Final Exam Solutions
Administered: Monday, December 9, 2025
10:30 AM – 12:45 PM
30 points

Problem 1. (14 points)

The one-dimensional heat equation can describe heat transfer in silver with both heat conduction and radiative heat loss.

$$0 = \frac{d^2T}{dz^2} - \frac{\varepsilon\sigma S}{k_c}\left(T^4 - T_s{}^4\right)$$

A silver cylindrical rod has diameter 0.04 m and length 0.5 m. One end of the rod is maintained at $T(z=0) = 600$ K. The other end of the rod is maintained at $T(z=L) = 800$ K. The variables and parameters are defined below.

- T is the temperature, K
- z is the axial position, m
- k_c is the thermal conductivity, $k_c = 429.0 \frac{W}{m \cdot K}$
- d is the diameter of the rod, $d = 0.04\ m$
- l is the length of the rod, $l = 0.5\ m$
- A is the surface area of the rod, $A = \pi d l$
- V is the volume of the rod, $V = \frac{\pi}{4} d^2 l$
- S is the surface area to volume ratio of the rod, $S = {}^{A}/_{V} = \frac{4}{d} = 100.0\ m^{-1}$
- T_{surr} is the surrounding temperature, $T_{surr} = 300\ K$
- σ is the Stefan–Boltzmann constant, $\sigma = 5.670373x10^{-8} \frac{W}{m^2 \cdot K^4}$
- ε is the gray body emissivity $\varepsilon = 0.04$

Answer the following questions and perform the following tasks.

(a) Is this ODE problem linear or nonlinear?
(b) Is this ODE problem an initial value problem or a boundary value problem?
(c) Convert this second order ODE into a system of two first order ODEs.
(d) Find the initial temperature gradient at $z = 0$. Important hint: a good initial guess for the initial temperature gradient is the average value of the slope of the temperature.
(e) Sketch the temperature profile.
(f) Verify that your discretization resolution was sufficient.
(g) What is the temperature in the middle of the rod?

Solution

(a) Is this ODE problem linear or nonlinear?

The problem is nonlinear due to exponent of 4 in the radiation term.

(b) Is this ODE problem an initial value problem or a boundary value problem?

This problem is a boundary value problem because both conditions are not given at the same value of the independent variable, *z*.

(c) Convert this second order ODE into a system of two first order ODEs.

This conversion follows a three step process.

Step 1. Define new variables.

$$y_1 = T \qquad y_2 = \frac{dT}{dz}$$

Step 2. Write ODEs for the new variables.

In this transformation, the first equation is always

$$\frac{dy_1}{dz} = y_2$$

The second equation is substituting the variables in Step 1 into the original ODE.

$$0 = \frac{d^2T}{dz^2} - \frac{\varepsilon\sigma S}{k_c}\left(T^4 - {T_s}^4\right)$$

or

$$\frac{d^2T}{dz^2} = \frac{\varepsilon\sigma S}{k_c}\left(T^4 - {T_s}^4\right)$$

$$\frac{dy_2}{dz} = \frac{\varepsilon\sigma S}{k_c}\left(y_1^4 - {T_s}^4\right)$$

Step 3. Write initial conditions for the new variables.

$$y_1(z=0) = T(z=0) = 600\ K \qquad y_2(z=0) = \left.\frac{dT}{dz}\right|_0 \text{ (not given)}$$

(d) Find the initial temperature gradient at *z* = 0. Important hint: a good initial guess for the initial temperature gradient is the average value of the slope of the temperature.
(e) Sketch the concentration profile.

$$y_2(z=0) \approx \frac{T(z=L)-T(z=0)}{L-0} = \frac{800-600}{0.5} = 400\ \frac{K}{m}$$

This is a boundary value problem. I used as the starting points the two Python functions, rk4n.py and nrnd1.py, distributed on the course website in the odesolver_bvp folder.

I modified the input file for rk4n.py, which uses the classical fourth-order Runge-Kutta method to solve a system of n ODEs.

```
def funkeval(x, y):
    dydx = np.zeros_like(y)
    sig = 5.670373e-8 # W/m^2/K^4
    eps = 0.04 #
    kc = 429.0 # W/m/K
    S = 100.0 # 1/m
    Tsurr = 300.0 # K
    con = eps*sig*S/kc
    dydx[0] = y[1]
    dydx[1] = con*(y[0]**4-Tsurr**4);
    return dydx
```

I also modified the input for nrnd1.py, which uses the Newton Raphson method with numerical derivatives,

```
def funkeval(x):
    xo = 0.0 # initial value of independent variable
    yo_1 = 600.0 # first initial condition at xo
    yo_2 = x # guess second initial condition at xo
    xf = 0.5 # final value of independent variable
    yf = 800.0 # boundary condition at xf
    n = 1000 # number of intervals
    # call Runge Kutta method to integrate ODEs
    x, y =rk4n(n,xo,xf,[yo_1,yo_2])
    # final value of first variable
    yf_calc = y[n,0]
    # this must equal the boundary condition
    f = yf_calc-yf
    return f
```

I wrote a small script to sole the problem. This code calculates the initial guess, calls the Newton-Rapshon Method to find the true correct initial temperature gradient, then calls the Runge-Kutta method to solve the ODEs with the correct initial temperature gradient. It generates a plot and reports the final value of the temperature to confirm that the boundary condition at z = L was satisfied and it reports the temperature in the middle of the rod as requested.

```
from nrnd1 import *
from rk4n import *

# generate initial guess for initial slope with average slope
xo = 0.0 # initial value of independent variable
yo_1 = 600.0 # first initial condition at xo
xf = 0.5 # final value of independent variable
yf = 800.0 # boundary condition at xf
```

```
slope_avg = (yf-yo_1)/(xf-xo)
x0 = slope_avg;

# call Newton Raphson Method
result, error = nrnd1(x0)
print(f"the initial slope is {result:.6e} ")

# evaluate solution at final result
yo_2 = result
n = 1000 # number of intervals
# call Runge Kutta method to integrate ODEs
x, y =rk4n(n,xo,xf,[yo_1,yo_2])
yf_calc = y[n,0]

print(f"the final value of y(1) is {yf_calc:.6e} ")

nmid = int(n/2)
xmid = x[nmid]
ymid = y[nmid,0]
print(f"the midpoint value of y(1) at {xmid } is {ymid} ")
```

This command generated the following output:

```
icount = 1, xold = 4.000000e+02, f = 1.310588e+01, df = 5.156843e-01, xnew = 3.745855e+02, err = 1.000000e+02
icount = 2, xold = 3.745855e+02, f = 5.750334e-03, df = 5.152335e-01, xnew = 3.745743e+02, err = 2.979552e-05
icount = 3, xold = 3.745743e+02, f = 1.095486e-09, df = 5.152333e-01, xnew = 3.745743e+02, err = 5.676234e-12
the initial slope is 3.745743e+02
the final value of y(1) is 8.000000e+02
the midpoint value of y(1) at 0.25 is 696.1593108819674
```

The code converged because the error was less than the stated tolerance of 10^{-6}. The initial slope is

$$\left.\frac{dT}{dz}\right|_0 = 374.57\,\frac{K}{m}$$

When we ran the Runge Kutta code with this initial condition at the end of the script, it generated the result.

```
the initial slope is 3.745743e+02
the final value of y(1) is 8.000000e+02
```

When we use 1000 intervals in the Runge Kutta code, we obtain a temperature at the far end of the rod of 800.0 K. That matches our specified boundary condition.

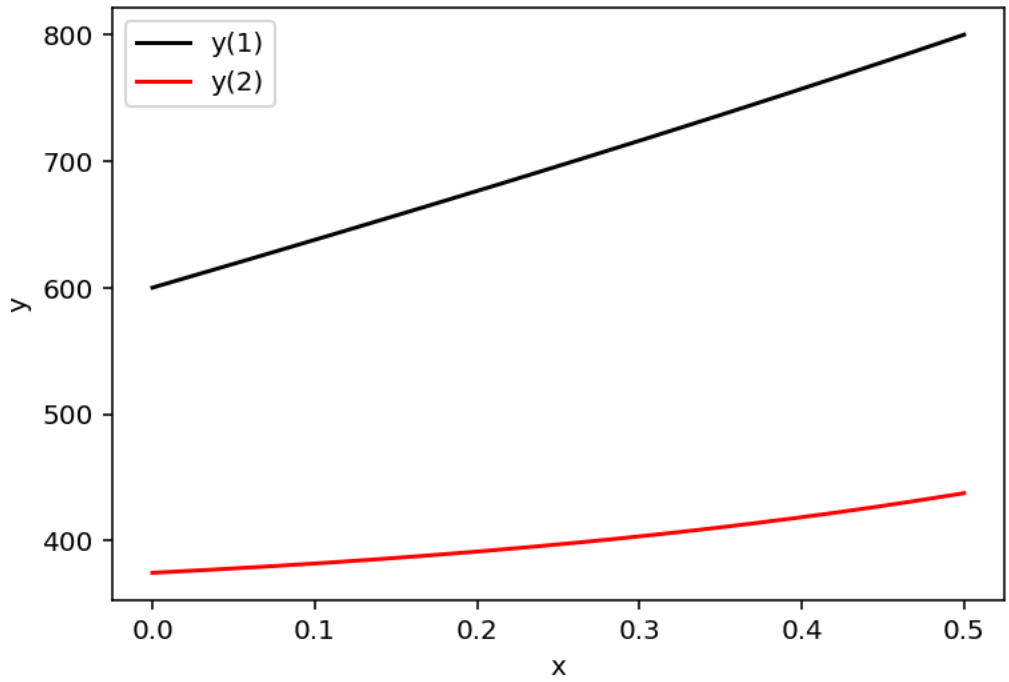
(f) Verify that your discretization resolution was sufficient.

In order to verify that the spatial discretization was sufficiently fine, we also use 10,000 intervals in the Runge Kutta code. For this finer resolution, we obtain a concentration at the far end of the

```
the final value of y(1) is 8.000000e+02
the midpoint value of y(1) at 0.25 is 696.159310881967
<Figure size 864x576 with 0 Axes>
```

The two results agree, so we had a good discretization resolution.

A plot of the profile is shown below. The black line marked "1" is the temperature. The red line marked "2" is the temperature gradient.



The temperature profile (black) appears fairly linear but you can see from the variation in the gradient (red) that there is a slight curvature due to radiative losses.

(g) What is the temperature in the middle of the rod?

The script we wrote also reported the temperature in the middle of the rod.

```
the midpoint value of y(1) at 0.25 is 696.1593108819674
```

The temperature in the middle of the rod (z = 0.25 m) is 696.159 K.

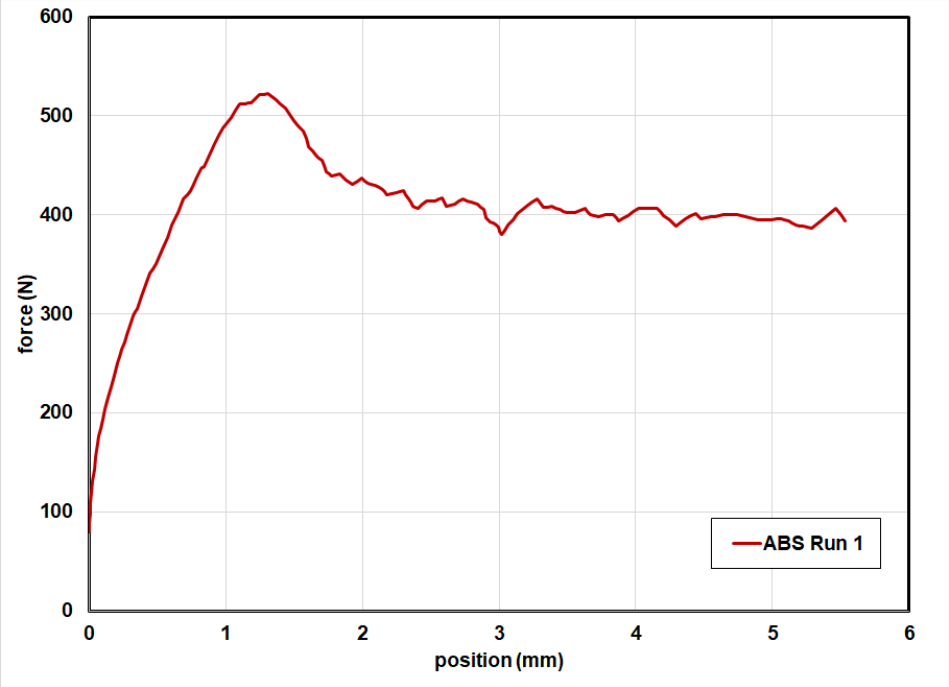
If there were no radiative losses, the temperature profile would be linear and the midpoint temperature would be 700 K. The difference is due to radiative heat loss.

Problem 2. (6 points)

Work, W, is defined as the integral of the force, F, over distance, x,

$$W = \int_{x_o}^{x_f} F dx$$

The data for a tensile test experiment of Acrylonitrile Butadiene Styrene (ABS) is reported in the file "xm4p02_f25.txt" on the exam portion of the course website. The first column of the data is position in mm and the second column of the data is force in Newtons. A plot of the data is shown below.



(a) What method can you use to numerically integrate this data to obtain the work done during the tensile test?
(b) Report the numerical integral of this data.
(c) Convert this result to SI units, i.e. Joules.

Solution:

(a) What method can you use to numerically integrate this data to obtain the work done during the tensile test?

The Trapezoidal rule can be used to numerically integrate this data.

(b) Report the numerical integral of this data.

To integrate this data, I used the following script

```
import numpy as np
from trapzd import *

# Read data from the file
# The file should contain two columns: x and f(x)
nheader = 1
datamat = np.loadtxt('xm4p02_f25.txt', skiprows=nheader)

# Extract the x values and f(x) values from the data matrix
x = datamat[:, 0]
f = datamat[:, 1]

# Perform trapezoidal integration
integral = trapzd(x,f)
print(f" Trapezoidal rule from data = {integral}")
```

When I executed this script, the code returned the following output.

```
Using the Trapezoidal method to integrate data
the integral is 2.252086e+03

Trapezoidal rule from data = 2252.0862500000017
```

So, the integral is 2,252 N-mm.

(c) Convert this result to SI units, e.g. Joules.

A Joule is a N-m. There are 1,000 mm/m, so we divide our result from (b) by 1,000.

The work done during the HIPS tensile test is 2.252 J.

Problem 3. (8 points)

Consider a set of three reactions occurring in a closed pot (a batch reactor) involving compounds, A, B and C.

number	reaction	rate expression	rate constant
1	A+A → B	$r_1 = k_1 A^2$	$k_1 = 7\ l \cdot mol^{-1} s^{-1}$
2	B+A → C	$r_2 = k_2 A \cdot B$	$k_2 = 13\ l \cdot mol^{-1} s^{-1}$
3	C → 3A	$r_3 = k_3 C$	$k_3 = 11\ s^{-1}$

These equations give rise to the following steady state (at infinite time) mass balances.

compound	rate expression
A	$0 = 3k_3 C - 2k_1 A^2 - k_2 A \cdot B$
B	$0 = k_1 A^2 - k_2 A \cdot B$
C	$0 = k_2 A \cdot B - k_3 C$

We also recognize that the sum of the mass fractions equal unity.

$$A + B + C = 1$$

Your goal is to find the steady state composition in this reactor. To do so, answer the following questions.
(a) Are these equations linear or non-linear?
(b) Since you have three unknowns, which three of the four equations should be used to solve for the composition?
(c) What solution technique should use you to solve this problem?
(d) Provide a solution.

Solution:

(a) Are these equations linear or non-linear?

These three mass balances are nonlinear because they contain terms in which the unknown variables are multiplied together and/or squared.

(b) Since you have three unknowns, which three of the four equations should be used to solve for the composition?

The third balance is a linear combination of the first two balances. (EQN1 = -2*EQN2 -3*EQN3) Therefore, you need two of the mass balances and the constraint that the sum of the mass fractions is unity.

(c) What solution technique should use you to solve this problem?

I would use multivariate Newton Raphson with Numerical Approximations to the derivatives to solve this system of nonlinear algebraic equations.

(d) Provide a solution.

I modified the input file for nrndn.py as follows

```
def funkeval(x):
    n = len(x)
    f = np.zeros(n)
    A = x[0];
    B = x[1];
    C = x[2];

    k1 = 7.0;
    k2 = 13.0;
    k3 = 11.0;

    rate1 = k1*A*A
    rate2 = k2*A*B
    rate3 = k3*C

    f[0] = 3.0*rate3 - 2.0*rate1 - rate2
    f[1] = rate1 - rate2
    f[2] = A + B + C - 1.0
    return f
```

The script to execute the code was as follows. I used an initial guess of 1/3, 1/3, 1/3 for A, B & C.

```
import numpy as np
from nrndn import *

x0 = np.array([1.0/3.0, 1.0/3.0, 1.0/3.0])
tol = 1.0e-6
iprint = 1
x, err, f = nrndn(x0, tol, iprint)
```

This code produced the following output.

```
iter =    1, err =  1.96e-01, f =  4.63e+00
iter =    2, err =  5.96e-02, f =  6.81e-01
iter =    3, err =  7.58e-03, f =  5.96e-02
iter =    4, err =  1.13e-04, f =  8.65e-04
iter =    5, err =  2.43e-08, f =  1.87e-07
Solution converged.
x(1) =  5.32646346e-01
x(2) =  2.86809571e-01
x(3) =  1.80544083e-01
```

Therefore the steady state solutions are

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 0.5326 \\ 0.2868 \\ 0.1805 \end{bmatrix}$$

The convergence of the root was not especially sensitive to the initial guesses. Other initial guesses that led to the same solution included.

```
x0 = np.array([1.0/2.0, 1.0/4.0, 1.0/4.0])
```

```
x0 = np.array([1.0/4.0, 1.0/2.0, 1.0/4.0])
x0 = np.array([1.0/4.0, 1.0/4.0, 1.0/2.0])
```

Even bad guesses where the mole fractions didn't sum to one actually converged.

```
x0 = np.array([1.0, 1.0, 1.0])
x0 = np.array([10.0, 20.0, 30.0])
```

Even horribly bad guesses where the mole fractions didn't sum to one actually converged and some were negative converted.

```
x0 = np.array([10.0, -20.0, 30.0])
```

Problem 4. Project Question. (2 points)

Regarding the course project, answer the following question.

(a) In your opinion, what was the most significant result of your project?

.