Final Exam
Administered: Monday, December 9, 2019
5:00 PM – 7:00 PM
28 points

Problem 1. (6 points)

The error function, erf(x) is defined as

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad \text{for } x > 0$$

(a) Evaluate the error function for x = 1.7 using the intrinsic gamma function in Matlab, `erf`. You likely will need to use the `format long` statement in MatLab to get enough digits to display.
(b) How many intervals so you need in the second-order Simpson's method to obtain this result to six significant digits?
(c) In statistics, for nonnegative values of x, the error function has the following interpretation: for a random variable Y that is normally distributed with mean 0 and variance 1/2, erf(x) is the probability of Y falling in the range [−x, x]. Knowing this, show how can you use the Matlab `cdf` command to evaluate this integral.

Solution:

To solve all three parts of this problem, I wrote the following script:

```
clear all;
format long;
% part a
x = 1.7;
parta = erf(x)
% part b
a = 0;
b = x;
% loop over number of intervals
for nintervals = 6:2:16
  integral = simpson2(a,b,nintervals);
  relative_error = (integral - parta)/parta;
  fprintf(1,'For %i intervals, the integral is %f & the relative error is %e \n',
nintervals, integral, relative_error);
end
% part c
mean = 0;
variance = 0.5;
stdev = sqrt(variance);
partc = cdf('normal',x,mean,stdev) - cdf('normal',-x,mean,stdev)
```

This script returns the following output:

```
>> xm4p01_f19
```

```
parta =   0.983790458590775
For 6 intervals, the integral is 0.983745 & the relative error is -4.585052e-05
For 8 intervals, the integral is 0.983777 & the relative error is -1.414098e-05
For 10 intervals, the integral is 0.983785 & the relative error is -5.720551e-06
For 12 intervals, the integral is 0.983788 & the relative error is -2.739818e-06
For 14 intervals, the integral is 0.983789 & the relative error is -1.472696e-06
For 16 intervals, the integral is 0.983790 & the relative error is -8.609076e-07

partc =   0.983790458590775
```

(a) Evaluate the error function for x = 1.7 using the intrinsic gamma function in Matlab, `erf`. You likely will need to use the `format long` statement in MatLab to get enough digits to display.

From the output we see

```
parta =   0.983790458590775
```

erf(1.7) = 0.98379046

(b) How many intervals so you need in the second-order Simpson's method to obtain this result to six significant digits?

I used the simpson2.m code and changed the integrand function to

```
function f = funkeval(x)
pi = 2.0*asin(1.0);
f = 2.0/sqrt(pi)*exp(-x^2);
```

From the output we find that

```
For 14 intervals, the integral is 0.983789 & the relative error is -1.472696e-06
For 16 intervals, the integral is 0.983790 & the relative error is -8.609076e-07
```

In order to get six good significant digits, we need the absolute value of the relative error to be less than 10^{-6}. So, we see that we need 16 intervals to reach this level of accuracy.

(c) In statistics, for nonnegative values of x, the error function has the following interpretation: for a random variable Y that is normally distributed with mean 0 and variance 1/2, erf(x) is the probability of Y falling in the range [−x, x]. Knowing this, show how can you use the Matlab cdf command to evaluate this integral?

The cdf function provides the cumulative probability distribution, or the integral from negative infinity to x. So to calculate the integral from -x to x, we take the difference of two cdf evaluations.

```
mean = 0;
variance = 0.5;
stdev = sqrt(variance);
partc = cdf('normal',x,mean,stdev) - cdf('normal',-x,mean,stdev)
```

This part of the script yields the output

```
partc =   0.983790458590775
```

This result is the same as the complementary erf(x).

Problem 2. (14 points)

A cylindrical titanium rod, of diameter, *d*, and length *L*, is horizontally suspended between two heat reservoirs, which maintain the temperature at one end (z=0) at 500 K and at the other end (z=l) at 1000 K. Between them a fan flows on the rod to conduct heat away. The steady state heat equation describing this set up is given below as

$$0 = \frac{k_c}{\rho C_p}\frac{d^2T}{dz^2} - \frac{h}{\rho C_p}\frac{A}{V}(T - T_{surr})$$

where

- k_c is the thermal conductivity, $k_c = 21.9\frac{W}{m\cdot K}$
- ρ is the mass density, $\rho = 4506.0\ \frac{kg}{m^3}$
- C_p is the specific heat capacity, $C_p = 523.5\frac{J}{kg\cdot K}$
- d is the diameter of the rod, $d = 0.05\ m$
- l is the length of the rod, $l = 0.5\ m$
- A is the surface area of the rod, $A = \pi dl$
- V is the volume of the rod, $V = \frac{\pi}{4}d^2l$
- A/V is the surface area to volume ratio of the rod, $A/V = \frac{4}{d}$
- T_{surr} is the surrounding temperature, $T_{surr} = 300K$
- h is an emprical heat transfer coefficient, $h = 40.0\frac{W}{m^2\cdot K}$

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.
(e) Sketch the temperature profile.
(f) Verify that your discretization resolution was sufficient.
(g) What is the temperature in the middle of the rod at steady state?

Solution

(a) Is this ODE problem linear or nonlinear?

The problem is linear.

(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.

$$0 = \frac{k_c}{\rho C_p}\frac{d^2T}{dz^2} - \frac{h}{\rho C_p}\frac{A}{V}(T - T_{surr})$$

or

$$\frac{k_c}{\rho C_p}\frac{d^2T}{dz^2} = \frac{h}{\rho C_p}\frac{A}{V}(T - T_{surr})$$

$$\frac{d^2T}{dz^2} = \frac{\rho C_p}{k_c}\frac{h}{\rho C_p}\frac{A}{V}(T - T_{surr}) = \frac{h}{k_c}\frac{A}{V}(T - T_{surr})$$

$$\frac{dy_2}{dz} = \frac{h}{k_c}\frac{A}{V}(y_1 - T_{surr})$$

Step 3. Write initial conditions for the new variables.

$$y_1(z=0) = T(z=0) = 400 \quad y_2(z=0) = \left.\frac{dT}{dz}\right|_0 \text{ (not given)}$$

(d) Find the initial temperature gradient at *z* = 0.
(e) Sketch the concentration profile.

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

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

```
function dydx = funkeval(x,y);
h = 40.0; % W/m^2/K
kc = 21.9; % W/m/K
d = 0.05; % m
AoV = 4.0/d; % 1/m
Tsurr = 300.0; % K
con = h/kc*AoV; % 1/m^2
dydx(1) = y(2);
dydx(2) = con*(y(1)-Tsurr);
```

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

```
function f = funkeval(x)
xo = 0.0;
yo_1 = 500.0;
yo_2 = x;
xf = 0.5;
yf = 1000.0;
n = 1000;
[x,y]=rk4n(n,xo,xf,[yo_1,yo_2]);
yf_calc = y(n+1,1);
f = yf_calc-yf;
```

At the command line prompt, I needed an initial guess for the initial slope. I used the average slope as my initial guess.

```
>> average_slope = (1000 - 500)/0.5
average_slope =   1000

>> [x0,err] = nrnd1(average_slope)
```

This command generated the following output:

```
>> [x0,err] = nrnd1_xm4f19(average_slope)
icount = 1 xold = 1.000000e+03 f = 5.889433e+04 df = 1.743738e+01 xnew = -2.377475e+03  err = 1.000000e+02
icount = 2 xold = -2.377475e+03 f = -3.467352e-06 df = 1.743738e+01 xnew = -2.377475e+03  err = 8.363747e-11

x0 =    -2.377475108168817e+03
err =     8.363747287904395e-11
```

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

$$\left.\frac{dT}{dz}\right|_0 = -2377.4751 \frac{K}{m}$$

We can run the Runge Kutta code with this initial condition to generate the result.

```
>> [x,y]=rk4n(1000,0,0.5,[500,-2377.4751]);
```

```
>> >> y_n1000 = y(1001,1)

y_n1000 =     1.000000142442697e+03
```

When we use 1000 intervals in the Runge Kutta code, we obtain a temperature at the far end of the rod of 1000.0 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

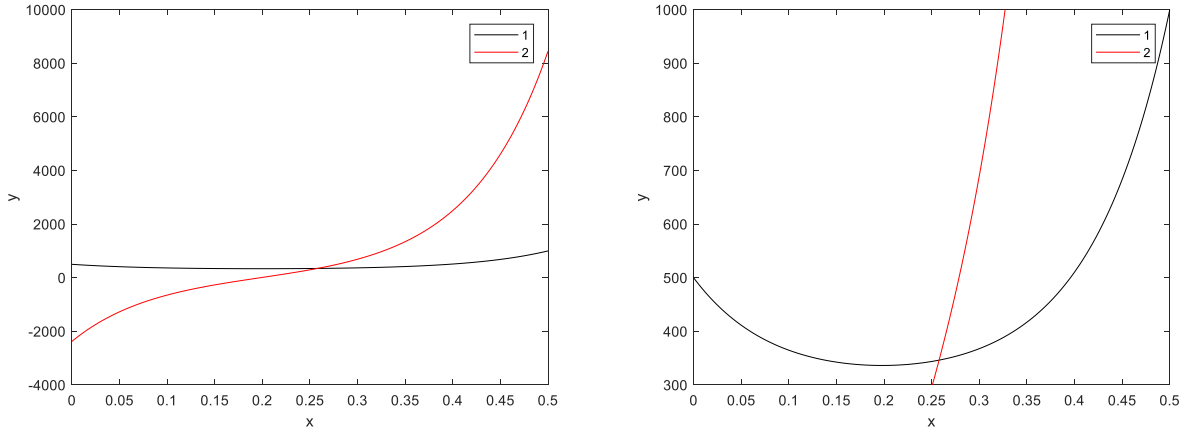
```
>> >> [x,y]=rk4n(10000,0,0.5,[500,-2377.4751]);
>> y_n10000 = y(10001,1)

y_n10000 =     1.000000142489317e+03
```

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.

A second plot is also shown which zooms in on the temperature. It is clear that there is sufficient cooling that the temperature actually drops well below 500 near the lower temperature reservoir.



(g) What is the temperature in the middle of the rod at steady state.

```
>> [x,y]=rk4n_xm4f19(1000,0,0.5,[500,-2377.4751]);
>> x(501)
ans =   0.250000000000000
>> y(501,1)
ans =     3.437300803429285e+02
```

The temperature in the middle of the rod (x = 0.25 m) is 343.73 K.

Problem 3. (8 points)

Consider the isomerization reaction:

$$A \rightarrow B$$

The reaction rate is given by

$$rate = C_A k_o e^{-\frac{E_a}{RT}} \qquad [\text{moles/m}^3\text{/sec}]$$

where

concentration of A: C_A [moles/m^3]
prefactor: k_o [1/sec]
activation energy for reaction: E_a [Joules/mole]
constant: $R = 8.314$ [Joules/mole/K]
temperature: T [K]

The reaction is measured at a constant concentration of A, C_A = 1000 mol/m^3, over a range of temperatures. The rate is recorded. The rate as a function of temperature is given in tabular form in the file "xm4p03_f19.txt" on the exam portion of the course website.

(a) Linearize this equation in the unknown reaction parameters.
(b) Perform a linear regression and report the measure of fit.
(c) Determine the rate constants, k_o and E_a, from experimental data.

Solution:

(a) Linearize this equation in the unknown reaction parameters.

Convert the data into a linear form necessary for a linear regression.

$$\ln(rate) - \ln(C_A) = -\frac{E_a}{RT} + \ln(k_o)$$

This equation is of the form: $y = b_1 x + b_0$ where

$$y = \ln(rate) - \ln(C_A),\ b_1 = E_a,\ x = -\frac{1}{RT},\ \text{and}\ b_0 = \ln(k_o).$$

I used the code linreg1.m for linear regression with one independent variable.

(b) Perform a linear regression and report the measure of fit.
(c) Determine the rate constants, k_o and E_a, from experimental data.

I wrote the small script xm4p03_f19.m

```
clear all;

%Temperature    rate of A loss
%K              mole/m^3/s

M = [500    0.000402223
525 0.001902014
550 0.005855487
575 0.020039784
600 0.053936753];

R = 8.314; % J/mol/K
CA = 1000;0; % mol/m^3
n = max(size(M));
for i = 1:1:n
    x(i) = -1.0/(R*M(i,1));
    y(i) = log(M(i,2)) - log(CA);
end
[b,bsd,MOF] = linreg1(x, y)
Ea = b(2)
ko = exp(b(1))
```

At the command line prompt, I executed the script

```
>> xm4p03_f19
```

This generated the following output for the means, standard deviations and Measure of Fit.

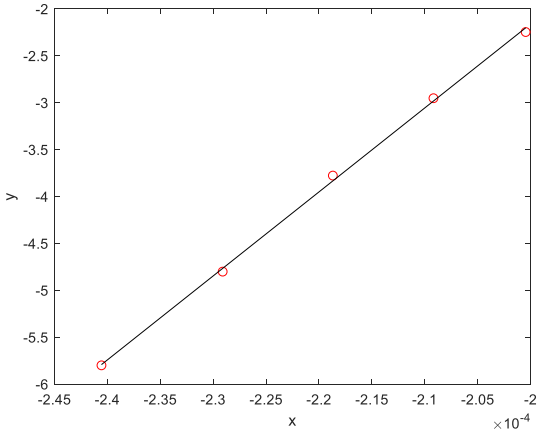
```
b =   1.0e+04 *
   0.001575019663364
   8.955208938846178

bsd =   1.0e+03 *
   0.000344626777448
   1.566083424270229

MOF =   0.999083354695982

Ea =     8.955208938846178e+04

ko =     6.921870770633748e+06
```



The MOF is 0.999.
The activation energy was 89,552 J/mol.
The rate constant was 6.922×10^6 1/sec.

The code also generated a plot.