EXTENDING SERPENT

Staffan Qvist
Department of Physics & Astrophysics
Divison of Applied Nuclear Physics

Serpent capabilities

What it can do:

— Almost everything

What it can’t do (right now):

— Equilibrium cycle analysis

— (Direct) DPA calculations

— Make your input deck and design your core for you

Equilibrium cycle

- Most cores load, shuffle and discharge fuel over and over
- Eventually, an “equilibrium” is reached where each new cycle looks like the preceeding one
- The equilibrium state can be defined by looking at the relative difference between suceeding cycles of a few variables:
  1. Multiplication factor (at each burnup step within cycle)
  2. Isotopic concentrations (at each burnup step within cycle)
  3. Cross-sections (at each burnup step within cycle)
  4. Flux/Spectrum/Power parameters etc..
- Logial convergence criteria: Differences < Statistics

Equilibrium cycle

- **2 possible ways to approach the problem:**

  1. Get to equilibrium state as quickly as possible

  2. Accurately model approach to equilibrium

- **EDIS (Every Day I'm Shuffling)**

Runs a 2-stage scheme (to be explained)

Considers 3 things:

- Convergence of k_{eff} and isotopic compositions
- Criticality over cycle (adjusts cycle time to try to find it)
- Peak DPA & burnup over cycle

The two-step approach (1)

- **For quickly reaching the equilibrium state**
  - S1: Serpent settings tailored for *fast results*, low accuracy, bad statistics,

    Runs to move fuel <x> times through its path through the core
  - S2: Run at “production” statistics and settings for a maximum of <y> times through the core (or, hopefully, until convergence settings are reached)

- **To accurately model approach to equilibrium**
  - S1 = S2

<x> and <y> depend heavily on the number of batches in the system!

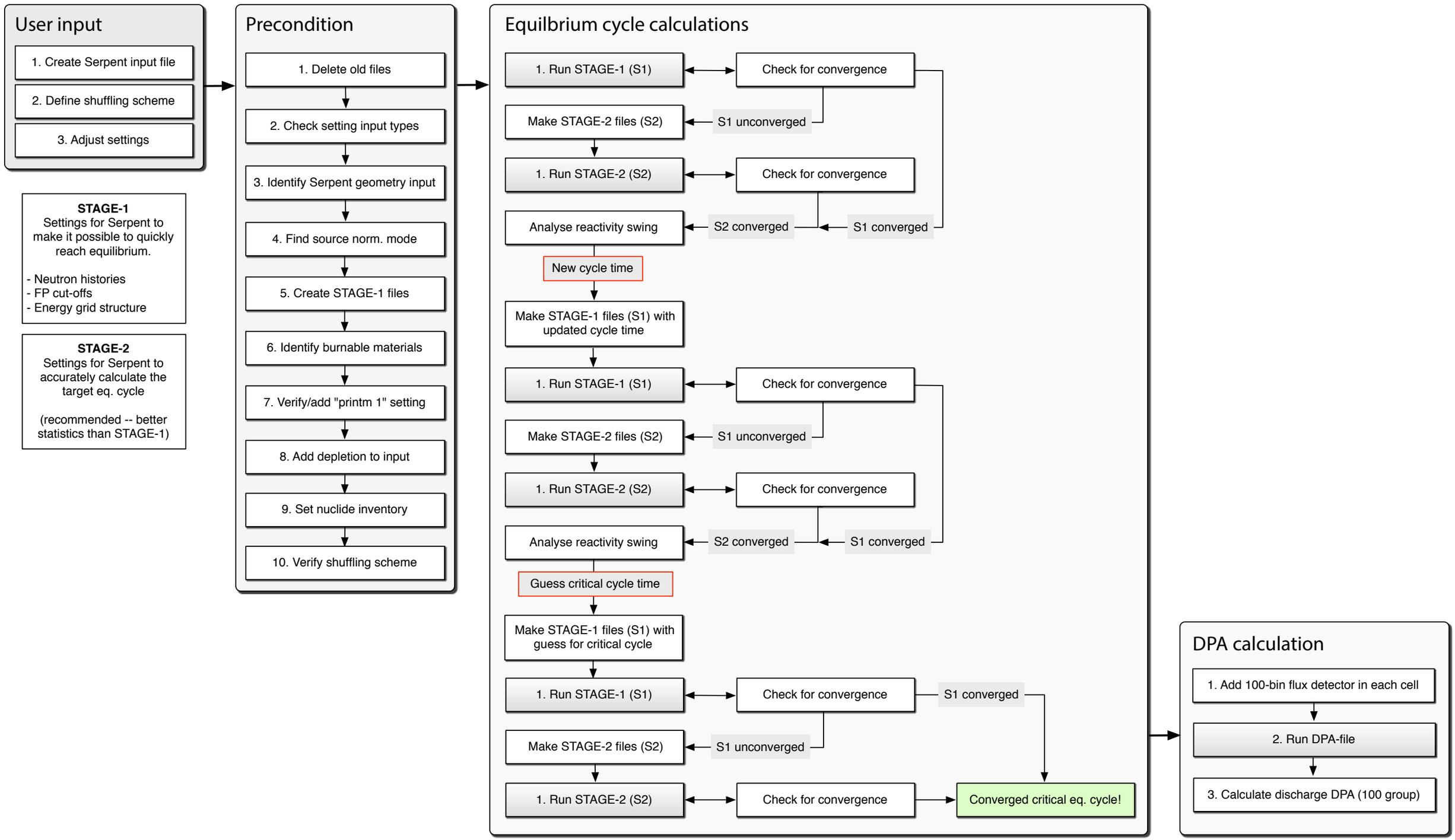
The critical cycle approach

1. Run S1-S2 for an initial guess for cycle time
2. Define min/max k_{eff} and reactivity swing type:

   1: Positive, 2: Negative, 3: Local extreme
3. Adjust cycle time based on swing type
4. Run S1-S2 for adjusted cycle time
5. Interpolate between results from (2) and (4)
6. Run S1-S2 for new guess for k_{eff}
7. If initial cycle time guess was good, k_{eff} should be at or very near target (usually = 1.00 + Δx)

   Otherwise, oops! Try again with better guess for cycle time

EDIS structure

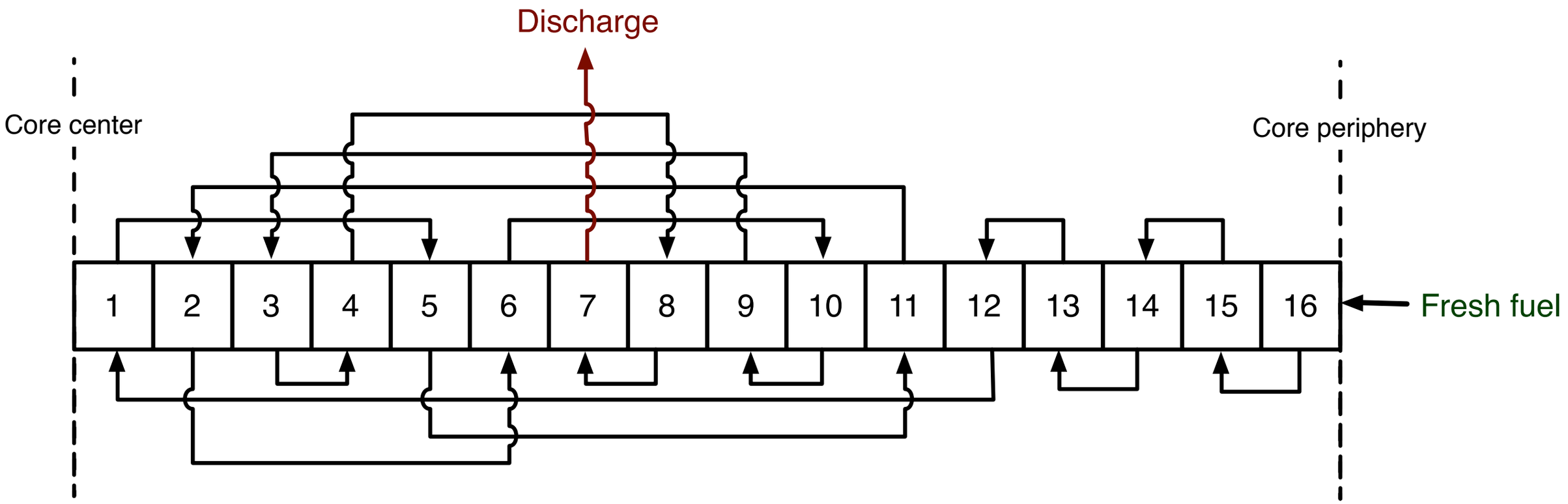


Example problem

Large 16-batch (TWR-type) B&B core fed by depleted uranium

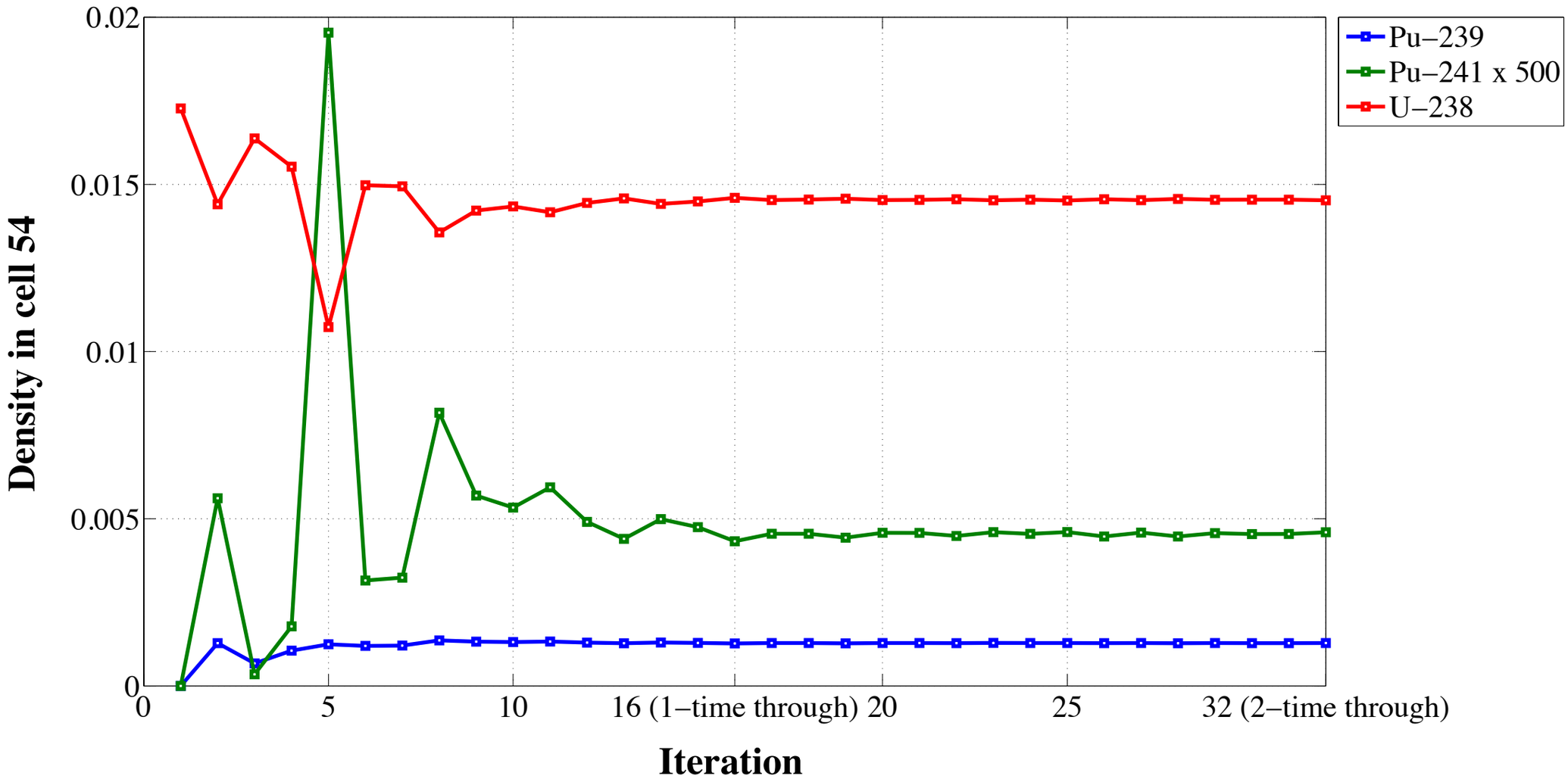
Complicated 2D shuffling scheme for power flattening

Problem: Find equilibrium core performance, critical cycle time, burnup and peak DPA



Material convergence

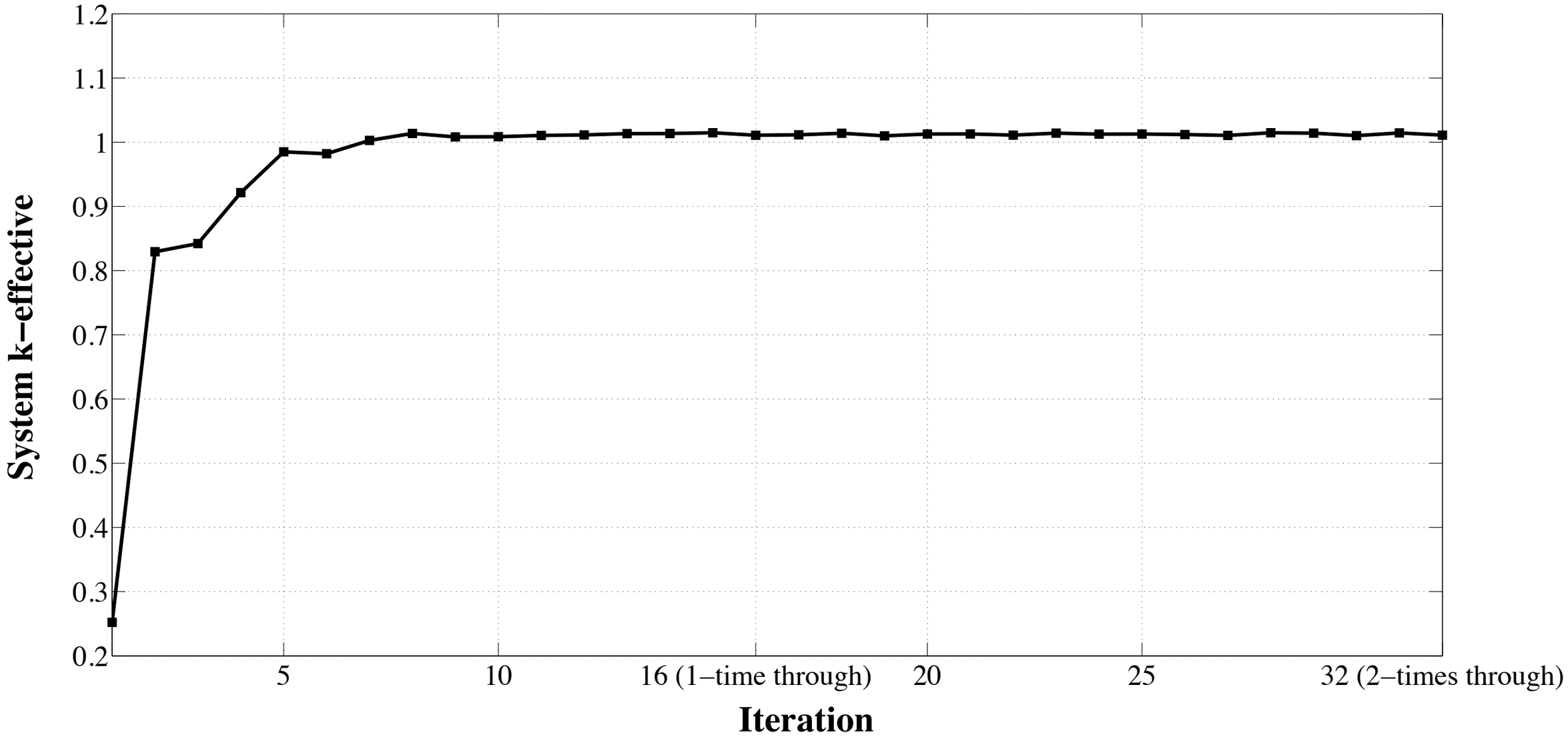
Isotope density in low-flux cell of 16-batch system, modeled as 80 homogenized core cells. Error in ^{241}Pu effectively sets the number of required iterations



Multiplication convergence

System (fed by DU) k_{eff} converged within statistics before

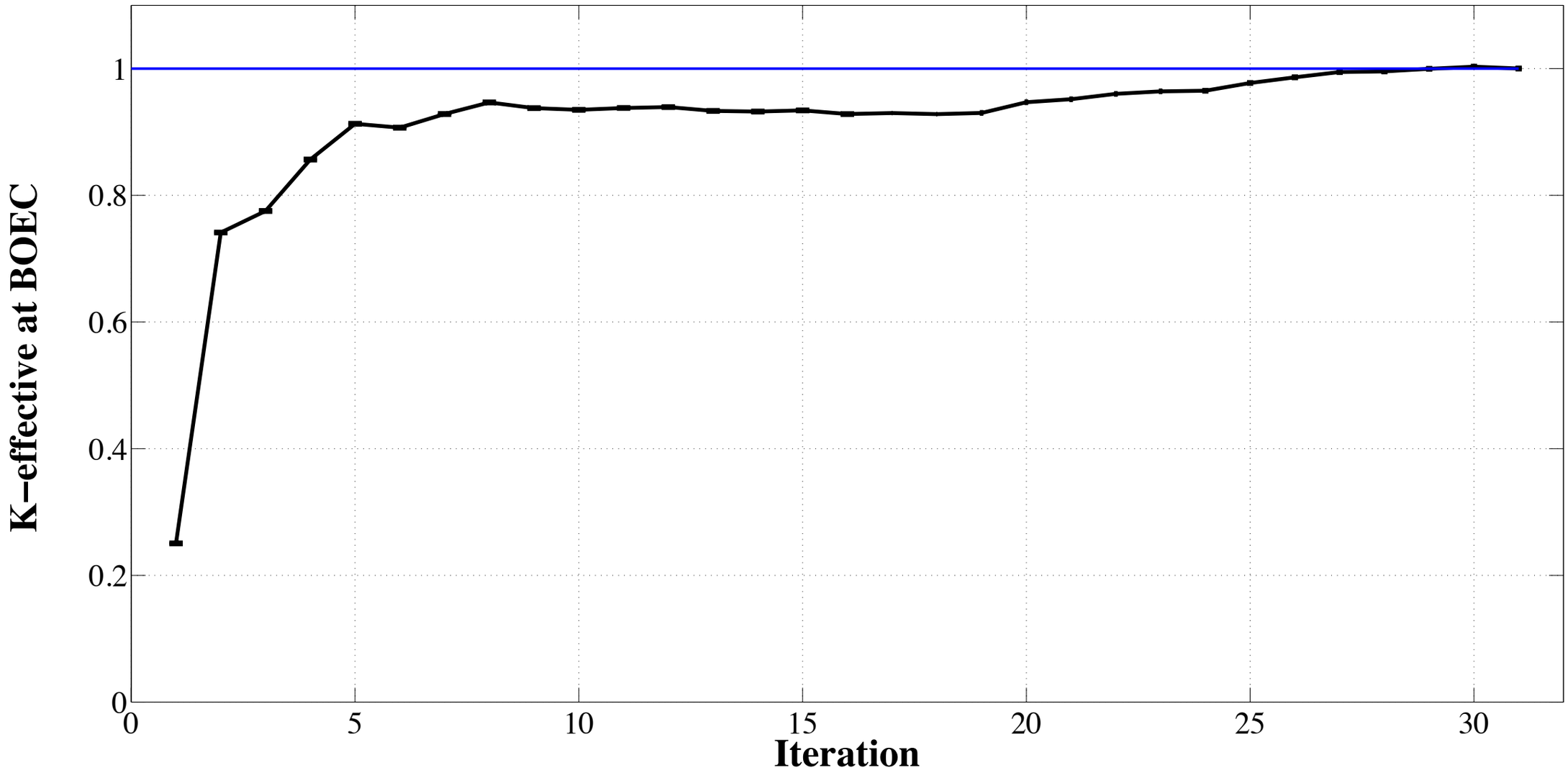
(rare) isotopic compositions converge (as expected)



Critical cycle mode

k_{eff} convergence criteria: 200 pcm, Material convergence criteria = 5%,

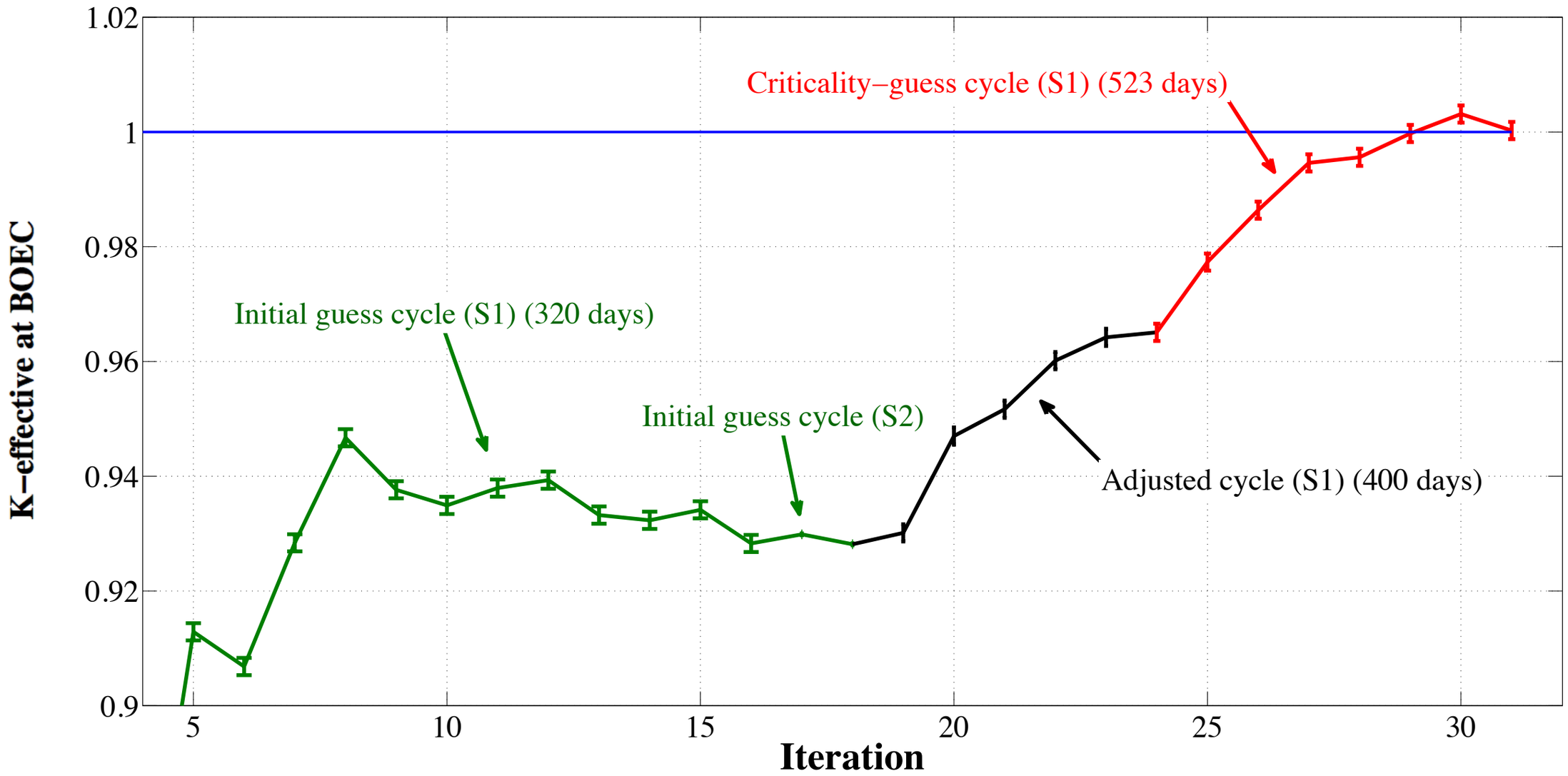
5 burnup steps, Criticiality search = on



Critical cycle mode

k_{eff} convergence criteria: 200 pcm, Material convergence criteria = 5%,

5 burnup steps, Criticiality search = on



Peak HT9 steel dpa: 407 -> 482 -> 555 (critical cycle)

Standalone EDIS

Needs 2 settings: Shuffling scheme and S1/S2

Needs 2 files: Main file(s) + seperate file with shuffled materials

Right now: maintains geometry, shuffles materials

(could/should be adjusted to shuffle universes)

Arbitrary shuffling scheme is possible

Arbitrary reprocessing at each shuffling step possible

Possible extension:

Shuffling scheme optimization? Needs *good theory* + genetic algorithms.. definitely impractical, maybe impossible..

Shuffling possibilites — 16 batch (cylinder) model = 7.6×10^{12} paths

Calculating DPA

DPA tallying for structural materials implemented in ADOPT

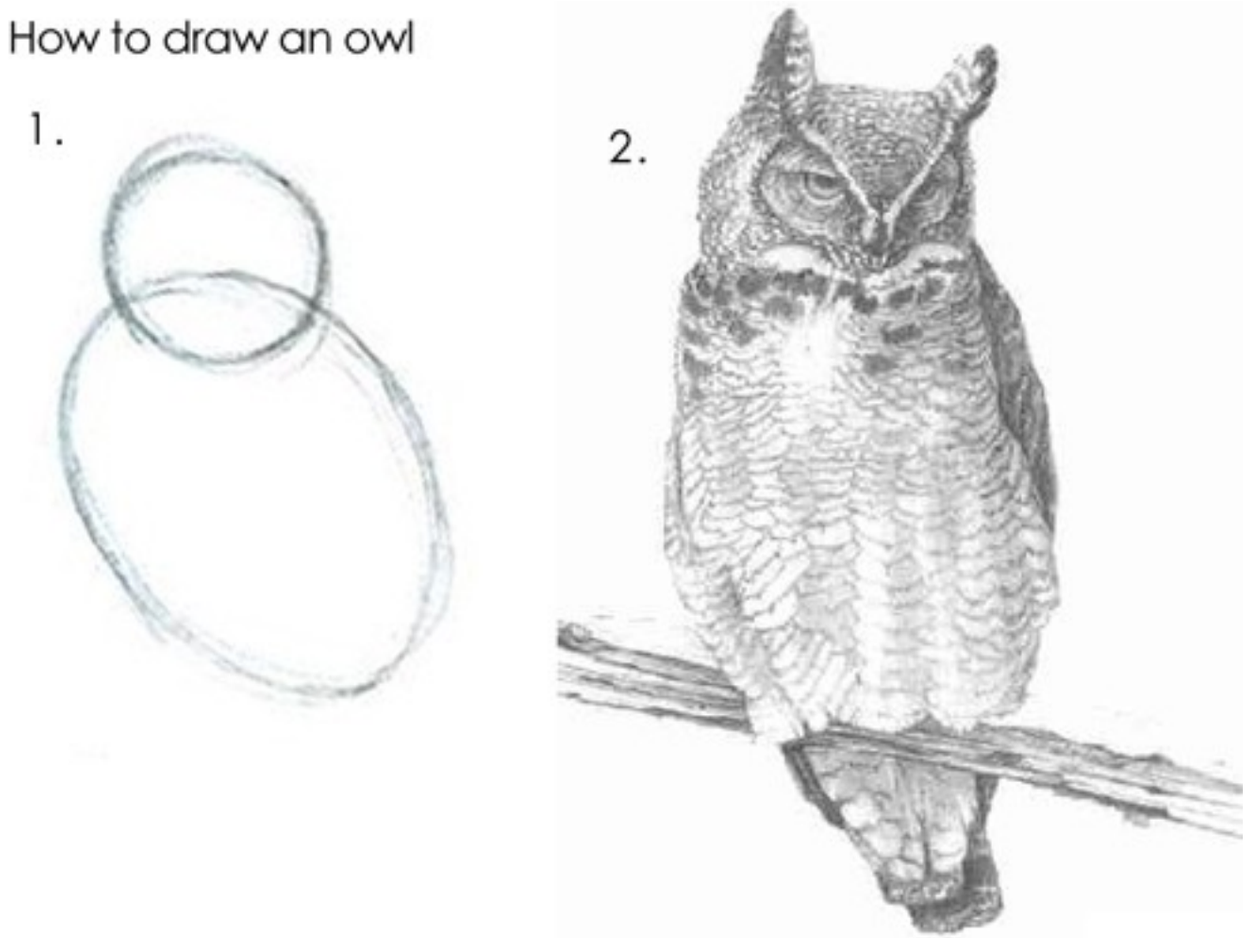
(and ADOPT stand-alone module *EDIS*)

Uses 100-group table effective dpa cross-section structure developed for the *ANL SPECTER* code for each element implemented. When compound cross-section is available (ex. for SiC), these are used.

To use outside of ADOPT or stand-alone *EDIS*:

1. Add pre-defined 100-group flux detector structure to Serpent input
2. Run Serpent
3. Run 1500-line Python script on the detector output

Core (owl) design process



1. Draw some circles

2. Draw the rest of the [redacted] owl

Core (owl) design process

How to draw an owl

1.

2.

1. Draw some circles

2. Draw the rest of the owl

Core (owl) design process

How to draw an owl

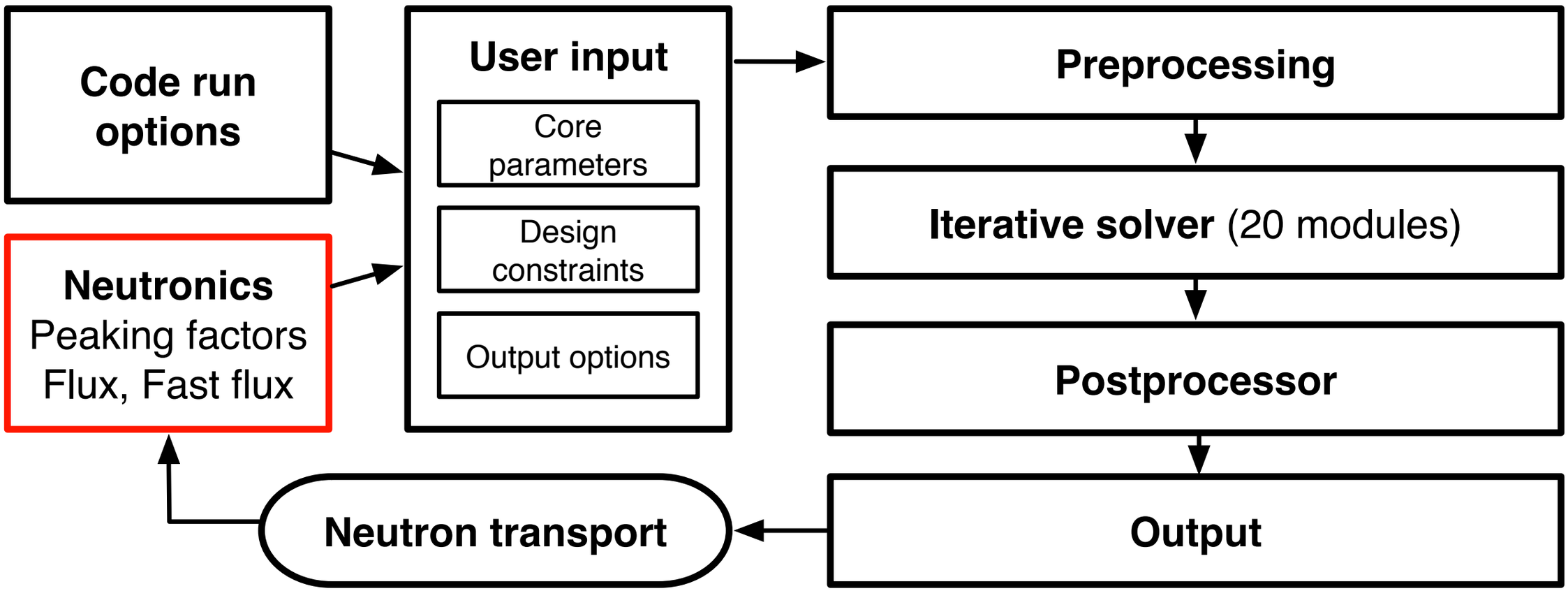
1.

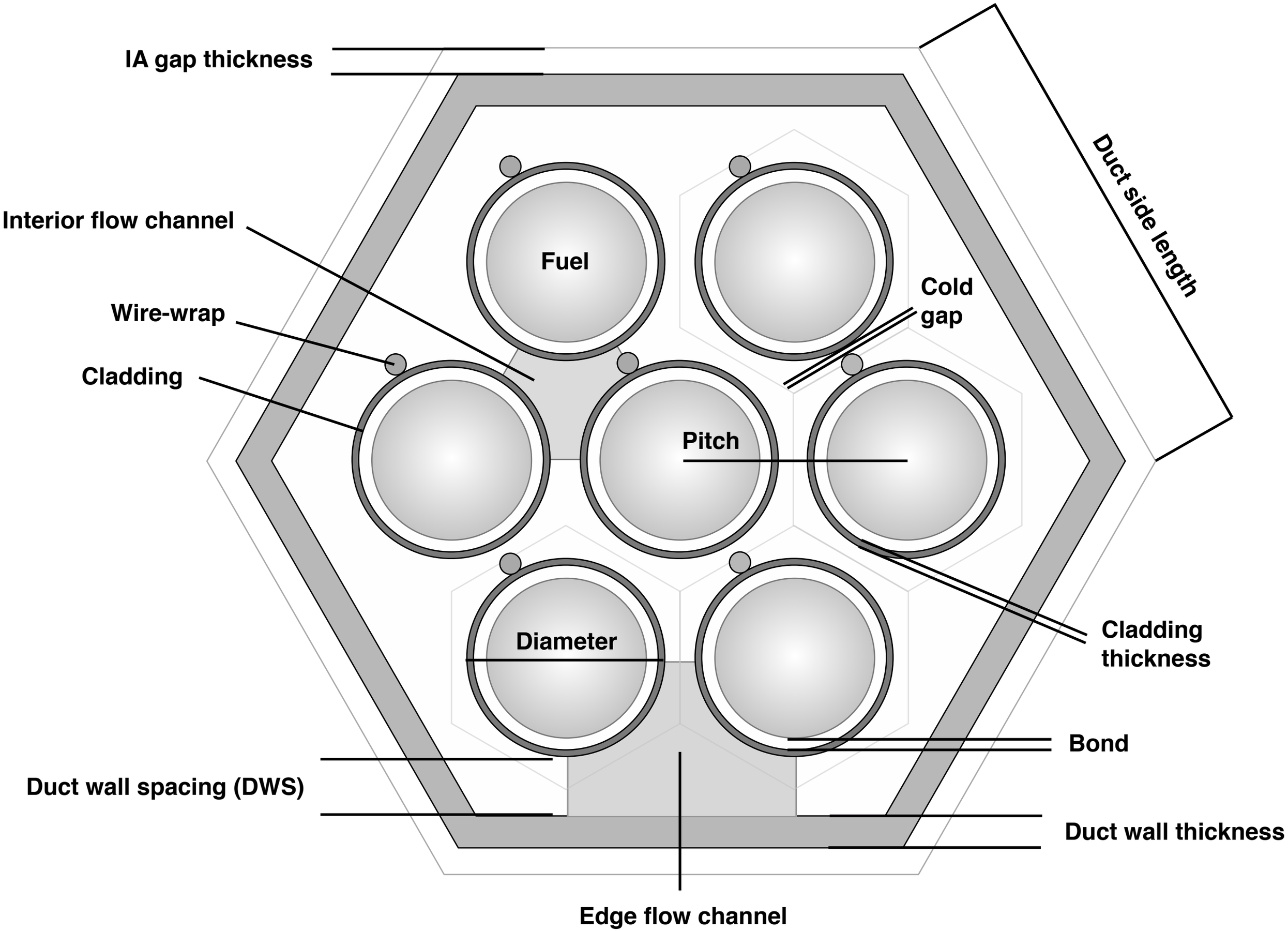
2.

1. Draw some circles

2. Draw the rest of the owl

ADOPT


Code run options
User input
Core parameters
Design constraints
Output options
Neutronics
Peaking factors
Flux, Fast flux
Preprocessing
Iterative solver (20 modules)
Postprocessor
Output
Neutron transport

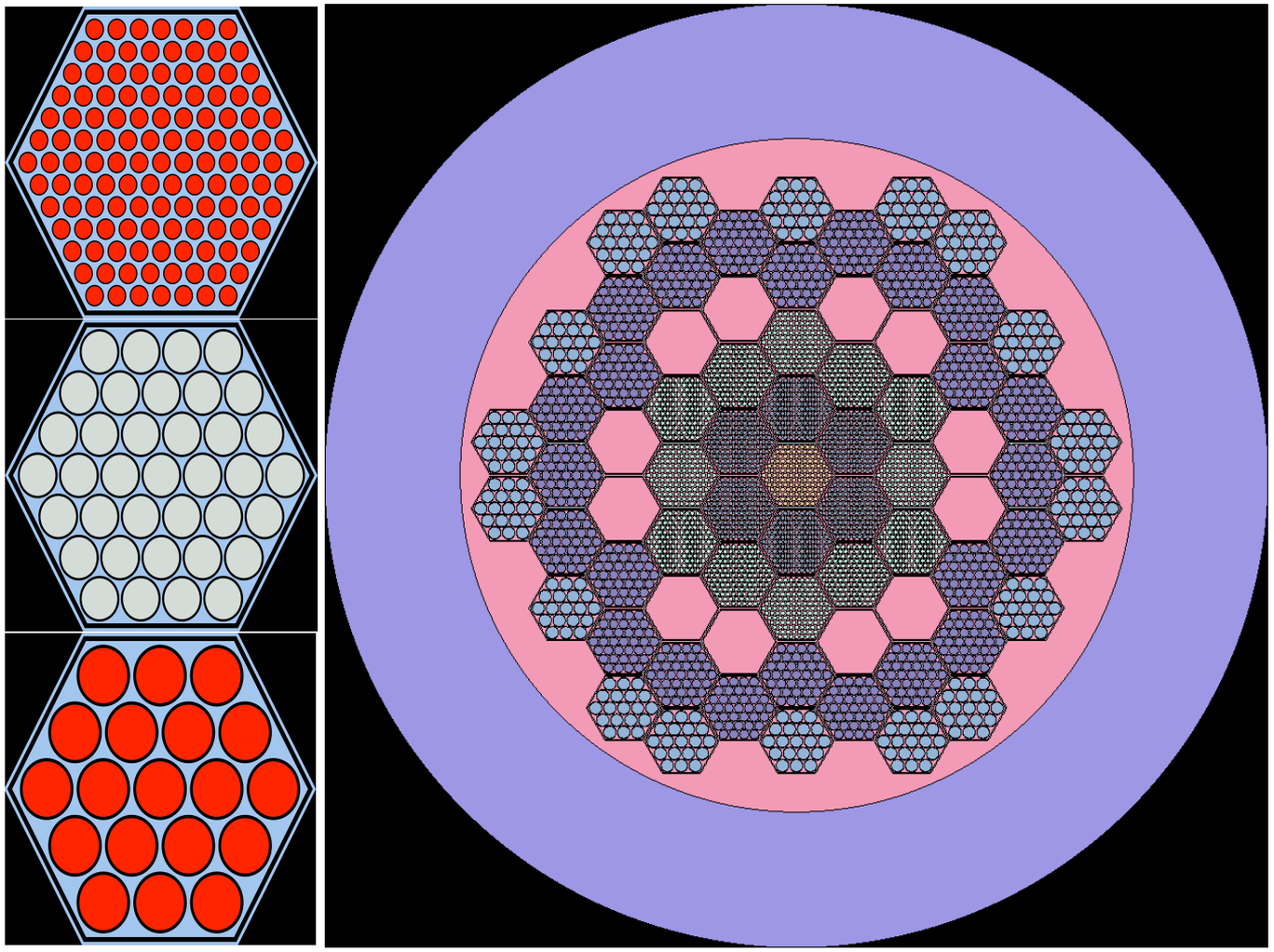

IA gap thickness
Interior flow channel
Wire-wrap
Cladding
Fuel
Cold gap
Duct side length
Pitch
Diameter
Cladding thickness
Bond
Duct wall spacing (DWS)
Duct wall thickness
Edge flow channel

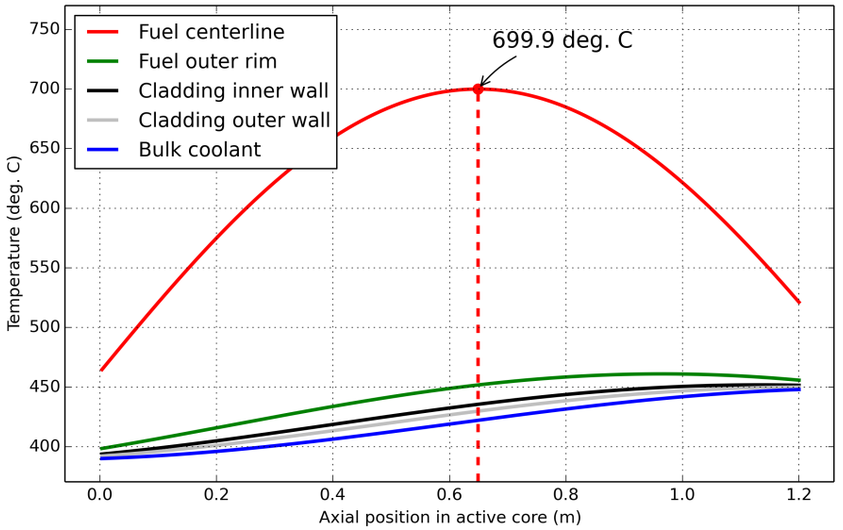
ADOPT

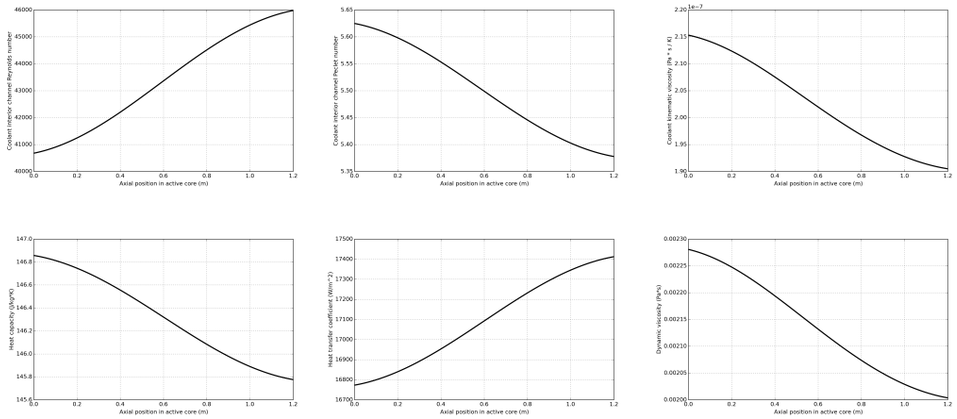
1. Reads a 100-parameter input file
2. Calculates thermal-hydraulic and structural-mechanical "optimal" solution adhering to 15 set constraints
3. Creates a full-core Serpent input deck
4. Runs Serpent and gathers needed data (Flux, Power distribution, DPA etc.)
5. (Optionally) runs a shuffling scheme until equilibrium convergence using the *EDIS* module
6. Reruns (2-4/5) until convergence
7. Plots and prints data for optimised and converged core design (several hundred output variables)

ADOPT output

- Full-detail core geometry
- Thermal-hydraulic, structural mechanic and neutronic (incl. fuel cycle) performance







ADOPT limitations

<u>Design code (not analysis code) — All methods are crude</u>

1. Cannot evaluate core transient safety performance
2. Does not create or model control systems
3. Geometry options are limited
4. Larger cores are modelled by concentric cylinders rather than individual assemblies, potentially introducing errors
5. Optimisation strategy only directly applicable for breeder-type cores
6. (Currently) no detailed model for FCMI
7. (Currently) only one fuel assembly design (at peak power conditions) per run

Serpent+Friends capabilities

What it can do:

— Almost everything
— Equilibrium cycle analysis
— (Direct) DPA calculations
— Make your input deck and design your core for you :)

Core design challenge!

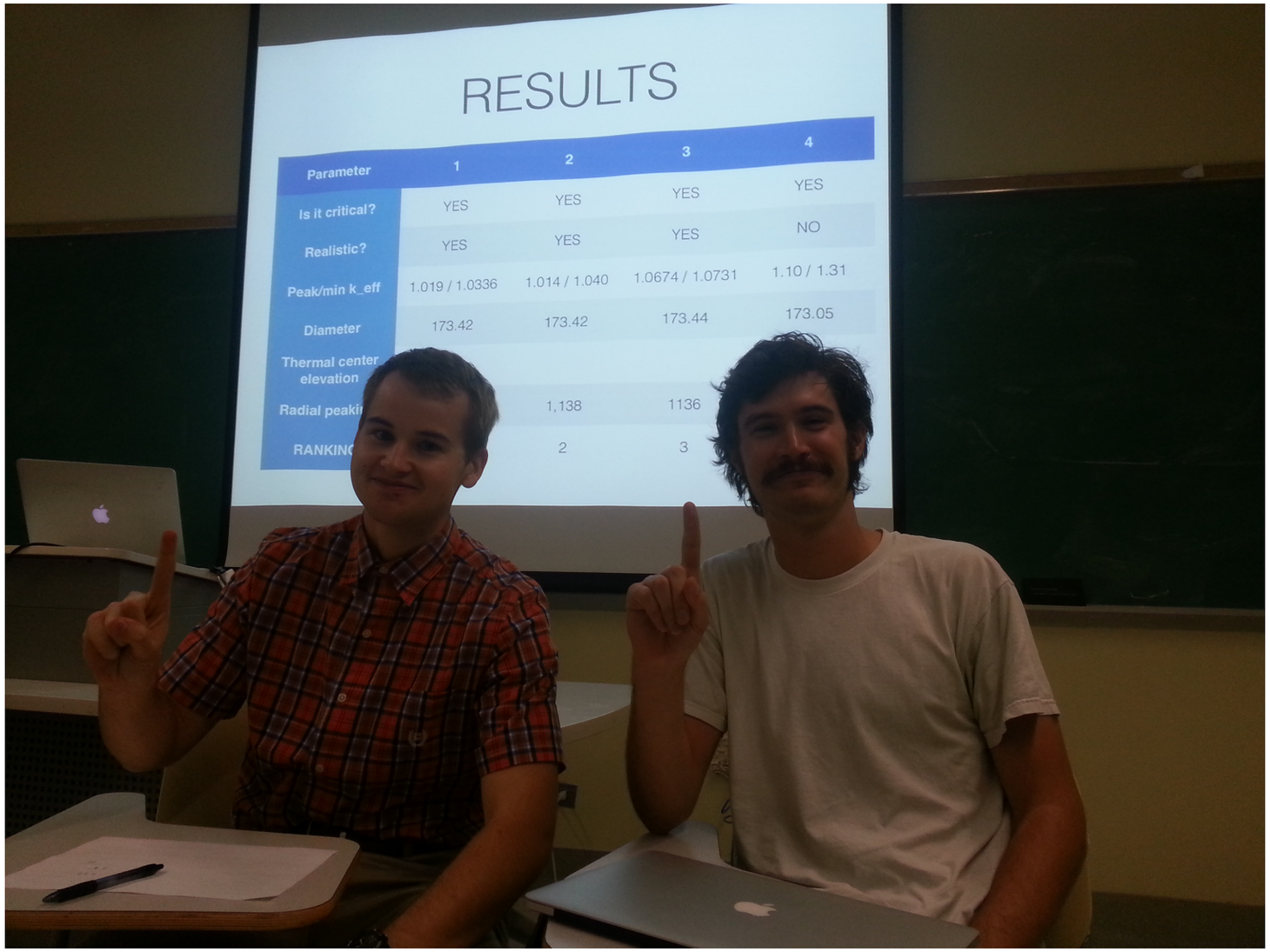
Application — For what are we designing this core?

- Long-life (30 year) 50 MWe “battery type” core
- Autonomous operation, “zero” reactivity swing
- No in-cycle fuel shuffling / reprocessing / conditioning
- Small and modular for mass production in factory
- Near-term “realistic” and licensable design

- Cheap, clean, deployable, proliferation resistant and safe energy production for the world

ADOPT boot camp
5th November 2013

Congrats Ryan and Kyle, winners of the worlds first (!) live core design competition!



Thank you!

(Serpent wish list addition)

Possibility to split up xs-loading, transport and depletion with seperate calls:
sss2 inputfile -transport, sss2 inputfile -depletion

Keep xs-data in memory while allowing changes to the input and re-running transport and/or depletion (from something eq. to .burn-file)