Accident Tolerant Fuels: Neutronic Aspects

Francesco Zazzu

MSc in Nuclear Engineering
Dipartimento di Ingegneria Civile e Industriale
Università di Pisa

A.Y. 2024/2025

Introduction

As a result of the Fukushima Daiichi accident, enhancing the safety of fuel and cladding became a topic of serious discussion.

- Why are Accident Tolerant Fuels being developed, and **how do they compare with conventional fuels**?
- What are the **neutronic characteristics** of ATFs?
- How can **Monte Carlo simulations** support the development of ATFs?

The aim of this work is to answer these questions through the use of **OpenMC**, a community-developed Monte Carlo neutron and photon **transport** simulation code.

FeCrAl alloy vs Zircaloy

Relevant properties and characteristics:

- **compatibility with the coolant chemistry**;
- **lower oxidation rate** than Zircaloy;
- **hydrogen has a higher mobility** in FeCrAl alloys and will not accumulate in the alloy, **reducing the embrittlement**;
- **higher specific heat** than Zircaloy that **decreases the peak cladding temperature**.

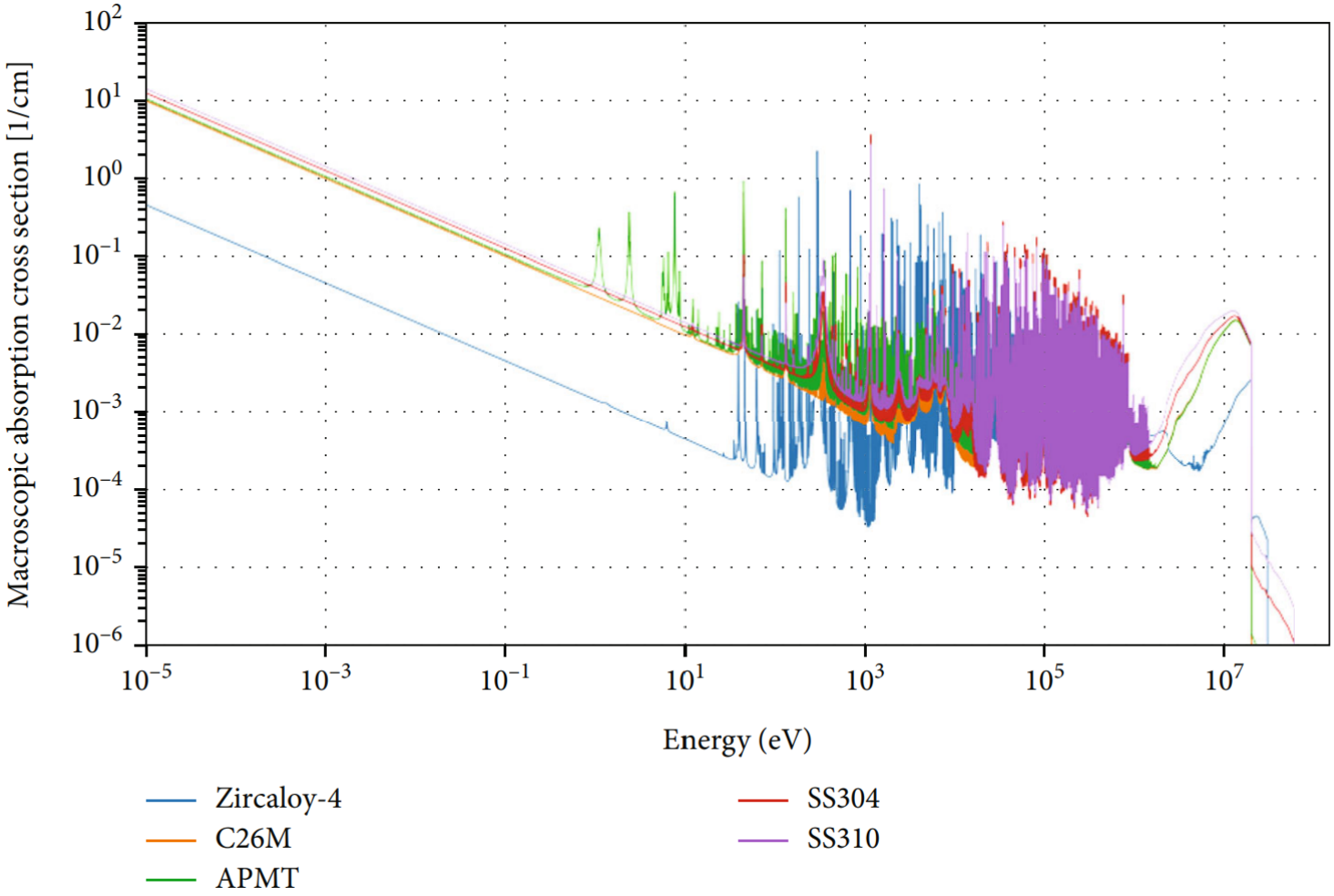
Challenges:

- **permeability of tritium**, which poses a **radiation concern**;
- increased **parasitic neutron absorption** of the FeCrAl.

FeCrAl alloy vs Zircaloy

Property	**Zircaloy-4**	**FeCrAl**
Melting Point (°C)	1850	1500
Specific Heat (J/kg·K)	285-368 [25° C-700° C]	480-710 [20° C-600° C]
Thermal Conductivity (W/m·K)	14.5-14.2 [25° -425°]	11-21 [50° C-600° C]
Thermal Neutron Absorption Cross Section (barns)	0.18-0.20	1.3-2.0
Density (g/cm^3)	6.52	7.1
Component		
Zr	98.43%	-
Sn	1.2%	-
Fe	0.21%	78.68%
Cr	0.12%	13.02%
Al	-	5.1%
Mo	-	2.1%
Si	-	2%
Nb	-	1%
Y	-	0.3%
C	-	0.003%

FeCrAl alloy vs Zircaloy



On OpenMC

- After initializing the particle and determining its current cell, the **distance to the nearest boundary** d_b of current cell is determined.
- The **distance to the next collision** is extrapolated thanks to a **pseudo-random** number ξ, d is then compared with d_b:

$$d = -\frac{\ln \xi}{\Sigma_t}, \ \xi \in [0, 1).$$

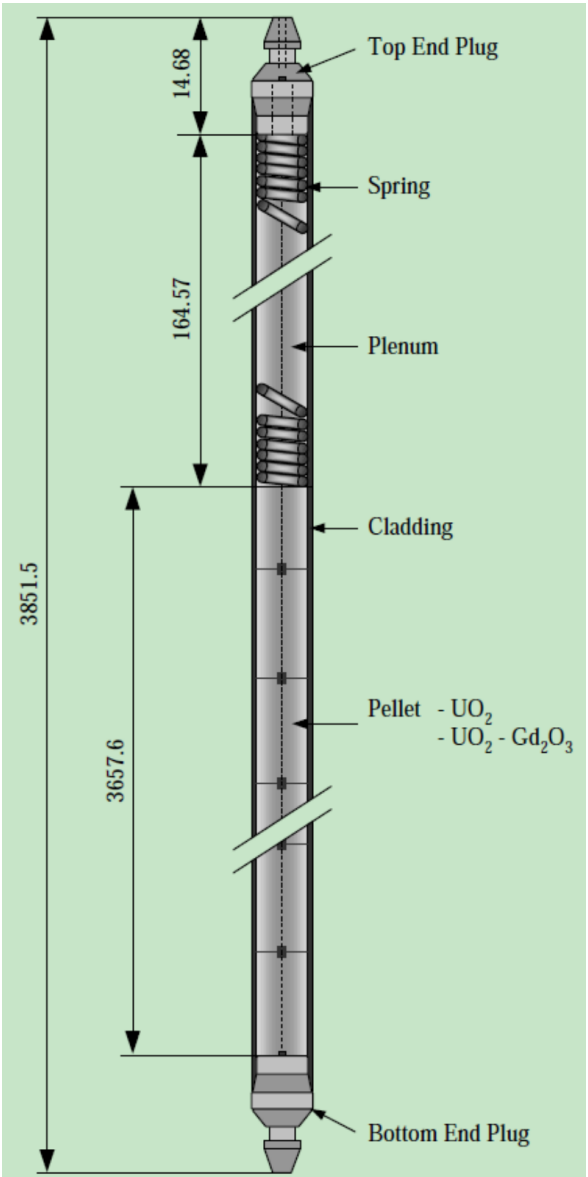
- To determine the **probability of interaction** of the particle **with each nuclide** i:

$$P(i) = \frac{\Sigma_{t,i}}{\Sigma_t}.$$

- A **reaction** for that nuclide is randomly sampled based on the microscopic cross sections:

$$P(x) = \frac{\sigma_x}{\sigma_t}.$$

Materials and Geometries



Some **approximations**:

- simplified geometries;
- only helium is present inside the plenum;
- the fuel pellet stack has been constructed as a single column.
- burnable poisons are not dispersed in the water.

Materials and Geometries

```
###Uranium###
uo2=openmc.Material(material_id=1)
uo2.add_element('U',1,enrichment=5.0)
uo2.add_element('O',2)
uo2.set_density('g/cm3',10.3)

###Coolant###
water=openmc.Material(material_id=2)
water.add_nuclide('H1', 2.0)
water.add_nuclide('O16', 1.0)
water.set_density('g/cm3',0.650)

###Assembly###

#gap and plenum
helium=openmc.Material(material_id=3)
helium.add_element('He',1)
helium.set_density('g/cm3',0.0043)


###Cladding###
zircaloy=openmc.Material(material_id=4)
zircaloy.add_element('Zr',0.9843,'wo')
zircaloy.add_element('Sn',0.0120,'wo')
zircaloy.add_element('Fe',0.0021,'wo')
zircaloy.add_element('Cr',0.0012,'wo')
zircaloy.add_element('O',0.0003,'wo')
zircaloy.set_density('g/cm3',6.57)

#plugs and thimbles
steel= openmc.Material(material_id=5)
steel.add_element('C',0.0005,'wo')
steel.add_element('Ti',0.005,'wo')
steel.add_element('Cr',0.1625,'wo')
steel.add_element('Mn',0.02,'wo')
steel.add_element('Fe',0.65,'wo')
steel.add_element('Ni',0.14,'wo')
steel.add_element('Mo',0.022,'wo')
steel.set_density('g/cm3',7.55)
```

A square cell region is used to surround each pin, with **transparent boundary conditions.**

```
###Geometry###

rod_pitch=1.26
frod_height=385.866

r_p=0.857/2
r_g=r_p+0.016
r_c=r_g+0.0570

#Fuel Pin: lateral surfaces, infinite cylinders centered in z axis with radius in cm
s_pellet=openmc.ZCylinder(0,0,r_p)
s_gap=openmc.ZCylinder(0,0,r_g)
s_clad=openmc.ZCylinder(0,0,r_c)

#Top and bottom surface
s_top=openmc.ZPlane(frod_height/2., boundary_type='vacuum')
s_bottom=openmc.ZPlane(-frod_height/2, boundary_type='vacuum')

#Planes internal to the pins
bottom_plug_pos=-frod_height/2+1.465
bottom_plenum_pos=frod_height/2-16.457-1.468
top_plenum_pos=frod_height/2-1.468

s_bottom_plenum= openmc.ZPlane(bottom_plenum_pos)
s_top_plenum= openmc.ZPlane(top_plenum_pos)
s_b_top_end_plug= openmc.ZPlane(bottom_plug_pos) #surface between bottom plug and fuel pellet stack

square_cell = openmc.model.RectangularPrism(rod_pitch,rod_pitch)
```

Cells and Universes

```
###Cell construction ###

c_pellet=openmc.Cell(name="Pellet")
c_pellet.region=+s_b_top_end_plug & -s_bottom_plenum & -s_pellet
c_pellet.fill=uo2

c_gap=openmc.Cell(name="Gap") #gas in the gap along the fuel stack
c_gap.region=+s_pellet & +s_b_top_end_plug & -s_bottom_plenum & -s_gap
c_gap.fill=helium

#gas plenum
c_gas= openmc.Cell(name='gas plenum')
c_gas.region=  +s_bottom_plenum & - s_top_plenum & -s_gap
c_gas.fill= helium

c_clad=openmc.Cell(name="Clad")
c_clad.region=+s_gap & +s_bottom & -s_top& -s_clad
c_clad.fill=zircaloy

#coolant
c_coolant=openmc.Cell(name="Coolant")
c_coolant.region= +s_bottom & -s_top & +s_clad & -square_cell
c_coolant.fill=water

#bottom plug
c_b_plug= openmc.Cell(name='Bottom plug')
c_b_plug.region=  -s_b_top_end_plug & + s_bottom & -s_gap
c_b_plug.fill= steel

#top plug
c_t_plug= openmc.Cell(name='top plug')
c_t_plug.region=  +s_top_plenum  & -s_top & -s_gap
c_t_plug.fill= steel

fuel_rod_universe=openmc.Universe(cells=[c_pellet,c_gap,c_gas,c_clad,c_coolant,c_t_plug,c_b_plug])
```

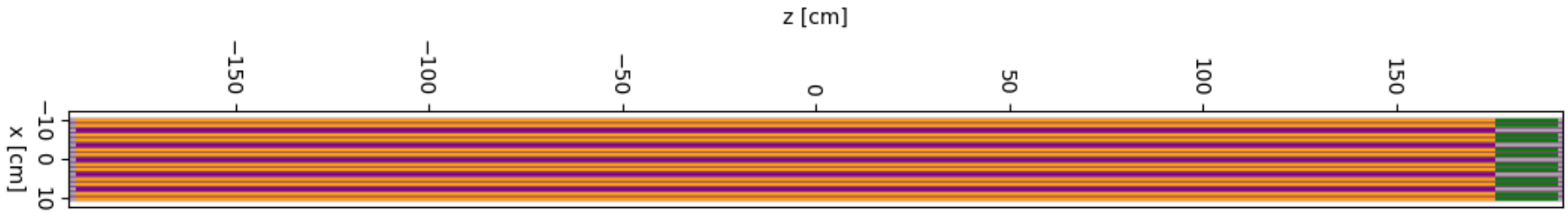
Regions of uniform composition are assigned to each **cell**. These regions can be defined through the use of Boolean operators. The possibility to use **universes** enables to model any identical repeated structures and then fill them with different elements.

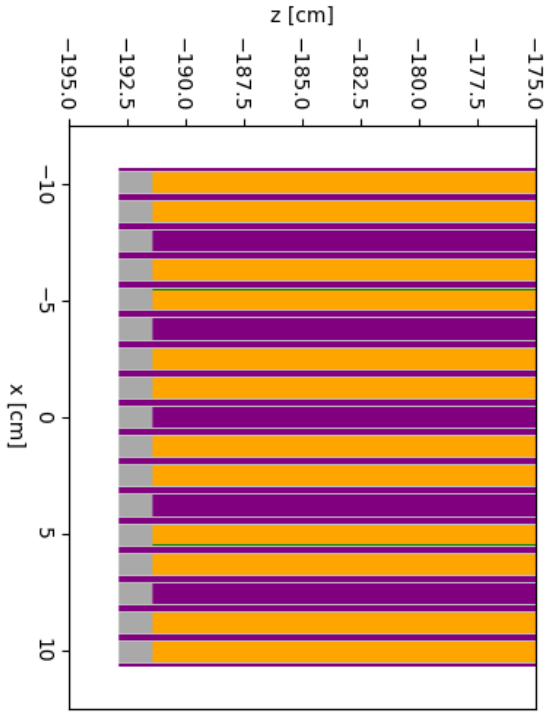
Fuel Assembly

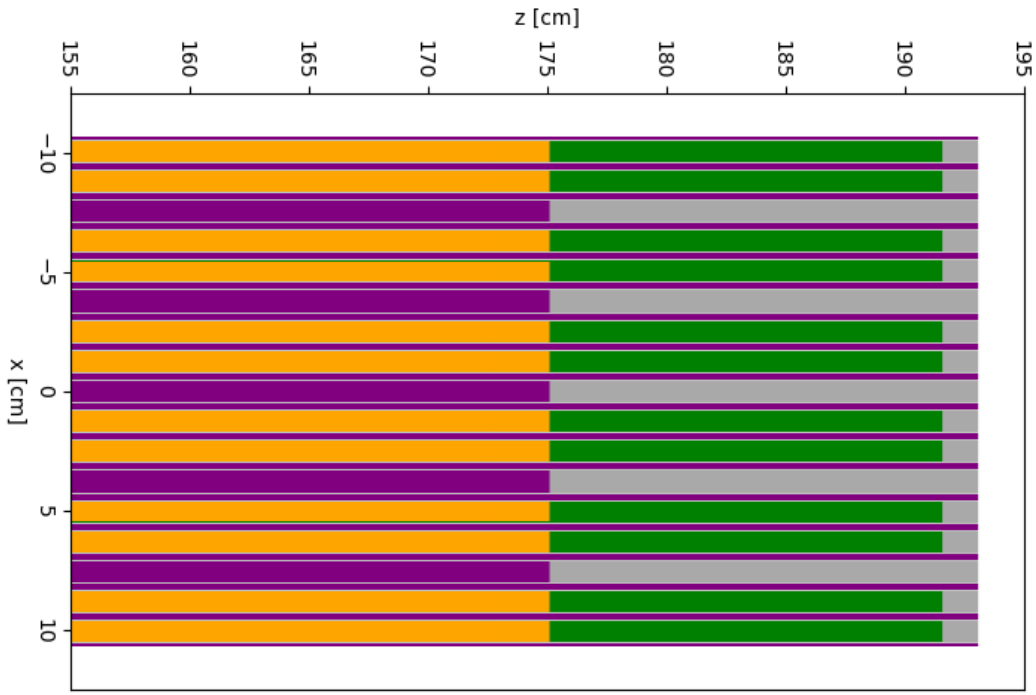
Finally, to obtain the Fuel Bundle the **lattice** has been filled with all the universes defined.

```
###The Bundle###

bundle_pitch=17*rod_pitch

outer_boundary = openmc.model.RectangularPrism(bundle_pitch,
                                               bundle_pitch, boundary_type='reflective')

#bundle lattice
bundle_lattice= openmc.RectLattice(name='Fuel Assembly')
bundle_lattice.pitch = (rod_pitch, rod_pitch)
bundle_lattice.lower_left = (-bundle_pitch/2, -bundle_pitch/2)

bundle_lattice.universes = np.full((17, 17), fuel_rod_universe)
bundle_lattice.universes[guide_tubes_pos[:, 0], guide_tubes_pos[:, 1]] =guide_tubes

bundle_cell = openmc.Cell(fill=bundle_lattice, region= -outer_boundary & -s_top & +s_bottom)

bundle_universe= openmc.Universe (cells= [bundle_cell])
```

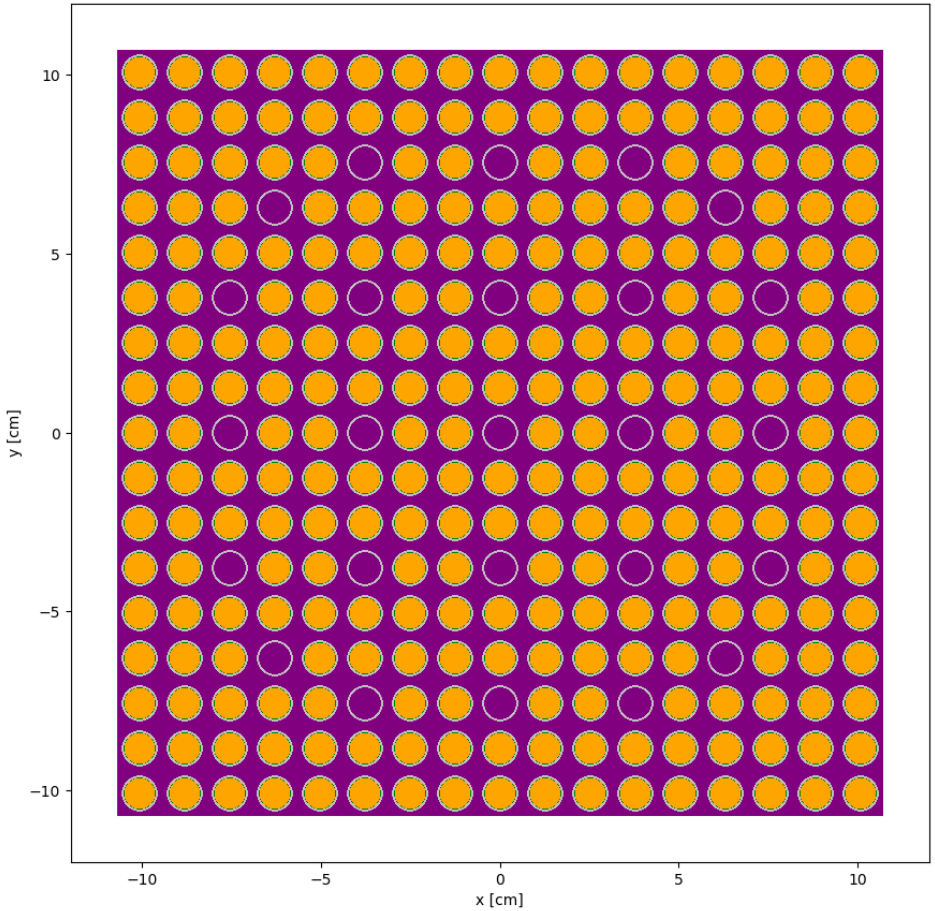
Plots

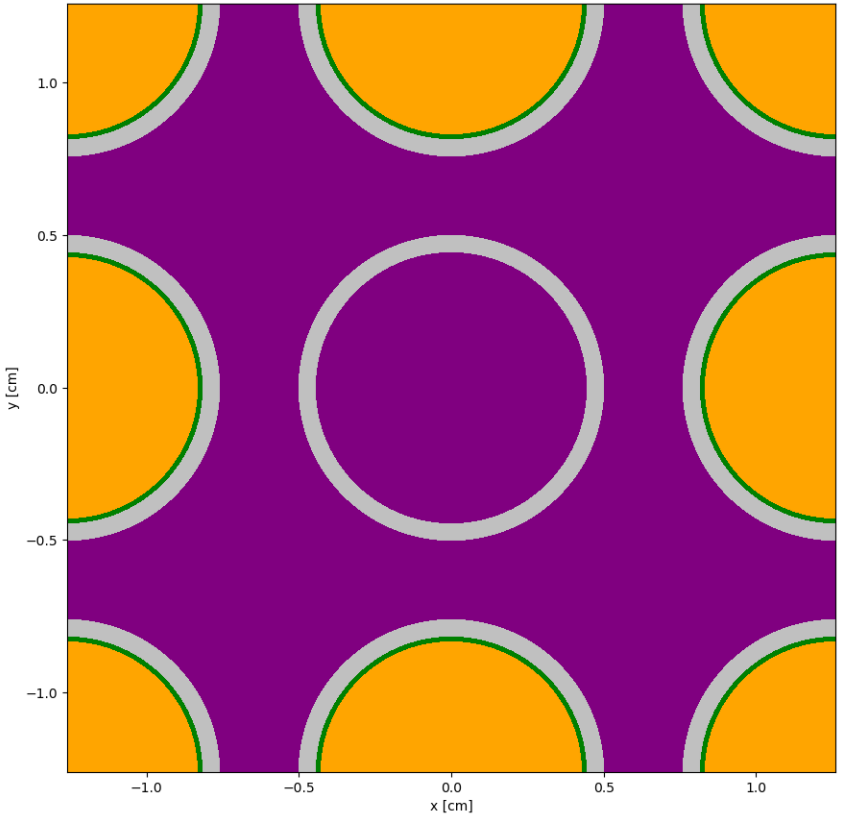

z [cm]
−150
−100
−50
0
50
100
150
x [cm]
−10
0
10


z [cm]
−195.0
−192.5
−190.0
−187.5
−185.0
−182.5
−180.0
−177.5
−175.0
x [cm]
−10
−5
0
5
10


z [cm]
155
160
165
170
175
180
185
190
195
x [cm]
−10
−5
0
5
10

Plots





Tallies

Any of the **tally** X could be defined as:

$$X = \underbrace{\int d\mathbf{r} \int d\mathbf{\Omega} \int dE}_{\text{filters}} \underbrace{f(\mathbf{r}, \mathbf{\Omega}, E)}_{\text{scores}} \psi(\mathbf{r}, \mathbf{\Omega}, E)$$

A **filter** identify which regions of phase space should score to a given tally as well as the **scoring function** f.
The **estimator** selected in this case is the **Track-lenght** estimator.

$$V\phi = \frac{1}{W} \sum_{i \in T} w_i \ell_i$$

Tallies

Filters:

- **energy filter**, two energy intervals are used $[0, 0.68]eV$ and $[0.68, 2 \times 10^6]eV$
- **mesh filter**, the number of mesh cells is equal to the number of fuel pins in the assembly. Along the axial direction is used a single bin.

Scores:

- **flux**, rappresents the total flux, measured in *particles-cm per source particles*;
- **kappa-fission**, the recoverable energy production rate due to fission: sum of fission product kinetic energy, kinetic energy of prompt and delayed neutrons, prompt and delayed γ-rays. Units are eV per source particle.

Tallies

```
###Tallies###

energy_filter=openmc.EnergyFilter([.0,.68,20e6])
msh_filter=openmc.RegularMesh(mesh_id=1)
msh_filter.dimension=(17,17,1)

# define the most negative and positive cprner creating a parallelepide that contains everything
msh_filter.lower_left= [-bundle_pitch/2, -bundle_pitch/2,-frod_height/2]
msh_filter.upper_right= [bundle_pitch/2, bundle_pitch/2,frod_height/2]
mesh_filter= openmc.MeshFilter(msh_filter,filter_id=2)

flux= openmc.Tally(tally_id=1,name= 'Flux')
flux.scores= ['flux'] #total flux
flux.filters= [energy_filter, mesh_filter]

#define tally for fission power
depos_energy = openmc.Tally(tally_id=2,name='Fission_power')
depos_energy.scores = ['kappa-fission']  # tracks energy deposition per fission
depos_energy.filters = [mesh_filter]

fission_rr =openmc.Tally (tally_id=3,name='Fission_reaction_rate')
fission_rr.scores = ['fission']  #total fission reaction rate
fission_rr.filters = [energy_filter, mesh_filter]

tallies = openmc.Tallies([flux,depos_energy,fission_rr])
tallies.export_to_xml()
```

Simulation Settings

Specify **execution settings**:

- **run mode**, has been choosen the **eigenvalue** mode. The transport equation becomes an eigenvalue equation: since then the source will depend on the flux of neutrons itself.
- **source distribution**;
- **batches**, rappresents the number of groups in which the source particle histories are broken up. By default, a batch will consist of only a single fission generation. **Inactive**, refers to the number of initial batches during which the code does not score tallies.
- number of **particles**, will determine the statistical uncertainty. A higher number of particles will reduce the uncertainty.

Sensitivity analysis

```python
def optimize_batches(name, N, batch_range, inactive, source, particles):

    cleaner()

    batch_sizes = np.linspace(batch_range[0], batch_range[1], N, dtype=int)

    keffs = []
    relative_errors = []
    keff_stddev = []

    print(f"\n Running batch optimization for {name}...")

    for n_batches in batch_sizes:

        #rename the summary file to not interrupt the loop
        summary_new_name = f"summary_{n_batches}.h5"
        !mv summary.h5 {summary_new_name}

        n_inactive = (inactive*n_batches)//100  # inactive% of batches

        settings = openmc.Settings()
        settings.batches = n_batches
        settings.inactive = n_inactive
        settings.particles = particles
        settings.source = source
        settings.export_to_xml()

        openmc.run()

        #extract keff from the latest statepoint file
        sp = openmc.StatePoint(f"statepoint.{n_batches}.h5")

        #storing values

        keff_stddev.append(sp.keff.std_dev)   #mean and standard deviation of keff
        keffs.append(sp.keff.nominal_value)

        #compute relative error correctly

        keff_relative_error = np.array(keff_stddev) / np.array(keffs)
        relative_errors.append(keff_relative_error)
```

- **N**, number of cycles, $\mathrm{N} = 5$;
- **batch range**, interval (min batches, max batches) to test, $(50, 100)$;
- **inactive**, the percentage of inactive batches $(20\%/40\%)$;
- **source**, a source region equal to the whole fuel bundle, selecting only the fissionable material;
- **particles**, number of particles per batch.

```python
###Sensitivity analysis###

name="Inactive batches: 40%, Particles: 10^5 "
N=5
batch_range=(50,100)
particles=100000
inactive=40
source=openmc.IndependentSource(space=source_region,energy=source_energy,
                                constraints={"fissionable":True})
optimize_batches(name, N, batch_range, inactive, source, particles)
```

Sensitivity analysis

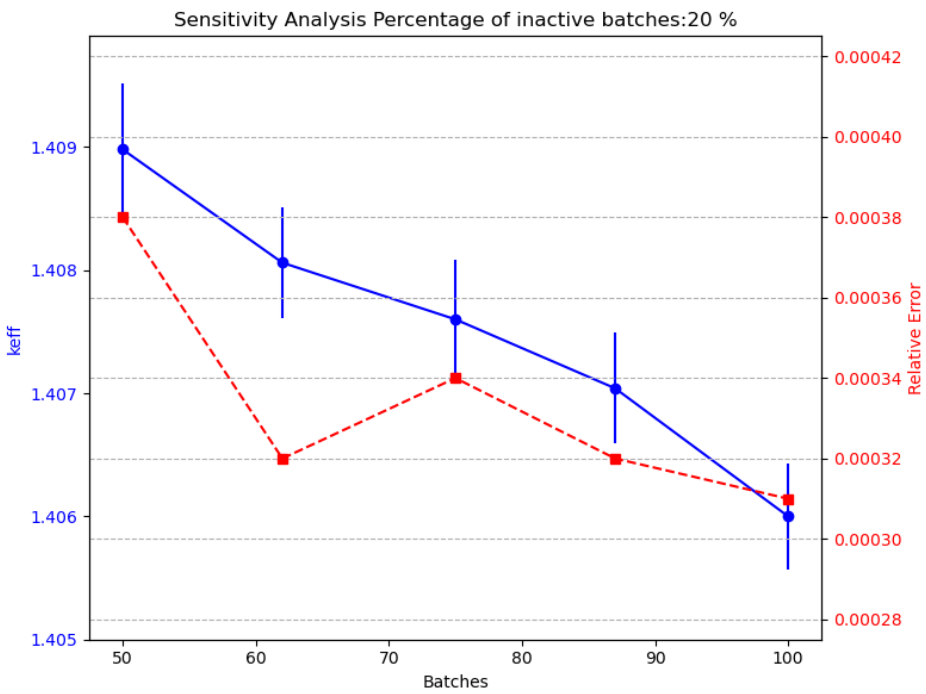


Figure: $k_{eff} = 1.4060 \pm 0.0004$ that means a relative error $\gamma_k = 3.1 \times 10^{-4}$

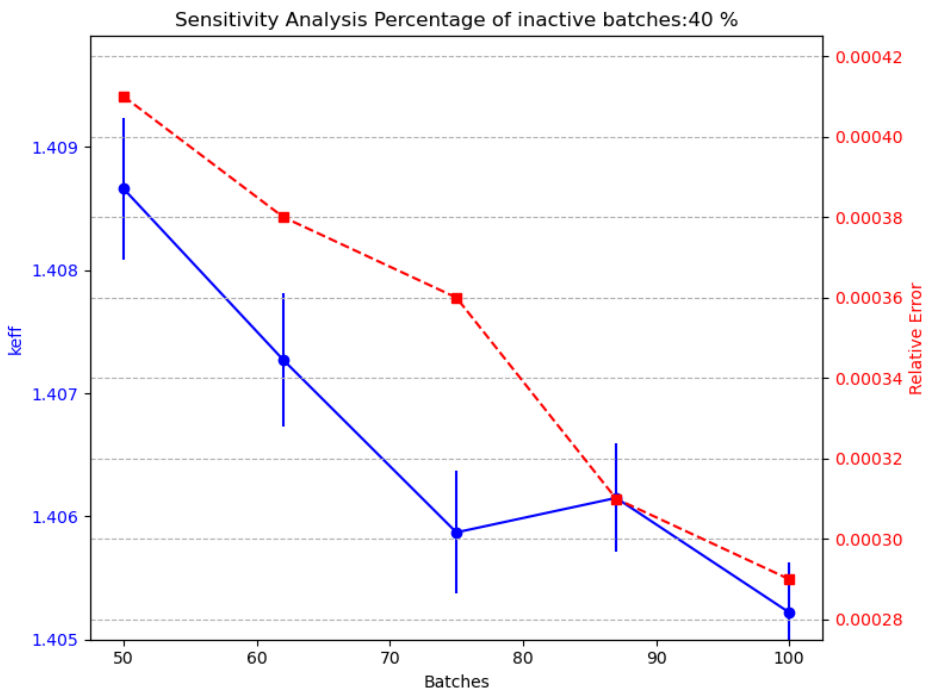


Figure: $k_{eff} = 1.4052 \pm 0.0004$ that means a relative error $\gamma_k = 2.8 \times 10^{-4}$

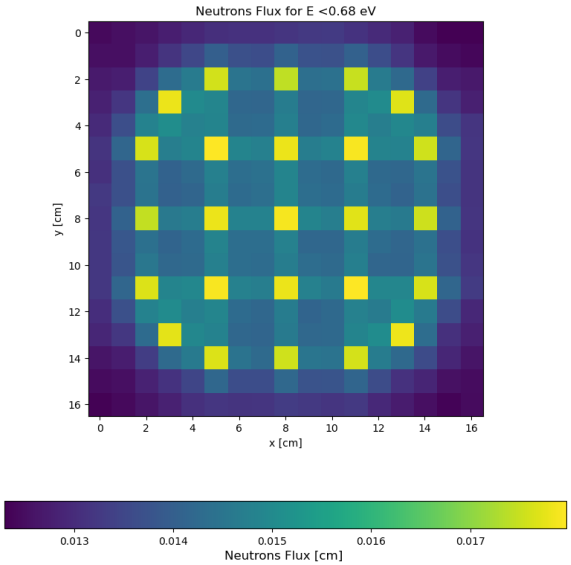
Sensitivity analysis

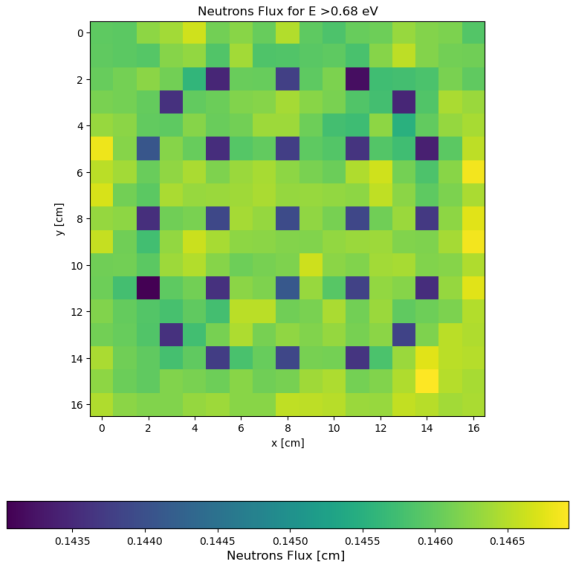
From previous graphs:

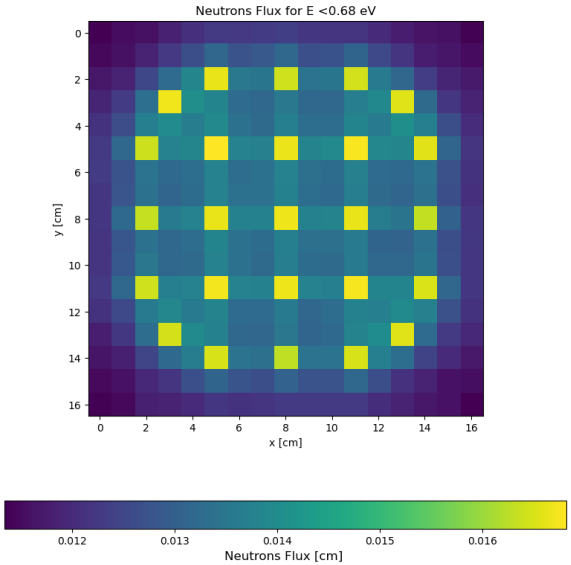
- The value of γ_k decrases as the number of batches increases;
- A higher γ_k may indicate an influence from source instabilities. However, the **relative error** remains more stable with a lower number of inactive batches.
- 10^5 particles were used because of limited computational capabilities.

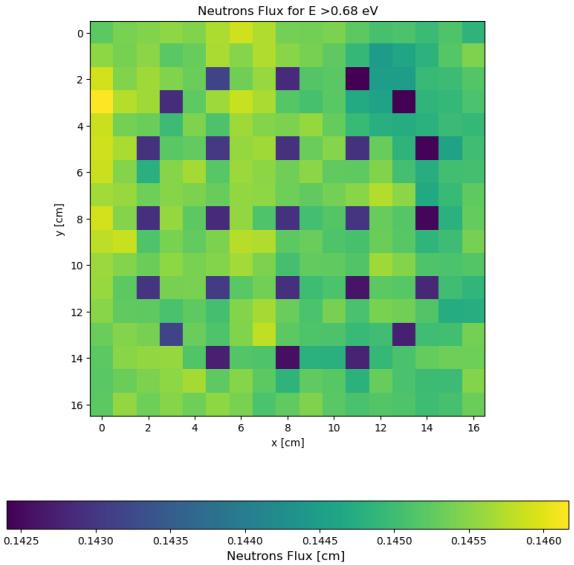
```
###Simulation Settings###

#this represents where the first gen of neutrons will appear
source_region= openmc.stats.Box((-bundle_pitch/2,-bundle_pitch/2,-1.),
                                (bundle_pitch/2,bundle_pitch/2,1.))
source_energy= openmc.stats.Watt( a=988000.0,b=2.249e-06)
source=openmc.IndependentSource(space=source_region,energy=source_energy,
                                constraints={"fissionable":True})


sim_settings= openmc.Settings()
sim_settings.run_mode='eigenvalue'
sim_settings.particles =100000
sim_settings.inactive=20
sim_settings.batches=100
sim_settings.source= source
```

Fluxes comparison







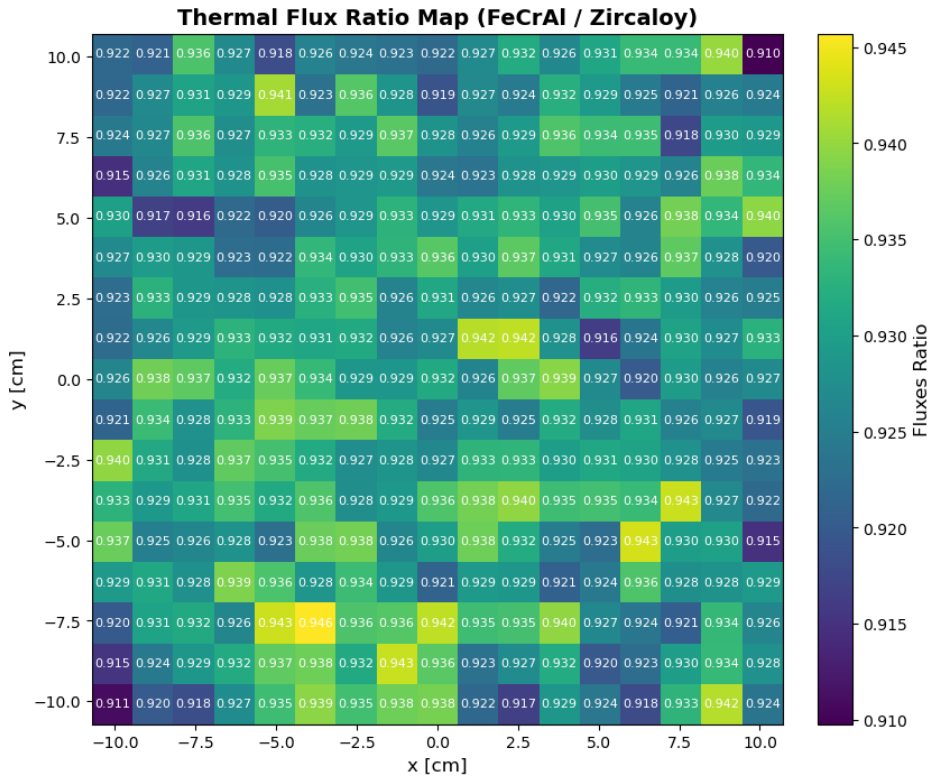


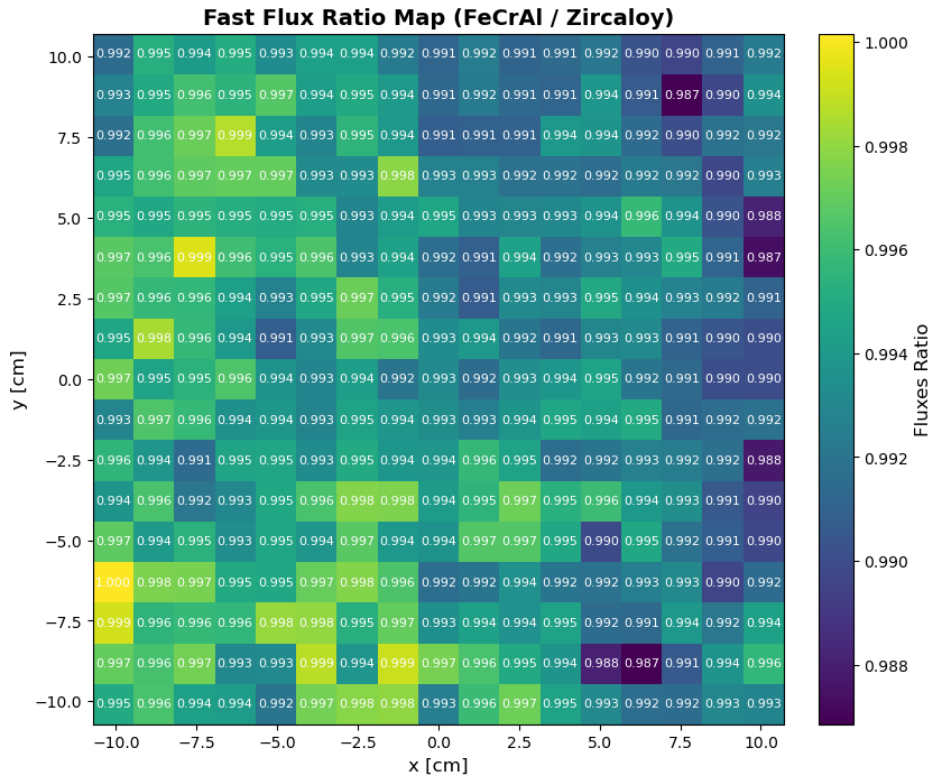
Upper map Zircaloy bundle; lower map Fecral bundle.

Assembly	**Thermal flux** [cm]	**Fast flux** [cm]
Zircaloy	0.012-0.018	0.143-0.147
FeCrAl	0.011-0.017	0.142-0.146

To better visualize the variation between the two bundles, the ratio of the fluxes for both energy bins has been plotted.

Fluxes comparison





Thermal flux	Fast flux
0.910-0.946	0.987-1.000

PPF evaluation

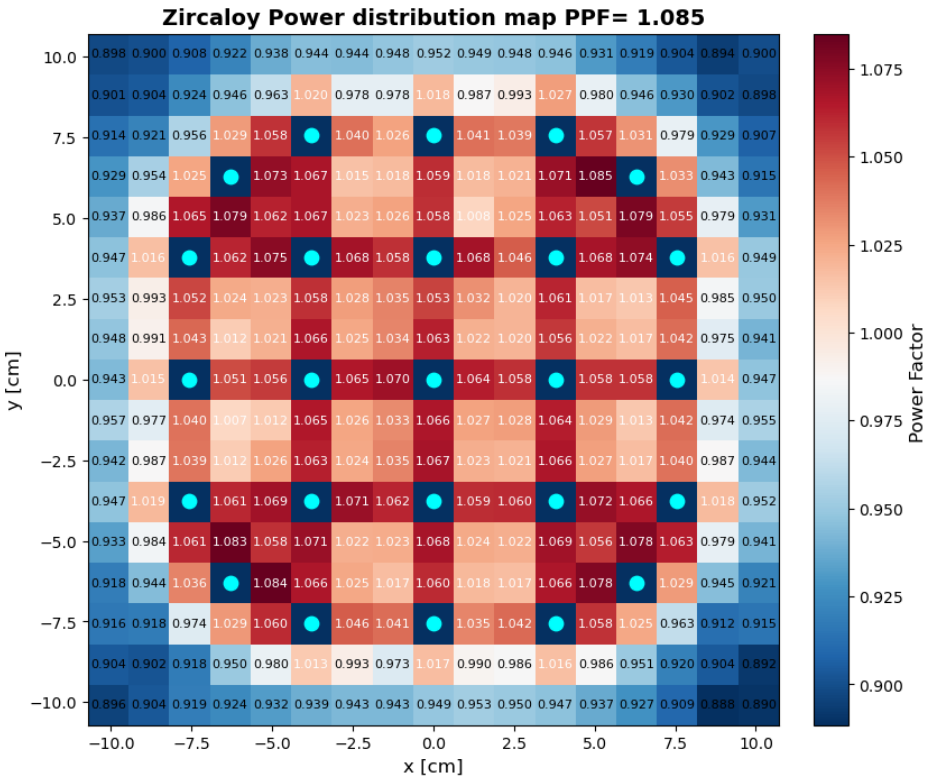
The PPF is defined as:

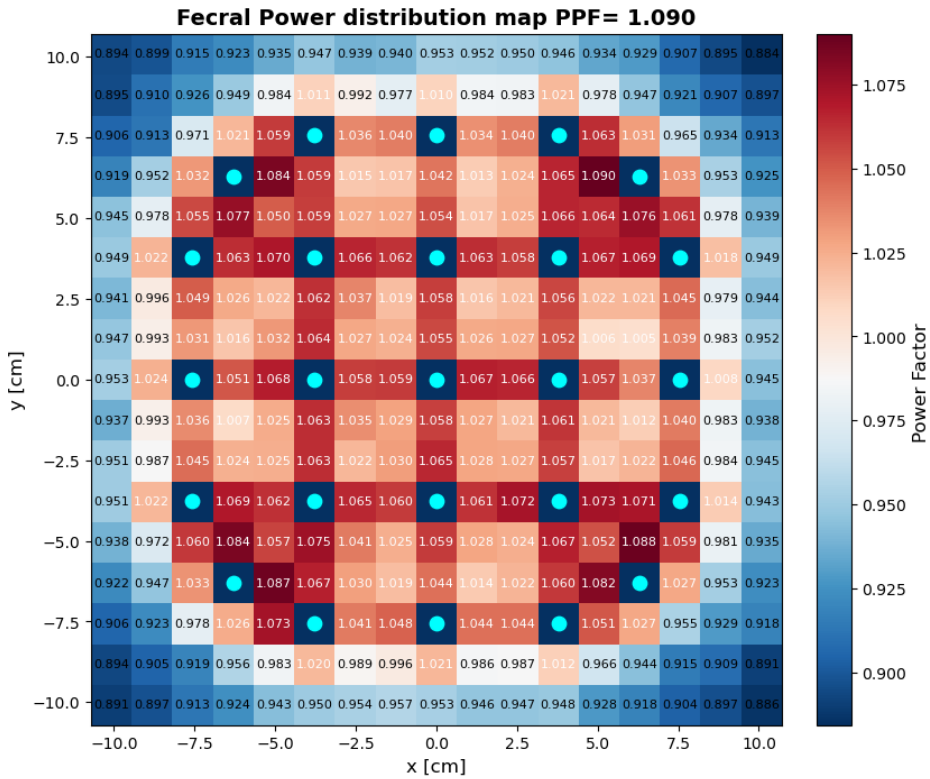
$$\mathrm{PPF} = \frac{\mathrm{P}_{max}}{\mathrm{P}_{avg}}.$$

Since the power produced per pin is proportional to the flux, the PPF should be independent of it, implying that FeCrAl and Zircaloy PPFs should be similar. The tally score used was the **kappa-fission**.

```
### Evaluate the ppf###

ppf_tally = sp_file.get_tally(id=2)
fission_power = ppf_tally.get_values()

max_power = fission_power.max()
avg_power = fission_power.sum()/264
ppf = max_power / avg_power
```

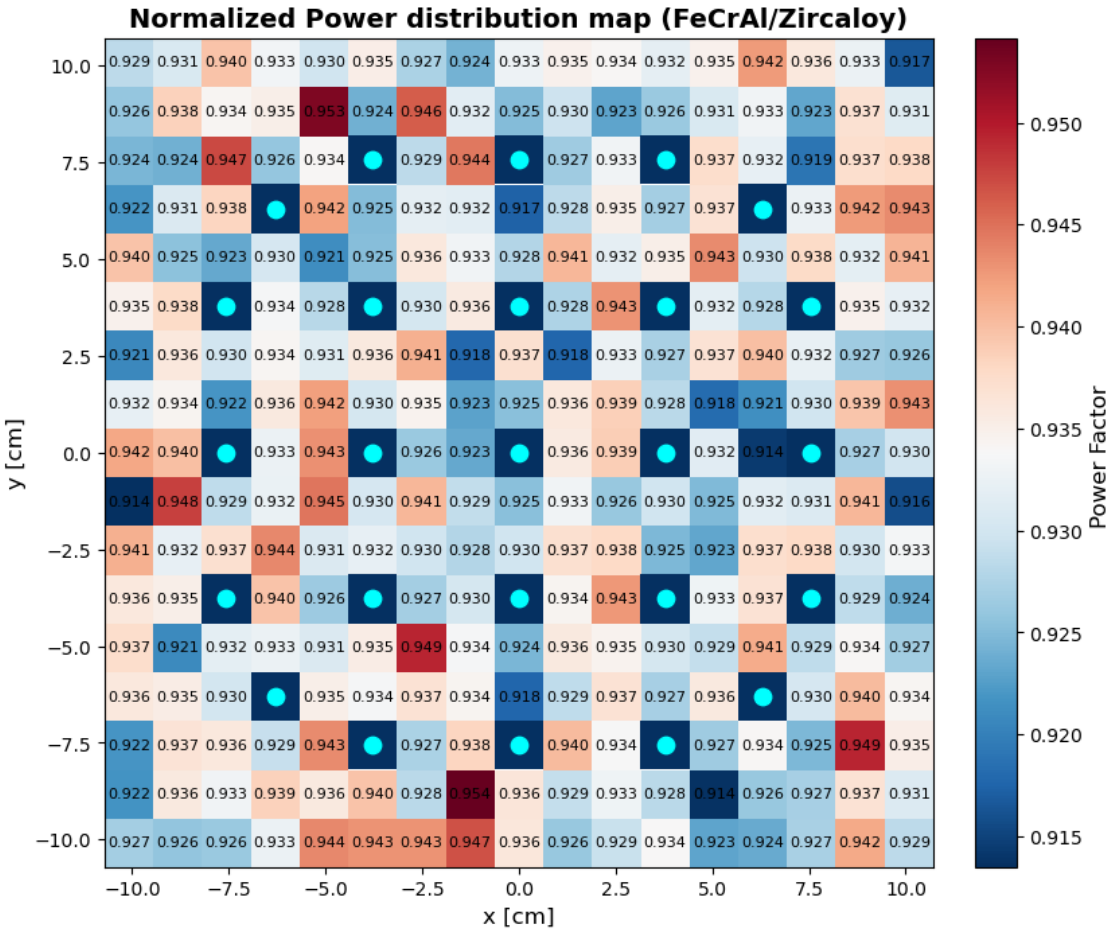
PPF evaluation





Zircaloy range	FeCrAl range
0.888-1.085	0.884-1.090

Power distribution comparison



The power distribution of Fecral bundle has been normalized respect the power distribution of Zircaloy.

Range: $0.914 - 0.954$

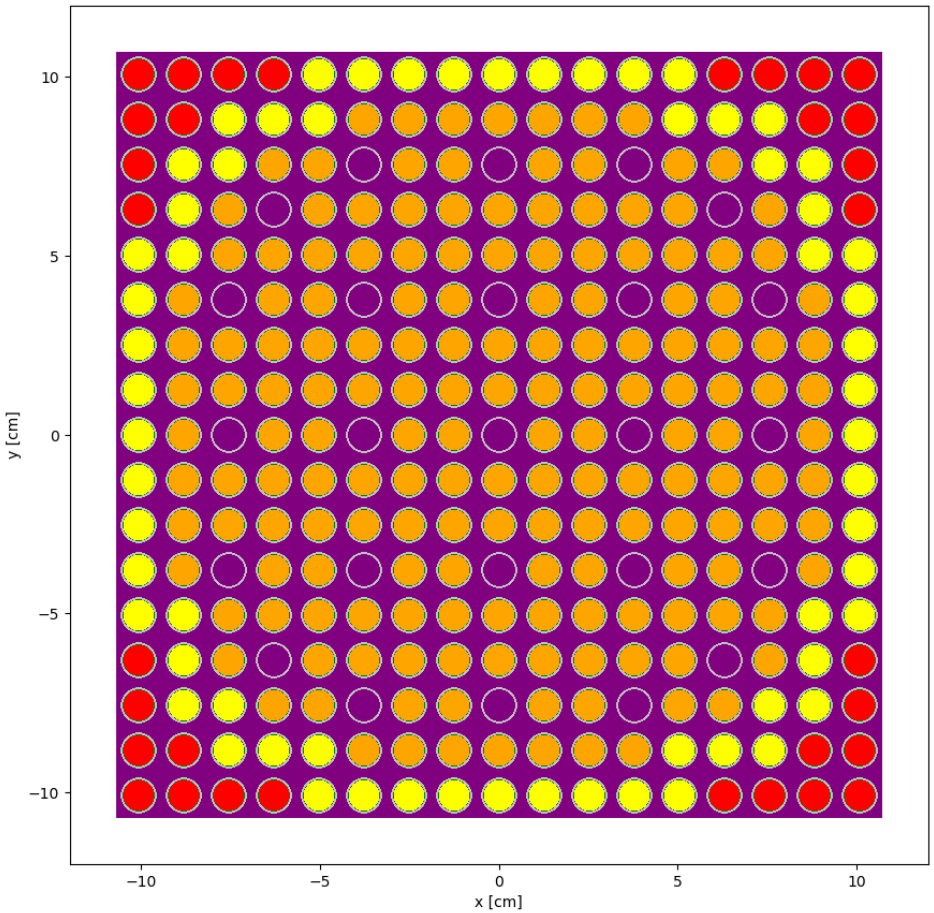
The values in the power map are similar to the thermal flux map. There is a slightly mismatch due to the **kappa-fission** score.

Enhancement of the assembly: strategies

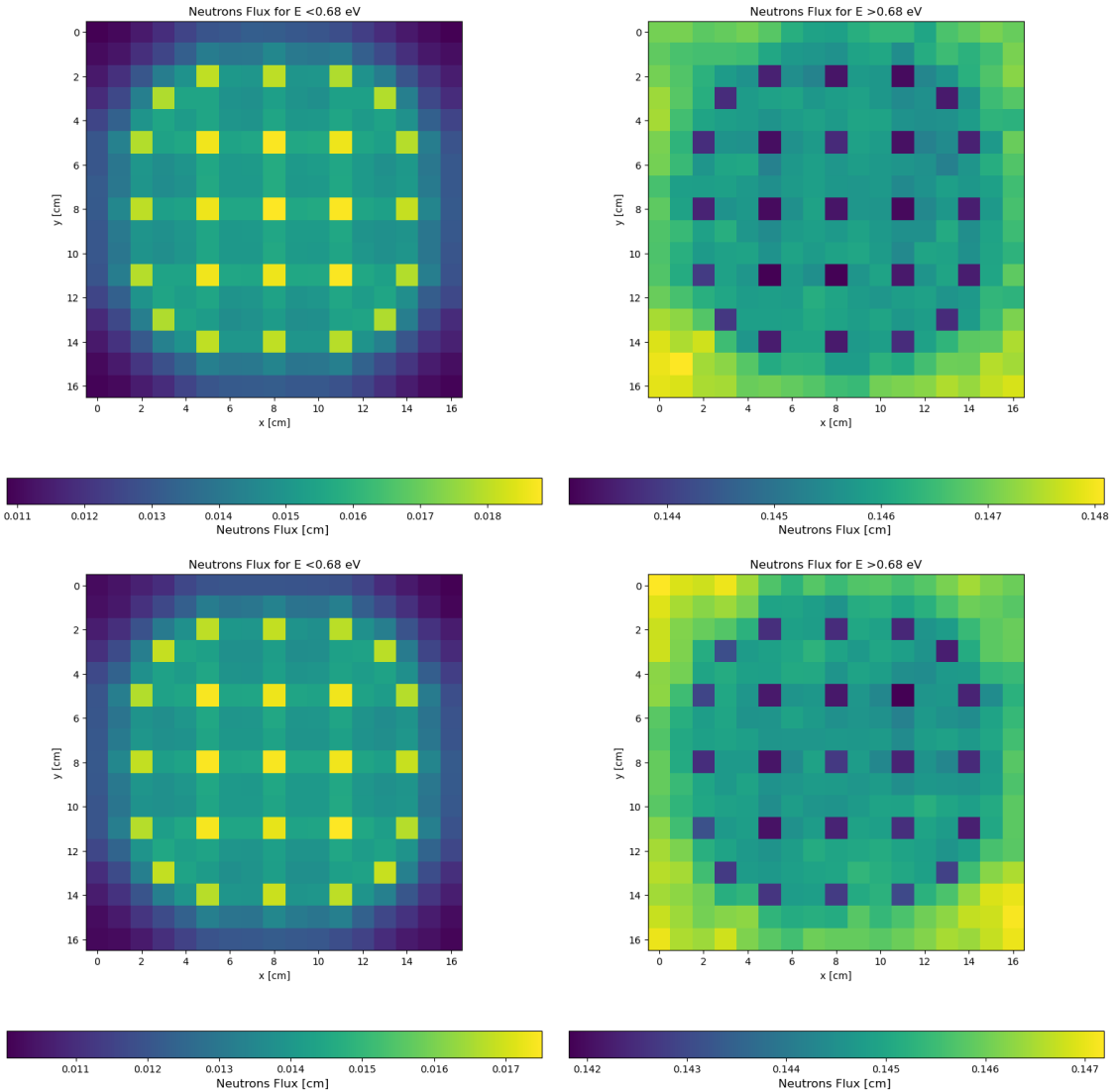
Another enrichment distribution has been adopted for the fuel assembly, following come constraints:

- respect the simmetry of the problem;
- mantain the **average enrichment** at 5%;
- try to **flatten the flux and reduce PPF**;

The attempt with three different enrichments $(6.20\%, 5.45\%, 4.6\%)$ is shown in the figure.



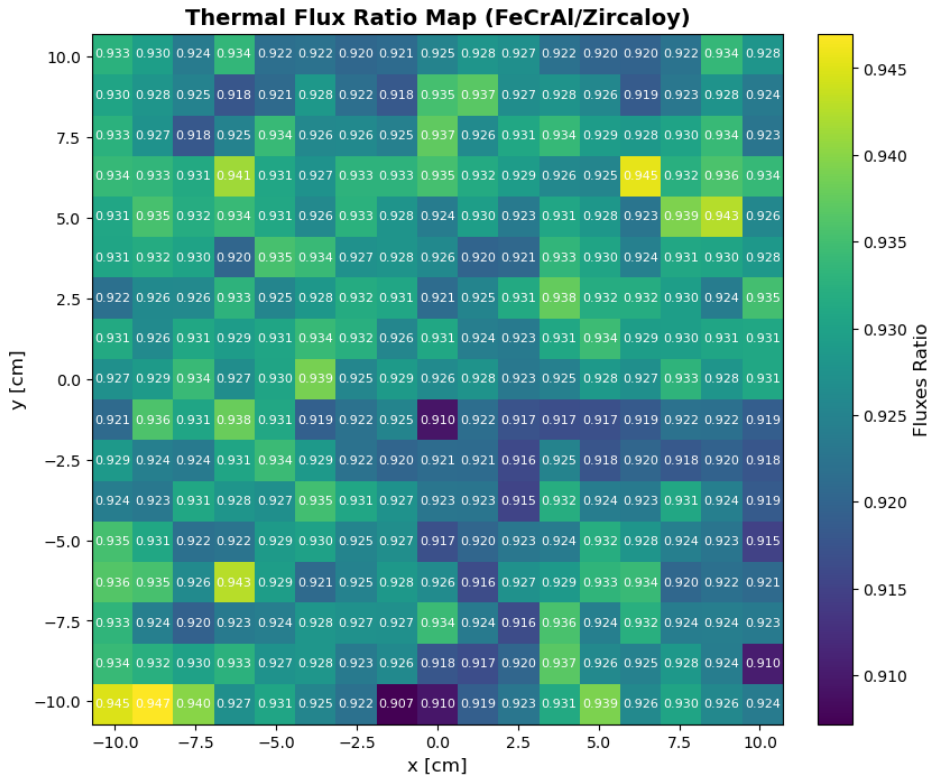
Enhancement of the assembly: Fluxes

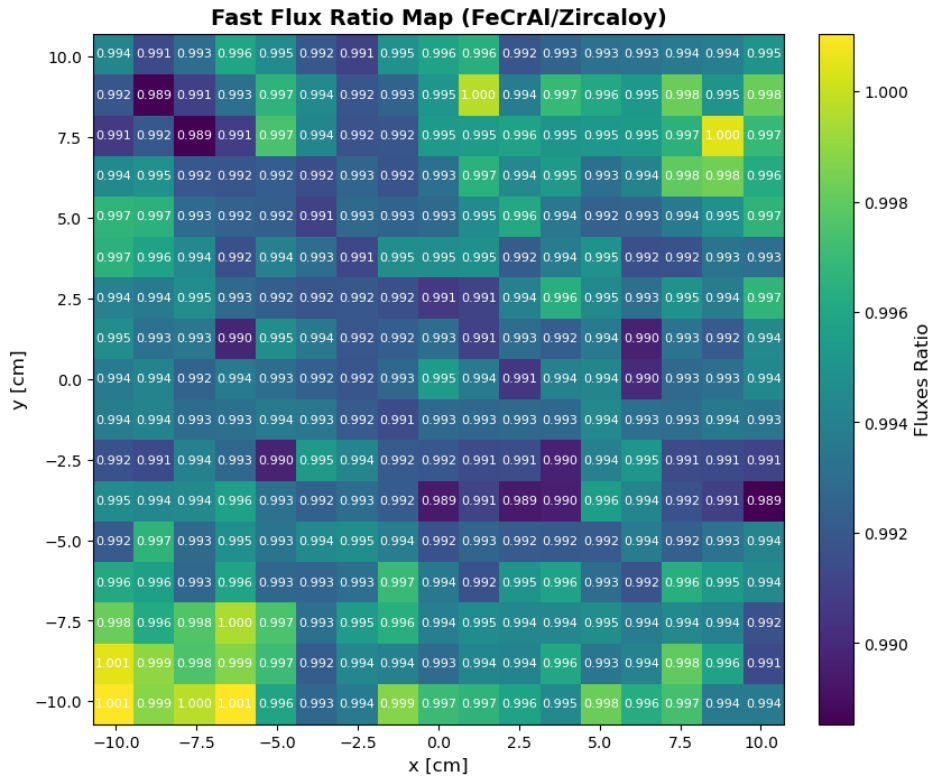


Upper map Zircaloy bundle; lower map Fecral bundle.

Assembly	**Thermal flux** [cm]	**Fast flux** [cm]
Zircaloy	0.011-0.019	0.143-0.148
FeCrAl	0.010-0.018	0.142-0.147

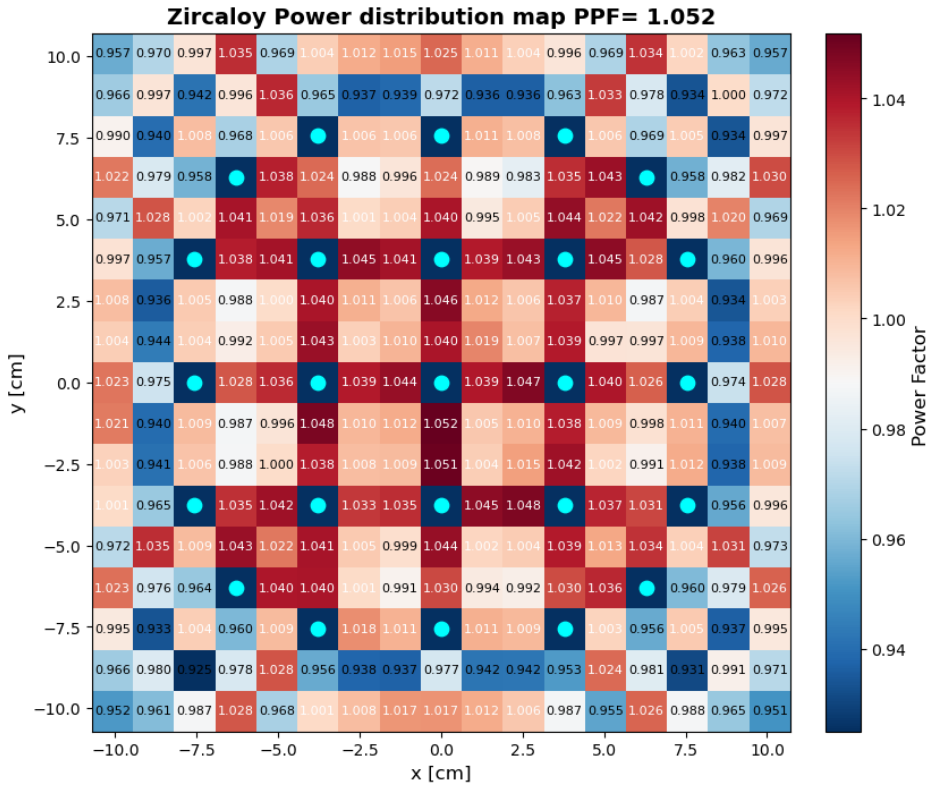
Enhancement of the assembly: Fluxes

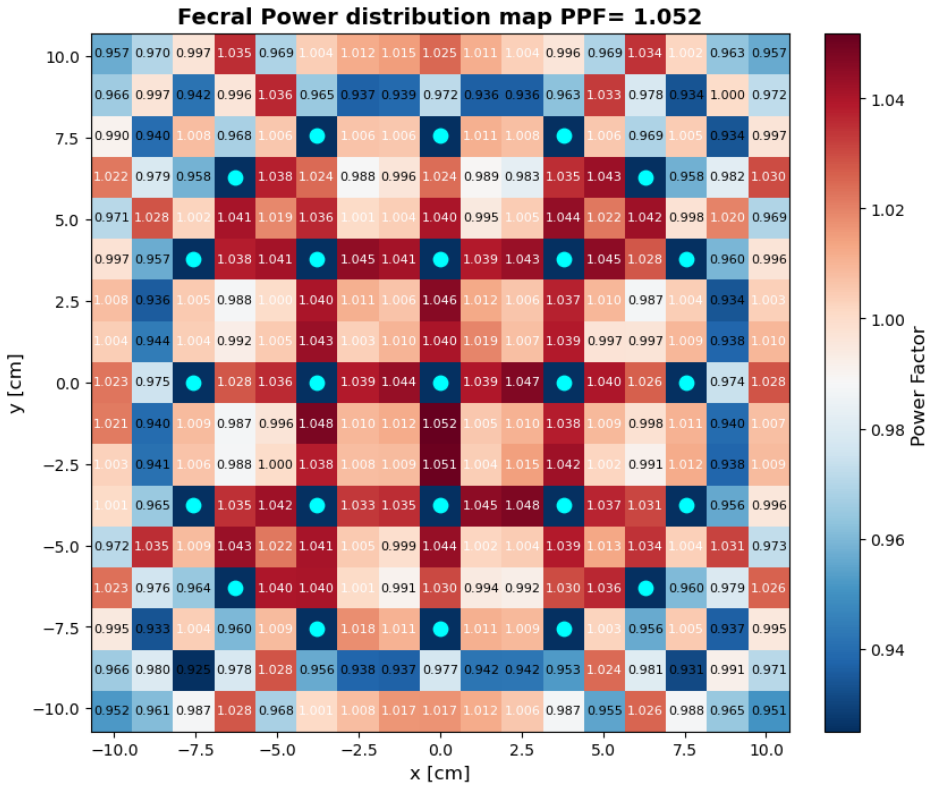




Thermal flux	Fast flux
0.907-0.947	0.989-1.001

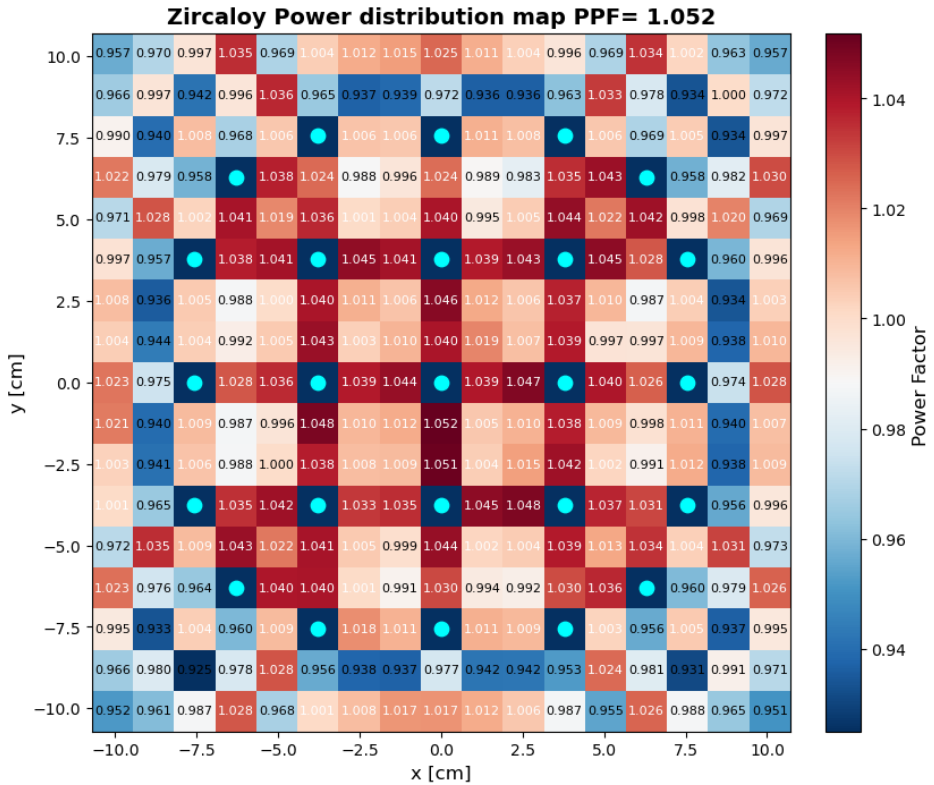
Enhancement of the assembly: Power map

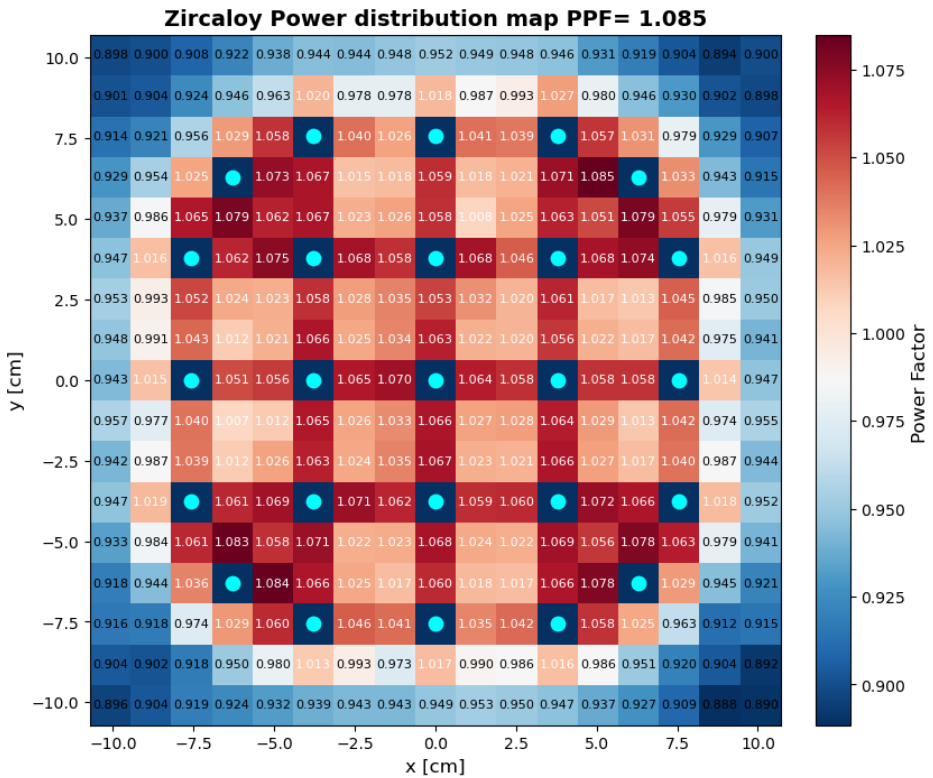




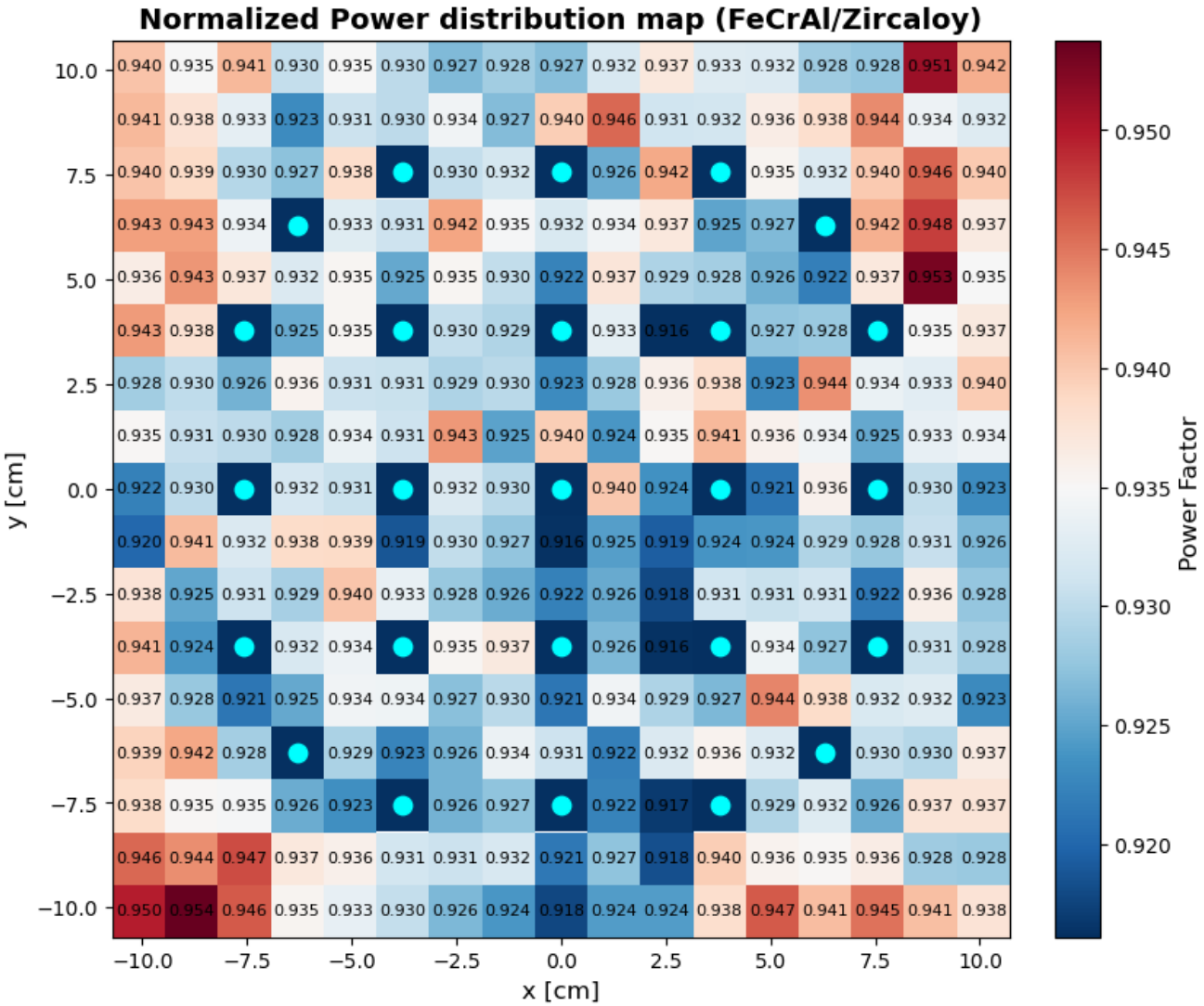
Zircaloy range	FeCrAl range
0.925-1.052	0.925-1.052

Enhancement of the assembly: Power map





Enhancement of the assembly: Normalized map



Range: $0.916 - 0.954$

Further Improvements

- The geometries adopted were strongly simplified, a **higher degree of detail** could be achieved;
- a **higher number of particles** should be implemented in order to reduce fluctuation effects. These effects were mainly visible plotting the flux maps;
- increasing the number of particle means also to increase the computational cost. Then, this solution should be accomplished exploiting the symmetries of the problem.

Conclusion

Primarily, due to its **absorption microscopic cross section**, FeCrAl exhibits a lower k_{eff} and reduced power production per fuel pin.

- However, evaluating the **PPF for both bundles**, a similar behavior has been obtained, implying **a similar consumption of fissile material**. **This is true also using a different enrichment pattern**, but with a uniform consumption of fuel.
- To overcome the problem related to the microscopic CS is possible to **reduce the thickness of the clad**, respecting the mechanical limit of the material.
- Finally, the main objective of ATF should be the capability to withstand against a accident condition, thanks to some solutions proposed also here, the FeCrAl alloys remains promising candidates.

Bibliography

[1] NEA Expert Group on Accident-tolerant Fuels for Light Water Reactors (EGATFL). *NEA State-of-the-Art Report on Light Water Reactor Accident-Tolerant Fuels*. Tech. rep. Nuclear Science, OECD Publishing, 2018. DOI: 10.1787/9789264308343-en.

[2] Kevin G. Field et al. *Handbook on the Material Properties of FeCrAl Alloys for Nuclear Power Production Applications (FY18 Version: Revision 1)*. Tech. rep. 2018. DOI: 10.2172/1474581.

[3] Paul K. Romano et al. "OpenMC: A state-of-the-art Monte Carlo code for research and development". In: *Annals of Nuclear Energy* (2015). DOI: 10.1016/j.anucene.2014.07.048.