10.34 – Fall 2006

Homework #1

Due Date: Wednesday, Sept. 13th, 2006 – 9 AM

Problem 1: Bessel Functions

Bessel functions are commonly encountered in heat and mass transfer problem, where the geometry is cylindrical. Bessel functions of the first kind ($J_\nu(x)$) and second kind ($Y_\nu(x)$) are the two linearly independent solutions of the differential equation

$$x^2\frac{d^2y}{dx^2}+x\frac{dy}{dx}+(x^2-\nu^2)y=0$$

The solution of the equation is exactly $J_\nu(x)$ for boundary condition $y(0)=1$ and $\frac{dy}{dx}(0)=0$.

This second order differential equation can be converted into a system of two coupled first order differential equations by defining new variable u_1 and u_2 as follows:

$$u_1=y \qquad u_2=\frac{dy}{dx}$$

The two differential equations thus obtained are

$$\frac{du_1}{dx}=u_2$$

$$\frac{du_2}{dx}=-\frac{u_2}{x}-\left(1-\frac{\nu^2}{x^2}\right)u_1 \qquad (1)$$

A matlab code is presented below which solves the above problem with boundary conditions.

$u_1(0)=1$ and $u_2(0)=0$

(Notice that at x =0, $\frac{du_2}{dx}$ in equation 1 becomes singular. To write the matlab code we have used the property of Bessel functions $J'_\nu(0)=0.5$, for $\nu\geq 1$).

```
%////////////////////////////////////////////////////////////
% problem 1, HW set 1
% Solution of bessel's equation function using ode23s.

function plot_bessel_using_ode(x_end)

%u_init is the intial condition at t=0
u_init = [1 0];

%solve the differential equation using ode23s to generate vectors for x
and
%u.

[x,u]=ode23s(@diff,[0 x_end],u_init,[]);

plot(x,u(:,1));
title('Bessel function of first kind');
xlabel('x');
ylabel('y(x)');
return;
%////////////////////////////////////////////////////////////


%////////////////////////////////////////////////////////////
%function diff evaluates the derivatives of u1 ad u2.
function f = diff(x,u)

f = zeros(size(u));

f(1) = u(2);
if (x ==0)
    f(2)=-0.5;
else
    f(2) = -u(2)/x - u(1);
end

return;
%////////////////////////////////////////////////////////////
```

1. Copy the matlab code and run it to generate the plot of Bessel functions. [**Hint:** Make sure that the file is saved in file name plot_bessel_usinig_ode.m, although this is not essential, it is a good practice to name file names the same as the function they contain. Also make sure that the matlab present directory is the same as the directory which contains the file plot_bessel_using_ode.m. Type the command plot_bessel_using_ode on the matlab prompt.]

2. The next task is to plot the function $J_0(x)$ using the built-in matlab command `besselj`. First make a vector of x using the command `linspace,` to generate 40 equally spaced values between 0 and 10. Then use a `for` loop to iterate over all

the values of *x* to calculate $J_0(x)$ using the command `besselj`. [Look at matlab help to figure out how to use `besselj` and `linspace` commands].

3. If you looked at the help for `besselj` you will realize that value of $J_0(x)$ can be calculated using the infinite series:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

The good news for Bessel functions is that each consecutive term of the Bessel Series quickly approaches 0. Write a fuction `my_bessel(nu,x)` which takes in a value of ν and *x* respectively and returns the value of $J_\nu(x)$ and also the number of terms in the series used to calculate it. Then plot the value of $J_\nu(x)$ for *x* = [0,10] and compare the graph to the ones that you got in part 1 and part 2. Also plot the number of terms *n* required to achieve a converged value of $J_\nu(x)$ against *x*.

[**HINT:**
Let,
$t_k = \frac{(-1)^k (x/2)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$, then

$J_\nu(x) = \sum_{k=0}^{\infty} t_k$, but for our purposes we will approximate $J_\nu(x)$ with the expression,

$$J_\nu(x) \approx \sum_{k=0}^{n} t_k ,$$

where n is sufficiently big. We will say that the value of $J_\nu(x)$ has converged when:

$$\left| \frac{t_{n+1}}{\sum_{k=0}^{n} t_k} \right| \leq 0.001.$$

You might want to use the `while` loop in matlab to check when the *(n+1)*th term becomes less than 0.1% of the overall sum.]

Problem 2 – Equation Solvers

Cubic equations of state are often good at describing vapor and liquid behavior, but it is typically difficult to find the volume roots analytically. Use the *fzero*, *fsolve*, and *fminsearch* Matlab functions to determine the *vapor* volume root of the following equation of state for water (Redlich/Kwong) at P = 1 atm and T = 300 K.

$$P = \frac{RT}{V-b} - \frac{a(T)}{(V+\varepsilon b)(V+\sigma b)}$$

with:

$$a(T) = \psi \frac{\alpha(T_r)(RT_c)^2}{P_c} \qquad \textit{and} \qquad b = \frac{RT_c}{P_c}$$

where:

$$\alpha(T_r) = T_r^{-1/2} \qquad \sigma = 1 \qquad \varepsilon = 0 \qquad \psi = 0.42748 \qquad \Omega = 0.08664$$

for water:

$$T_c = 647.1\ K \qquad P_c = 217.7\ atm$$

Compare the performance of these solvers over a range of initial volume guesses from 1 to 50 L/mole. Plot the values returned for each solver as a function of the initial guess. Do some solvers appear to be more robust than others?

Some useful Matlab tools for this problem:

```
fzero, fsolve, fminsearch, global, linspace, plot, function
```

Problem 3 – Reading and Writing Files

In this problem you will have to use some of the I/O utilities in Matlab that allow one to read and write text and Excel files. Write a Matlab script that reads the text and excel files posted on the website (*p2_text.txt* and *p2_excel.xls*) containing the modified Arrhenius parameters of a reaction and plots (y-axis log scale) the rate constant over the temperature range from 300 K to 1500 K. Also, print the rate constant in intervals of 100 K to both a text file and an Excel file in the following form:

Rate constant data for:
NH2OH + HO2 --> NH2O. + HOOH

Temp (K)	**k(T) (cm^3/mole-s)**
300	**XXX**
…	**…**
1500	**YYY**

Some useful Matlab tools for this problem:

`textscan, xlsread, fprintf, xlswrite, char, fopen, fclose, length, strmatch, for, plot`