Simulation of Vehicle Drivetrain with Modelica

Dynamic Simulation in Vehicle Engineering 2012
Contents

- Modeling language Modelica
- Simulation tool Dymola
- SmartElectricDrives Library
- PowerTrain Library
- VehicleDynamics Library
Modeling and Simulation

- **Modeling:**
  Simplifying description of reality by means of
  - Algebraic equations
    Resistance: \( v = R \cdot i \)
  - Ordinary (time dependent) differential equations
    Inductance:
    \( v = L \cdot \frac{di}{dt} \)

- **Simulation:**
  Solving the DAE (Differential Algebraic Equation system) with suitable Solvers
**Solver**

- **Simple**: Euler (Runge-Kutta 1\(^{st}\) order)
  
  \[
  \frac{dy}{dt} = \dot{y} = f(y,t)
  \]

  - Given:
  - Solve: \( y(t + dt) = y(t) + \dot{y} \cdot dt \)
  - Take care of the stepsize \( dt \)!

- **Important**: consistent solution of the whole system after each step

- **Standard**: Dassl
  - Multistep
  - Variable stepsize
  - Suitable for stiff systems
Modelica

- Modeling language:
  - Object-oriented
  - Acausal
  - Multiphysical
- Modelica Standard Library
Developed and maintained by the Modelica Association  www.modelica.org
All proceedings / papers available online!

Literature:
- Michael Tiller, Introduction to Physical Modeling with Modelica
  Kluwer Academic Publisher
- Peter Fritzson, Principles of Object Oriented Modeling and Simulation with Modelica
  Wiley IEEE press
Object-oriented Modelling

- Each icon represents a physical component, e.g. electrical resistance
- Connectors are interfaces to the surrounding, e.g. electrical plug
- Connections represent physical links, e.g. electrical wire

Anton Haumer
Connector = physical Interface

- Defines potential and flow - variables
- Connected connectors have the same potential (mesh rule).
- Flow-variables fulfill the node rule.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Potential</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrotechnics</td>
<td>Voltage / Potential</td>
<td>Current</td>
</tr>
<tr>
<td>Rotatory Mechanics</td>
<td>Angle</td>
<td>Torque</td>
</tr>
<tr>
<td>Translatory Mechanics</td>
<td>Position</td>
<td>Force</td>
</tr>
<tr>
<td>Thermodynamics</td>
<td>Temperature</td>
<td>Heat flow</td>
</tr>
<tr>
<td>Fluid flow</td>
<td>Pressure</td>
<td>Mass flow</td>
</tr>
</tbody>
</table>

...
Acausal modeling

- Equations vs. Assignments
- Example: electrical resistance
  - Assignment: Unknown depending on usage!
    ```algorithm
    i := v/R;
    v := R*i;
    ```
  - Equation: solved by the tool equation
    ```
    R\cdot i = v;
    ```
- Derivative operator
  - Example: electrical inductance
    ```
    v = L*\text{der}(i);
    ```
Multiphysical Modeling

Example: vehicle model
- Vehicle Dynamics (3D mechanics)
- Drive train (1D mechanics)
- Combustion engine
- Air conditioning (thermo-fluid)
- Hydraulics
- Electrical systems and drives
- Controller
- …
Libraries

Commercial libraries:
- SmartElectricDrives
  Electric drives, power electronics and control
- PowerTrain
  1D mechanical power transmission (automotive)
- VehicleDynamics
  3D mechanics for vehicles’ dynamic behaviour
- AirConditioning
  Air conditioning cycles
- Other …
Dymola

- Modeling- and simulation – tool based on Modelica
  - Graphical editor
  - Symbolic transformation + translation
  - Simulation and graphical representation
  - Windows or Linux

Developed and maintained by Dassault Systemes AB Lund, a subsidiary of Dassault Systemes (Catia), based on Hilding Elmqvist’s work, Lund, Sweden
Translation

- LookUp: Search the class definitions of instantiated components.
- Creating “Flat Modelica Code” (i.e. one big DAE) and conditioning of the system.
- Translating Modelica to C – Code.
- Compiling C – Code with Microsoft Visual C/C++ (Express).
- Linking program control and solver, creating an executable model.
- The simulation GUI calls this model and reads the results for plotting.
Dymola – Modeling

- Icon
- Diagram
- Code/Text
- Opened Libraries
- Documentation
- Toggle Modeling/Simulation
Dymola – Simulation

Translate

Simulate

Settings

Plot results

Parameters & Variables

Toggle Modeling/Simulation

Anton Haumer
Variables

- **Basic types:**
  - Real continuous
  - Integer discrete
  - Boolean false, true
  - String

- **Prefixes:**
  - constant
  - parameter
  - discrete only changed at “events”
  - input, output

- **Derived Types:**
  - Physical quantities with (SI-) units, pre-defined in Modelica.SIunits
e.g. Modelica.SIunits.Current
type ElectricCurrent = Real (final quantity="ElectricCurrent", final unit="A");
  - Further attributes: min, max, start, …
Inheritance

Goal:  avoid redundant code (maintenance)

- partial / extends
  - Definition of an incomplete model
    ```
    partial model OnePort
        PositivePin p;
        NegativePin n;
        Modelica.SIunits.Voltage v = p.v - n.v;
        Modelica.SIunits.Current i = p.i;
    equation
        p.i + n.i = 0;
    end OnePort;
    ```

- Usage:
  ```
  model Resistor extends OnePort;
  equation
      v = R*i;
  end Resistor;
  ```

  ```
  model Inductor extends OnePort;
  equation
      v = L*der(i);
  end Inductor;
  ```
Object oriented modeling of electric machines

- Electric machine = electro mechanical energy converter
- Interfaces:
  - Electrical terminals (windings)
  - Mechanical flanges (shaft and housing)
  - Thermal ports: dissipating losses, defining operational temperatures
- Each separable effect is encapsulated:
  - Winding resistances
  - Winding stray inductances
  - Airgap model:
    - Magnetic coupling between stator and rotor (main field inductance)
    - Torque generation
  - Loss models: core losses, friction losses, stray load losses
Object oriented modeling of electric machines

[Diagram of electric machines]
Inverter: controlled speed

Asynchronous  

\[ \omega_m = \frac{(1 - s) \cdot 2\pi f}{p} \]

⇒ speed control by stator frequency

In the region of base speed the magnetic flux is held constant.

Limits: induced voltage = magnetic flux * speed

If induced voltage exceeds maximum voltage, the control has to reduce the flux

⇒ field weakening

Torque = Magnetic flux * Current

Synchronous  

\[ \omega_m = \frac{2\pi f}{p} \]
Inverter: controlled speed

constant flux / torque

field weakening
Inverter: Field Oriented Control

- Torque = Magnetic Flux \times Current
  - maximum, if magnetic flux \perp current
  - i.e. current phasor is **oriented** with respect to the magnetic field
  - Control of torque by means of active current control

- Magnetic field is generated by:
  - Synchronous machine: Permanent magnets
    - Field weakening by means of reactive current: limited
    - Position of rotor (i.e. magnetic field) : Position encoder
  - Asynchronous induction machine:
    - Magnetic field has to be controlled by means of reactive current control
    - Position of magnetic flux: Machine model
### SmartElectricDrives: Different levels of abstraction

<table>
<thead>
<tr>
<th>Models of controlled machines</th>
<th>Quasi stationary models (only mechanical transients)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical transients and electrical transients, including power balance converter</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Converters</th>
<th>Power balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal switches</td>
<td></td>
</tr>
</tbody>
</table>
Electric Drive Structure

Anton Haumer
Compare results: power balance vs. ideal switching

Anton Haumer
PowerTrain

- 1-dimensional components for a drivetrain
- Modeling longitudinal dynamics of a car
- Performance
- Consumption
- Hybridization concepts
PowerTrain: Car Consumption

[Graph showing consumption data with axes labeled and data points plotted.]
VehicleDynamics

- 3D driveability
Thank you very much for your attention!

A.Haumer@Haumer.at
www.Haumer.at
www.catia-austria.at