PennState
College of Engineering

NucE 497: Reactor Fuel Performance

Lecture 11: Coolant temperature change, power generation, and melting

February 3, 2017
Michael R Tonks
Mechanical and Nuclear Engineering

Today we will discuss solving for the temperature profile in the fuel using 2D transient FEM

- Module 1: Fuel basics
- Module 2: Heat transport
  - Intro to heat transport and the heat equation
  - Analytical solution of the heat equation
  - Numerical solution of the heat equation
  - **1D solution of the heat equation using Matlab**
  - **2D solution of the heat equation using Matlab**
  - Coolant temperature change, power generation, and melting
- Module 3: Mechanical behavior
- Module 4: Materials issues in the fuel
- Module 5: Materials issues in the cladding
- Module 6: Accidents, used fuel, and fuel cycle

Here is some review from last time

- What kind of solution is done by the PDE toolbox in Matlab?
  a) 3D
  b) Transient 2D axisymmetric, smeared pellets
  c) Transient 1D axisymmetric
  d) Steady state 1D axisymmetric
- How do you define your BCs in the PDE toolbox?
  a) You define a single function the sets the values for all the BCs
  b) You don't need BCs
  c) You call applyBoundaryCondition at least once for each boundary
  d) You define them in a separate file

We make the solution fit the heat equation by correctly setting the parameters

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(c\frac{\partial u}{\partial z}\right) + au = f$$

$$\rho\, c_p\, r\frac{\partial T}{\partial t} = \frac{\partial}{\partial r}\left(rk(T)\frac{\partial T}{\partial r}\right) + \frac{\partial}{\partial z}\left(rk(T)\frac{\partial T}{\partial z}\right) + r\,Q(r, z)$$

- $m = 0$
- $d = \rho\, c_p\, r$
- $c = r\, k(T)$
- $a = 0$
- $f = r\, Q(r, z, t)$

There are seven steps required to set up a solve using the PDE toolbox (or any other FEM code)

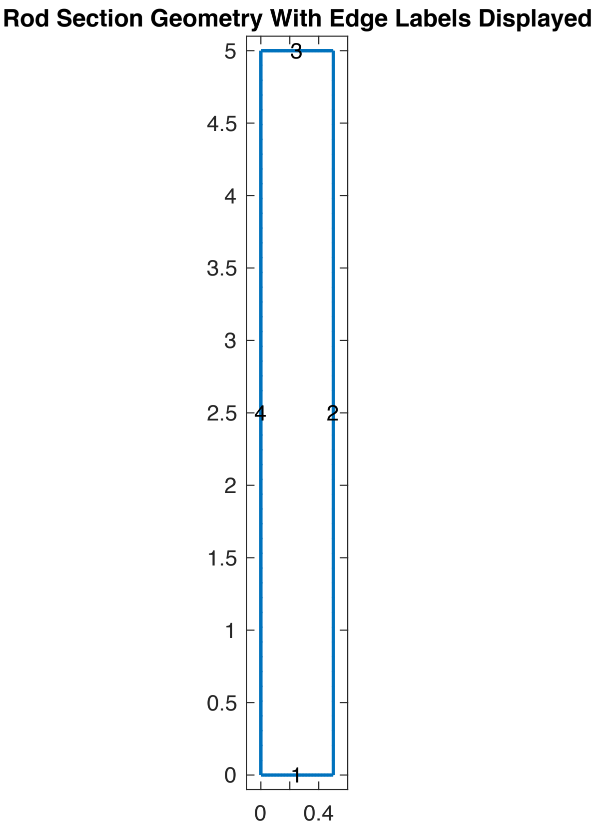
1. Define how many and what variables you are solving for
2. Define your geometry
3. Create a mesh
4. Define your PDE and material properties
5. Set up boundary conditions
6. Set up the initial condition and time steps (for a transient solve)
7. Details about executing the solve

The first step is to define the problem you are trying to solve

- In the PDE toolbox in Matlab, you define an object that contains all the information about your solve.
- You create it with the createpde command that takes one input
  - Input is the number of PDEs you wish to solve
  - model = createpde(numberOfPDE);

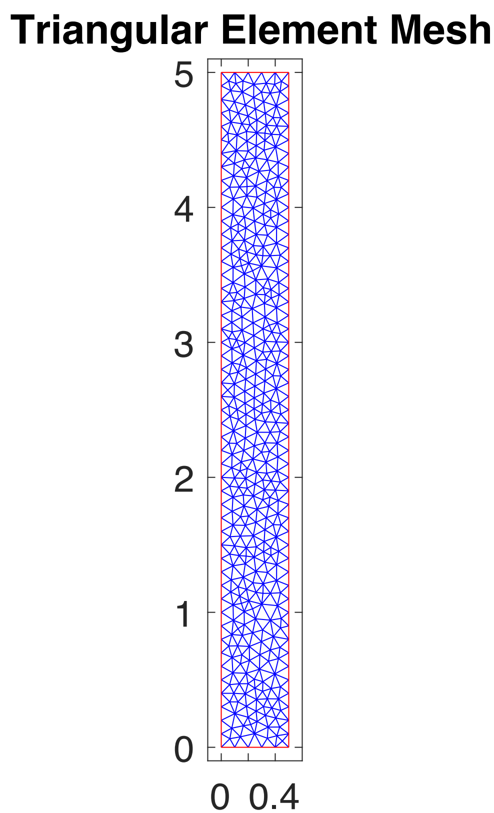
The first step is to create the geometry

- We will model geometry using symmetry around the center axis in the r direction and about the half plane in the z direction
- g = decsg([3 4 0 Rf Rf 0 0 0 length/2 length/2]');
    - First number tells it type of geometry, 3 = rectangle
    - Second is how many points define the geometry
    - The next four are the y coordinates of the points
    - The last four are the z coordinates
- Then, convert the geometry to the correct form and append it to the pde model
- geometryFromEdges(model,g);
    - We add the geometry to our model, called “model”
- We define the geometry in the x and y coordinates in Matlab



Now we need to create a mesh over the geometry

- The PDE toolbox uses triangle elements in 2D
- We control the mesh size by giving the target max element size Hmax
  - generateMesh(model,'Hmax',0.1);



Next, we define the coefficients

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(c\frac{\partial u}{\partial z}\right) + au = f$$

- specifyCoefficients(model,'m',0,'d',@dFunc,'c',@cFunc,'a',0,'f',@fFunc);
  - We are defining the coefficients for “model”
  - The coefficients are defined in pairs. The first is the coefficient name and the second is the corresponding function (or 0)
- For each function, you pass in region and state and pass out the coefficient
  - Region contains the values of x, y, and t: r = region.x
  - State contains the variable values: state.u

```
function c = cFunc(region, state)
global k
c = k*region.x;
end
```

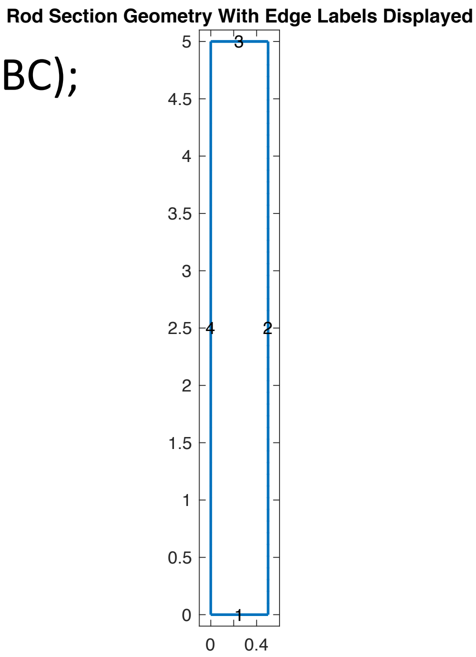
```
function d = dFunc(region,state)
global density cp;
d = density*cp*region.x;
end
```

```
function f = fFunc(region,state)
global Q;
f = Q*region.x;
end
```

Next, we define the boundary conditions for each boundary

- For each boundary we have to either specify the variable value or its derivative.
  - Dirichlet condition: Set the value of the variable
    bbottom = applyBoundaryCondition(model,'Edge',1,'u',Ts);
  - Neumann condition: Set the values of the expression $c\nabla u + q\,u = g$
    bmid = applyBoundaryCondition(model,'Edge',3,'g',0.0);
  - You can also use functions to define boundaries
    bbottom = applyBoundaryCondition(model,'Edge',1,'u',@TBC);
    u_value = TBC(region, state)

```
bbottom = applyBoundaryCondition(model,'Edge',1,'u',Ts);
bouter = applyBoundaryCondition(model,'Edge',2,'u',Ts);
bmid = applyBoundaryCondition(model,'Edge',3,'g',0.0);
bcenter = applyBoundaryCondition(model,'Edge',4,'g',0.0);
```



Now, because our problem is transient, we need to define our time behavior and initial condition

- We create a vector with our times we wish to simulate the solution at
  - tlist = linspace(0, tmax, M);
- Then we set the initial condition
  - setInitialConditions(model, Ts);

Once we have all the pieces set up, we can solve the system

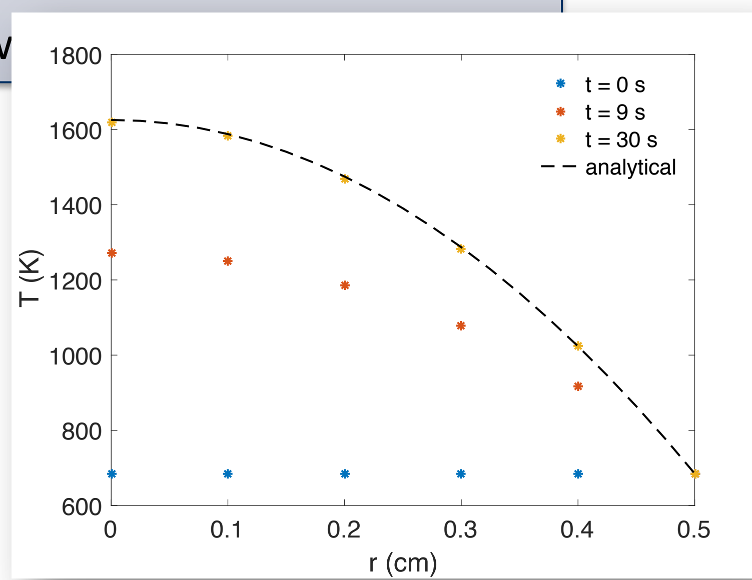
- For the solve, you pass in the model object and the time vector
  - result = solvepde(model,tlist);
  - result is an object containing the coordinate, the times, and the solution
- When the solve is complete, you can extract the solution
  - u = result.NodalSolution;
  - u is a list of the temperature at every node at all the solution times

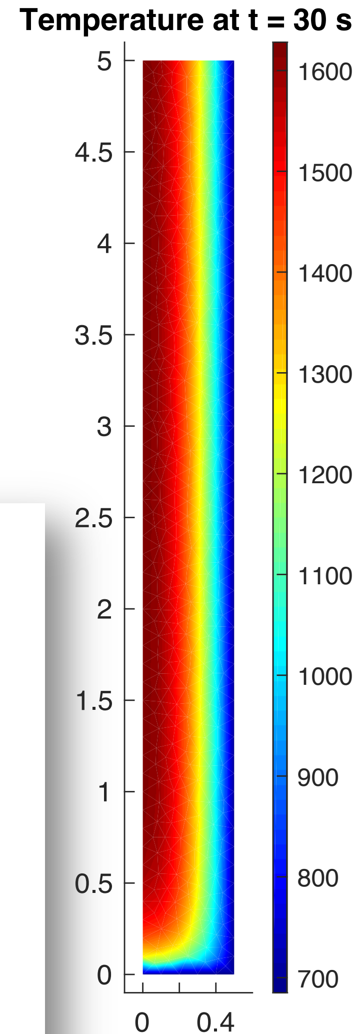
Once the solve is complete, you can plot the solution

- You can plot the solution with this command
  - pdeplot(model,'XYData',u(:,end),'Contour','on');
- You can make line plots like this

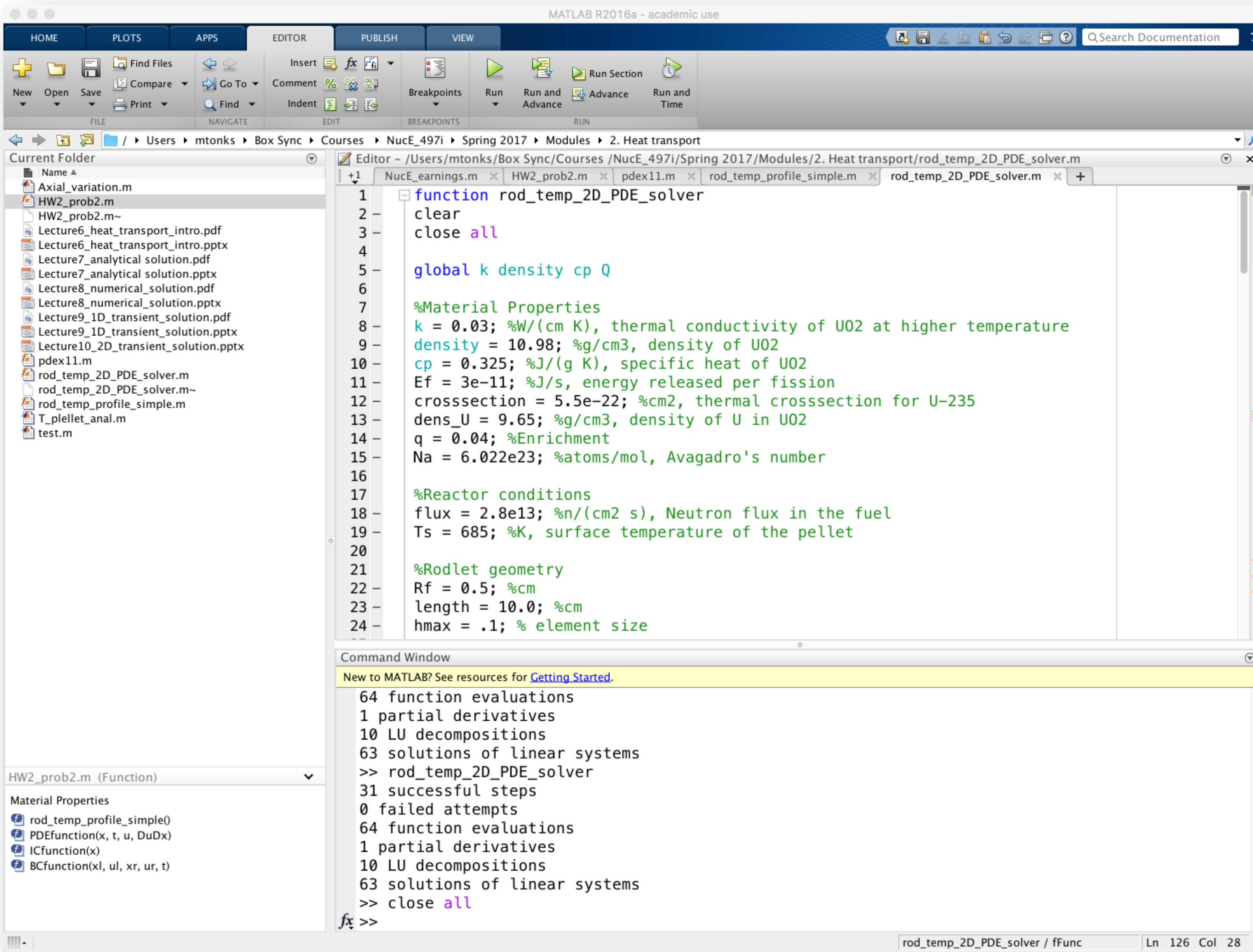
```
p = model.Mesh.Nodes;
top_nodes = find(p(2,:) == 5.0);
top_radius = p(1,top_nodes);

plot(top_radius, top_T,'*','linew
```

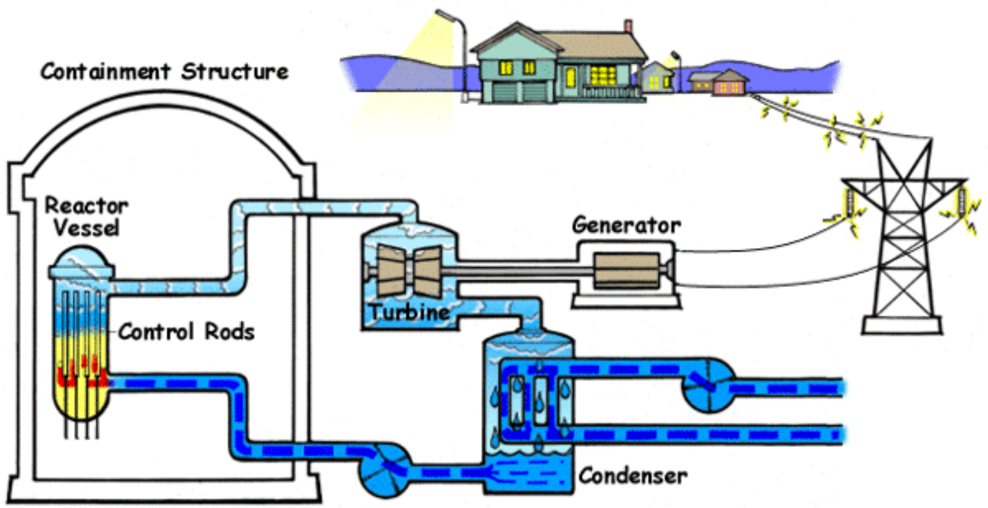


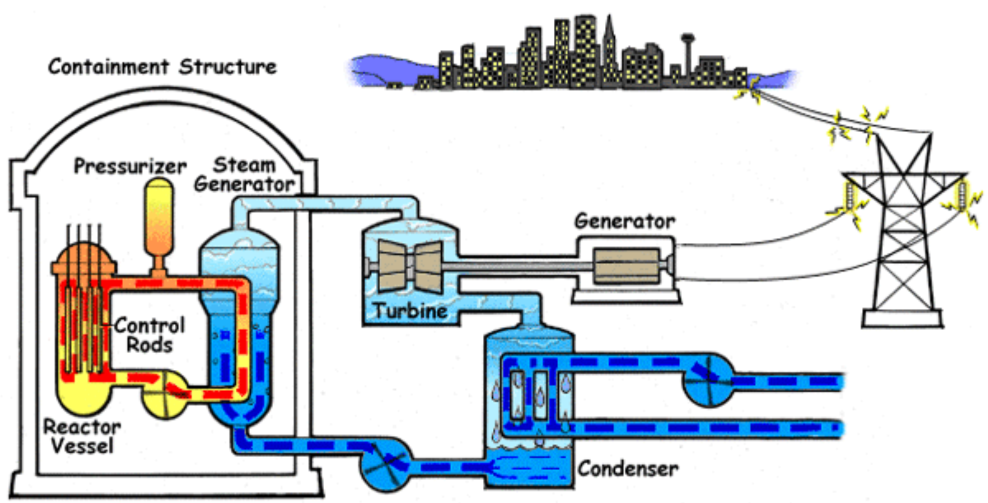


Now, we will look at the code



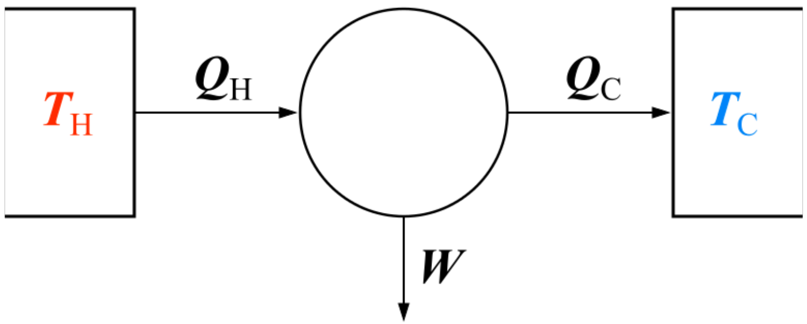
In all current commercial reactors, power is generated from steam



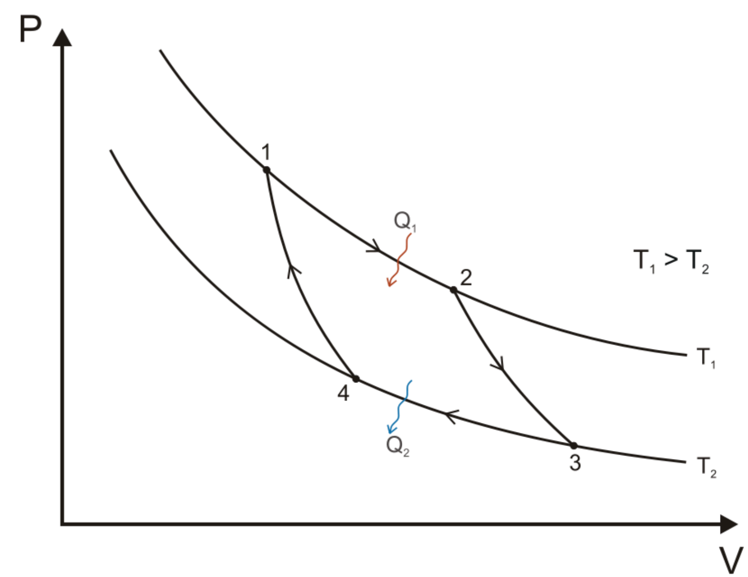


The efficiency of our power generation is a function of the inlet and outlet temperatures

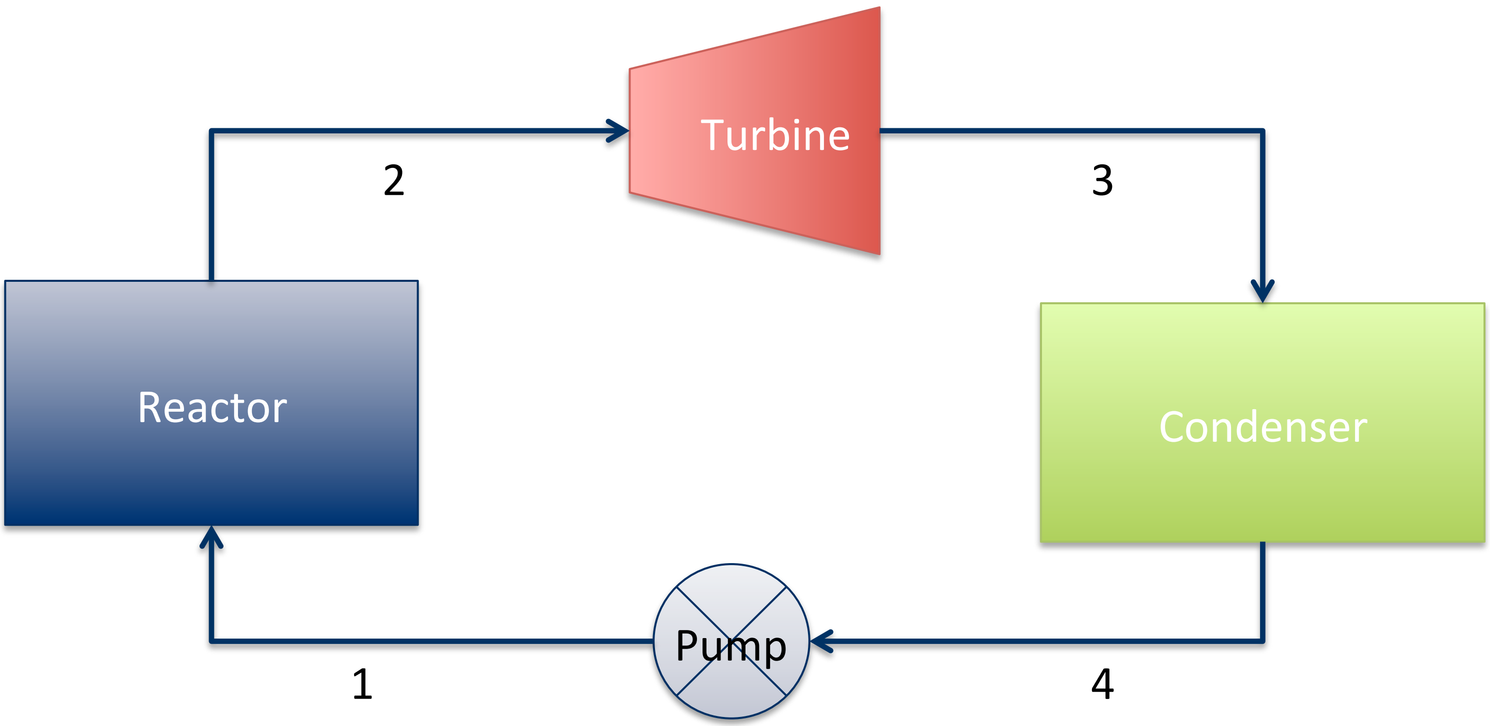
- The maximum possible efficiency of power generation comes from the Carnot cycle



$$\eta_C = \frac{W}{Q_H} = \frac{T_H - T_C}{T_H}$$



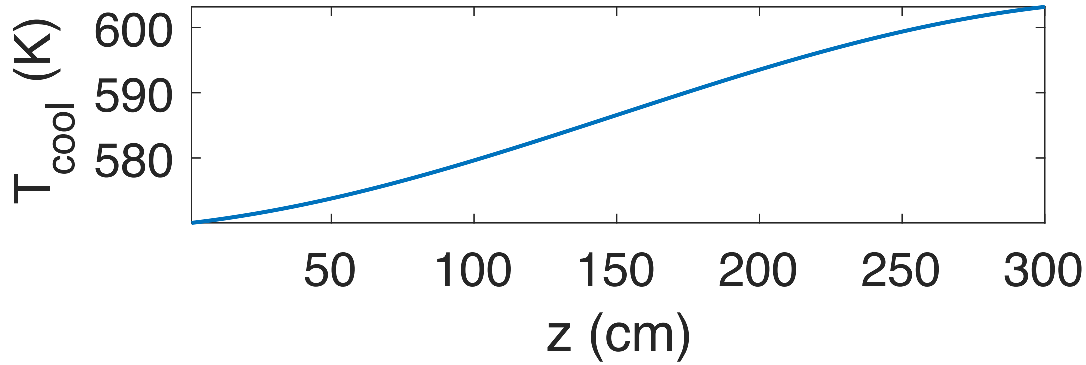
The reactor cycle is less efficient than the Carnot cycle, but still follows the same general idea



$$\eta_R = \frac{(h_2 - h_3) + (h_4 - h_1)}{h_2 - h_1}$$

In the flow past a fuel rod, the temperature gradually increases before leaving the rod

$$T_{cool} - T_{cool}^{in} = \frac{2\gamma}{\pi}\frac{Z_0 LHR^0}{\dot{m}C_{pw}}\left(\sin\left(\frac{\pi}{2\gamma}\right) + \sin\left(\frac{\pi}{2\gamma}\left(\frac{z}{Z_0} - 1\right)\right)\right)$$

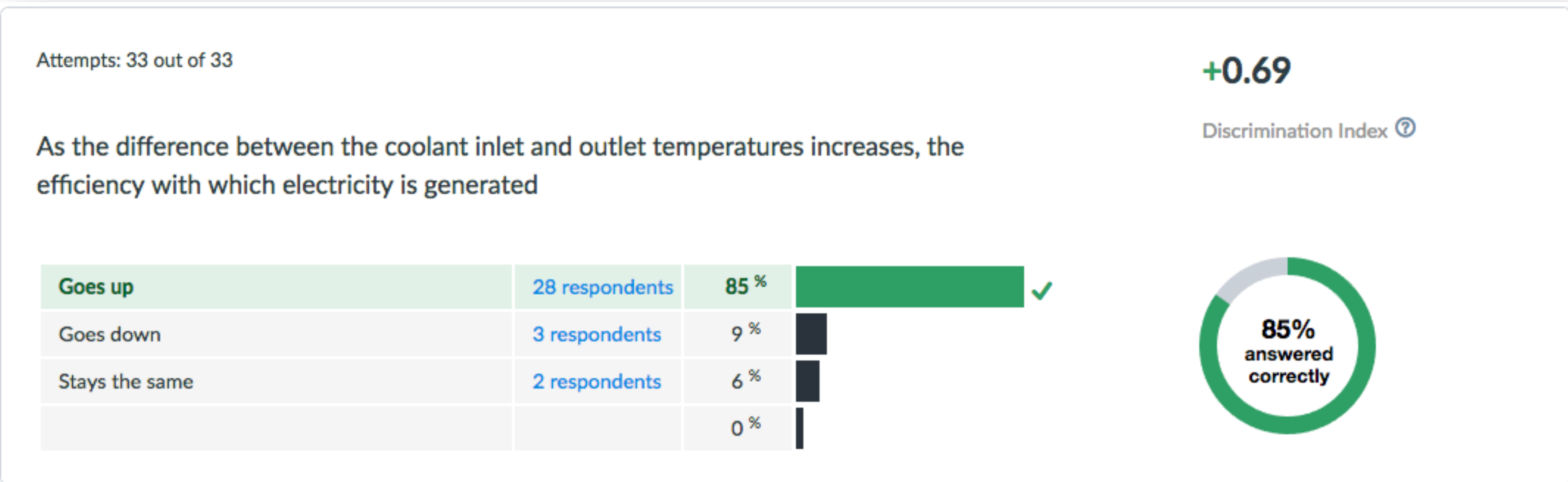


- T_{in} = 570 K, T_{out} = 603 K
- What is the maximum efficiency of this reactor?
  - η_C = (603 – 570)/603 = 0.055 = 5.5%

$$\eta_C = \frac{W}{Q_H} = \frac{T_H - T_C}{T_H}$$

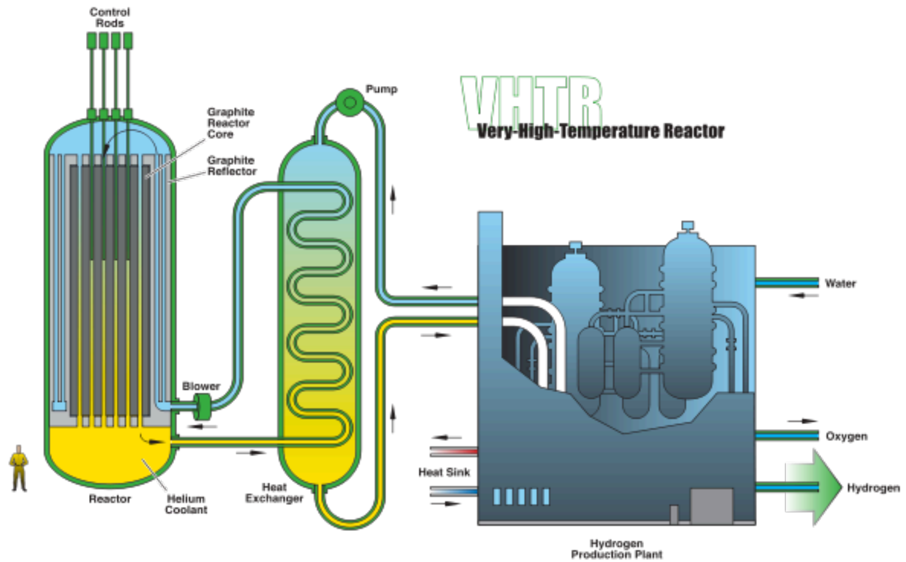
Quiz question: As the difference between the coolant inlet and outlet temperatures increases, the efficiency with which electricity is generated…

- Goes up
- Goes down
- Stays the same



To increase the efficiency of our reactor, we need to increase the difference in the inlet and outlet temps

- Is there a limit to the difference in a PWR?
- Is there a limit to the difference in BWR?
- Are there Gen IV reactor designs that have a larger difference?



Are there any side effects in the fuel that could result from maximizing this temperature difference?

- Fuel melting
- Increased fission gas release
- Higher plenum pressures
- Break away oxidation of the cladding
- More creep in the cladding and fuel
- Less safety margin in the case of an accident

The max differences depends on the safety margin, but also on the fuel material

Property	*Metal*	*UO_2*	*UC*	*UN*	*U_3Si_2*
A. Chemical					
Corrosion resistance in water	Very poor	Excellent	Very poor	Poor	Moderate
Compatibility with clad materials	Reacts with normal clad	Excellent	Variable	Variable	Variable
Thermal stability	Phase change at 665 and 770 °C	Good	Good in reducing atmosphere	Good, decomposes at 2600 °C	Good
B. Physical					
Uranium (metal) density (g/cm^3)	19.04	9.65	12.97	13.52	11.31
Melting point (°C)	**1132**	**2865**	**2850**	**2860**	**1665**
Thermal conductivity (W/cmK)	**0.38 at 430 °C**	**0.03 at 1000°C**	**0.25 at 100 – 700°C**	**0.2 at 750°C**	**0.23 at 773°C**

after Garzarolli in [Rudling, et al. 2007]

We can estimate the maximum temperature difference that will provide a given safety margin

- What is the max temperature difference between the inlet temperature (550 K) and the outlet rod coolant temperature for which the fuel centerline temp. in a UO_2 fuel rod stays below 80% of melting? We will use analytical solution with the same conditions from the example in Lecture 7.
    - Melting temperature of UO_2 is 2865, so T_{max} = (2865+273)*0.8 = 2510.4 K
    - With only He in the cladding, our T differences we calculated were
        - $T_{CO} - T_{cool} = 25.5$ K, $T_{CI} - T_{CO} = 22.5$ K, $T_s - T_{CI} = 73.5$ K, $T_0 - T_s = 530.5$ K
- If we put all these differences together, we get
    - $550 + \Delta T_{max} + 25.5 + 22.5 + 73.5 + 530.5 = T_{max} = 2510.4$
    - $\Delta T_{max} = 2510.4 - (550 + 25.5 + 22.5 + 73.5 + 530.5)$
    - $\Delta T_{max} = 1308$ K

Now you try it when 30% of the gap is filled with Xe

- T_{in} = 550 K, T_{max} = 2510 K
- With 30% Xe in the cladding, our T differences we calculated were
  - $T_{CO} - T_{cool}$ = 25.5 K, $T_{CI} - T_{CO}$ = 22.5 K, $T_s - T_{CI}$ = 188.0 K, $T_0 - T_s$ = 530.5 K

- If we put all these differences together, we get
- $550 + \Delta T_{max} + 25.5 + 22.5 + 188.0 + 530.5 = T_{max} = 2510$
- $\Delta T_{max} = 2510 - (550 + 25.5 + 22.5 + 188.0 + 530.5)$
- ΔT_{max} = 1193 K

Summary

- The Matlab function PDE toolbox allows you to solve PDEs in 2D or 3D, transient or steady-state.
- The power generation efficiency of a reactor depends on the difference between the coolant inlet and outlet temperatures
- Raiding the outlet temperature increases the efficiency but can negatively impact the fuel