

Mathematical Methods (10/24.539)
Solving BVPs with COMSOL Multiphysics

Problem Description

COMSOL Multiphysics was designed specifically to solve problems written in the form of PDEs involving 1, 2, or 3 spatial coordinates and time. For steady state problems and 1-D geometry, the general problem description reduces to a simple BVP. Thus, although it was designed to handle much more complex situations, COMSOL Multiphysics can also be used to study the BVPs that we have been solving with the Shooting Method and the Finite Difference Method.

In fact, since we are now quite familiar with solving 1-D time-invariant BVPs using other methods, this might be a good place to start learning how to use COMSOL Multiphysics -- since we have alternate techniques that can be used for comparison purposes.

As a starting point, let's consider the linear and nonlinear nuclear fuel pin problems that we have already solved using other methods, and compare the COMSOL Multiphysics procedure and results to the previous solutions obtained using an analytical method, the Shooting Method, and the Finite Difference Method. Then, once we have established confidence in doing this, we can also extend the base problem slightly to include some additional factors that affect the real situation of interest in practical applications. Thus, we will do a series of cases to illustrate how COMSOL Multiphysics can be used to solve a variety of general BVPs of interest.

Recall that the nuclear fuel pin problem can be described mathematically, as follows:

$$\text{ODE:} \quad \frac{d}{dr} \left(k_f r \frac{dT}{dr} \right) + q''' r = 0 \quad (1)$$

$$\text{BCs:} \quad \left. \frac{dT}{dr} \right|_{r=0} = 0 \quad (\text{symmetry at } r = 0) \quad (2)$$

$$T(r) \Big|_{r=R_f} = T_s \quad (\text{fixed temperature at surface}) \quad (3)$$

where we note that the first BC can also be viewed as “no heat flow at the fuel pin centerline”, or

$$q \Big|_{r=0} = -k_f r \left. \frac{dT}{dr} \right|_{r=0} = 0 \quad (4)$$

A typical set of parameters used to get numerical results for this problem was given as:

$$q''' = 1.35e+7 \text{ BTU/hr-ft}^3 \quad (\text{volumetric heat source due to nuclear reactions})$$

$$R_f = 0.0175 \text{ ft} \quad (\text{fuel pin outer radius})$$

$$T_s = 650 \text{ F} \quad (\text{outer surface temperature of fuel pin})$$

$$\text{Case 1: } k_f = 1.1 \text{ BTU/hr-ft-F} \quad (\text{constant thermal conductivity of fuel})$$

$$\text{Case 2: } k_f = a_1 T^{a_2} \text{ BTU/hr-ft-F} \quad (\text{temperature dependent thermal conductivity})$$

with $a_1 = 94.4$ and $a_2 = -0.556$ (obtained from curve fit to experimental data for UO₂ fuel).

Case 1: The Linear Fuel Pin Problem

To solve this problem (and most BVPs) within COMSOL Multiphysics, we must perform the following steps:

1. define the geometry of interest (including discretization)
2. set the equation constants
3. specify the BCs
4. set solution parameters and solve the problem
5. perform post processing and analysis of the results

As our first introduction to COMSOL Multiphysics, we will be very deliberate and identify each explicit step as follows:

Start COMSOL Multiphysics

Use the “COMSOL x.xx with Matlab” icon on the desktop (or under the Programs list) to start a new session with COMSOL Multiphysics, where x.xx represents the current version number. If Matlab is not available, COMSOL Multiphysics 3.0 and higher can run independent from Matlab. In this case, just click on the “COMSOL Multiphysics x.xx” icon. Under this scenario, the option to export the “FEM structure”, with subsequent post-processing within Matlab, and the capability to plot to a Matlab file with the COMSOL Multiphysics GUI, are not available.

Set Application Mode

In the **Model Navigator** under the **New** tab always select the appropriate space dimension first, since this sets many of the options that are available for the chosen dimension. For the current case, our problem is one-dimensional, so select the **1D** case.

Under the **New** tab, expand the **Application Modes** for the **COMSOL Multiphysics, PDE Modes, PDE, Coefficient Form**, and select the **stationary analysis** option. Notice that this gives only one dependent variable, $u(x)$, within the following very general PDE:

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c \nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + a u = f \quad (5)$$

where e_a and d_a will be set to zero for stationary (time-independent) problems.

Draw Geometry

For a new model, COMSOL Multiphysics automatically puts you in **Draw Mode**. Since we selected the 1-D geometry case, only a few options are available: creating a point or a line. From the **Draw** menu, select **Draw/Specify Objects/Line** and specify the coordinates of the 1-D geometry of interest. Enter the start and end dimensions of the desired geometry: **0 0.0175**. This creates an object denoted as **l1**. This single subdomain represents the 1-D geometry for this problem. Hit the **Zoom Extents** icon to see the full geometry in the main COMSOL Multiphysics window.

Set Model Parameters

Go to **Subdomain Mode** and double click the object of interest (I1 in this case). A GUI window appears that allows you to enter the desired model parameters for the specific PDE of interest. Notice that, since we selected the stationary form, the time derivatives in the general equation are missing. Your job now is to match the ODE of interest to the general form that is available in COMSOL Multiphysics. In the current case, the following correspondence is needed:

$$\begin{aligned} \text{independent variable: } r &\rightarrow x & \text{dependent variable: } T &\rightarrow u \\ \text{coefficients: } \alpha = \beta = \gamma = a &= 0 \\ f = -q'' x & \quad \text{or} \quad f = -1.35e7 * x \\ c = -k_f x & \quad \text{or} \quad c = -1.1 * x \end{aligned}$$

Thus, we simply enter these values into the **Value/Expression** field under the **Coefficients** tab on the GUI. The options available under the other tabs are either not needed or do not need to be changed for this problem.

Set the BCs

Go to **Boundary Mode** and double click one of the boundary points for the problem. The GUI that appears allows you to select the appropriate BC for each boundary in the system. For the current problem, boundary point #1 on the left side has a Neumann BC. COMSOL Multiphysics models this as a generalized Neumann BC (or mixed condition) that involves the function and its gradient evaluated at the given boundary point. Thus, we can handle this by simply setting $q = 0$ and $g = 0$ in the general formulation:

$$\hat{n} \cdot (c \nabla u - \alpha u + \gamma) + qu = g \quad (6)$$

Noting that $\alpha = \gamma = 0$ and $c = -k_f x$ have already been set in the previous step, we see that eqn. (6), with $q = g = 0$, does indeed give us the BC from eqn. (4). Thus, for BC #1, simply set $\mathbf{q} = \mathbf{g} = \mathbf{0}$ in the GUI window.

The right BC for the problem is a Dirichlet condition. The representation for this situation in COMSOL Multiphysics is very general, and includes a Lagrange multiplier, μ , which is a free variable in most problems. For the usual case, where one simply wants to set the value of the function on the boundary, only the h and r variables in the general formulation are needed:

$$\hat{n} \cdot (c \nabla u - \alpha u + \gamma) + qu = g - h^T \mu \quad \text{and} \quad hu = r \quad (7)$$

Thus, for the case of interest, we set $\mathbf{q} = \mathbf{g} = \mathbf{0}$, and the flexibility associated with the Lagrange multiplier essentially eliminates the Neumann condition in the first part of eqn. (7) from consideration. In the second part, setting $\mathbf{h} = \mathbf{1}$ and $\mathbf{r} = \mathbf{650}$, give the desired BC at the right endpoint [as specified in eqn. (3)].

Select Mesh Grid

Once the geometry, equation coefficients, and BCs have been specified, we are now ready to discretize the independent variables -- that is, to create a finite element mesh for the geometry of interest. COMSOL Multiphysics also makes this step quite easy for most problems. Here we

simply select the **Initialize Mesh** icon to create the initial grid, and hit the **Refine Mesh** icon as many times as necessary to double the number of mesh uniformly throughout the geometry until we get the desired grid for the problem. For 1-D problems, this is relatively straightforward, and a single **Refine Mesh** for the current problem is quite sufficient (gives 30 elements).

Note: In more complex 2-D and 3-D geometries, this step may require more care to get an accurate solution throughout the full system, and COMSOL Multiphysics has a number of user options for refining the mesh grid for particularly difficult problems.

Solve the Problem

Many general PDE problems are difficult to solve, but most 1-D BVPs are not. Thus, simply hitting the **Solve** icon will solve the current problem (in a fraction of a second)! The **Restart** icon does the same thing, except that it uses a previous solution as a starting guess, if one is available.

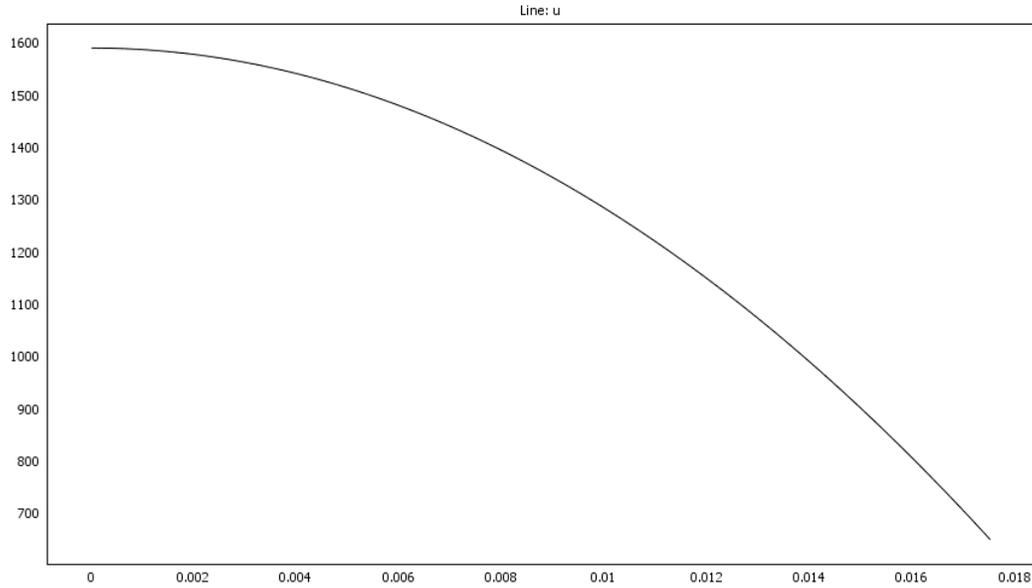
Note: The **Solver Parameters** and the **Solver Manager** icons can be used, as needed, for more difficult problems. These icons open additional GUI windows with a number of user-adjustable parameters that one can modify to help the overall solution process. However, until you get comfortable with using COMSOL Multiphysics, I suggest that you leave most of the default values alone -- they have been preset to work efficiently for the most common problems.

Perform Some Post Processing

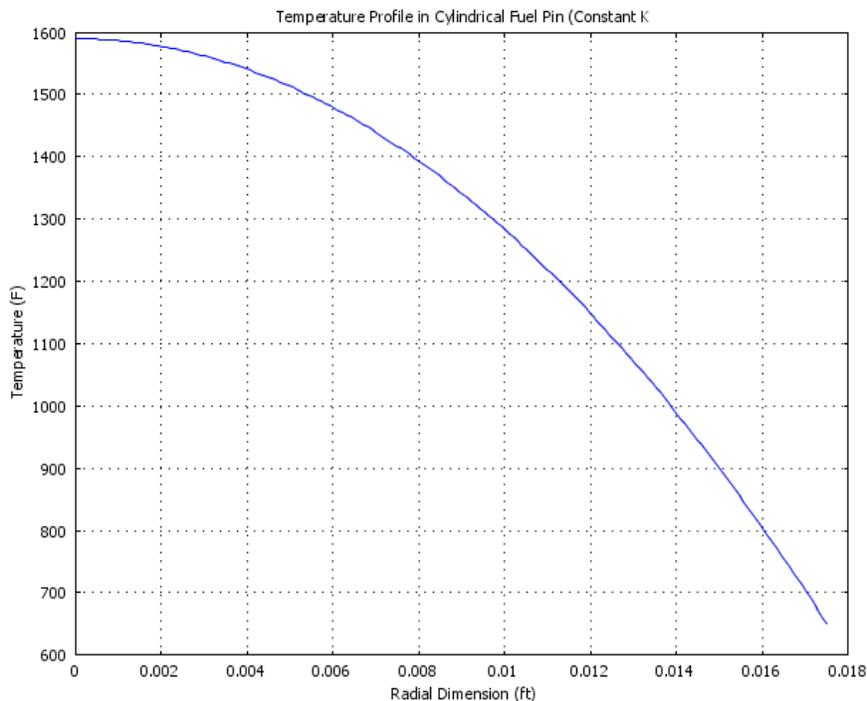
Once COMSOL Multiphysics has solved the problem of interest, you can go into the **Postprocessing Mode** to help you visualize and analyze the system. In the simplest form, you can plot the solution $u(x)$ versus x [or $T(r)$ vs. r in our case]. This can be visualized in the COMSOL Multiphysics main plot screen or it can be placed into a **New Figure** in COMSOL Multiphysics or a **Matlab Figure** if you are running with the Matlab interface open. Plotting within a Matlab figure window gives the most flexibility, since you have the full power of Matlab to customize things as needed. However, for quick access to the primary results from your COMSOL Multiphysics analyses, you can simply use the GUIs in COMSOL Multiphysics to do most of the initial analyses.

Note: For more complicated problems, where a fair amount of analysis is required, I usually use COMSOL Multiphysics to take a quick look at things and then use Matlab to perform the desired analysis and to customize the results for formal documentation purposes. An example of a relatively simple Matlab post processing file that illustrates some typical tasks is available on the course website for the cylindrical fin problem that is treated within Section V and Section VIII of the formal class notes. For the current example, we will simply use the COMSOL Multiphysics GUI to do some post processing.

Continuing our example, you should already have a view of the solution, $u(x) = T(r)$, within the main COMSOL Multiphysics GUI -- this should have appeared just after you hit the **Solve** icon in the previous step. This plot has the Geometry Edges shown by default. These can be removed and the plot scaled more appropriately by hitting the **Plot Parameters** icon, and deselecting the **Geometry Edges** button under the **General** tab. Upon doing this, you should have a plot similar to the one shown below. This is pretty basic, but it displays the same solution that we generated previously using the Shooting and FD Methods -- and we got this with no mathematics and no programming!!!



We can generate a better visualization using the post processing capability in COMSOL Multiphysics. In particular, select the **Postprocessing** option along the top menu bar of the GUI and select **Domain Plot Parameters**. In the dialog box that appears, select **Line/Extrusion Plot** under the **General** tab and tell the plot to appear in a **New Figure** (in the lower left corner of the dialog box). You should also check the **Line/Extrusion** tab dialog box to make sure that the **Line Plot** and **Subdomain 1** entries are selected and that the expression to be plotted is $u(\mathbf{x})$. This plot has a default grid and some simple labels for the title and x and y axis labels. Using the **Edit Plot** icon on the top of this figure, you can improve upon the labels, modify the axes, turn off/on the grid etc. After editing the labels somewhat the plot should appear as shown below.



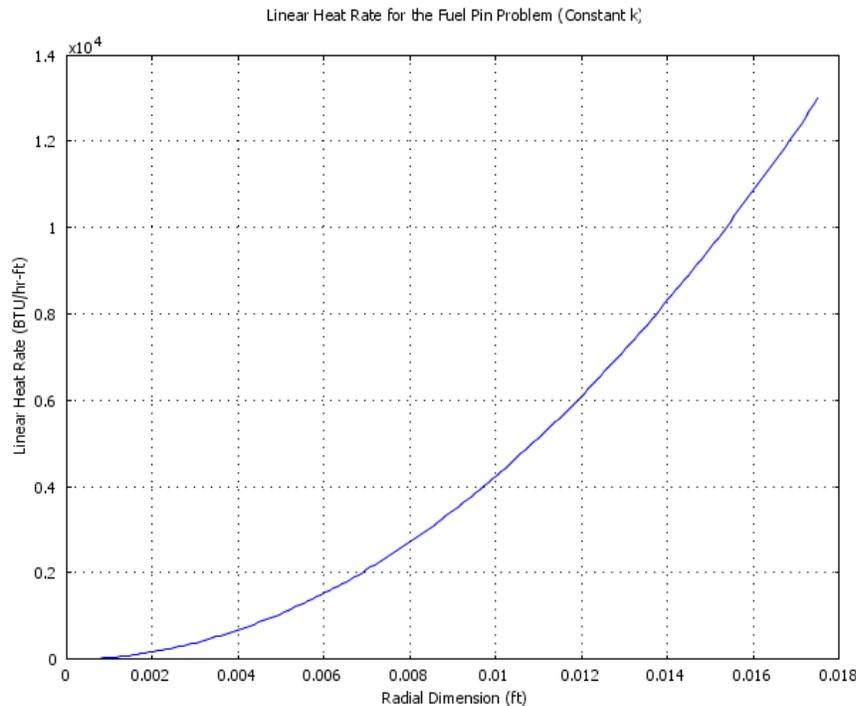
Note that you can also plot any function of the solution that may be of interest. For example, in this problem, the energy flow rate per unit height of the fuel pin (BTU/hr-ft) is given by

$$q'(r) = -k_f A \frac{dT}{dr} = -k_f 2\pi r \Delta z \frac{dT}{dr} = -(1.1)(2\pi x)(1) u_x$$

where u_x represents the gradient of the temperature profile (using the default independent and dependent variables within COMSOL Multiphysics). We can plot this function by selecting the **Postprocessing/Domain Plot Parameters** option again, being sure to select **New Figure** again so that we don't write over Figure 1 containing the temperature profile. In the **Line/Extrusion** tab dialog box type the above formula into the expression field, as follows:

$$\mathbf{-1.1*2*pi*x*ux}$$

where a Matlab-like notation is used (except that we don't need to worry about "dot" arithmetic). Figure 2 should be generated and, after modifying the titles and labels somewhat, it should resemble the plot given below.



Recall that all the energy generated within the fuel pin is removed via radial conduction through the pin, which is eventually deposited into the coolant. Thus, the heat transfer rate out of the pin per unit height is given by

$$q'(R_f) = -k_f 2\pi r \Delta z \left. \frac{dT}{dr} \right|_{r=R_f} = q'''(\pi R_f^2) = (1.35e7)\pi(0.0175)^2 = 12988.5 \text{ BTU/hr-ft}$$

and we see that this is indeed the value of the linear heat rate at the outer radius of the fuel pin as shown in the above plot.

Note that numerical evaluations can also be performed directly from the COMSOL Multiphysics GUI. For example the above computation can be performed using the internal code variables by selecting **Postprocessing/Point Evaluation** and typing in the above equation, $-1.1*2*pi*x*ux$ into the expression field for boundary point 2. The result is returned as 12988.52 in the message log at the bottom of the user interface -- just what we got from our hand calculation.

Subdomain integrations are also possible. Although the volumetric heat source is constant in this problem (making the integration over the volume pretty simple), we can still go through the formal integration within COMSOL Multiphysics to compute the total energy production rate per foot of length -- just as a demo of this capability. Mathematically, the linear heat rate can be written as

$$q'_{\text{total}} = \int_0^{R_f} q''' 2\pi r \Delta z dr$$

where $\Delta z = 1$ ft. In COMSOL Multiphysics, this can be evaluated by selecting **Postprocessing/Subdomain Integration** and entering the expression $1.35e7*2*pi*x$ as the integrand. Of course, the result that is returned to the message log is exactly as computed above, 12988.52 BTU/hr-ft.

As a final post processing step in this first example, let's get COMSOL Multiphysics to plot $u(x) = T(r)$ within a Matlab figure and then, within the Matlab command window, we can add the analytical solution to the plot and include some annotations to identify the two solutions, where the exact solution to this simple linear BVP is given by

$$T(r) = T_s + \frac{q'''}{4k_f} (R_f^2 - r^2)$$

To do this, we again select the **Postprocessing/Domain Plot Parameters** option and, under the **General** tab we can set the **Plot in** entry to **Matlab Figure** and set the titles and axis labels to

Comparison of Analytical and Numerical Temperature Profiles (Constant k)

Radial Dimension (ft) and Temperature (F)

Also, under the **Line/Extrusion** tab, we can modify the default **Line Settings** to plot red circles with no line for the numerical solution. Doing this gives a Matlab figure with the discrete temperature solution and the line desired settings.

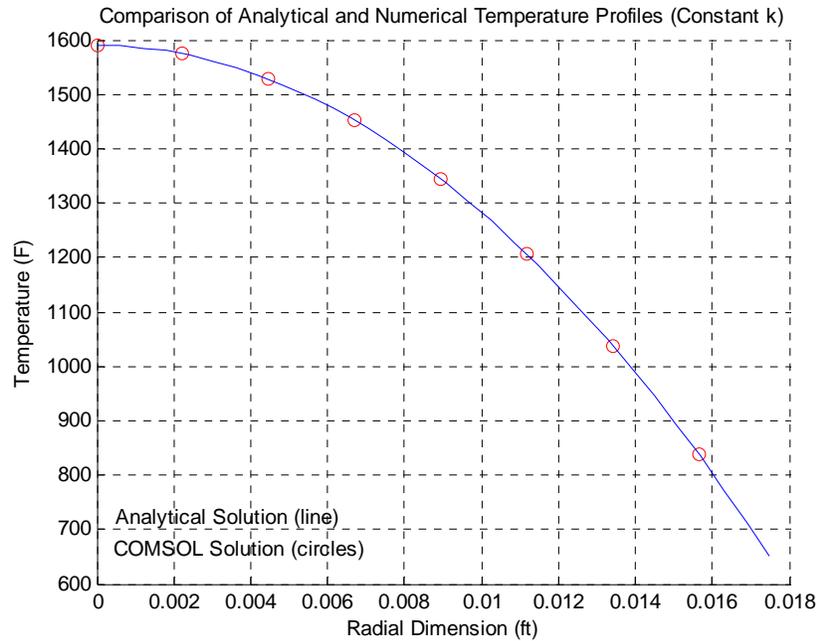
Now, from within the Matlab command window, issue the following commands:

```

Ts = 650; qppp = 1.35e7; kf = 1.1; Rf = 0.0175;
r = linspace(0,Rf,30);
T = Ts + (qppp/4/kf)*(Rf^2 - r.^2);
hold on
plot(r,T,'b-')
gtext({'Analytical Solution (line)';'COMSOL Solution (circles)'})

```

The result of this operation is the figure on the following page. Of course, all this could have been done directly within the COMSOL Multiphysics GUI (by plotting the analytical expression along with the numerical solution), but this illustration shows a simple application of the Matlab interface to COMSOL Multiphysics. We could do a lot more, but this is enough for our first example with COMSOL Multiphysics...



Save Your Work

Well, this completes our first demo with COMSOL Multiphysics. Before exiting the program, make sure you save your work for later use. To do this, simply select **File/Save As** from the main menu and enter the desired file name, such as **fuel_temp_1**. COMSOL Multiphysics will save the file as **fuel_temp_1.mph** in the selected directory. This can then be retrieved for subsequent use as needed.

Case 2: The Non-Linear Fuel Pin Problem

As a test of your understanding from the above example, let's modify the linear fuel pin problem to include a nonlinear thermal conductivity. The instructions for doing this will be given in outline form, so you will have to figure out on your own how to make this happen within the COMSOL Multiphysics interface. The steps needed are:

1. Run COMSOL Multiphysics and open the Case 1 model (which was saved above as **fuel_temp_1.mph**).
2. Go to the **Subdomain Mode** and change the value of the c coefficient to the following:
 $c = -(94.4 * u^{(-0.556)}) * x$.
3. Under **Solver Parameters** select **Stationary Nonlinear** since this is a nonlinear problem.
4. Solve the problem by hitting the **Restart** button (this uses the old solution as a guess).
5. In the **Postprocessing Mode**, get a well labeled plot of the numerical temperature profile. Also include the analytical solution,

$$T(r) = \left[T_s^{1+a_2} + \left(\frac{1+a_2}{a_1} \right) \frac{q'''}{4} (R_f^2 - r^2) \right]^{\frac{1}{1+a_2}} = \left[17.739 + 1.5874 \times 10^4 (3.0625 \times 10^{-4} - r^2) \right]^{2.2523}$$

on the same plot as a check on the numerical solution (do this in the user interface by selecting **Keep Current Plot** within the **Domain Plot Parameters** dialog window). Note that the numerical values were pre-computed here for ease in implementing this expression within the GUI.

- Use the subdomain integration capability within the GUI to find the average temperature within the fuel pin, where

$$T_{ave} = \frac{1}{\pi R_f^2} \int_0^{R_f} T(r) 2\pi r dr$$

- Save your new file as **fuel_temp_2.mph** (if desired).

The outcome from the above tasks produces the plot shown below and a numerical result for the average temperature of $0.84082/9.6211 \times 10^{-4} = 873.93$ F. Clearly, the numerical solution and analytical solution agree, and the amount of work needed to get the COMSOL finite element solution was rather trivial. This tool (and other similar software packages) certainly makes it easier to get solutions to some rather complex problems. As a practicing engineer, you will use your mathematics background in a number of ways, but I expect one application will be as a base for learning to use and to appreciate the computational analysis tools available for solving realistic engineering design and analysis problems -- such as the simple example illustrated here...

