# ME 2450 Lab 09a: System of Ordinary Differential Equations

## Objective

Two computer programs, `falling_ball` and `moving_train`, will be written. `falling_ball` is a modification of the program by the same name that was written in Lab 01. `moving_train` is a program that integrates the motion of a compressed air train through time. Each program requires an ordinary differential equation (ODE) solver capable of solving a system of first-order ODEs. To this end, `euler` from Lab 01 will be modified to process systems of ODE. To receive credit for the lab, complete each program and demonstrate their correctness to your TA. Specifically:

- ☐ (2 points) Modify `ball_motion` from Lab 01 so that it represents a system of two differential equations, use MATLAB or Python.
- ☐ (2 points) Modify `euler` from Lab 01 so that it can process a system of ODE with an arbitrary number of equations.
- ☐ (1 point) Decompose the second-order differential equation $eq.\ 4$ into a system of differential equations.
- ☐ (2 points) Write a function `train_motion` that evaluates the derivatives for the system of ODE– this will be used by your program `euler`.
- ☐ (2 points) Write programs `falling_ball` and `moving_train` that integrate $eq\ 4$, which models the motion of the falling ball and the train, respectively, forward through time. The programs should generate two plots showing the results of each simulation: position and velocity over time.
- ☐ (1 point) Write a short paragraph explaining why your numerical results are reasonable; i.e. does your plot make physical sense? Submit this in a separate pdf file.

## References

- A Brief Introduction to MATLAB, *MatlabIntro.pdf*.
- A Brief Introduction to Python, *PythonIntro.pdf*.
- An Introduction to Function Handles, *FunctionHandles.pdf*.
- Lectures 13/15, *Lectures 13/15.pdf*.
- Chapter 25 *Chapra, Numerical Methods for Engineers (7th Edition)*.

## Physical Models

### Falling Ball

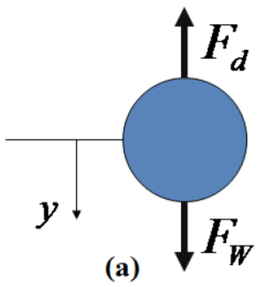

Figure 1: Forces Acting on a Falling Ball

In Lab 01, you analyzed the motion of a falling ball (Figure 1). The forces acting on the ball are the weight ($F_W$) and drag ($F_d$). For the purposes of this lab, assume that the drag force obeys the aerodynamic drag force law

$$F_d = \frac{\rho C_d A v^2}{2}.$$

Previously, the velocity of the ball was modeled using Conservation of Momentum, which results in the following model:

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

This model is in the standard ODE form:

$$\frac{dv}{dt} = f(t, v) \tag{1}$$

(1) is in the standard form that is used to represent the motion of the ball in the first lab, where the derivative of the state, $\frac{dv}{dt}$, is equal to some function $f$ of the independent variable, $t$, and the state, $v$.

Using a modeling technique such as Euler's Method, velocity of the ball could be estimated at a given time $t$. But, what if instead the position $y$ of the ball was of interest? Another differential equation that represents the position of the ball can be formulated: $\frac{dy}{dt} = v$. This gives a system of ODE that can be solved to estimate both the position and velocity of the ball at any time $t$:

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

To simplify representing this system computationally, write it as a vector equation:

$$\begin{Bmatrix} \frac{dy}{dt} \\ \frac{dv}{dt} \end{Bmatrix} = \begin{Bmatrix} v \\ g - \frac{1}{2}\frac{\rho C_d A v^2}{m} \end{Bmatrix} \Rightarrow \frac{d}{dt}\begin{Bmatrix} y \\ v \end{Bmatrix} = \begin{Bmatrix} f_1(t, y, v) \\ f_2(t, y, v) \end{Bmatrix} \tag{2}$$

One final change is to let $\boldsymbol{x} = [y \;\; v]^T$ which gives:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x}) \tag{3}$$

This final equation is in the form to be implemented computationally.

## Acceleration of a Simplified Train

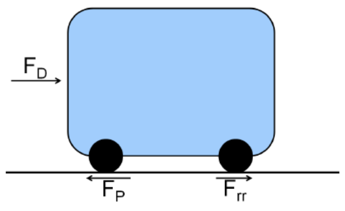

Figure 2: Forces Acting on a Simple Train

The conservation of momentum can be applied to the train in Figure 2. To make effective use of this model, however, the formulas for each of the force "sources" and force "sinks" shown in the Figure 2 need to be determined. In the case of the falling ball, for example, the force source was the weight, $F_W = mg$, while the force sink was the drag, $F_D = \frac{\rho C_D A V^2}{2}$. Other objects will have different sources and sinks, and a force that is a source in one model may instead be a sink in another. For example, imagine instead of falling, the ball were being propelled upward by some sort of rocket. The thrust of the rocket would become the force source, while the weight would now counteract the motion and become a force sink.

As you have no doubt determined in your AID assignments, your train has a single force source and two force sinks. The force source is the propulsion force $F_P$ supplied by the propulsion system you will choose to utilize the energy stored in your compressed air tank. For this lab, the propulsion force is assumed to be constant. You should recognize, however, that this is an extremely simple model and the actual force provided by your propulsion system will vary with time and the pressure left in your tank. This model will be improved in future assignments.

The two force sinks are the rolling resistance, $F_{rr}$, and the drag force, $F_D$. The resistive force due to rolling friction is $F_{rr} = mgC_{rr}$, where $m$ is the mass acting on each wheel, $g$ is the acceleration of gravity, and $C_{rr}$ is the coefficient of rolling resistance between the wheel and the track. Note that the normal force, $mg\cos\theta$, is estimated by $mg$, since it is assumed that the track is perfectly level. By now you should be familiar with the formula for drag force, $F_D = \frac{\rho C_D A V^2}{2}$, where $\rho$ is the density of the fluid, $C_d$ is the coefficient of drag, $A$ is the effective frontal area, and $V$ is the velocity.

With the source and sink forces now defined, conservation of momentum is used to find $m\boldsymbol{a} = \sum \boldsymbol{F}$:

$$\frac{d^2x}{dt^2} = \frac{F_P - F_D - F_{rr}}{m} \tag{4}$$

This is a second-order differential equation, where $x$ is the position of the train, measured with respect to the starting location.

## Numerical Methods

### Systems of Ordinary Differential Equations

Equation 1 can be adjusted to solve for the velocity and position simultaneously as a function of time as shown in equation 3. The function will still have two inputs ($t$ and $\boldsymbol{y}$), and will still evaluate the function $\boldsymbol{f}$ on the right-hand side to determine the output $\frac{d\boldsymbol{y}}{dt}$.

**Note:In MATLAB, the output from a function representing a system of ordinary differential equations *must* be a column vector. See the MATLAB documentation (especially the entry on "Creating Matrices") if you need help creating a column vector. If the output is not a column vector, MATLAB built-in functions for processing ODEs such as `ode45` will not work.**

With the updated `ball_motion` function now tracking position, the `euler` function from Lab 01 will need to be modified to solve for all state variables. Note the output will be a vector. The formula for Euler's method in terms of vectors of states is:

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

This formula is equivalent to writing Euler's method for each of the individual ODEs. In the case of the falling ball with two state variables:

$$\begin{aligned} z\left(t_{i+1}\right) &= z\left(t_i\right) + \left.\frac{dz}{dt}\right|_{t=t_i}\left(t_{i+1}-t_i\right) \\ v\left(t_{i+1}\right) &= v\left(t_i\right) + \left.\frac{dv}{dt}\right|_{t=t_i}\left(t_{i+1}-t_i\right) \end{aligned}$$

If $n$ state variables, $y_1$ through $y_n$ exist, Euler's method is used $m$ times:

$$\begin{aligned} y_1\left(t_{i+1}\right) &= y_1\left(t_i\right) + \left.\frac{dy_1}{dt}\right|_{t=t_i}\left(t_{i+1}-t_i\right) \\ y_2\left(t_{i+1}\right) &= y_2\left(t_i\right) + \left.\frac{dy_2}{dt}\right|_{t=t_i}\left(t_{i+1}-t_i\right) \\ &\vdots \\ y_{n-1}\left(t_{i+1}\right) &= y_{n-1}\left(t_i\right) + \left.\frac{dy_{m-1}}{dt}\right|_{t=t_i}\left(t_{i+1}-t_i\right) \\ y_n\left(t_{i+1}\right) &= y_n\left(t_i\right) + \left.\frac{dy_m}{dt}\right|_{t=t_i}\left(t_{i+1}-t_i\right) \end{aligned}$$

The `euler` function will still output two variables, but in this instance, the second output needs to be a matrix of values, with the columns representing the position and velocity at each time value in the first output. The two outputs would look like this:

$$\boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m,1} & y_{m,1} & \cdots & y_{m,n} \end{bmatrix} \tag{6}$$

For the falling ball, the two outputs would look like this:

$$\boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 & v_1 \\ y_2 & v_2 \\ \vdots & \vdots \\ y_m & v_m \end{bmatrix}$$

where $y_j = y(t_j)$ and $v_j = v(t_j)$. Now, imagine a system with 3, 5, 100, or $n$ differential equations. The second output, `y`, should be an array with $m$ rows and $n$ columns (one column for each of the differential equations.)

## Lab Assignment

### Ball Motion

Write a function `dydt = ball_motion(t, y, params)` that implements equation (2), for a scalar `t` and vector `y` for every m iterations. That is, a function that calculates $y' = f(t, y)$ where $f(t, y)$ is given by the right-hand-side of (2). This function is a modification of the function by the same name developed in Lab 01.

**Input Arguments:**

- `t` (scalar): Current time.
- `y` (vector): State of the falling ball at time `t`. The first entry of `y` is the position and the second entry its velocity.
- `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` (vector): The $f(t, y)$ from $y' = f(t, y)$ evaluated at time `t`. The first entry of `dydt` is the ball's velocity and the second its acceleration.

### Euler's Method

Write a function `[t,y] = euler(odefun, tspan, y0)` that implements Euler's Method equation (5) for a system of equations, 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 vector `y`, must return a vector `dydt` having the same shape as `y` 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` (vector) - Initial conditions. `y0` contains the initial conditions 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`.

## Train Motion

Write a function `dydt = train_motion(t, y, params)` that that implements the first-order system of ODE found by linearizing equation (4) into a system of first-order ODE, for a scalar `t` and vector `y`.

**Input Arguments:**

- `t` (scalar): Current time.
- `y` (vector): State of the moving train at time `t`. The first entry of `y` is the position and the second entry its velocity.
- `params` (array): Array of system properties
  - params(1): Gravity
  - params(2): Air density
  - params(3): Train mass
  - params(4): Frontal area
  - params(5): Drag coefficient
  - params(6): Rolling coefficient of friction
  - params(7): Propulsion force

**Output Arguments:**

- `dydt` (vector): The $f(t, y)$ from $y' = f(t, y)$ evaluated at time `t`. The first entry of `dydt` is the trains's velocity and the second its acceleration.

## Falling Ball and Moving Train

Write programs (driver scripts) `falling_ball` and `moving_train` that simulate the motion of the falling ball and moving train, respectively. Generate plots showing the results of each simulation for a duration of 10 seconds with a step size of 1 second. Plot the position of the object (ball or train) versus time and in the second subplot, plot the velocity of the object versus time. Each program can be generated by modifying the program `falling_ball` from Lab 01.

Use the following system properties:

- Acceleration of gravity: $9.8$ m/s$^2$
- Air density: $1.0$ kg/m$^3$
- Ball mass: $0.01$ kg.
- Ball radius: $0.03$ m.
- Train mass: $10$ kg
- Rolling resistance coefficient: $0.002$
- Drag coefficient: $0.4$
- Propulsion force: $1.0$ N
- Total frontal area: $0.05$ m$^2$
- Initial conditions on ball: $z_0 = -2$ and $v_0 = 0$.
- Initial conditions on train: $x_0 = 0$ and $v_0 = 0$.