ME 2450 Lab 07: System Modeling - Euler's Method

Objective

A computer program that simulates the velocity of a spherical object (ball) falling through a viscous medium will be completed. The velocity of the ball is modeled by Newton's 2nd law with viscous effects modeled by a quadratic drag model. The equations of motion are integrated forward through time using the first-order forward Euler's method. At the simulation's completion, a plot of the ball's velocity versus time is generated. To receive credit for the lab, you must implement `falling_ball` and demonstrate of its correctness to your TA. Specifically:

- ☐ (8 points) write a function ball_motion that models the motion of the falling ball. The function is to be implemented in MATLAB or Python;
- ☐ (8 points) write a function euler that performs first-order integration using Euler's method. The function is to be implemented in MATLAB or Python; and
- ☐ (2 point) run the falling_ball simulation and show your TA the resultant plot. falling_ball requires that ball_motion and euler be implemented.
- ☐ (2 point) write a short paragraph explaining why your numerical results are reasonable. Why does your plot look the way it looks? Turn in this paragraph with your code files in a single .zip file.

References

- Lecture 01 and Lecture 12 slides: *Lecture01.pdf* and *Lecture12.pdf*.
- Chapter 1.1 and Chapter 25.1 *Chapra, Numerical Methods for Engineers (7th Edition)*.

Physical Model

When analyzing a falling ball, it is helpful to begin with a free-body diagram (Figure 1a). The forces acting on the ball include gravity (F_W) and friction (F_d). When we attempt to model the behavior of the falling ball, we must make several modeling design decisions. Can we approximate the ball as a perfect sphere, or do we need to take into account irregularities in its shape? Is gravity constant, or does it vary with altitude or other factors? What effects of friction do we want to model? The answers to these and other questions will affect the level of detail we have in our mathematical model, as well as potentially changing the methods of solution we have available to us.

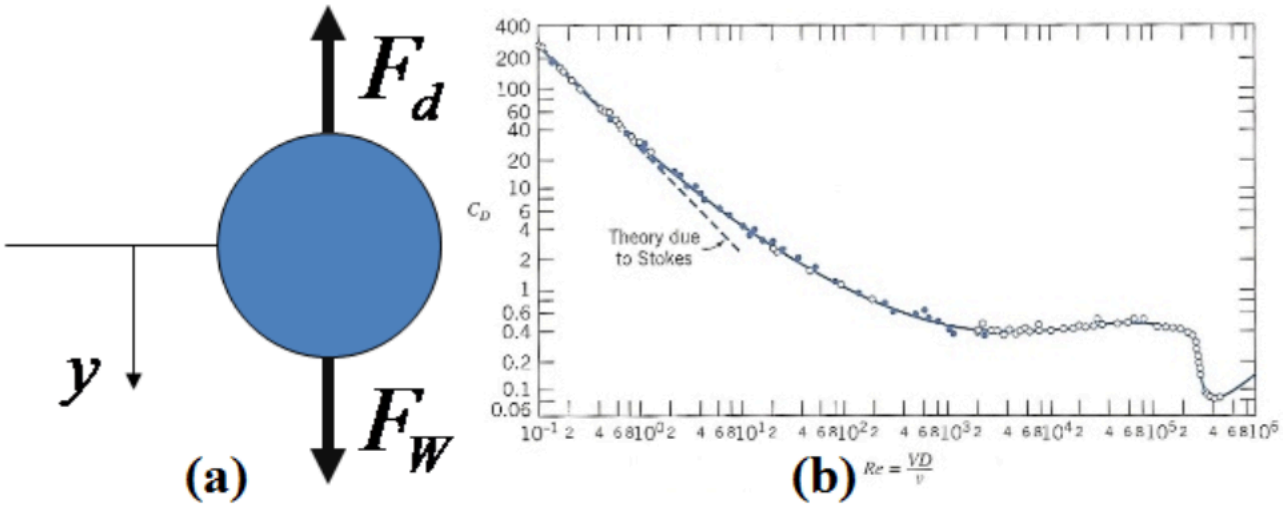

Figure 1: (a) Free-body Diagram of a Falling Ball (b) The drag coefficient for a sphere as a function of Reynolds Number, as given by Fox, McDonald and Pritchard. **Introduction to Fluid Mechanics**. 6th ed., John Wiley & Sons, Inc., 2006.

A general mathematical model of a falling ball which accounts for the forces of gravity and friction can be formed from the Law of Conservation of Momentum (Newton's 2nd Law), which, for Figure 1a, is:

$$m\frac{dv}{dt} = F_W - F_d$$

where m is the mass of the ball and $\frac{dv}{dt}$ is the rate of change of velocity with respect to time (i.e., acceleration) of the ball. (Since y is positive downward in Figure 1a, we likewise assume that v is positive downward in this formula and throughout this lab handout.) For this lab, we will assume that the ball remains close to the Earth's surface, so the acceleration of gravity remains constant ($F_W = mg$, $g \approx 9.81 \frac{\text{m}}{\text{s}^2}$). Substituting this formula for F_W and solving for the derivative, $\frac{dv}{dt}$, we obtain:

$$\frac{dv}{dt} = g - \frac{F_d}{m} \tag{1}$$

Part (b) of Figure 1 shows the classic plot of Drag Coefficient versus Reynolds Number. Reynolds Number, Re, is the ratio of the inertial forces to the viscous forces ($Re = \frac{VD}{\nu}$, where ν is the kinematic viscosity of the air). The Drag Coefficient, C_d, is proportional to the drag force the ball experiences. From the chart, you can see that finding the Reynolds Number allows us to determine the Drag Coefficient. In the study of Fluid Mechanics and other areas of engineering, we frequently make use of nondimensional parameters such as the Reynolds Number or the Drag Coefficient, because they allow us to relate similar situations.

Depending on what we intend to do with our model, we may choose to assume that the frictional forces have a negligible effect ($F_d = 0$), which results in a mathematical model of this form:

$$\frac{dv}{dt} = g \tag{2}$$

Note that integrating this equation twice results in our standard formulas for velocity ($v(t) = v_0 + gt$) and position ($x(t) = x_0 + v_0 t + \frac{1}{2}gt^2$) of a projectile. For Reynolds Numbers above 2×10^5 (i.e., above the sharp drop in C_d, this may be a valid approximation.

We may instead decide that the frictional force behaves like viscous friction ($F_d = cv$ for some coefficient of viscous friction c). The resulting mathematical model has this form:

$$\frac{dv}{dt} = g - \frac{cv}{m} \tag{3}$$

For Reynolds Numbers less than 10^3, (i.e., low speeds and/or very small objects), this is likely to be a good approximation.

Alternatively, we may decide to model the friction as a drag force ($F_d = 1/2\rho C_d A v^2$, where ρ is the density of the air, C_d is the drag coefficient, A is the cross-sectional area of the ball, and v is the velocity of the ball).

$$\frac{dv}{dt} = g - \frac{1}{2}\frac{\rho C_d A v^2}{m} \tag{4}$$

For Reynolds Numbers between 10^3 and 10^5, this is considered to be a good approximation of the behavior of a falling sphere.

Armed with these three models ((2), (3) and (4)), we might experimentally investigate the behavior of a falling ball and compare it to the results of the models. We would then determine which model best fits our experimental observations. In today's lab, you will learn how to use a numerical method to approximate the behavior of a falling ball using each of (2), (3) and (4) in MATLAB or Python.

Numerical Methods

Euler's Method

As discussed in Section 1.1 of the textbook, Euler's Method is a numerical method used to approximate a solution to ordinary differential equations. It is derived from the definition of the derivative:

$$\begin{aligned}
\frac{dy}{dt} &= \lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t} \\
\frac{dy}{dt} &\approx \frac{\Delta y}{\Delta t} \\
\frac{dy}{dt} &\approx \frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i}
\end{aligned}$$

If we have already calculated $y(t_i)$ at a given t_i, and we know the formula for the derivative $\frac{dy}{dt}$ (for example, from a model such as (1)), we can rearrange 5 to find $y(t_{i+1})$:

$$y(t_{i+1}) = y(x_i) + \left.\frac{dy}{dt}\right|_{t=t_i} (t_{i+1} - t_i) \tag{5}$$

Equation (5) is known as Euler's Method. If we wanted to, we could rewrite it in terms of our model given in (4):

$$v_{i+1} = v_i + \left(g - \frac{\rho A C_d}{2m} v_i^2\right)(t_{i+1} - t_i) \tag{6}$$

In this last equation, t_i is the time at the last time step, t_{i+1} is the time at the current time step, v_i is the velocity of the ball at the last time step and v_{i+1} is the velocity of the ball at the current time step. g, ρ, A, C_d and m are each as defined above. Given a set of initial conditions, such as $t_0 = 0$ and $v_0 = 0$, and a time step, such as $t_{i+1} - t_i = 0.25$, we could calculate v_k for any time t_k.

Lab Assignment

The program `falling_ball` runs the falling ball simulation by setting the appropriate physical parameters and initial conditions. In order for `falling_ball` to work, you'll need to write `euler` and `ball_motion` first.

Ball Motion

Write a function `dydt = ball_motion(t, y, params)` that implements (4), for a scalar `t` and scalar `y`. That is, a function that calculates $y' = f(t, y)$ where $f(t, y)$ is given by the right-hand-side of (4).

Input Arguments:

- `t` (scalar): Current time.
- `y` (scalar): Velocity of the falling ball at time `t`.
- `params` (array): Array of system properties
  - params(1): Gravity
  - params(2): Air density
  - params(3): Ball Mass
  - params(4): Ball radius
  - params(5): Drag coefficient

Output Arguments:

- `dydt` (scalar): The $f(t, y)$ from $y' = f(t, y)$ evaluated at time `t`.

Note: The units of the entries in `params` should all be consistent. That is, you need to use the same units for mass, length, time, etc. throughout. For example, if you use units of kg for ball mass, all other parameters that have mass in their units will also need to have kg. And if you provide gravity in $\frac{m}{s^2}$ and your ball mass in kg, then your air density will need to be in $\frac{kg}{m^3}$. As a counterexample, a ball mass in grams would not be consistent with air density in $\frac{kg}{s^2}$, and would produce erroneous results (likely off by a factor of 10^3).

Note: The decision to use the variable `y` in `ball_motion`, instead of `v` was deliberate. As the semester progresses, `ball_motion` will be modified to represent a system of ODE and `y` will represent an array of states (not just velocity). The name `y` is consistent with the documentation and implementation in MATLAB, Python, and other documentation commercial ODE solvers.

Euler's Method

Write a function `[t,y] = euler(odefun, tspan, y0)` that implements Euler's Method (5), where `tspan=[t0 t1 ... tf]`, that integrates a differential equation $y' = f(t, y)$ from `t0` to `tf` with initial conditions `y0`. Each row of the solution array `y` corresponds to a value in `t`.

Input Arguments:

- `odefun` (callable) - Function to solve. The function `dydt = odefun(t,y)`, for a scalar `t` and a scalar `y`, must return a scalar `dydt` that corresponds to $f(t, y)$. `odefun` must accept both input arguments `t` and `y`, even if one of the arguments is not used in the function.
- `tspan` (array) - Time values of integration. The elements in `tspan` must be all increasing.

  The solver imposes the initial conditions given by `y0` at the initial time given in the first entry of `tspan`, then integrates from the first to the last entry in `tspan`.
- `y0` (scalar) - Initial conditions. `y0` contains the initial condition for the equation defined in `odefun`.

Output Arguments:

- `t` (array) - Evaluation points. `t` is the same as `tspan`.
- `y` (array) - Solution returned as an array. Each row in `y` corresponds to the solution at the value returned in the corresponding row of `t`.