Homework No.5

Osamu Katagiri-Tanaka : A01212611

September 22, 2020

1 Part A : Vector Field

Matlab's *quiver(X, Y, U, V)* function plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y. When implementing *quiver*, the first arrow originates from the point $(X(1), Y(1))$, extends horizontally according to $U(1)$, and extends vertically according to $V(1)$. This function scales the arrow lengths so that they do not overlap.

On the other hand, Matlab's *contour(X,Y,Z)* function creates a contour plot containing the isolines of a matrix Z, where Z contains height values on the $x - y$ plane. *contour* automatically selects the contour lines to display. X and Y are the x and y coordinates in the plane, respectively.

Listing 1 implements functions *quiver* and *contour* to visualize the velocity field and pressure lines given a vector field. Figure 1 is the result.

```matlab
%% HW05 part A - Velocity Field, adapted from (jose lopez salinas)'s solution
clear;
close all;

% create points to visualize
xyLim  = 2.5;
xyStep = xyLim/10;
[x, y] = meshgrid(-xyLim : xyStep : xyLim);

VectorX = cos(y); % vector in the x direction
VectorY = sin(x); % vector in the y direction

V = sqrt(VectorX.^2 + VectorY.^2);
PHI = 6 + x.^3 / 3 - y.^2 .* x - y;
[Dx, Dy] = gradient(V, 0.2, 0.2);

% Display
figure;
quiver(x, y, VectorX, VectorY);
hold on;
contour(x, y, PHI);
colorbar;
hold off;
xlabel('x-axis');
ylabel('y-axis');
title('Velocity Field, and Pressure Lines');
```

Listing 1: Vector Field Visualization

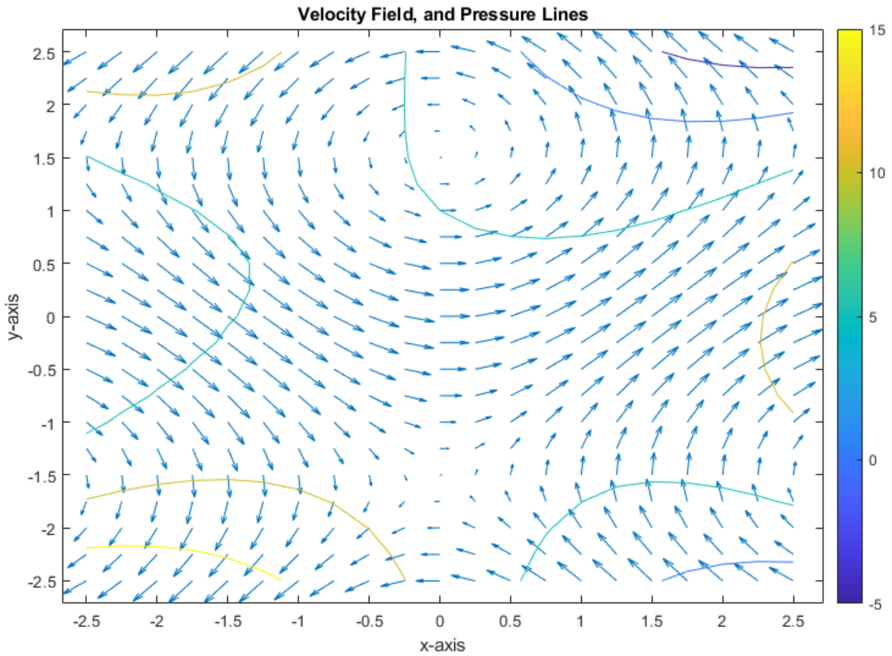

Figure 1: Visualization of a Velocity Field

2 Part B : 1-D PDE Tubular Chemical Reactor

The equation of conservation of chemical species under a chemical reaction of decomposition can be represented with the PDE given below.

$$\frac{\partial C}{\partial t} = \vec{\nabla} \cdot (D\vec{\nabla}C) - \vec{v} \cdot \vec{\nabla}C - kC^n$$

If a tubular catalytic chemical reactor initially filled with an inert solvent ($C = 0$) is fed by a stream of component "A" with a concentration of $1kmol/m^3$ ($C = 1$) and speed of $1m/s$ ($v = 1$), calculate the distribution of "A" across the reactor and as a function of time $C(x, t)$. The dispersion coefficient of the component "A" is $0.02m^2/s$ ($D = 0.001$), the kinetic decomposition coefficient $0.05s^{-1}$ ($k = 1.5$). The chemical decomposition kinetics is first order ($n = 1$).

The molar balance in axial direction for a 1D flow can be written as:

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} - v\frac{\partial C}{\partial x} - kC^n$$

The initial condition IC is:

$$C|_{t=0} = 0, \ 0 \leq x \leq 1$$

The boundary conditions BCs are:

$$C|_{x=0} = 1, \ t > 0$$

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

Where,
D is the diffusion coefficient
C is the injection concentration
v is the velocity of fluid injection
k is the first order kinetic coefficient

L is the length of domain
t is the simulation time
x is the distance mesh

The PDE shall be transformed into a set of ordinary differential equations ODEs using central finite differences in space, as depicted in the central differences discretization Equation 1.

$$\frac{\mathrm{d}C_i}{\mathrm{d}t} = D\frac{C_{i+1} - 2C_i + C_{i-1}}{(\Delta x)^2} - v\frac{C_{i+1} - C_{i-1}}{2\Delta x} - k{C_i}^n \tag{1}$$

Equations 2, 3 and 4 are the backward differences discretization for $O(h^2)$, $O(h^3)$ and $O(h^4)$, respectively. Listing 2 solves the ODEs with truncation errors of $O(h^2)$, $O(h^3)$ and $O(h^4)$. *ode45* results are displayed in Figures 2, 3 and 4 for which no noticeable differences can be depicted. The results from the three truncation errors as similar, as the difference between each one is around 2 percent. See Listings 4 and 5.

$$\begin{aligned} \frac{\mathrm{d}C_N}{\mathrm{d}x} = \frac{C_{N-2} - 4C_{N-1} + 3C_N}{2\Delta x} &= 0 \\ C_N = \frac{4C_{N-1} - C_{N-2}}{3}& \end{aligned} \tag{2}$$

$$\begin{aligned} \frac{\mathrm{d}^2C_N}{\mathrm{d}x^2} = \frac{-C_{N-3} + 4C_{N-2} - 5C_{N-1} + 2C_N}{(\Delta x)^2} &= 0 \\ C_N = \frac{C_{N-3} - 4C_{N-2} + 5C_{N-1}}{2}& \end{aligned} \tag{3}$$

$$\begin{aligned} \frac{\mathrm{d}^3C_N}{\mathrm{d}x^3} = \frac{3C_{N-4} - 14C_{N-3} + 24C_{N-2} - 18C_{N-1} + 5C_N}{2\,(\Delta x)^3} &= 0 \\ C_N = \frac{-3C_{N-4} + 14C_{N-3} - 24C_{N-2} + 18C_{N-1}}{5}& \end{aligned} \tag{4}$$

```
%% Runge-Kutta
p(1)  = 0.001; % Diffusion coefficient D
p(2)  = 1.0;   % Injection concentration c0
p(3)  = 1.5;   % First order kinetic coefficient k
p(4)  = 1.0;   % Velocity of fluid injection vo
M     = 2*640; % Number of nodes
p(5)  = M;
Tspan = [0 1]; % Domain of time
xi    = linspace(0, 1, M);

% Initial conditions of the resulting set of ODEs
Y0    = zeros(M, 1);
Y0(1) = 1.0;

% Solve differential equation (medium order method)
% use @reactub_2 for O(h^2) truncation error
% use @reactub_3 for O(h^3) truncation error
% use @reactub_4 for O(h^4) truncation error
OPTIONS    = [];
[time_2, Y_2] = ode45(@reactub_2, Tspan, Y0, OPTIONS, p);
[time_3, Y_3] = ode45(@reactub_3, Tspan, Y0, OPTIONS, p);
[time_4, Y_4] = ode45(@reactub_4, Tspan, Y0, OPTIONS, p);

% group all data / prepare to plot ...
time      = {time_2, time_3, time_4};
```

```
Y         = {    Y_2,     Y_3,     Y_4};
Yprime    = {   Y_2',    Y_3',    Y_4'};
plotName = {'Oh2_truncationError', 'Oh3_truncationError', '
    Oh4_truncationError'};
```

Listing 2: Reactor : Runge-Kutta ODE solver

```
% Plot limits
noOf_curvesToPlot = 10;
dlim              = 0.02;
time_lim          = [0 - dlim, 1 + dlim];
Y_lim             = [0 - dlim, 1 + dlim];
xi_lim            = [0 - dlim, 1 + dlim];
Yprime_lim        = [0 - dlim, 1 + dlim];

for plotCount = 1:1:3
    % Display concentration vs. time
    totalNoOf_curves  = size(Y{1, plotCount}, 2);
    noOf_curvesToSkip = fix(totalNoOf_curves/noOf_curvesToPlot);
    figure;
    subplot(1, 2, 1)
    for n = linspace(1, totalNoOf_curves, totalNoOf_curves)
        hold all
        if mod(n, noOf_curvesToSkip) == 0
            plot(time{1, plotCount}, Y{1, plotCount}(:, n));
        end
    end
    xlabel('time \tau');
    ylabel('Concentration mol/dm^3');
    axis([time_lim(1) time_lim(2) Y_lim(1) Y_lim(2)])

    % Display concentration vs. distance
    totalNoOf_curves  = size(Yprime{1, plotCount}, 2);
    noOf_curvesToSkip = fix(totalNoOf_curves/noOf_curvesToPlot);
    %figure;
    subplot(1, 2, 2)
    for n = linspace(1, totalNoOf_curves, totalNoOf_curves)
        hold all
        if mod(n, noOf_curvesToSkip) == 0
            plot(xi, Yprime{1, plotCount}(:, n));
        end
    end
    xlabel('distance x/L');
    ylabel('Concentration mol/dm^3');
    axis([xi_lim(1) xi_lim(2) Yprime_lim(1) Yprime_lim(2)])
end
```

Listing 3: Reactor : Plot the solutions

```
%% Print error between O(h)s
sprintf(                                                        ...
    '(O(h^2) - O(h^3))/O(h^2)*100 error: %f%%',                 ...
    (Yprime{1, 1}(end) - Yprime{1, 2}(end))/Yprime{1, 1}(end)*100 ...
)
sprintf(                                                        ...
```

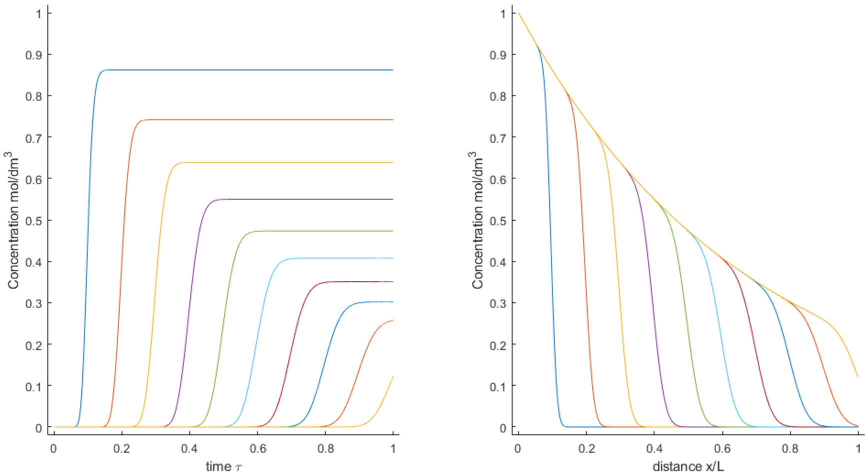

Figure 2: $O(h^2)$ Truncation Error

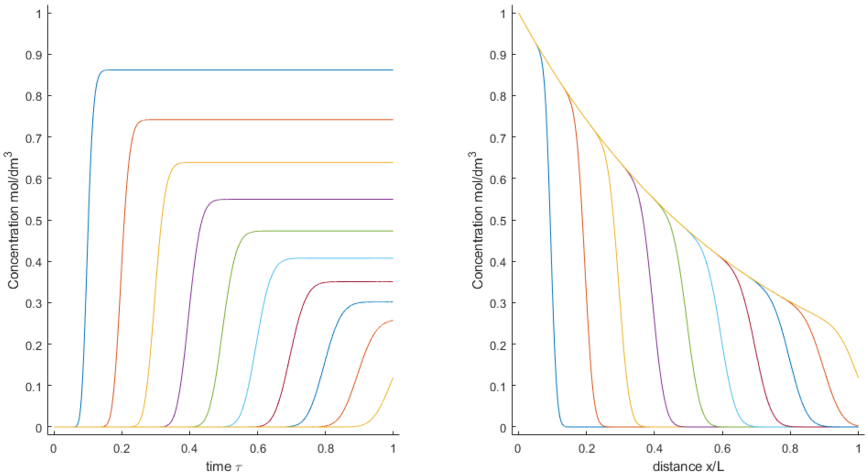

Figure 3: $O(h^3)$ Truncation Error

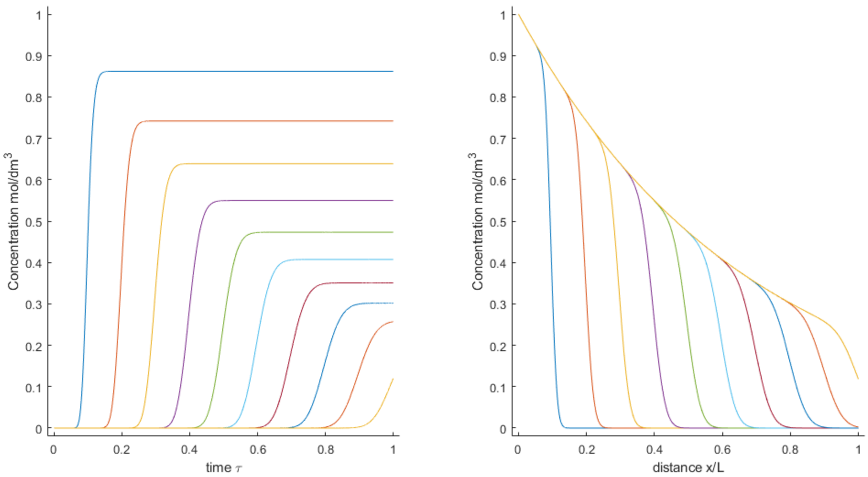

Figure 4: $O(h^4)$ Truncation Error

```
    '(O(h^2) - O(h^4))/O(h^2)*100 error: %f%%',                    ...
    (Yprime{1, 1}(end) - Yprime{1, 3}(end))/Yprime{1, 1}(end)*100 ...
)
```

```
sprintf(                                                        ...
    '(O(h^3) - O(h^4))/O(h^3)*100 error: %f%%',                 ...
    (Yprime{1, 2}(end) - Yprime{1, 3}(end))/Yprime{1, 2}(end)*100 ...
)
```

Listing 4: Reactor : Compute percent error for each implementation

```
ans =
    '(O(h^2) - O(h^3))/O(h^2)*100 error: 1.908447%'
ans =
    '(O(h^2) - O(h^4))/O(h^2)*100 error: 1.919032%'
ans =
    '(O(h^3) - O(h^4))/O(h^3)*100 error: 0.010792%'
```

Listing 5: Reactor : Percent errors for each truncation

3 Part C : Growing Bubbles

The growth and collapse of bubbles is given by Equations 5.

$$\begin{aligned} \ddot{y}y + \frac{3}{2}\left(\dot{y}\right)^2 &= -B(\tau) + \frac{1}{y^{3k}} \\ \ddot{y} &= \frac{\mathrm{d}^2 y}{\mathrm{d}\tau^2} \\ \dot{y} &= \frac{\mathrm{d}y}{\mathrm{d}\tau} \end{aligned} \tag{5}$$

Where y is the ratio of the actual radius and the initial radius of the bubble. τ is the dimensionless time: $y = \dfrac{R}{R_0}$ $\tau = \dfrac{t}{R_0\sqrt{\frac{\rho}{p_0}}}$. The initial conditions are: $y|_{\tau=0} = 1$ and $\dot{y}|_{\tau=0} = 0$

Listing 6 solves the ODEs to describe the evolution of the air bubbles in time given a pressure perturbation $B(\tau)$

$$B(\tau) = \begin{cases} \frac{1+cos\left(\pi\frac{\tau}{5}\right)}{5} & 0 \leq \tau \leq 10 \\ 1 & \text{otherwise} \end{cases}$$

```
% Runge-Kutta
% k = 1.4 adiabatic process, k = 1 isothermic
% alphaM = 0 inviscid
% betaM = 0 negligible surface tension
k      = {1.0, 1.4, 1.7};
alphaM = {0.0, 0.5, 0.1};
betaM  = {0.0, 0.2, 0.7};

% Solve and Plot
figure;
solveNplot_growingBubbles(k{1}, alphaM{1}, betaM{1}, sprintf('k = %1.1f', k
    {1}))
solveNplot_growingBubbles(k{2}, alphaM{1}, betaM{1}, sprintf('k = %1.1f', k
    {2}))
solveNplot_growingBubbles(k{3}, alphaM{1}, betaM{1}, sprintf('k = %1.1f', k
    {3}))
suptitle(sprintf('Gas Molecule Shape and Size Effect : \\alpha = %1.1f and \\
    beta = %1.1f', alphaM{1}, betaM{1}))
```

```matlab

figure;
solveNplot_growingBubbles(k{2}, alphaM{1}, betaM{1}, sprintf('\\alpha = %1.1f
    ', alphaM{1}))
solveNplot_growingBubbles(k{2}, alphaM{2}, betaM{1}, sprintf('\\alpha = %1.1f
    ', alphaM{2}))
solveNplot_growingBubbles(k{2}, alphaM{3}, betaM{1}, sprintf('\\alpha = %1.1f
    ', alphaM{3}))
suptitle(sprintf('Effect of the Viscosity : k = %1.1f and \\beta = %1.1f', k
    {2}, betaM{1}))

figure;
solveNplot_growingBubbles(k{2}, alphaM{1}, betaM{1}, sprintf('\\beta = %1.1f'
    , betaM{1}))
solveNplot_growingBubbles(k{2}, alphaM{1}, betaM{2}, sprintf('\\beta = %1.1f'
    , betaM{2}))
solveNplot_growingBubbles(k{2}, alphaM{1}, betaM{3}, sprintf('\\beta = %1.1f'
    , betaM{3}))
suptitle(sprintf('Effect of the Surface Tension : k = %1.1f and \\alpha =
    %1.1f', k{2}, alphaM{1}))

function[] = solveNplot_growingBubbles(k, alphaM, betaM, curveLabel)
    tspan = linspace(0, 35, 500);
    y1    = 1;
    y2    = 0;
    yo    = [y1, y2]';

    % Solve differential equation (medium order method)
    par(1) = k;
    par(2) = alphaM;
    par(3) = betaM;
    [t, Y] = ode45(@growingBubbles, tspan, yo, [], par);
    time   = t;
    Yout   = Y;

    %% Plot
    % Display radius vs. time
    subplot(2,1,1);
    hold all
    plot(time, Yout(:,1), 'DisplayName', curveLabel);
    xlabel('\tau');ylabel('R/Ro');
    legend

    % Display d(radius) vs. time
    subplot(2,1,2);
    hold all
    plot(time, Yout(:,2), 'DisplayName', curveLabel);
    xlabel('\tau');ylabel('d(R/Ro) /d\tau');
    legend
end
```

Listing 6: Growing Bubbles : Solve and Plot

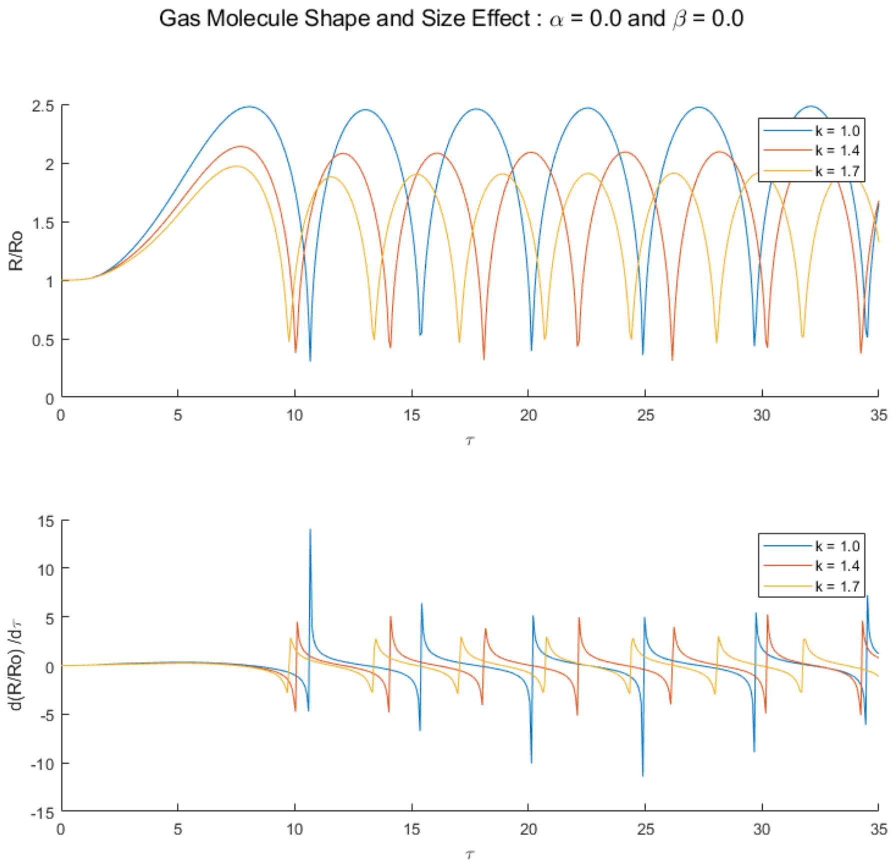

Figure 5: Gas molecule shape and size effect

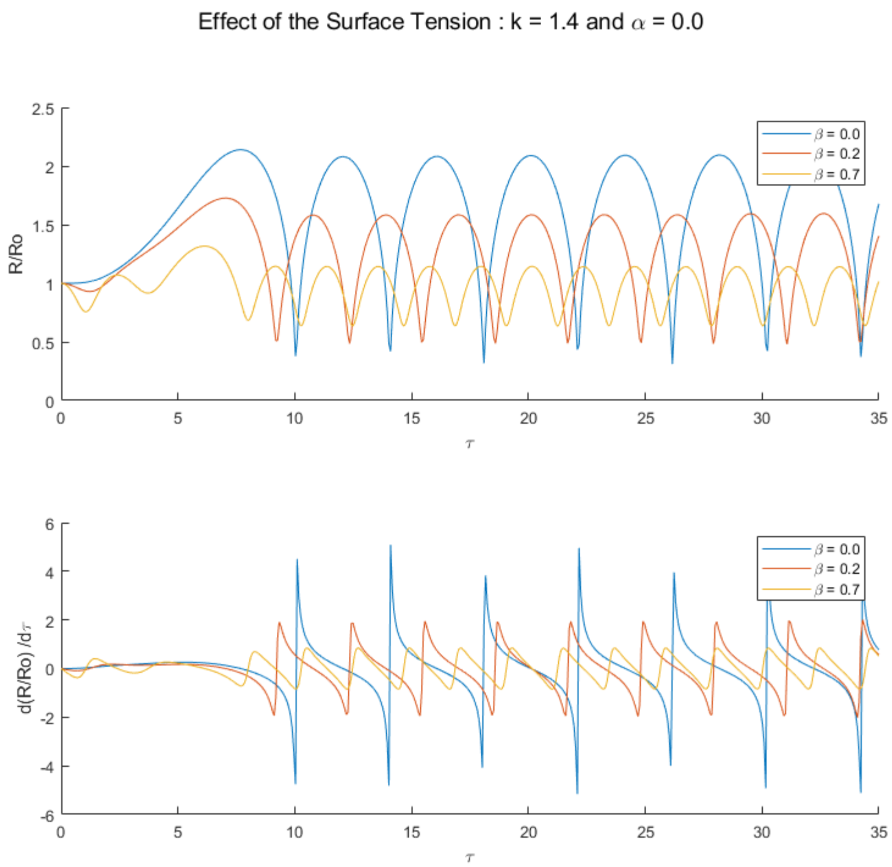

Figure 6: Effect of the surface tension

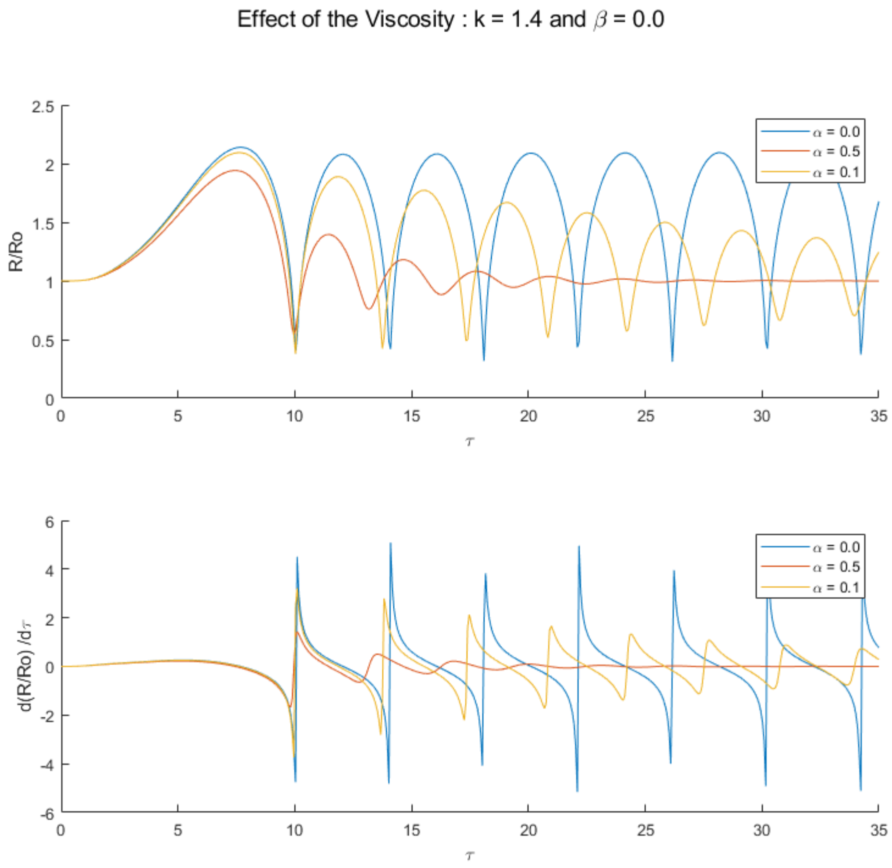

Figure 7: Effect of the viscosity

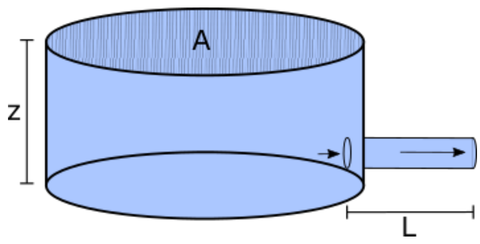

Figure 8: Effect of the viscosity

4 Part C : Draining Tank

For a tank (as in Figure 8), filled with an invisid fluid (no ciscous effect), the fluid height z can be tracked with two equations: conservation of mass $-\dot{m}$ and conservation of energy $\dfrac{\mathrm{d}(k+\Phi)}{\mathrm{d}t}$.

$$\frac{\mathrm{d}m}{\mathrm{d}t} = -\dot{m} = \frac{\mathrm{d}}{\mathrm{d}t}(\rho A z)$$

$$-\dot{m} = \rho A \frac{\mathrm{d}z}{\mathrm{d}t}$$

$$\frac{\mathrm{d}(k+\Phi)}{\mathrm{d}t} = -\dot{m}\left(\frac{1}{2}v^2 + gz\frac{p_0}{p}\right)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int\left[\frac{1}{2}\rho A v^2\,\mathrm{d}z + \frac{1}{2}\rho A_0 {v_0}^2\,\mathrm{d}L + \rho A g z\,\mathrm{d}z\right] = -\dot{m}\left(\frac{1}{2}{v_0}^2\right)$$

$$\frac{1}{2}\rho A v^2\frac{\mathrm{d}z}{\mathrm{d}t} + \frac{1}{2}\rho z A\frac{\mathrm{d}v^2}{\mathrm{d}t} + \frac{1}{2}\rho A_0 L\frac{\mathrm{d}{v_0}^2}{\mathrm{d}t} + \rho A g z\frac{\mathrm{d}z}{\mathrm{d}t} = -\dot{m}\left(\frac{1}{2}{v_0}^2\right)$$

$$\rho A\frac{\mathrm{d}z}{\mathrm{d}t}\left[\frac{1}{2}v^2 + gz + zv\frac{\mathrm{d}v}{\mathrm{d}z} + L\frac{A_0}{A}v_0\frac{\mathrm{d}v_0}{\mathrm{d}z}\right] = -\dot{m}\left(\frac{1}{2}{v_0}^2\right)$$

$$\rho A \frac{\mathrm{d}z}{\mathrm{d}t} = -\rho v_0 A_0$$

$$A \frac{\mathrm{d}z}{\mathrm{d}t} = -vA$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = -v$$

$$\frac{1}{2}v^2 + gz + zv\frac{\mathrm{d}v}{\mathrm{d}z} + L\frac{A_0}{A}v_0\frac{\mathrm{d}v_0}{\mathrm{d}z} = -\frac{1}{2}{v_0}^2$$

Therefore the equation to solve is:

$$\frac{\mathrm{d}^2 z}{\mathrm{d}t^2} = \frac{-gz + \left(\frac{\mathrm{d}z}{\mathrm{d}t}\right)^2 \frac{\lambda^4 - 1}{2}}{L\lambda^2 + z}$$

$$\lambda = \frac{D}{D_0}$$

$$\lambda^2 = \frac{A}{A_0}$$

where,

$$y_1 = z$$

$$y_2 = \frac{\mathrm{d}z}{\mathrm{d}t} = \frac{\mathrm{d}y_1}{\mathrm{d}t}$$

The ODEs to solve are:

$$\frac{\mathrm{d}y_2}{\mathrm{d}t} = \frac{-gy_1 + {y_2}^2 \frac{\lambda^4 - 1}{2}}{L\lambda^2 + y_1}$$

$$\frac{\mathrm{d}y_1}{\mathrm{d}t} = y_2$$

with the following initial conditions:

$$y_1|_{t=0} = 1\mathrm{m}$$

$$y_2|_{t=0} = 1\mathrm{m/s}$$

Listing 7 solves and plot the previous ODEs.

```
% Runge-Kutta
g      = 9.81; % gravity
lambda = 10;   % Area/Area0
L      = 2;    % length of the pipe

% Plot
figure;
solveNplot_drainingTank(g, lambda, L, '')
suptitle(sprintf('Draining Tank : g = %1.1f, A/A_{0} = %1.1f, and L = %1.1f',
    g, lambda, L))

function[] = solveNplot_drainingTank(g, lambda, L, curveLabel)
    tspan = linspace(0, 50, 500);
    y1    = 1; % initial height (1m)
    y2    = 0; % initial velocity (0m/s)
    yo    = [y1, y2]';

    % Solve differential equation (medium order method)
    par(1) = g;
```

```
    par(2) = lambda;
    par(3) = L;
    [t, Y] = ode45(@drainingTank, tspan, yo, [], par);
    time   = t;
    Yout   = Y;

    %% Plot
    % Display radius vs. time
    hold all
    plot(time, Yout(:,1), 'DisplayName', curveLabel);
    xlabel('\tau');ylabel('z/z_o');
    %legend
end
```

Listing 7: Draining Tank : Solve and Plot

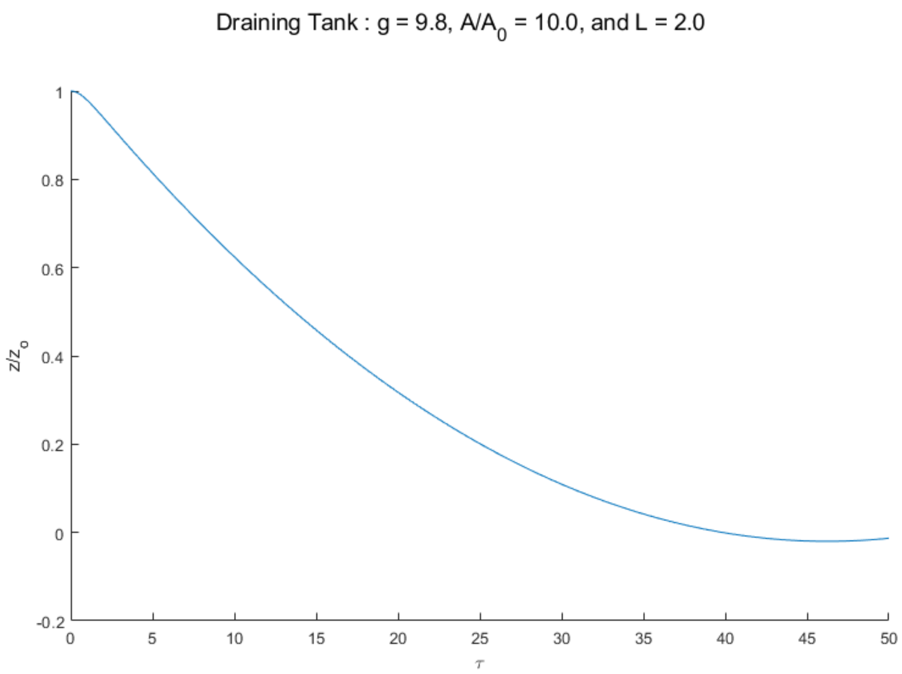

Figure 9: Draining Tank Height z

As depicted in Figure 9, the tank shall be emptied by time $\tau = 45$.

Appendix

```
function[ydot] = drainingTank(x, y, par)
    g      = par(1);
    lambda = par(2);
    L      = par(3);

    ydot(1) = y(2);
    ydot(2) = (-g*y(1) + ((y(2))^2*(lambda^4 - 1))/2) / (L*lambda^2  + y(1));
    ydot    = ydot';
end
```

Listing 8: drainingTank

```
function[ydot] = growingBubbles(x, y, par)
    k   = par(1);
    a1  = par(2);
    b1  = par(3);
    tau = x;
    B   = (1+cos(pi*tau/5))/2;
    if tau > 10
        B = 1;
    end

    ydot(1) = y(2);
    ydot(2) = (-B+1/(y(1))^(3*k)-(3/2)*y(2)^2-a1*y(2)/y(1)-b1/y(1))/y(1);
    ydot    = ydot';
end
```

Listing 9: growingBubbles

```
%% TUBULAR REACTOR adapted from (jose lopez salinas)'s solution
function yprime = reactub_2(t, y, p)
    c  = y;            % Concentration in kmol/m3 of 'A'
    D  = p(1);         % Diffusion coefficient D
    k  = p(3);         % First order kinetic coefficient
    vo = p(4);         % Velocity of fluid injection
    N  = p(5);         % Number of nodes
    m  = 1;            % Chemical decomposition kinetics is first order
    dx = 1 / (N - 1); % Step size

    % Initially filled with an inert solvent
    yprime    = zeros(1, N-1);
    yprime(1) = 0;

    % centered differences with truncation error proportional to the step
    % size to the power 2
    for i = 2 : N - 1
        sum0 = D * (c(i + 1) - 2 * c(i) + c(i - 1)) / (dx^2);
        sum1 = vo * (c(i + 1) - c(i - 1)) / (2 * dx);
        sum2 = k * c(i)^m;
        yprime(i) = sum0 - sum1 - sum2;
    end
    for i = N : -1 : 2
        yprime(N) = (4 * yprime(N - 1) - yprime(N - 2)) / 3;
```

```
    end
    yprime = yprime';
end
```

Listing 10: reactub2

```
%% TUBULAR REACTOR adapted from (jose lopez salinas)'s solution
function yprime = reactub_3(t, y, p)
    c  = y;             % Concentration in kmol/m3 of 'A'
    D  = p(1);          % Diffusion coefficient D
    k  = p(3);          % First order kinetic coefficient
    vo = p(4);          % Velocity of fluid injection
    N  = p(5);          % Number of nodes
    m  = 1;             % Chemical decomposition kinetics is first order
    dx = 1 / (N - 1); % Step size

    % Initially filled with an inert solvent
    yprime    = zeros(1, N-1);
    yprime(1) = 0;

    % centered differences with truncation error proportional to the step
    % size to the power 2
    for i = 2 : N - 1
        sum0 = D * (c(i + 1) - 2*c(i) + c(i - 1)) / (dx^2);
        sum1 = vo * (c(i + 1) - c(i - 1)) / (2*dx);
        sum2 = k * c(i)^m;
        yprime(i) = sum0 - sum1 - sum2;
    end

    for i = N : -1 : 3
        yprime(N) = (yprime(N - 3) - 4*yprime(N - 2) + 5*yprime(N - 1)) / 2;
    end
    yprime = yprime';
end
```

Listing 11: reactub3

```
%% TUBULAR REACTOR adapted from (jose lopez salinas)'s solution
function yprime = reactub_4(t, y, p)
    c  = y;             % Concentration in kmol/m3 of 'A'
    D  = p(1);          % Diffusion coefficient D
    k  = p(3);          % First order kinetic coefficient
    vo = p(4);          % Velocity of fluid injection
    N  = p(5);          % Number of nodes
    m  = 1;             % Chemical decomposition kinetics is first order
    dx = 1 / (N - 1); % Step size

    % Initially filled with an inert solvent
    yprime    = zeros(1, N-1);
    yprime(1) = 0;

    % centered differences with truncation error proportional to the step
    % size to the power 2
    for i = 2 : N - 1
        sum0 = D * (c(i + 1) - 2*c(i) + c(i - 1)) / (dx^2);
```

```
        sum1 = vo * (c(i + 1) - c(i - 1)) / (2*dx);
        sum2 = k * c(i)^m;
        yprime(i) = sum0 - sum1 - sum2;
    end

    for i = N : -1 : 4
        yprime(N) = (-3*yprime(N - 4) + 14*yprime(N - 3) - 24*yprime(N - 2) +
     18*yprime(N - 1)) / 5;
    end
    yprime = yprime';
end
```

Listing 12: reactub4