BIEN/CENG 2310
MODELING FOR CHEMICAL AND BIOLOGICAL ENGINEERING

HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY, FALL 2022

HOMEWORK #5 (DUE OCT. 29, 2022)

1. Consider the no-propulsion rocket example presented in the lecture video:

$$\frac{d^2y}{dt^2} = -\frac{gR^2}{(y+R)^2} - \frac{D}{m}\left(\frac{dy}{dt}\right)^2 \operatorname{sgn}\left(\frac{dy}{dt}\right); \quad y(t=0) = 0 \quad ; \quad \left.\frac{dy}{dt}\right|_{t=0} = v_0$$

where $y(t)$ is the altitude of the rocket at time t, D is the drag coefficient, m is the mass of the rocket, v_0 is the initial speed, and g is the acceleration due to gravity at the Earth's surface, $g = 9.8\ \text{ms}^{-2}$. The radius of the Earth at the launch site, R, is fixed at 6×10^6 m. The mass of the rocket is fixed at $m = 10{,}000$ kg. (Note that "sgn" denotes the signum function: sgn(k) is equal to +1 if k is positive, 0 if k is 0, and -1 if k is negative.)

(a) Write a MATLAB program to simulate the flight of the rocket, and plot $y(t)$ versus t. The user will specify D and v_0. The simulation should stop when the rocket hits the ground again.

(b) Suppose we do not know the initial speed v_0, but we want to specify time taken for the rocket to return to the ground, t_{ground}. Reformulate the ODE as a boundary value problem, with D and t_{ground} as input parameters. Using the shooting method, solve the boundary value problem, plot $y(t)$ versus t, and return the initial speed v_0.

(c) Solve the same boundary value problem in Part (b) by the finite difference method, and compare the solution to Part (b) for different number of nodes. A template `rocketFiniteDifference.m` is provided.

DELIVERABLES:

Parts (a) and (b) will be done in class.

Submit the program, renamed "`rocketFiniteDifference_<LastName>_<FirstName>.m`" for Part (c). Also, submit two plots, one using the shooting method and one using the finite difference method, for the parameter settings $D = 10\ \text{kg m}^{-1}$, $t_{ground} = 14.\text{yy}$ s, where yy is the last 2 digits of your student ID.

Put your full name and student ID in the title by the commands (using me as an example):

```
title('Shooting method by LAM, Henry 235587133');
title('Finite difference method by LAM, Henry 23587133');
```

2. In this problem we will write a MATLAB program to model the motion of charged objects held together by springs in the xy-plane:

At any moment, the coordinate of the k-th object is (x_k, y_k), and its instantaneous velocity is $\vec{v}_k$, which can be written as a vector:

$$\vec{v}_k = \left(\frac{dx_k}{dt}\right)\vec{\imath} + \left(\frac{dy_k}{dt}\right)\vec{\jmath}$$

Similarly, the acceleration can be written as:

$$\vec{a}_k = \left(\frac{d^2x_k}{dt^2}\right)\vec{\imath} + \left(\frac{d^2y_k}{dt^2}\right)\vec{\jmath}$$

Consider the case of only two objects, 1 and 2, first. The objects are similarly charged, so they repel each other. The forces experienced by each object due to the electrostatic interaction is given by:

$$\vec{F}_{e,1} = -q\left[\left(\frac{x_2 - x_1}{r^3}\right)\vec{\imath} + \left(\frac{y_2 - y_1}{r^3}\right)\vec{\jmath}\right]$$

$$\vec{F}_{e,2} = -q\left[\left(\frac{x_1 - x_2}{r^3}\right)\vec{\imath} + \left(\frac{y_1 - y_2}{r^3}\right)\vec{\jmath}\right]$$

where q is a constant and r is the distance between the two objects, $r = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. The force experienced by each object due to the spring is given by:

$$\vec{F}_{s,1} = \alpha[(x_2 - x_1)\vec{\imath} + (y_2 - y_1)\vec{\jmath}]$$

$$\vec{F}_{s,2} = \alpha[(x_1 - x_2)\vec{\imath} + (y_1 - y_2)\vec{\jmath}]$$

where α is a constant. Finally, there is a drag force which is proportional to the velocity:

$$\vec{F}_{d,1} = -\delta\left[\left(\frac{dx_1}{dt}\right)\vec{\imath} + \left(\frac{dy_1}{dt}\right)\vec{\jmath}\right]$$

$$\vec{F}_{d,2} = -\delta\left[\left(\frac{dx_2}{dt}\right)\vec{\imath} + \left(\frac{dy_2}{dt}\right)\vec{\jmath}\right]$$

All objects have the same mass, m. Initially, the objects are stationary.

(a) Write down the system of second-order ODEs to model the motion of the two objects. Write a MATLAB program to simulate the motion of the two objects, and show a 3-D plot of trajectory versus time, and a movie of how the objects move in the xy-plane over time. The user will specify the initial coordinates of the objects. The simulation should stop when the objects have come to a stop. The function definition should be:

```
function [] = forceDirectedGraph_<LastName>_<FirstName>(xinit, yinit)
```

where **`xinit`** is a column vector of the initial x-coordinates of the objects, and **`yinit`** is a column vector of the initial y-coordinates. Use the parameters $m = 1$, $q = 0.2$, $\alpha = 0.1$, $\delta = 0.5$.

(b) (*Warning: Tricky programming, but a lot of fun if it works.*) Generalize your program to simulate the motion of n objects, where every object is connected to every other object by a spring and experiences electrostatic repulsion from every other object.

Hint 1: To help compute the inter-object distances, use the `meshgrid` *function:*

```
[XX1, XX2] = meshgrid(x, x);
Xdist = XX2 - XX1;
```

Here, `x` *is a column vector of the* x_i*'s, and* `Xdist` will be an $n \times n$ matrix whose (i, j) element is equal to $x_j - x_i$. *Try to understand how this works. Also, check out the* `sum` *function, which can be used to compute column sums of a matrix.*

Hint 2: Watch out for the case when two objects are at the exact same location, which will result in a 0/0 when calculating the electrostatic force. To avoid this, add a small constant to all elements of your distance matrix.

DELIVERABLES:

Part (a) will be done in class.

Submit your MATLAB programs in Part (b)-- remember to follow ***exactly the function definition as specified***.

Test run your program with 7 objects starting at randomized locations, and submit a 3-D plot of the objects' trajectories versus time.

3. A textile factory located at the source of a river releases a dye pollutant continuously into the river. Fortunately, the dye is slowly degraded by bacteria in the river. Assuming the river flows in the $+x$ direction, and we can ignore any concentration variations in the y and z directions. At steady state, the concentration profile $C(x)$ can be described by the following ODE:

$$D\left(\frac{d^2C}{dx^2}\right) - v\left(\frac{dC}{dx}\right) - kC = 0$$

with the boundary conditions:

$$C(x = 0) = C_0$$

$$\left.\frac{\partial C}{\partial x}\right|_{x=L} = 0$$

where C_0 is the concentration of the dye at the source, D is the diffusion coefficient of the dye in water, v is the speed of the river flow, k is the rate constant for bacterial degradation. L is chosen to be large enough that at a faraway point $x = L$ the dye can be assumed to be non-existent. For this problem, setting $L = 10\ \max(\sqrt{D/k}, v/k)$ seems to work reasonably well.

Solve the boundary value problem with MATLAB. Your program should allow the user to specify D, v, k, and C_0, and plot C versus x. You may use the shooting method or the finite difference method. Your function definition should be:

`function [] = dyeRiver_<LastName>_<FirstName>(D, v, k, C0)`

To test your program, select all parameters to be between 0 and 1, and keep $v < D$.

DELIVERABLES:

Submit your MATLAB program `dyeRiver_<LastName>_<FirstName>.m`.

No need to submit write-up or plots. If you have time, try both methods to check answers against each other, as a practice.

4. (*Optional*) In the lecture video of Module 7, we modeled heat transfer within a cylindrical metal rod by dividing it into 3 pieces along its length, each assumed to have uniform temperature within it. We also learned about the "no flux" boundary condition, which specifies that the temperature gradients at the two ends of the rods are zero. This leads to the following system of ODEs:

$$\frac{dT_1}{dt} = \frac{k}{\rho C}\left(\frac{-T_1 + T_2}{l^2}\right)$$

$$\frac{dT_i}{dt} = \frac{k}{\rho C}\left(\frac{T_{i-1} - 2T_i + T_{i+1}}{l^2}\right) \quad for \;\; i = 2, 3, 4, \ldots, n-1$$

$$\frac{dT_n}{dt} = \frac{k}{\rho C}\left(\frac{T_{n-1} - T_n}{l^2}\right)$$

where k is the thermal conductivity of the metal, ρ is the mass density, C is the heat capacity of the metal, and $l = L/n$ is the length of each piece.

(a) As mentioned, the solution will be more accurate if we divide into smaller pieces. Write a MATLAB program that can simulate the same problem for n pieces, which can have different initial temperatures. The user will specify the initial temperatures of all the pieces (counting from the left end to the right end) in a vector of n elements, the thermal diffusivity $\alpha = k/(\rho C)$, and the total length of the rod L. Assume that all pieces have the same length and mass, the thermal diffusivity is uniform everywhere, and there is no heat exchange with the surroundings. Apply the "no flux" boundary conditions on the two ends. Stop the simulation when the temperature of all pieces becomes steady, and plot the temperature versus time profiles of all n pieces in one plot. Your function should have the following definition:

```
function [] = heatRod_<LastName>_<FirstName>(alpha, L, Tinit)
```

Your program should expect that `Tinit` is a column vector of n elements, each element containing the initial temperature of one piece.

Note: You may realize that by posing the problem this way, our MATLAB program is exactly the same as if we try to solve the PDE with the "method of lines" using n nodes:

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C}\left(\frac{\partial^2 T}{\partial x^2}\right)$$

(b) Suppose we now put a heat source on the left end of the rod, such that there is a constant heat flux, F, going into the rod. At steady state, the same amount of heat would flow out of the rod per unit time. Modify the program to account for the following two possibilities. You may still assume that temperature only varies along the length of the rod, and not radially.

(i) The curved surface along the length of the rod is insulated, such that the only heat loss will be on the right end of the rod, with heat flux equal to F. Your function definition should be:

```
function [] = heatRodAxialFlow_<LastName>_<FirstName>(alpha, L,
Tinit, phi)
```

where `phi` is a combined parameter, $\phi = F/(\rho C)$. You should see this combination emerge when you write the energy balance equation.

(ii) There is heat loss from the curved surface along the rod, with the heat flux, q_{env}, modeled by Newton's Law of cooling:

$$q_{env} = h(T - T_{env})$$

where h is the heat transfer coefficient (a constant), T is the temperature of the rod at that location, and T_{env} is the temperature of the environment. Because the rod is long, we can assume that most heat is lost radially in this way, and that the "no flux" condition still applies for the right end of the rod. Your function definition should be:

```
function [] = heatRodRadialFlow_<LastName>_<FirstName>(alpha, L,
Tinit, phi, Tenv, beta)
```

where `beta` is a combined parameter, $\beta = 2h/(kR)$; R is the radius of the rod. You should see this combination emerge when you write the energy balance equation.

Note: This model can be used to design cooling fins, such as the ones used to keep computer chips from overheating.

DELIVERABLES:

This problem will be done in class as a demo, if time allows.