Modelling Nuclear Reactors with Julia

Sterling Harper

December 2015

My objective

To see how well Julia fits my use cases

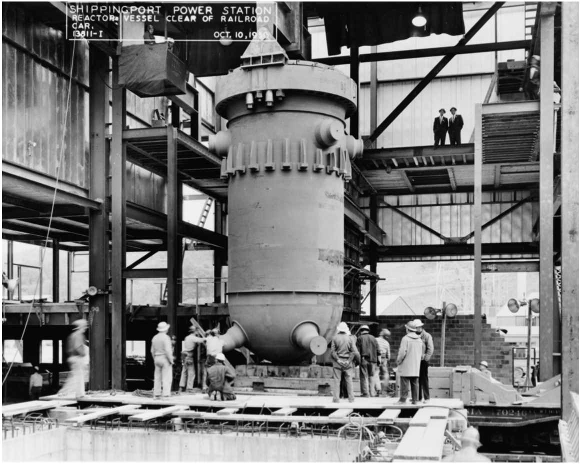
My objective

To see how well Julia fits my use cases

Benchmark problem: a nuclear reactor simulator (specifically method of charactersitics)

What a nuclear reactor looks like

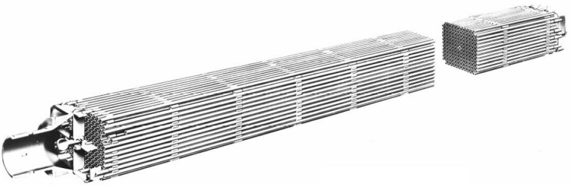
The heart of a typical nuclear powerplant is a pressure vessel where water is heated



Credit: US DOE

What a nuclear reactor looks like

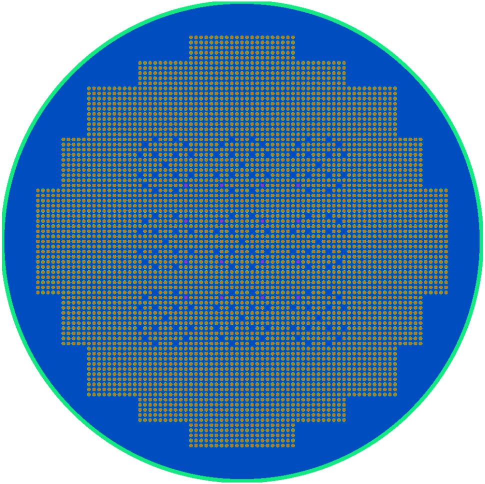
Inside that vessel are about 200 fuel assemblies—bundles of rods containing uranium



Credit: US DOE

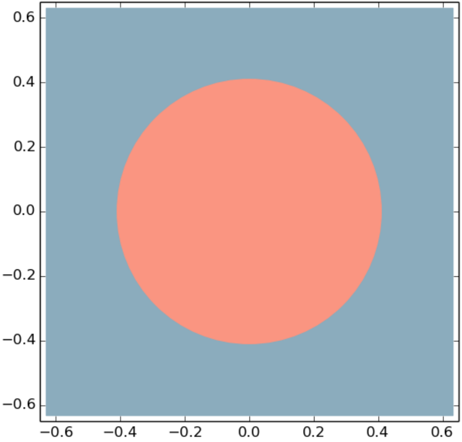
What a nuclear reactor looks like

Reactors have a lot of axial symmetry which usually allows us to model them in 2D



What my geometry looks like

For this project, I further simplified down the model to a single "pin cell" with reflective boundaries



The problem

Goal: neutron interaction rate

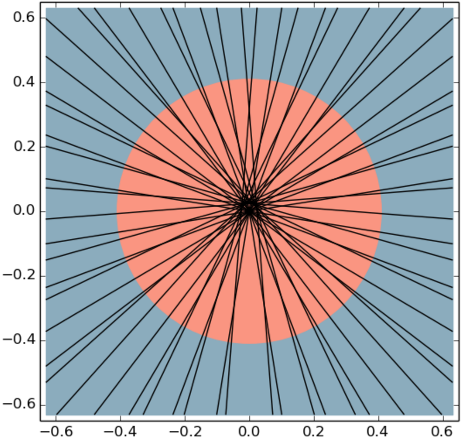
Complication: strong anisotropy

Diffusion can't solve this problem!

Solution: integrate over characterstic paths

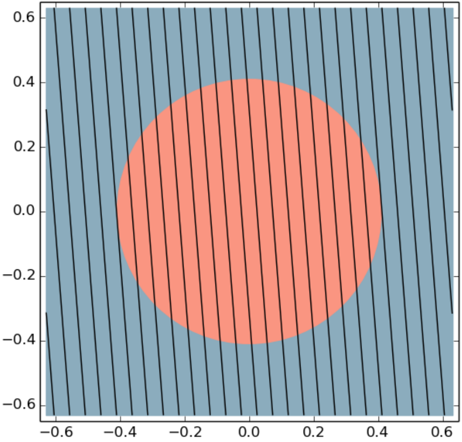
Neutron characteristics

Select a set of azimuthal angles (32 in this case)



Neutron characteristics

Repeat each azimuthal ray with a regular spacing (0.5 mm in this case)



The algorithm

Algorithm 2 Transport Sweep Algorithm

$\Phi_{i,g} \leftarrow 0 \quad \forall\, i, g \in \{I, G\}$ # Initialize FSR scalar fluxes to zero
while $\Phi_{i,g}\ \forall\, i$ not converged **do**
  for all $m \in M$ **do** # Loop over azimuthal angles
    for all $k \in K(m)$ **do** # Loop over tracks
      for all $s \in S(k)$ **do** # Loop over segments
        for all $g \in G$ **do** # Loop over energy groups
          for all $p \in P$ **do** # Loop over polar angles
            $i \leftarrow I(s)$ # Get FSR for this segment
            $\Delta\Psi_{k,i,g,p} \leftarrow \left(\Psi_{k,g,p} - \frac{Q_{i,g}}{\Sigma^T_{i,g}}\right)\left(1 - e^{-\tau_{k,i,g,p}}\right)$ # Compute angular flux change along segment
            $\Phi_{i,g} \leftarrow \Phi_{i,g} + \frac{4\pi}{A_i}\omega_m\omega_p\omega_k \sin\theta_p \Delta\Psi_{k,i,g,p}$ # Increment FSR scalar flux
            $\Psi_{k,g,p} \leftarrow \Psi_{k,g,p} - \Delta\Psi_{k,g,p}$ # Update track outgoing flux
          end for
        end for
      end for
    end for

Credit: Will Boyd

The codes: a three-way face-off

C++

Well developed parallel research code with many features

Python

My own homework code, slow

Julia

Similar to the Python code, but fast

	C++	Python	Julia
Baseline	2.8 s (1×)	117.0 s (41×)	14.4 s (5×)
Serial opt	-	-	-
2 Procs	-	-	-

Direct translation Python → Julia gives 8× speed up. Awesome!

	C++	Python	Julia
Baseline	2.8 s (1×)	117.0 s (41×)	14.4 s (5×)
Serial opt	-	-	9.4 s (3×)
2 Procs	-	-	-

Direct translation Python → Julia gives 8× speed up. Awesome!

A little serial optimization gets us close the C++ reference

	C++	Python	Julia
Baseline	2.8 s (1×)	117.0 s (41×)	14.4 s (5×)
Serial opt	-	-	9.4 s (3×)
2 Procs	-	-	46.1 s (16×)

Direct translation Python → Julia gives 8× speed up. Awesome!

A little serial optimization gets us close the C++ reference

Sadly my parallel efforts didn't turn out