NUEN 618

Homework Assignment #4

October 24, 2019

Robert Turner

NUEN 618

Homework #4

Exercise 1: 1D Steady-State Nonlinear Heat Conduction
Model Problem:

$$-\frac{d}{dx}\left(k(T)\frac{dT}{dx}\right) = q''' \qquad \forall x \in [0, L] \tag{1}$$

$$\left.\frac{dT}{dx}\right|_0 = 0 \quad \text{(NeumannBC)} \tag{2}$$

$$T(L) = T_{\text{right}} \quad \text{(DirichletBC)} \tag{3}$$

with

$$k(T) = 1.5 + \frac{2510}{215+T}$$
$$L = 0.45$$
$$T_{\text{right}} = 300$$

Goal: To discretize and solve this problem using FEM (see provided 1D FEM Python code on ecampus, but feel free to write your own from scratch.)

Tasks:

- **[20 pts]** Build a function to compute the nonlinear residual $F(T)$ (i.e., a vector-valued function whose length is equal to the number of unknowns in your mesh). Once F has been created (including the natural, i.e. Neumann and Robin BCs), you need to impose the Dirichlet BC, which is simply $F_0(T) = T_0 - T_{\text{right}}$, that is, you completely overwrite the entry in F_0 from the loop over elements by the Dirichlet formula. Once again, this happens after the regular FEM process.

  In my code, the residual is computed element by element, where a 2×1 list is created for each element. Then, these lists are placed in a global residual vector of length N if N is the number of nodes in the problem. However, these 2×1 lists overlap by 1 element on the interior of the global residual vectors. These elements of the overlapping lists are added. The effect of this is the same as computing the residual as $R = b - Ax$ after creating the A matrix in the linear fashion.

- **[10 pts]** Create a function to build a numerical Jacobian (obtained by finite difference in a column-by-column manner).

  In my code, the Jacobian is built by evaluating the 3 column vectors that make up the entirety of the tridiagonal Jacobian matrix. These columns are then appropriately placed for all of the interior columns with the first and last two elements placed by hand.

- **[5+10 pts]** Solve the nonlinear problem using Newton's method. For the linear solves, use both a direct solver and a `gmres` solve. The answer should be identical.

- **[20 pts]** Implement a JFNK approach using `gmres` for the linear solves but do not pass J. Instead, create a `LinearOperator` object to create a function the returns the action of J on any vector ν using only calls to the nonlinear function F. At this stage, `gmres` is not preconditioned.

Results to present:

- **[5 pts]** Find the analytical solution and use it in your plots.

  Begin by taking the indefinite integral with respect to x of both sides of the original equation:

$$-k(T)\frac{dT}{dx} = q'''x + c.$$

  At this point, the Neumann condition at the left boundary can be used to solve for c:

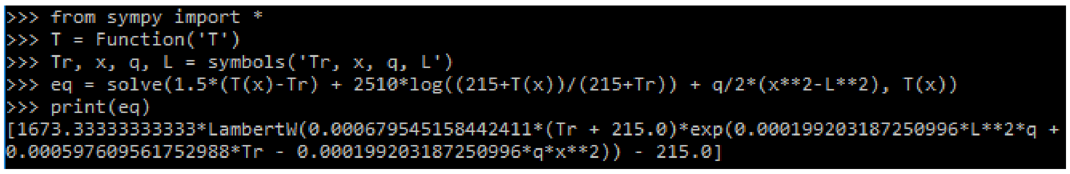
$$-k(T(0))\cancelto{0}{\frac{dT(0)}{dx}} = q'''0 + c$$

  So, $c = 0$. Both sides are again indefinitely integrated with respect to x:

$$1.5T(x) + 2510\ln(215 + T(x)) = -q'''\frac{x^2}{2} + c.$$

  Now, the Dirichlet condition is imposed on the right boundary to find c, and `Sympy` can be used to find an equation for $T(x)$ as seen in Figure 1:

$$c = 1.5T(L) + 2510\ln(215 + T(L)) + q'''\frac{L^2}{2}.$$



```
>>> from sympy import *
>>> T = Function('T')
>>> Tr, x, q, L = symbols('Tr, x, q, L')
>>> eq = solve(1.5*(T(x)-Tr) + 2510*log((215+T(x))/(215+Tr)) + q/2*(x**2-L**2), T(x))
>>> print(eq)
[1673.33333333333*LambertW(0.000679545158442411*(Tr + 215.0)*exp(0.000199203187250996*L**2*q +
0.000597609561752988*Tr - 0.000199203187250996*q*x**2)) - 215.0]
```

Figure 1: Result of using `Sympy` to find an analytical solution for $T(x)$.

- **[2.5 pts]** Verify that each of the above numerical options yields the same final result.

  Figures 2, 3, and 4 show all results at a nonlinear tolerance of 1E-8 with spatial discretizations of 25, 50, and 100 elements, respectively.

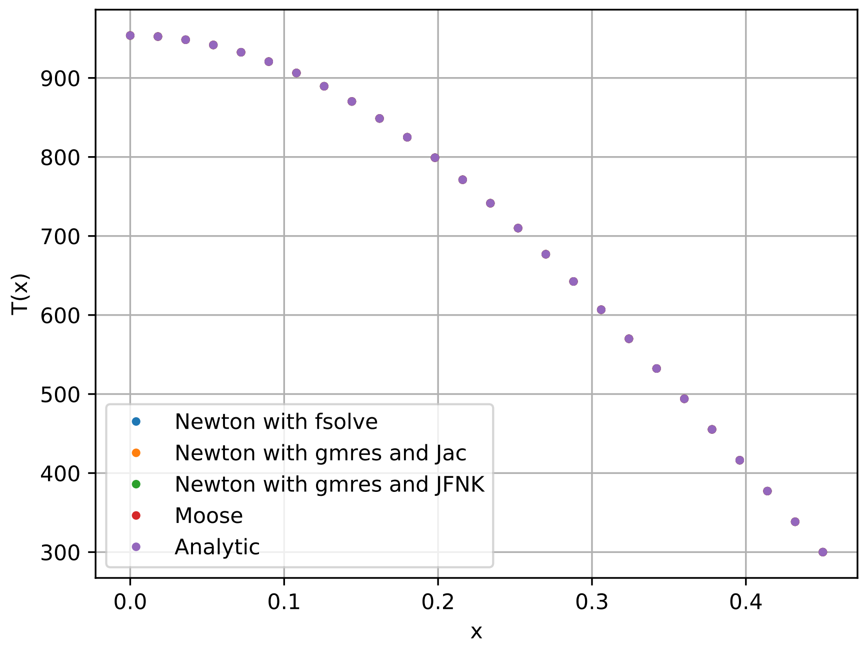


Figure 2: Finite element temperature profile with all solve methods on a spatial domain of 25 elements.

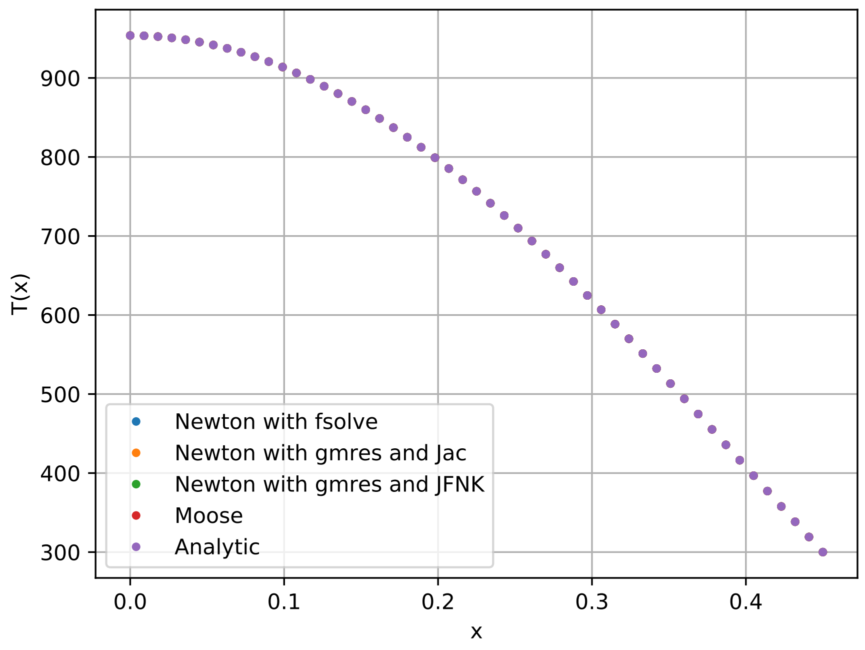


Figure 3: Finite element temperature profile with all solve methods on a spatial domain of 25 elements.

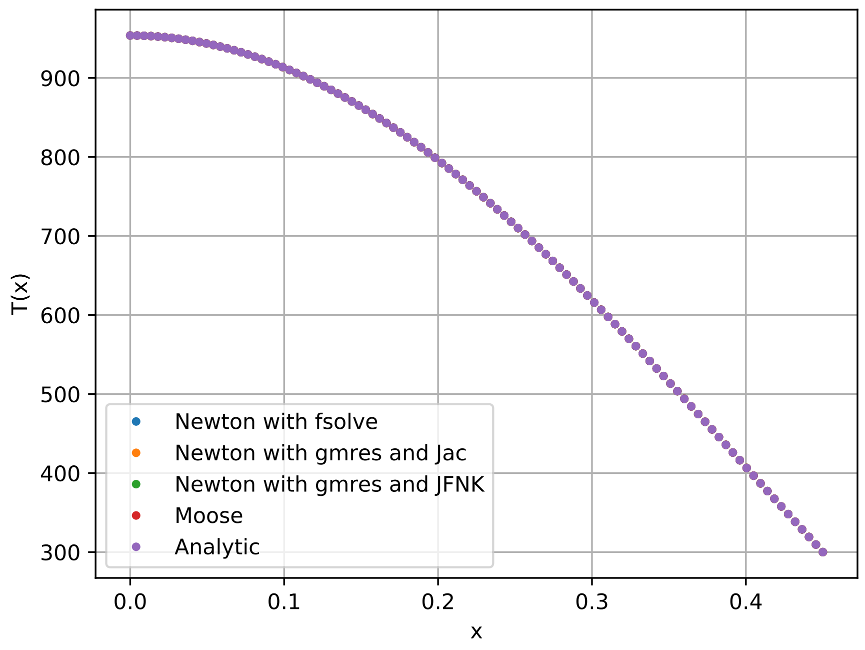


Figure 4: Finite element temperature profile with all solve methods on a spatial domain of 25 elements.

- **[2.5 pts]** Compare the number of nonlinear iterations between a direct solver and using `gmres`. Compare the number of linear iterations needed for `gmres` when using a numerical Jacobian and the matrix-free approach.

  The iteration results for spatial discretizations of 25, 50, and 100 elements can be seen in Tables 1, 2, and 3, respectively.

Table 1: Iterations needed to reach nonlinear convergence tolerance of 1E-8 for the nonlinear solves and 1E-5 for linear solves with a spatial discretization of 25 elements.

Nonlinear Iterations w/ direct solve	Nonlinear Iteration w/ Jacobian	Linear Iterations	Nonlinear Iteration w/ JFNK	Linear Iterations
5	1	257	1	260
	2	163	2	193
	3	118	3	130
	4	256	4	249
	5	0	5	0

Table 2: Iterations needed to reach nonlinear convergence tolerance of 1E-8 for the nonlinear solves and 1E-5 for linear solves with a spatial discretization of 50 elements.

Nonlinear Iterations w/ direct solve	Nonlinear Iteration w/ Jacobian	Linear Iterations	Nonlinear Iteration w/ JFNK	Linear Iterations
5	1	510	1	510
	2	510	2	510
	3	510	3	510
	4	510	4	510
	5	510	5	0

Table 3: Iterations needed to reach nonlinear convergence tolerance of 1E-8 for the nonlinear solves and 1E-5 for linear solves with a spatial discretization of 100 elements.

Nonlinear Iterations w/ direct solve	Nonlinear Iteration w/ Jacobian	Linear Iterations	Nonlinear Iteration w/ JFNK	Linear Iterations
5	1	1010	1	1010
	2	1010	2	1010
	3	1010	3	1010
	4	1010	4	1010
	5	1010	5	1010
	6	1010	6	1010
	7	1010	7	1010
	8	0	8	1010

- Provide your Python code.

Exercise 2 [25 pts]: 1D Steady-State Nonlinear Heat Conduction
Repeat Exercise 1 using MOOSE. Compare the solution between your FEM code and MOOSE. Provide the MOOSE input file in your report. You should not need to implement anything in MOOSE at this stage.

The MOOSE solutions are compared with the other results in Figures 2, 3, and 4.