

# Modeling 1DoF System

ARO (@art\_of\_electronics\_)

June 19, 2025

## 1 Introduction

This document presents the modeling of a simple 1 degree-of-freedom (DoF) system using Lagrangian mechanics and co-energy analysis. It also provides a method for including a friction model. A Simulink block diagram is also provided for simulation purposes.

## 2 1DoF System Description

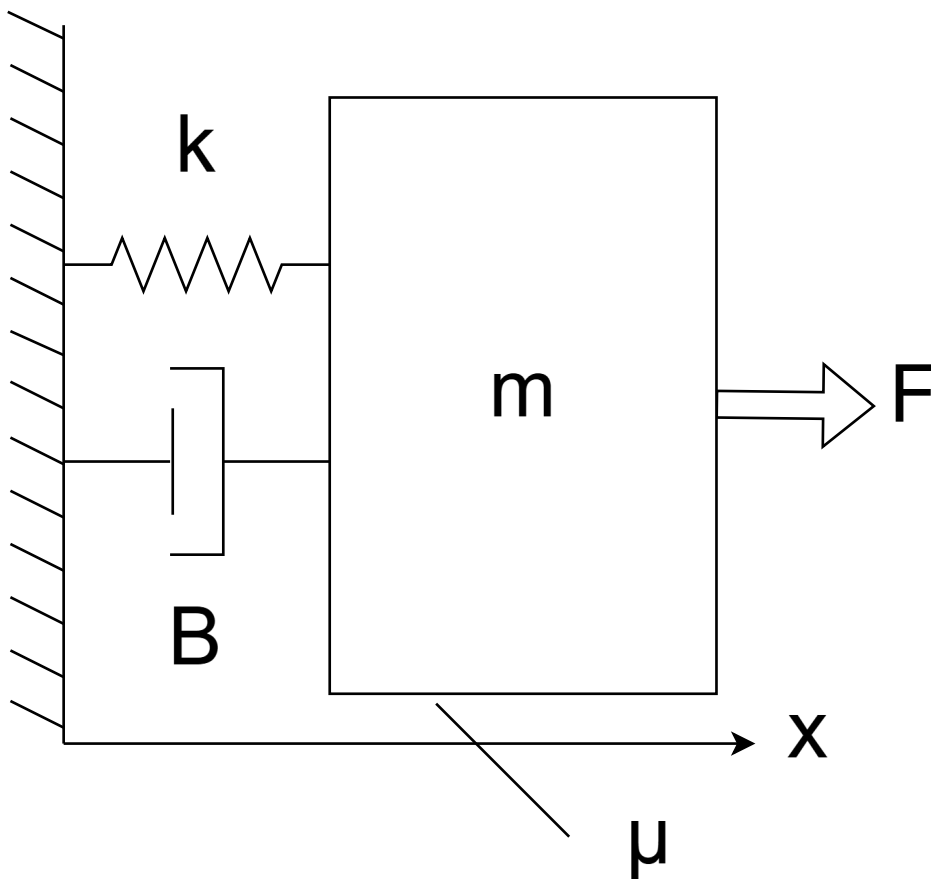


Figure 1: Overview of the 1DoF system

We consider a 1DoF system in 2 main cases: first, where a plain, frictionless model is presented; second, where friction submodel is included into 1DoF system.

### 3 Lagrangian Formulation - General Case

The kinetic and potential energies of 1DoF system are defined as:

$$T = \frac{1}{2}m\dot{x}^2$$

$$U = k \int_0^x x dx = \frac{1}{2}kx^2$$

The system has a damper, which is a dissipative element:

$$D_B = B \int_0^{\dot{x}} \dot{x} d\dot{x} = \frac{1}{2}B\dot{x}^2$$

Now, we can include the friction model, as it is also a dissipative element:

$$D_\mu = B_\mu \dot{x}$$

In that case, the total dissipation in 1DoF system can be described as:

$$D = \frac{1}{2}B\dot{x}^2 + B_\mu \dot{x}$$

With this approach, we can either define the friction submodel, or equate it to 0.

The Lagrangian of 1DoF system is:

$$\mathcal{L}(x, \dot{x}) = T - U = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2$$

The Rayleigh dissipation function can be written as:

$$\mathcal{D} = D - P = \left( \frac{1}{2}B\dot{x}^2 + B_\mu \dot{x} \right) - F\dot{x}$$

Using Lagrange's equation with a non-conservative force (damper), the equation expands into the following form:

$$\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{q}} \right) - \frac{\partial \mathcal{L}}{\partial q} + \frac{\partial \mathcal{D}}{\partial \dot{q}} = 0$$

This leads to the following differential equation:

$$\frac{d}{dt} \left( \frac{\partial (\frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2)}{\partial \dot{x}} \right) - \frac{\partial (\frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2)}{\partial x} + \frac{\partial ((\frac{1}{2}B\dot{x}^2 + B_\mu \dot{x}) - F\dot{x})}{\partial \dot{x}} = 0$$

$$\frac{d}{dt} \left( \frac{\partial (\frac{1}{2}m\dot{x}^2 - 0)}{\partial \dot{x}} \right) - \frac{\partial (0 - \frac{1}{2}kx^2)}{\partial x} + \frac{\partial ((\frac{1}{2}B\dot{x}^2 + B_\mu \dot{x}) - F\dot{x})}{\partial \dot{x}} = 0$$

$$\frac{d}{dt} \left( 2 \cdot \frac{1}{2} m \dot{x}^{2-1} \right) - \left( - \left( 2 \cdot \frac{1}{2} k x^{2-1} \right) \right) + \left( \left( 2 \cdot \frac{1}{2} B \dot{x}^{2-1} + \cdot B_\mu \dot{x}^{1-1} \right) - F \dot{x}^{1-1} \right) = 0$$

$$\frac{d}{dt} (m \dot{x}) - (- (k x)) + ((B \dot{x} + B_\mu) - F) = 0$$

$$m \ddot{x} + k x + B \dot{x} + B_\mu - F = 0$$

Sorting yields the equation's final form:

$$m \ddot{x} + B \dot{x} + k x = F - B_\mu$$

## 4 Co-energy and Energy Expressions

### Mass

The general inertia co-energy for mass is:

$$W'_m = W_m = \int_0^v m v' dv' = \frac{1}{2} m v^2 = \frac{1}{2} m \dot{x}^2$$

In linear systems, energy and co-energy are equal.

### Spring

The general energy stored in the spring  $k_1$  is:

$$W'_s = W_s = \int_0^x k x' dx' = \frac{1}{2} k x^2$$

Since the spring is conservative (no energy loss), co-energy equals energy.

### Damper and Friction

The general damper dissipates energy as friction:

$$W_d = \int_0^t B v^2(\tau) d\tau = \frac{1}{2} B v^2 = \frac{1}{2} B \dot{x}^2$$

The general friction dissipates energy in the following manner:

$$W_f = \frac{d}{d\dot{x}} \left[ \int_0^{\dot{x}} B_\mu(v) v dv \right] = B_\mu(\dot{x}) \dot{x} = B_\mu \dot{x}$$

Dampers, as friction, do not store energy, so they do not have co-energy.

Using the co-energy equation with a non-conservative force, we can write:

$$\frac{d}{dt} \left[ \frac{\partial T}{\partial \dot{y}} \right] - \frac{\partial T}{\partial y} + \frac{\partial U}{\partial y} + \frac{\partial D}{\partial \dot{y}} = f_i$$

Adjusting the general co-energy expressions to 1DoF systems specific configuration gives:

$$\begin{aligned}W_m &= \frac{1}{2}m\dot{x}^2 \\W_s &= \frac{1}{2}kx^2 \\W_d + W_f &= \frac{1}{2}B\dot{x}^2 + B_\mu\dot{x}\end{aligned}$$

Finally, we can assign the following assumptions:

$$\begin{aligned}T &= W_m \\U &= W_s \\D &= W_d + W_f \\f_i &= F\end{aligned}$$

By substituting the coefficients into the equation, we obtain the following:

$$\begin{aligned}\frac{d}{dt} \left[ \frac{\partial \left( \frac{1}{2}m\dot{x}^2 \right)}{\dot{x}} \right] - \frac{\partial \left( \frac{1}{2}m\dot{x}^2 \right)}{x} + \frac{\partial \left( \frac{1}{2}kx^2 \right)}{x} + \frac{\partial \left( \frac{1}{2}B\dot{x}^2 + B_\mu\dot{x} \right)}{\dot{x}} &= F \\ \frac{d}{dt} \left[ \frac{\partial \left( \frac{1}{2}m\dot{x}^2 \right)}{\dot{x}} \right] - 0 + \frac{\partial \left( \frac{1}{2}kx^2 \right)}{x} + \frac{\partial \left( \frac{1}{2}B\dot{x}^2 + B_\mu\dot{x} \right)}{\dot{x}} &= F \\ \frac{d}{dt} [m\dot{x}] + kx + B\dot{x} + B_\mu &= F \\ m\ddot{x} + B\dot{x} + kx &= F - B_\mu\end{aligned}$$

## 5 Simulink Model for Lagrangian and Co-energy Expression

In order to model the equation properly, we must translate the dynamic equation into a form understandable for Simulink:

$$m\ddot{x} + B\dot{x} + kx = F - B_\mu$$

$$\ddot{x} = \frac{1}{m} (F - B\dot{x} - kx - B_\mu)$$

Before coding the solution in Simulink, we should consider the friction submodel. The most versatile friction model is the Coulomb function:

$$B_\mu = \mu mg \cdot \text{sign}(\dot{x}) + c\dot{x}$$

$\mu$  – dry friction coefficient;  
 $c$  – viscous friction coefficient;  
 $g$  – gravitational force;

The screenshot shows a Simulink model of a mass-spring-damper system. The model consists of the following blocks and connections:

- Step**: A step function input block.
- Add**: A summing junction with three inputs: the Step block, the output of the **Gain3** block, and the output of the **Add1** block.
- Gain3**: A gain block labeled  $1/in.m$ .
- Integrator1**: An integrator block labeled  $1/s$ .
- Gain2**: A gain block labeled  $in.k$ .
- Gain5**: A gain block labeled  $in.c$ .
- Sign**: A signum function block.
- Gain4**: A gain block labeled  $-K$ .
- Add1**: A summing junction with two inputs: the output of the **Gain4** block and the output of the **Gain5** block.
- Integrator2**: An integrator block labeled  $1/s$ .
- Scope**: A scope block for monitoring the output.
- To Workspace**: A block to save the output to the MATLAB workspace.

The signal flow is as follows: The Step input is fed into the Add block. The output of the Add block is fed into Gain3. The output of Gain3 is fed into Integrator1. The output of Integrator1 is fed into Gain2 and Gain5. The output of Gain2 is fed into the Sign block. The output of the Sign block is fed into Gain4. The output of Gain4 is fed into Add1. The output of Gain5 is also fed into Add1. The output of Add1 is fed into Integrator2. The output of Integrator2 is fed into the Scope and To Workspace blocks.

A dialog box titled "Source Block Parameters: Step" is open on the left side of the image. It contains the following fields:

- Step**: A dropdown menu.
- Output a step.**: A checkbox.
- Parameters**: A section with a text field for "Step time".
- Initial value:**: A text field.
- Final value:**: A text field.
- Duration**: A text field.
- Sample time:**: A text field.
- Interpret vector parameters as 1-D**: A checkbox.
- Enable zero-crossing detection**: A checkbox.
- Buttons**: OK, Cancel, Help, and Apply.

At  $\dot{x} = 0$ , the sign function has a discontinuity (jump from -1 to +1), and Simulink's solver repeatedly attempts to find the exact zero-crossing time, failing to progress. This causes the 1000 consecutive zero-crossings error. In order to solve the issue, use either:

- 5

## 6 Matlab and Python script

For script-based system modeling, we must take a slightly different approach to translating mathematical equations into code. Let  $x(t) = x(1)$ ,  $\dot{x}(t) = x(2)$ , in that case the matrix of the 1DoF equation will be written as:

$$\frac{d}{dt} \begin{bmatrix} x(1) \\ x(2) \end{bmatrix} = \begin{bmatrix} x(2) \\ (F - B \cdot x(2) - k \cdot x(1) - friction) / m \end{bmatrix}$$

Now, the equation can be coded into Matlab:

```
1 Dx = [x(2); (F-B * x(2) - k * x(1) - friction) / m];
```

Python counts matrix elements from 0, so the equivalent Python code can be written as:

```
1 def sindle_dof(x, t):
2     friction = Mu * m * g * np.sign(x[1]) + c * x[1]
3     return x[1], (F - B * x[1] - k * x[0] - friction) / m
```

## 7 Simulation Results

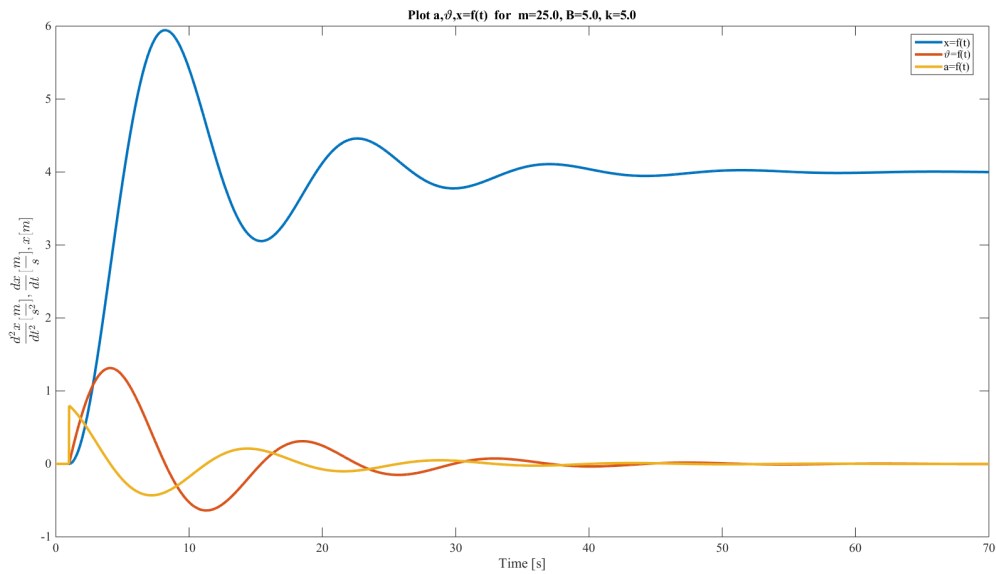


Figure 4: Result without friction

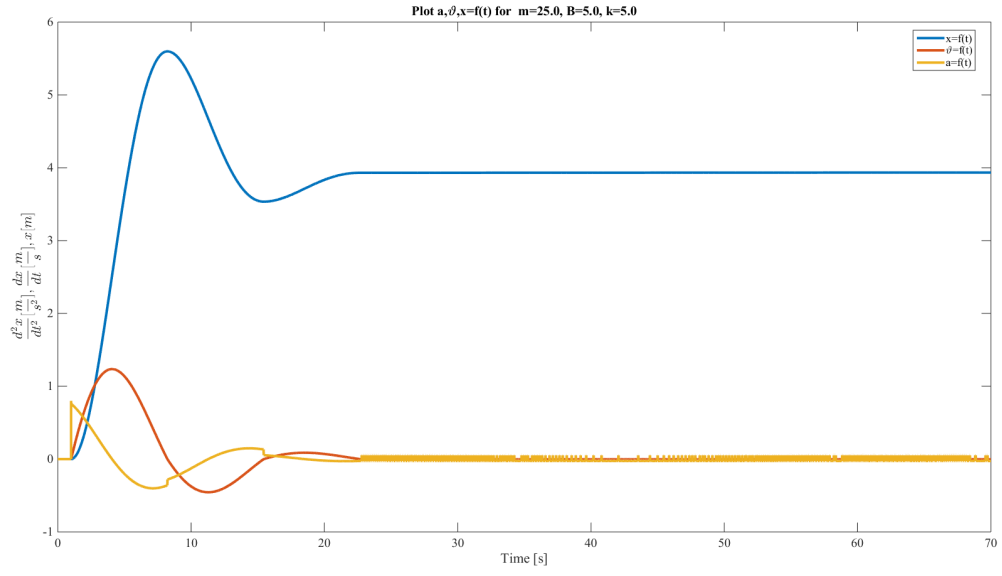


Figure 5: Result with friction

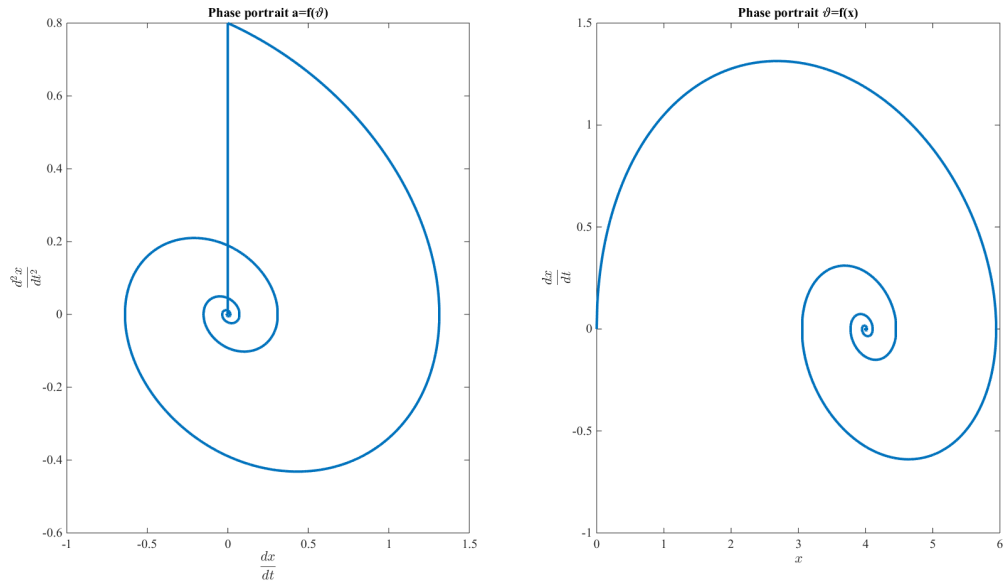


Figure 6: Phase portrait without friction

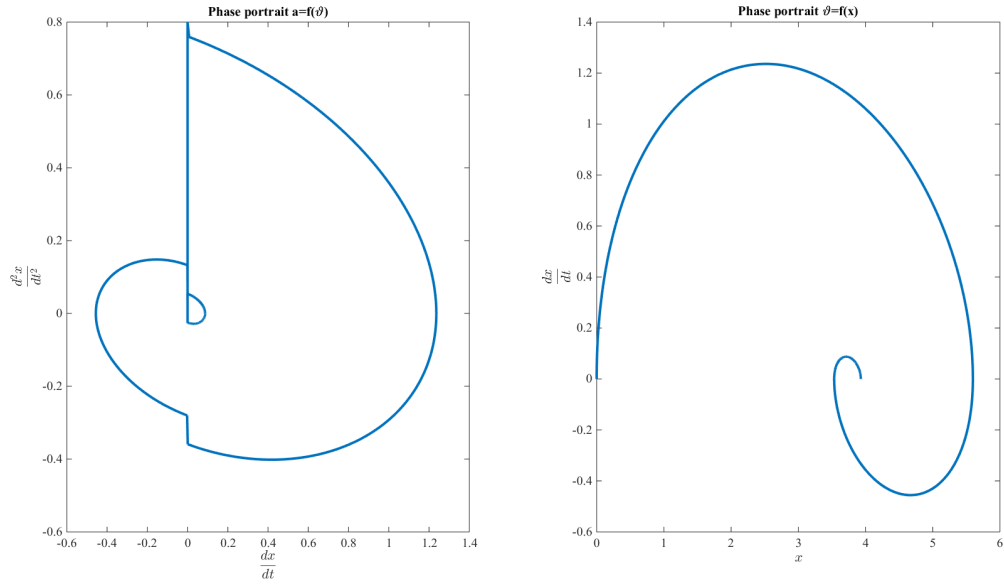


Figure 7: Phase portrait with friction

## 8 Conclusion

This document demonstrates the Lagrangian and co-energy approach to modeling a 1DoF system. It also provides a visual block diagram suitable for simulation in Simulink and basic code for scripting.