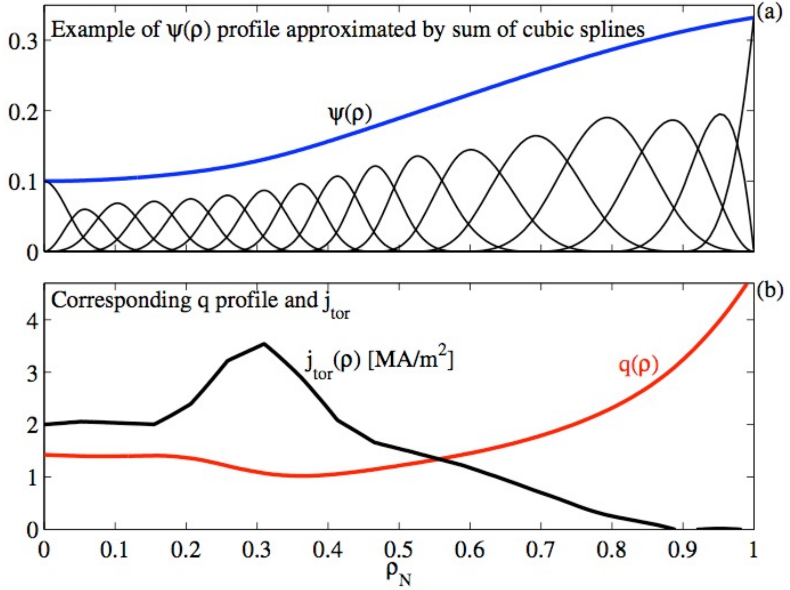
- Other physics (optional)
  - H-mode pedestal, *e.g. setting BC at ρ=0.8 with scaling law*
  - MHD: Sawteeth and NTM module

Swiss Plasma Center

Some background: spatial discretization of profiles

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

- Express profile as sum of *spline basis functions* (finite element method)
- Note: time and space dependent factor



Swiss Plasma Center

EPFL

From Continuous time PDE to Discrete-time matrix ODE

- Introducing the spline representation of y(ρ,t) (e.g. ψ):

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

- Projecting onto the spline basis functions and integrating by parts, a set of ODEs for the time evolution of the spline coefficients is obtained:

$$\sum_{\alpha=1}^{n_{sp}} \frac{d\hat{y}_\alpha(t)}{dt} \underbrace{\int_0^{\rho_e} m \Lambda_\beta \Lambda_\alpha \, \mathrm{d}\rho}_{M_{\beta\alpha}(t)} = -\Sigma_{\alpha=1}^{n_{sp}} \hat{y}_\alpha(t) \underbrace{\left[\int_0^{\rho_e} g \frac{\partial \Lambda_\beta}{\partial \rho} \frac{\partial \Lambda_\alpha}{\partial \rho} \mathrm{d}\rho \right]}_{=D_{\beta\alpha}} + \underbrace{\left[g \Lambda_\beta \frac{\partial y}{\partial \rho} \right]_0^{\rho_e}}_{=l_\beta} + \underbrace{\left[\int_0^{\rho_e} \Lambda_\beta kj \mathrm{d}\rho \right]}_{=s_\beta}.$$

Matrix-vector ODE:

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

NB: M,D,l,s depend non-linearly on (y,u)

- For the various eq. and compacting into non-linear state evolution equation:

$$f(\dot{x}(t), x(t), u(t)) = 0 \; \forall \; t$$

Swiss Plasma Center

Implicit time stepping

$$f(\dot{x}(t), x(t), u(t)) = 0 \; \forall \; t$$

- Temporal discretization *(usually θ=1, fully implicit method)*:

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

- Non-linear eq. at each time step:

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

- At each time step, given state x_k, inputs u_k at time step k,
  - Take steps in Newton descent direction *d:* $\boxed{\mathcal{J}_{k+1}^k d = \tilde{f}_k,}$
  - Need Jacobian, obtained from analytical expression for all the derivatives (copious application of chain rule): $\boxed{\mathcal{J}_{k+1}^k = \frac{\partial \tilde{f}_k}{\partial x_{k+1}}}$
  - Iterate until residual f_k < *tolerance*
  - Go to next time step
- Store Jacobians at each time step

Swiss Plasma Center

EPFL

Parameter sensitivity of profile evolution

- Time evolution depends on model and input parameters
  - One example: a transport model parameter
  - Another example: a parameter defining the input trajectory
- Differentiating with respect to parameter *p*, we get the *sensitivity equation*

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

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

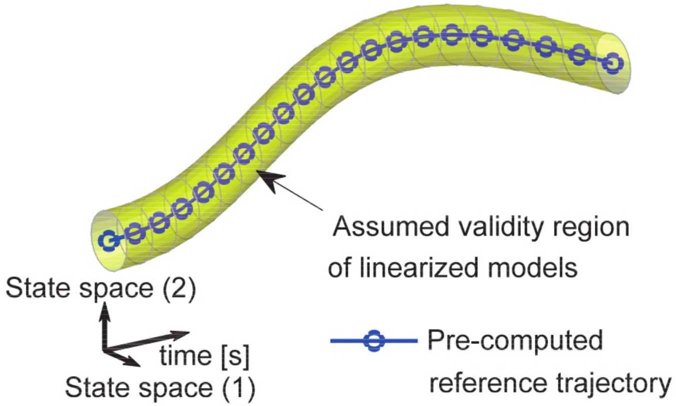
  - Linear ODE for dx_k/dp, solve while evolving non-linear PDE: **Forward sensitivity analysis**
  - Jacobians df_k/dx_k, df_k/dx_{k+1} are known from Newton iterations
  - Computational cost proportional to *p*

Swiss Plasma Center

Local linearization and cost/constraint gradients

- dx_k/dp gives the **linearization** of the state trajectories in the parameter space

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



- With optimization variables p parametrizing an input trajectory, dx_k/dp enables evaluation of **analytical cost and constraint gradients**

Swiss
Plasma
Center

RAPTOR data objects: structures

- `config`: configuration for generating model and parameter structures
  - Specify which **equations** are solved for,
    e.g. `config.ne.method = 'state';`
  - Specify equilibrium data
  - **Physics modules** specified though `RAPTOR_module` class
    e.g. `config.echcd = RAPTORmodule('echcd_gaussian');`
    or `config. chi_e = RAPTORmodule('chi_BgB');`
  - **Numerics** (radial and temporal grid, spline order, solver tolerances, etc.)
  - Environment (paths etc.), run-time plot and display options
- `model`: pre-calculated data (matrices etc.), should not be changed manually
  - Spline basis matrices, radial grid, physics module config
- `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 state x_0

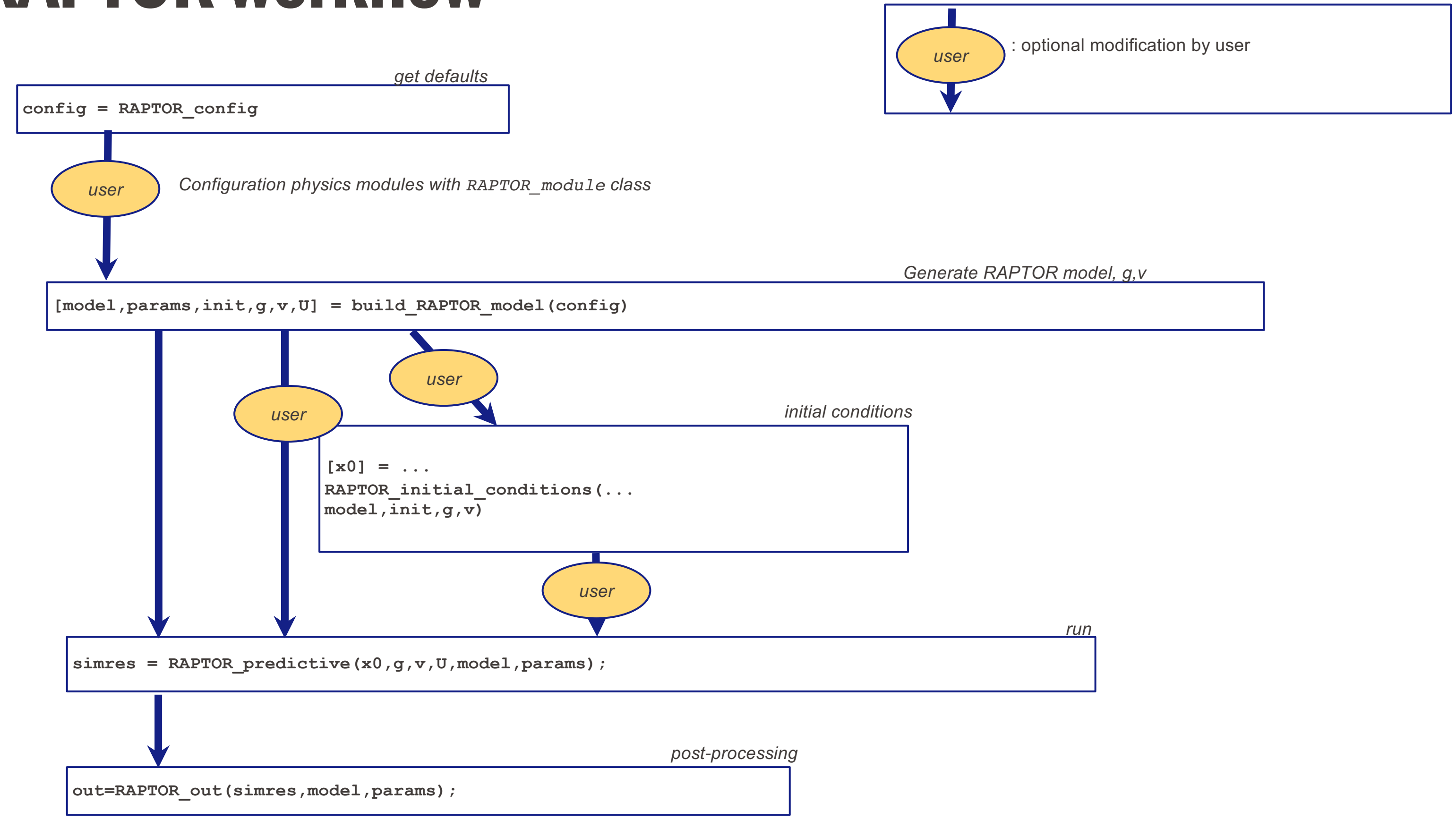
RAPTOR data objects: time-varying vectors

- `x`: state quantities which the code solves for
- `v`: pre-known (time-varying) kinetic profile quantities
- `g`: geometry-related quantities (calculated from equil config)
- `u`: actuator inputs

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

Swiss Plasma Center

RAPTOR workflow



Swiss Plasma Center

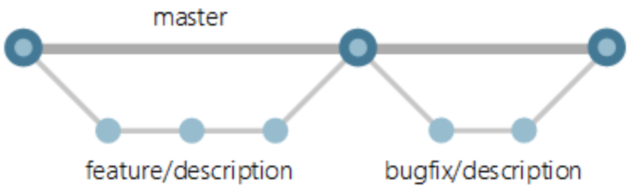
Directory structure

- 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 non-linear optimization
  - `data`: store temporary data here (not versioned)
  - `personal`: store personal scripts here (not versioned)
  - `license`: license agreement
  - `RT`: Real-time implementations (Simulink)

Swiss Plasma Center

Version control - git

- Web-based interface for the code repository: (https://gitlab.epfl.ch/spc/raptor)
- git: distributed version control system
- Protected branches: `development` and `master`
- Short-lived development branches per feature/bug, integrated to `master` via merge request
- Many tutorials exist for git



EPFL

Where to get help

- Getting started: tutorials and demo files
  - in the `/demos` directory, execute `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
  - [Felici PPCF 2012, Felici PhD thesis, Felici NF 2018]
- Equation details
  - RAPTOR equation document in `/doc` directory

Swiss Plasma Center

User's license and conditions

EPFL-SPC
Station 13
1015 Lausanne (Switzerland)
Fax: +41 21 69 35176

Authorisation to use and apply

RAPTOR

at the ____ (insert association name)

The computer code RAPTOR has been developed at the Swiss Plasma Center, Ecole Polytechnique Fédérale de Lausanne (EPFL-SPC), Switzerland.

RAPTOR (RApid Plasma Transport Simulator) is a 1D tokamak transport code specially designed for rapid execution compatible with needs for real-time execution or for use in nonlinear optimization schemes. RAPTOR is an open-source code available only for non-commercial usage.

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 SPC.

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 predictive simulations of poloidal flux and electron temperature, or its use in nonlinear optimization routines:
F. Felici et.al. Plasma Physics and Controlled Fusion **54**(2), p.025002 (2012)
For predictive multi-channel simulations including ion energy and particle transport equations:
F. Felici et al, Nuclear Fusion **58** p.096006 (2018)

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

5.- The user accepts that the code is provided on an as-is basis without any warranty or conditions of any kind.

Responsible person

Name: _________________________________

email: _________________________________

______________________ Place and Date

__________________________ Signature

Code_transfer_RAPTOR, version July 2018

Swiss Plasma Center