PennState
College of Engineering

NucE 497: Reactor Fuel Performance

Lecture 9: 1D transient solution of heat equation using Matlab

January 27, 2017

Michael R Tonks
Mechanical and Nuclear Engineering

Some material taken from Matlab documentation

Today we will discuss solving for the temperature profile in the fuel using 1D 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

Let’s review some from last time:

- What is one strength and one weakness of the finite element method?
  a. Can model any geometry but very complicated
  b. Simple to implement, but limited on boundary conditions
  c. Can model any geometry but can’t hand heterogeneous properties well
  d. Can model any boundary condition, but it required periodic boundary conditions
- What temperature solution assumes axisymmetry and that the pellets act as a single body?
  a. 1D transient
  b. 1.5D transient
  c. 2D transient, smeared pellets
  d. 2D transient, discrete pellets

Quiz question: Which is not actually true about LWR nuclear fuel, but is assumed to make the analytical model possible?

a) The fuel system has a cylindrical geometry
b) The thermal conductivity of UO2 is constant with temperature
c) Heat is generated throughout the fuel pellet
d) The fuel is solid throughout the fuel pellet during normal operation

Attempts: 33 out of 33

+0.42

Discrimination Index

Which is not actually true about LWR nuclear fuel, but is assumed to make the analytical model possible?

The fuel system has cylindrical symmetry		0 %
The thermal conductivity of UO2 is constant with temperature	29 respondents	88 % ✓
Heat is generated throughout the fuel pellet		0 %
The fuel is solid throughout the fuel pellet during normal operation	4 respondents	12 %

Quiz question: Numerical solutions for the temperature profile throughout the fuel have both strengths and weaknesses compared to analytical solutions. Which is a correct set of a strength and a weakness?

a) Strength: Accounts for human error; Weakness: More boring

Attempts: 33 out of 33

+0.24

Discrimination Index

Numerical solutions for the temperature profile throughout the fuel have both strengths and weaknesses compared to analytical solutions. Which is a correct set of a strength and a weakness?

Strength: Accounts for the human errorWeakness: More boring		0 %
Strength: Accounts for temperature dependence of the thermal conductivity Weakness: Does not correctly represent geometry	2 respondents	6 %
Strength: Accounts for cylindrical symmetryWeakness: Higher computational complexity	1 respondents	3 %
Strength: Accounts for temperature dependence of the thermal conductivityWeakness: Higher computational complexity	30 respondents	91 % ✓

Quiz question: Early fuel performance codes used one dimensional predictions in space of the temperature profile in a fuel rod, to simplify the calculation. The dimension used in these simulations was the one that experienced the most temperature change. This dimension was the

a) Radial position

b) Axial position

c) Location with the fuel assembly

Attempts: 33 out of 33

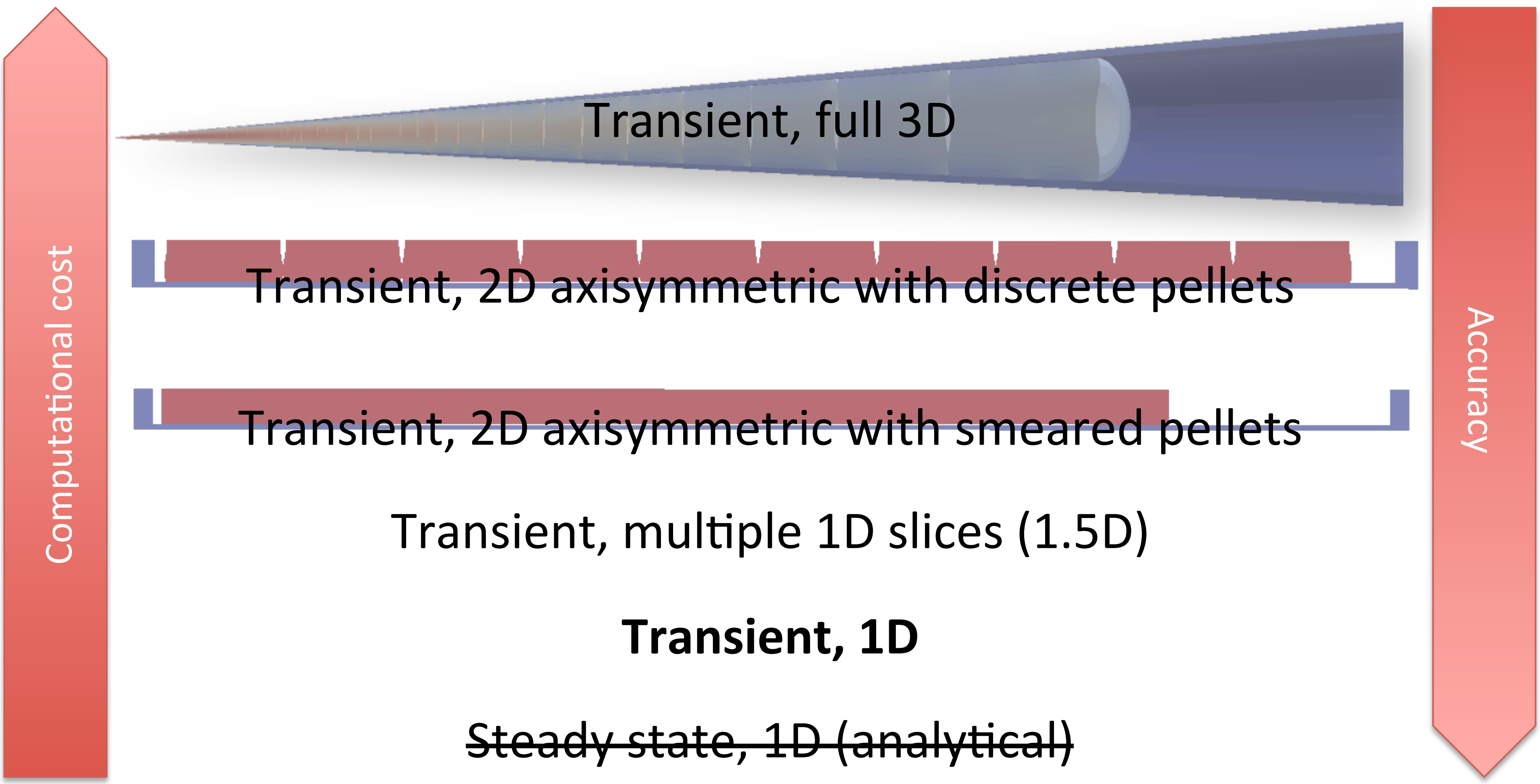
+0.42

Discrimination Index

Early fuel performance codes used one dimensional predictions in space of the temperature profile in a fuel rod, to simplify the calculation. The dimension used in these simulations was the one that experienced the most temperature change. This dimension was the

Radial position ✓	29 respondents	88 %
Axial position	3 respondents	9 %
Location within the fuel assembly		0 %
Location within the core	1 respondents	3 %

We talked about numerical methods, now we will use one for 1D transient solutions



Remember our two assumptions

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + Q$$

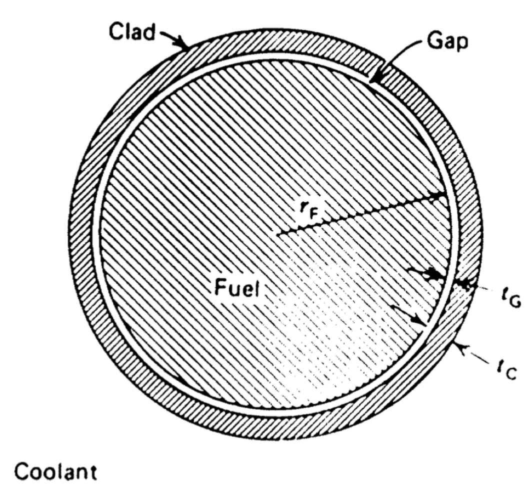
- *Assumption 1:* The behavior is axisymmetric

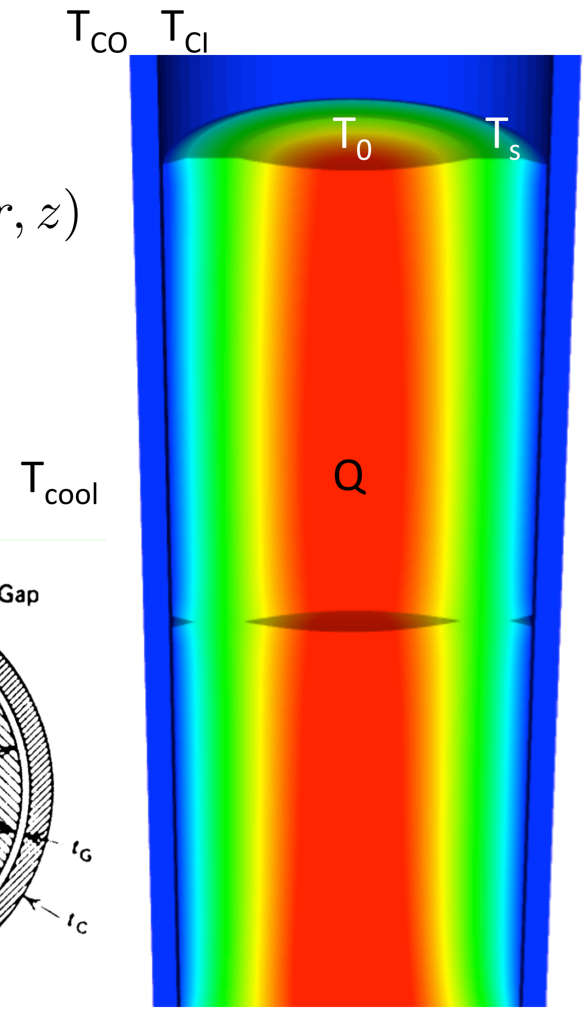
$$\rho c_p \frac{\partial T}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r k(T) \frac{\partial T}{\partial r}\right) + \frac{\partial}{\partial z}\left(k(T)\frac{\partial T}{\partial z}\right) + Q(r, z)$$

- *Assumption 2: T is constant in z*

$$\rho\, c_p \frac{\partial T}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r k(T) \frac{\partial T}{\partial r}\right) + Q(r)$$

- **We will solve this using FEM and implicit time integration in Matlab**





In Matlab, you save files of commands, and when you run the file, it runs the commands

Matlab easily deals with vectors and matrices

- You can easily create vectors or matrices
  - A = [1 2 3 4];
  - B = [1 3 4 5; 1 3 5 3];
  - C = [8 26 24 32];
- Operations with a . first, like .* or ./ operate on corresponding values in the vectors or matrices
  - C./A = [2 2 2 2]

Your saved file can define multiple functions

```
Function RunStuff
    %This function just runs another function
    vec1 = [1, 4];
    vec2 = [4, 7];
    [a, b] = test(vec1, vec2);
end

Function [a, b] = test(c, d)
    a = c + d;
    b = c.*d;
end
```

If you need to use the same quantity in multiple functions, you can make it global

- Normally, a variable only exists in the function it was created in.

```
Function RunStuff
    vec1 = [1, 4];
    vec2 = [4, 7];
    [a, b] = test(vec1, vec2);
    mult = 5;
end
```

THIS DOESN'T WORK!

```
Function [a, b] = test(c, d)
    a = mult*(c + d);
    b = c.*d;
end
```

- However, you can declare a variable global in each function and now it can be accessed

```
Function RunStuff
    global mult;
    vec1 = [1, 4];
    vec2 = [4, 7];
    [a, b] = test(vec1, vec2);
    mult = 5;
end
```

THIS WORKS

```
Function [a, b] = test(c, d)
    global mult;
    a = mult*(c + d);
    b = c.*d;
end
```

Matlab has lots of documentation

- If you want to do something but you don’t know the command, google it
- If you know the command, you can use the help command
  - In matlab > help plot

Matlab has great capability for plotting.

- For bar plots, use the “bar” command (type help bar to learn about it)
- For line plots, use the “plot” command.
    - You pass the command two vectors of the same length, and it uses the first one for the x axis of points and the second one for the y axis
    - For example plot([1:5], sqrt(1:5])
- If you pass in a matrix, you can plot a surface using the “surf” command

Matlab’s 1D PDE solver is called pdepe

- pdepe solve a 1D PDE in time and in one spatial direction using a transient implicit solution in time and the finite element method in space

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+s\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

  - u is the variable we are solving for, it is a function of x and t
  - x is the space variable, a ≤ x ≤ b
  - t is the time variable, $t_0 \leq t \leq t_f$
  - m = 0, 1, or 2
- How can we make this equation match our heat equation?

$$\rho\, c_p\frac{\partial T}{\partial t}=\frac{1}{r}\frac{\partial}{\partial r}\left(rk(T)\frac{\partial T}{\partial r}\right)+Q(r)$$

- What is u? What is x? What is t? What is m? What is c? What is f? What is s?

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

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+s\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

$$\rho\, c_p\frac{\partial T}{\partial t}=\frac{1}{r}\frac{\partial}{\partial r}\left(rk(T)\frac{\partial T}{\partial r}\right)+Q(r)$$

- x = r, t = t, u = T, dU/dx = dT/dr
- m = 1
- c(x, t, u, du/dx) = ρ c_p
- f(x, t, u, du/dx) = k(T) dT/dr
- s(x, t, u, du/dx) = Q(r)

In order to use PDEPE, we have to provide six inputs

sol = pdepe(m, @PDEfunction, @ICfunction, @BCfunction, r, t);

- The input m sets the coordinate system (m = 0 for Cartesian, m = 1 for cylindrical, and m = 2 for spherical)
- We need to create three functions
  - [c, f, s] = PDEfunction(x, t, u, dudx) – This function defines the three constants in the PDE
  - u = ICfunction(x) – This function defines the initial condition of u
  - [pl, ql, pr, qr] = BCfunction(xl, ul, xr, ur, t) – This function defines boundary conditions on the left and right side of the domain
- We create the mesh using the command r = linspace(a, b, N)
  - This creates r as a vector that goes from a to b with N points
- We create the time domain t = linspace(t_0, t_s, M)
  - This creates t as a vector that goes from t_0 to t_s with M time steps

Here is more detail about the function that creates the three terms of the PDE

[c, f, s] = PDEfunction(x, t, u, dudx)

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right) + s\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

- Four inputs are passed in by the solver
  - x – the current location in space (r in our case)
  - t – the current time
  - u – the current value of the variable (T in our case)
  - dudx – the current value of the derivative of the variable (dT/dr in our case)
- You then define c, f, and s as functions of these variables

```
function [c,f,s] = PDEfunction(x,t,u,DuDx)
    global density cp Q k

    c = density*cp;
    f = k*DuDx;
    s = Q;
end
```

Here is more detail about the function that creates the initial condition

u = ICfunction(x)

- One input is passed in by the solver
  - x – the current location in space (r in our case)
- You then define the value of the variable throughout the mesh at t = 0

```
function u0 = ICfunction(x)
    global Ts;
    %Initial temperature
    T0 = Ts; %K

    %Assign values
    u0 = T0*ones(size(x));
end
```

Here is more detail about the function that creates the boundary condition

[pl, ql , pr, qr] = BCfunction(xl, ul, xr, ur, t)

- Five inputs are passed in by the solver
  - xl – the coordinate location on the left side (r = 0 in our case)
  - ul – the value of u on the left side (T(r=0) in our case)
  - xr – the coordinate location on the right side (r = Rf in our case)
  - ur – the value of u on the right side (T(r=Rf) in our case)
- You define the boundary conditions on the left and right side in this form:
  $p(x,t,u) + q(x,t) f\left(x,t,u,\frac{\partial u}{\partial x}\right) = 0$
  - pl and ql set p and q on the left. These are ignored for cylindrical and spherical coordinates
  - pr and qr set p and q on the right.
  - What are p and q for $T_r = T_s$?
    - pr = ur - T_s

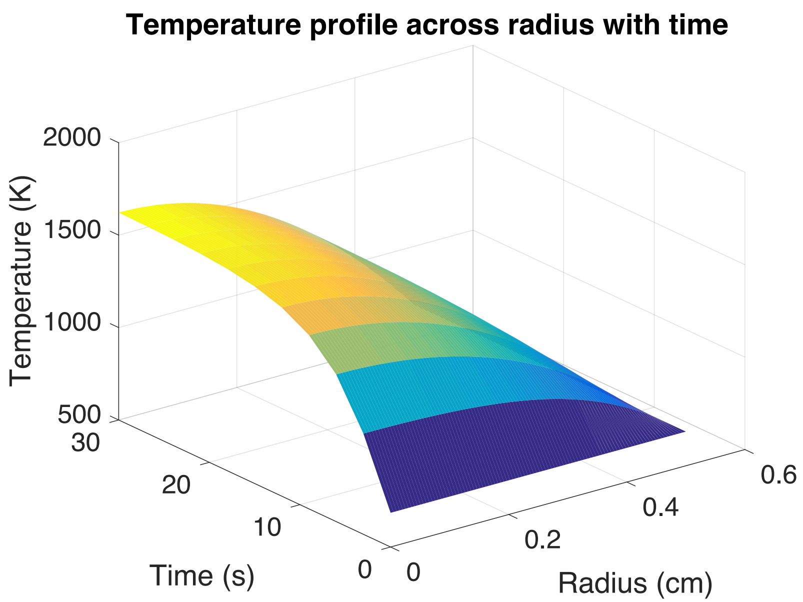
```
function = BCfunction(xl,ul,xr,ur,t)
    global Ts;

    pl = 0; %This gets ignored
    ql = 0; %This gets ignored
    pr = ur - Ts;
    qr = 0;
end
```

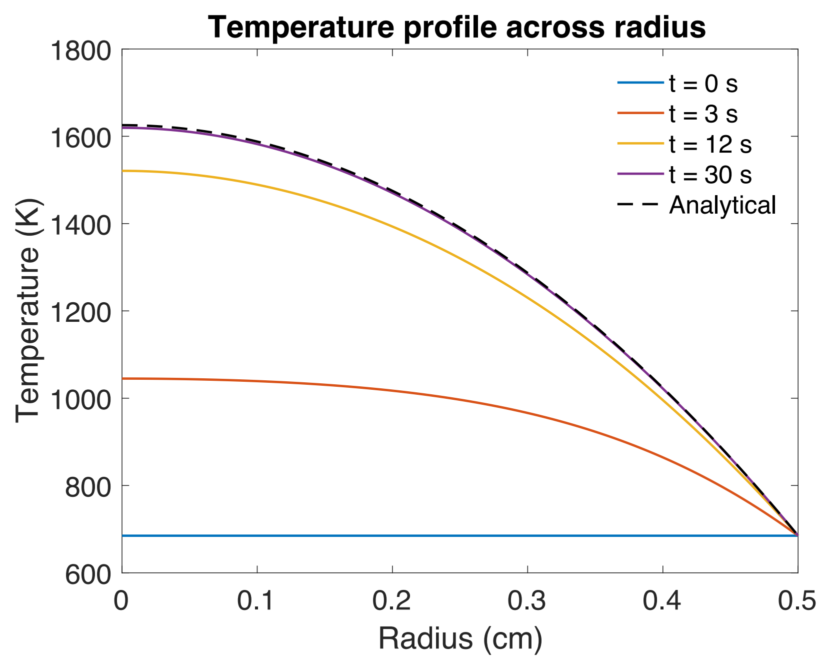
The solution comes out as a 3D array of data

T = pdepe(1,@PDEfunction,@ICfunction,@BCfunction,r,t);

- The 1st dimension is the spatial coordinate, the 2nd is the time coordinate, and the third is for additional variables you may be solving for.
- So, if we solve just for T on a mesh with N nodes and M times steps, T is a N × M matrix
- You could plot it in various ways

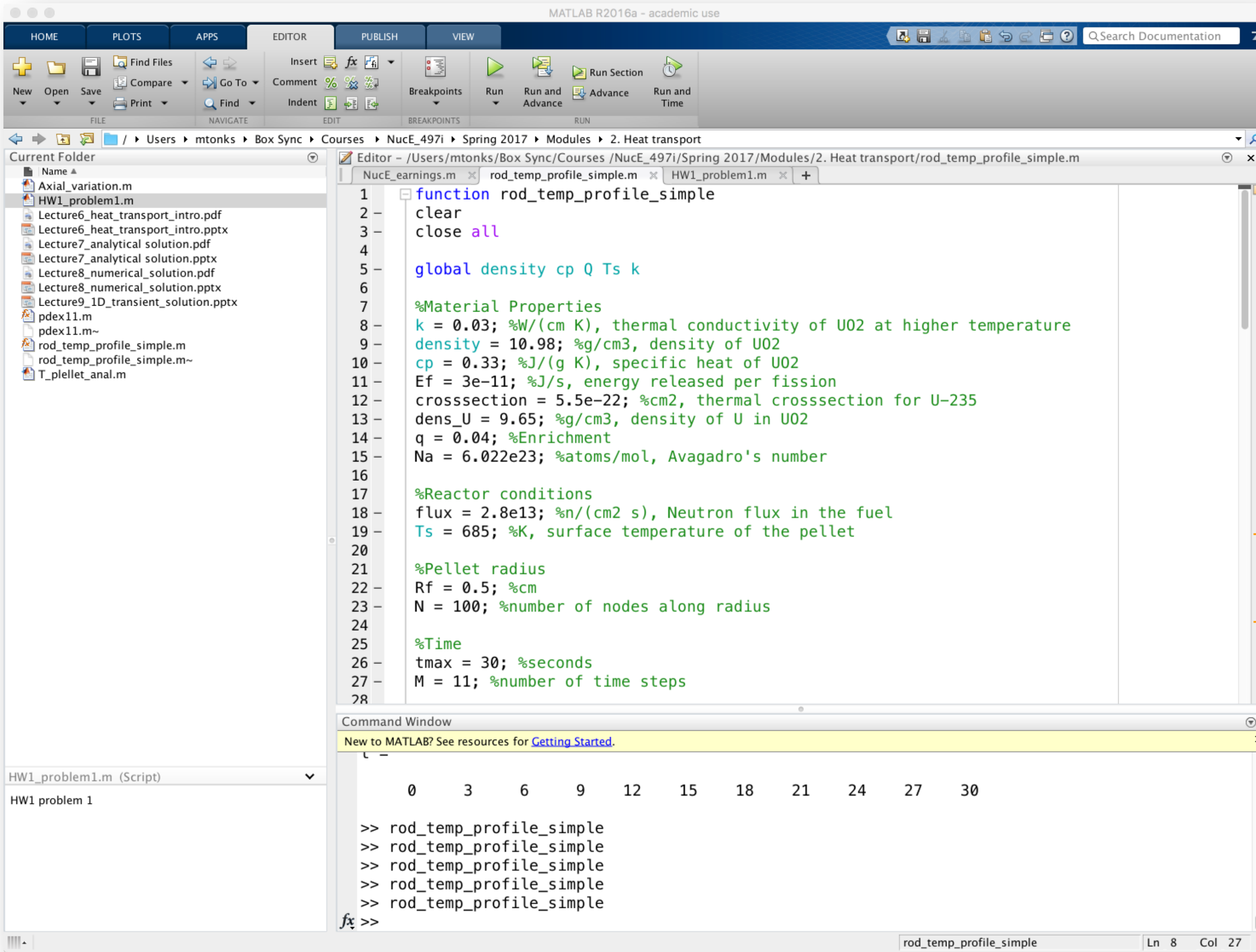


surf(r, t, T, 'edgecolor','none')



plot(r, T([1,2,5,10],:),'linewidth',1.5)

Now, let's look at the code



Summary

- The Matlab function PDEPE solves partial differential equations in 1 spatial dimension and in time using implicit FEM
- You define the PDE your are solving by setting parameters using a function
- Other functions define the initial condition and the boundary conditions