Lab Exercise 7 - Hierarchical & Modular Modeling of Mechanical System (Text View Method)

1. AIM

To design a hierarchical and modular mechanical system in OpenModelica using Text View by creating reusable mechanical subsystems and integrating them into a top-level model.

2. OBJECTIVES

After completing this lab, students will be able to:

Create reusable mechanical Modelica models
Build hierarchical mechanical structures
Connect mechanical subsystems properly
Simulate a modular mechanical system
Understand advantages of modular modeling

3. SOFTWARE REQUIREMENT

OpenModelica (OMEdit)

---

4. SYSTEM TO BE DEVELOPED

A Modular Mechanical System consisting of:

• Force Subsystem

• Mass Subsystem

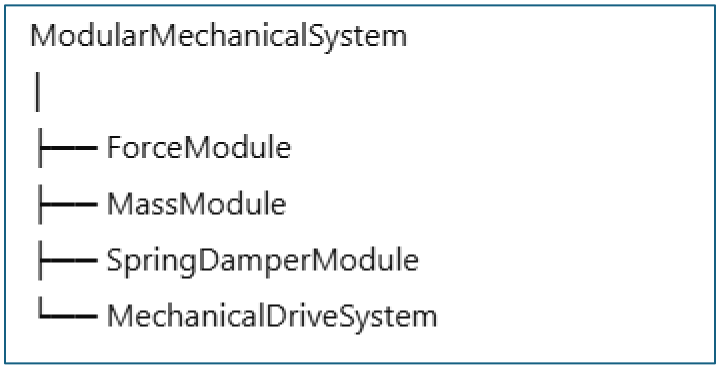
• Spring-Damper Subsystem

• Top-Level Integrated Mechanical Model

The system represents a translational second-order mechanical system.

---

5. STEP-BY-STEP PROCEDURE (TEXT VIEW METHOD)



```
ModularMechanicalSystem
│
├── ForceModule
├── MassModule
├── SpringDamperModule
└── MechanicalDriveSystem
```

STEP 1: CREATE A NEW PACKAGE

Open OMEdit

Click File → New Modelica Class

Name the class: **ModularMechanicalSystem**

Select restriction: **package**

Click OK

Switch to Text View

Leave the package empty initially.

STEP 2: CREATE FORCE SUBSYSTEM

Right-click the package → New Modelica Class

Name: **ForceModule**

Restriction: **model**

Switch to Text View

Paste the following code:

```
model ForceModule
  Modelica.Mechanics.Translational.Sources.ConstantForce force(f=150);
  Modelica.Mechanics.Translational.Interfaces.Flange_b outputFlange;


equation
  connect(force.flange, outputFlange);
end ForceModule;
```

This subsystem provides a constant force of 150 N and acts as a reusable force module.

STEP 3: CREATE MASS SUBSYSTEM

Right-click package → New Modelica Class

Name: **MassModule**

Restriction: **model**

Switch to Text View

Paste the following code:

```
model MassModule


  Modelica.Mechanics.Translational.Components.Mass mass(m=5);
  Modelica.Mechanics.Translational.Interfaces.Flange_a inputFlange;
  Modelica.Mechanics.Translational.Interfaces.Flange_b outputFlange;
```

```
equation
  connect(inputFlange, mass.flange_a);
  connect(outputFlange, mass.flange_b);


end MassModule;
```

This subsystem represents a mass of 5 kg with input and output mechanical connectors.

STEP 4: CREATE SPRING-DAMPER SUBSYSTEM

Right-click package → New Modelica Class

Name: **SpringDamperModule**

Restriction: **model**

Switch to Text View

Paste the following code:

```
model SpringDamperModule


  Modelica.Mechanics.Translational.Components.Spring spring(c=1000);
  Modelica.Mechanics.Translational.Components.Damper damper(d=50);
  Modelica.Mechanics.Translational.Components.Fixed fixed;


  Modelica.Mechanics.Translational.Interfaces.Flange_a inputFlange;


equation
```

```
  connect(inputFlange, spring.flange_a);

  connect(spring.flange_b, damper.flange_a);

  connect(damper.flange_b, fixed.flange);


end SpringDamperModule;
```

This subsystem represents a spring (1000 N/m), damper (50 Ns/m), and fixed support.

STEP 5: CREATE TOP-LEVEL MECHANICAL SYSTEM

Right-click package → New Modelica Class

Name: **MechanicalDriveSystem**

Restriction: **model**

Switch to Text View

Paste the following code:

```
model MechanicalDriveSystem


  ForceModule forceModule;

  MassModule massModule;

  SpringDamperModule springDamperModule;

equation

  connect(forceModule.outputFlange, massModule.inputFlange);

  connect(massModule.outputFlange, springDamperModule.inputFlange);
```

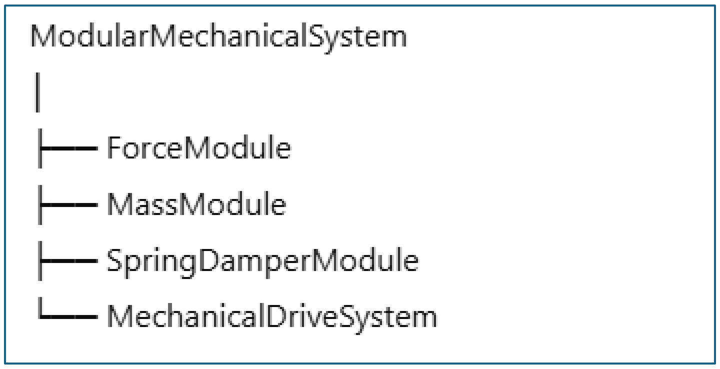
```


end MechanicalDriveSystem;
```

This top-level model connects Force → Mass → Spring-Damper in a hierarchical manner.

STEP 6: VERIFY PACKAGE STRUCTURE

Your final structure must appear as:



```
ModularMechanicalSystem
│
├── ForceModule
├── MassModule
├── SpringDamperModule
└── MechanicalDriveSystem
```

STEP 7: SIMULATION PROCEDURE

Right-click MechanicalDriveSystem

Click Simulation Setup

Set:

Start Time = 0
Stop Time = 10

Click Simulate

STEP 8: RESULTS TO PLOT

After simulation, plot:

massModule.mass.s (Displacement)

massModule.mass.v (Velocity)

massModule.mass.a (Acceleration)