How to use

This directory contains 3 codes that achieve different tasks:

- `CriticalityFudgeFactor.py`: computes the fudge factor of a given reactor configuration,
- `1SpeedReactor.py`: integrates over time the neutron flux dynamics of a given 1 speed reactor configuration,
- `1SpeedReactorCR.py`: as the above but also *control rods* dynamics is included in the integration.

All the parameters for these codes must be inserted in apposite `.dat` files in this directory:

- `Parameters.dat`: contains all the simulation parameters and options,
- `grid.dat`: contains the geometrical information of the reactor configuration space and the initial conditions,
- `Sigma_absorption.dat`: contains the absorption values of the moderator over the configuration space,
- `Sigma_fuel.dat`: contains the production values of the fuel over the configuration space,
- `Control_rods.dat`: contains the absorption values of the control rods, when fully inserted, over the configuration space.

All these files are shared between all the codes.

Parameters insertion

`Parameters.dat` contains the following variables:

- *t_max* is the time duration (*s*) of time integration,
- *delta_t* is the time discretization step,
- *omega* is the weight used in *successive relaxation method*
- *D* is the diffusion parameter,
- *v* is the average speed of neutrons,
- *fudge* is the fudge factor,
- *Delta* is the spacial discretization step,
- *Integration_mode* can be:
  - `nopython`: using Numba this option precompiles the matrix operations (usually faster for small grids),
  - `parallel`: using Numba this option parallelizes the matrix operations (don't use this I just tried),
  - `Scipy`: uses Scipy sparse matrix optimization methods for integration (significantly faster on larger grids, e.g. 40x40).
- *Steady_state_Control_rods_level*: insertion level of control rods when the reactor is at steady state (for *CriticalityFudgeFactor.py* this is used as fixed parameter to evaluate the fudge with such rods insertion level).

Steady_state_Control_rods_level = .2 All these parameters can be inserted in any order and spaces can be omitted. Any unknown parameter will be ignored by the code and for every missing parameter the code will use default values.

```
    t_max = 10
    delta_t = 0.1
    omega = 1
    conv_criterion = 1E-5
    D = 10
    v = 1
    fudge = 0.045210616027732034
    Delta = 0.2
    Integration_mode = Scipy
```

`grid.dat` must be set in a matrix form, in the following way:

```
E E E 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E E E
E E 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E E
E 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
E 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E
E E 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E E
E E E 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E E E
```

Each element of the matrix must be separated by a `space` char. Numerical entries will be interpreted as initial conditions (or guesses) of the neutron flux, the char `E` is instead interpreted as an empty cell which flux is always zero (Boundary condition). Any other value is invalid.

`Sigma_absorption.dat`, `Sigma_fuel.dat` and `Control_rods.dat` must be set as the above but only numerical values are allowed (set zero values for the corresponding empty cells).

On control rods dynamics

The dynamics of control rods is generated by the package `Reactpy`, which provides different operational schemes. By default, the *linear_cycle* scheme is used (see Reactpy documentation for more info). To change the scheme it is necessary to modify the code of `1SpeedReactorCR.py` at line 79.

Output

Each code will generate as output un image/animation using Matplotlib. The animations are saved ad `.gif` files automatically (the old ones in the source directory are replaced each time so copy them somewhere else) while static images (generated by `CriticalityFudgeFactor.py`) must be saved manually. This last one also outputs the evaluated fudge factor needed to reach criticality: this number is printed into the terminal but also on the graphs.