Lab Exercise 8 - Extending Mechanical Core Library Component in OpenModelica

---

AIM

To extend a core translational mechanical component from the Modelica Standard Library, build a complete mass–spring–damper system using the extended component, compute energy and damping losses, and simulate system behavior in OpenModelica.

---

2. OBJECTIVES

After completing this lab, students will be able to:

- Extend a core Modelica library component using inheritance
- Add custom physical calculations
- Build a modular mechanical system
- Compute kinetic and potential energy
- Analyze damping power loss
- Perform simulation and interpret system response

---

3. THEORY OVERVIEW

A translational mechanical system follows Newton's Second Law:

$$m \frac{d^2s}{dt^2} + c \frac{ds}{dt} + k s = F$$

Where:

m = mass
c = damping coefficient
k = spring stiffness
s = displacement
F = applied force

Energy relationships:

Kinetic Energy = ½ m v²
Potential Energy = ½ k s²
Damping Power Loss = c v²

We will extend the core library component:

This demonstrates professional modeling practice because we do NOT modify library code — we extend it safely.

PART A – EXTENDING A CORE LIBRARY COMPONENT

MAJOR STEP 1 – CREATE A NEW PACKAGE

Step 1.1 Open OMEdit

Step 1.2 Click File → New Modelica Class

Name: **MechanicalHierarchicalLab**
Restriction: **package**

Click OK.

This package will contain all models for this experiment.

MAJOR STEP 2 – EXTEND THE CORE MASS COMPONENT

We now extend the Mass model from the Modelica Standard Library.

Create a new model:

Name: **ExtendedMass**
Restriction: **model**

Insert the following code:

```
model ExtendedMass


  extends Modelica.Mechanics.Translational.Components.Mass;


  Real kineticEnergy "Kinetic Energy";

  Real momentum "Linear Momentum";


equation

  kineticEnergy = 0.5 * m * v^2;

  momentum = m * v;

end ExtendedMass;
```

DETAILED EXPLANATION OF THE CODE

1. extends Modelica.Mechanics.Translational.Components.Mass;

This line means:

• We inherit all equations and connectors of the standard Mass component
• The mass already contains:

- Displacement (s)
- Velocity (v)
- Acceleration
- Force balance equations

- Mechanical flanges

We are not rewriting physics.
We are reusing the existing core component safely.

2. Real kineticEnergy;

This declares a new variable to calculate the kinetic energy of the mass.

3. Real momentum;

This declares a variable to calculate linear momentum.

4. kineticEnergy = 0.5 * m * v^2;

This calculates:

$\frac{1}{2} m v^2$

Where:
m = mass parameter
v = velocity of mass

So during simulation, this continuously computes energy based on motion.

5. momentum = m * v;

This computes linear momentum of the moving mass.

What this extended model actually does:

It behaves exactly like a normal Mass component,
but now additionally calculates:

• Kinetic energy
• Momentum

This makes it reusable in advanced systems.

PART B – BUILDING THE COMPLETE MECHANICAL SYSTEM

MAJOR STEP 3 – CREATE COMPLETE MECHANICAL DRIVE SYSTEM

Create new model:

Name: **MechanicalDriveSystem**
Restriction: **model**

Insert the following code:

```
model MechanicalDriveSystem
  Modelica.Mechanics.Translational.Sources.ConstantForce force(F=15);
  ExtendedMass massBlock(m=3);
  Modelica.Mechanics.Translational.Components.Spring springElement(c=1500);
  Modelica.Mechanics.Translational.Components.Damper damperElement(d=8);
  Modelica.Mechanics.Translational.Components.Fixed fixedSupport;

  Real potentialEnergy;
  Real dampingPower;
  Real totalEnergy;

equation
  connect(force.flange, massBlock.flange_a);
  connect(massBlock.flange_b, springElement.flange_a);
  connect(springElement.flange_b, damperElement.flange_a);
```

```
  connect(damperElement.flange_b, fixedSupport.flange);

  potentialEnergy = 0.5 * springElement.c * massBlock.s^2;

  dampingPower = damperElement.d * massBlock.v^2;

  totalEnergy = massBlock.kineticEnergy + potentialEnergy;


end MechanicalDriveSystem;
```

DETAILED EXPLANATION OF SYSTEM CODE

1. ConstantForce force(F=15);

Applies a constant 15 N force to the system.
This is the input driving force.

2. ExtendedMass massBlock(m=3);

A 3 kg mass that:

• Moves under applied force
• Computes kinetic energy
• Computes momentum

3. Spring springElement(c=1500);

Spring stiffness = 1500 N/m

This stores mechanical energy when stretched.

4. Damper damperElement(d=8);

Damping coefficient = 8 Ns/m

This dissipates mechanical energy as heat.

5. Fixed fixedSupport;

Provides a fixed mechanical reference point.

CONNECT STATEMENTS EXPLAINED

connect(force.flange, massBlock.flange_a);

Force acts on the mass.

connect(massBlock.flange_b, springElement.flange_a);

Mass connected to spring.

connect(springElement.flange_b, damperElement.flange_a);

Spring connected to damper.

connect(damperElement.flange_b, fixedSupport.flange);

Damper connected to ground.

This forms:

Force → Mass → Spring → Damper → Fixed Support

A classical mass–spring–damper system.

ENERGY EQUATIONS EXPLAINED

potentialEnergy = 0.5 * k * s^2

This calculates spring stored energy.

dampingPower = d * v^2

This calculates power dissipated by damper.

totalEnergy = kinetic + potential

This tracks system mechanical energy.

WHAT THE COMPLETE SYSTEM IS ACTUALLY DOING

1. Force pushes the mass.
2. Mass accelerates.
3. Spring resists motion and stores energy.
4. Damper removes energy from system.
5. Motion gradually stabilizes.

You observe oscillation that decays over time due to damping.

PART C – SIMULATION

MAJOR STEP 4 – SIMULATION SETTINGS

Start Time = 0
Stop Time = 10

Simulate the model.

Plot the following:

massBlock.s (Displacement) massBlock.v (Velocity) massBlock.kineticEnergy potentialEnergy totalEnergy dampingPower