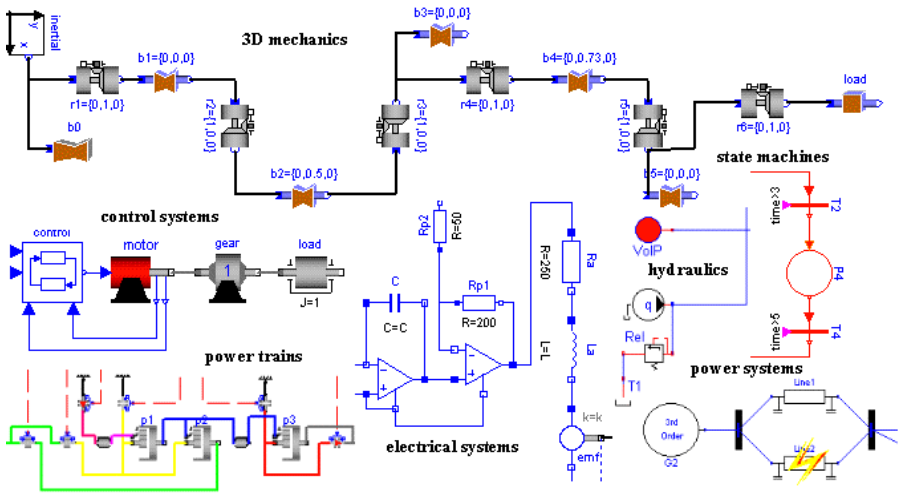
Exercises for Advanced Modelica Tutorial

Hilding Elmqvist, Dynasim
Sven Erik Mattsson, Dynasim
Martin Otter, DLR
Hubertus Tummescheit, Lund Institute of Technology

Modelica 2002, March 18-19, Oberpfaffenhofen, Germany



Exercises

Exercise 1: Modeling and initialization of a **steam superheater**
(only 5 equations, numerical properties influenced by stateSelect, initialization of higher index system, differentiating of functions)

Exercise 2: Modeling and initialization of a **robot** with **gear elasticity**
(many equations, mixed state and steady state initialization, **approximate steady state initialization**, initialization of discrete controller).

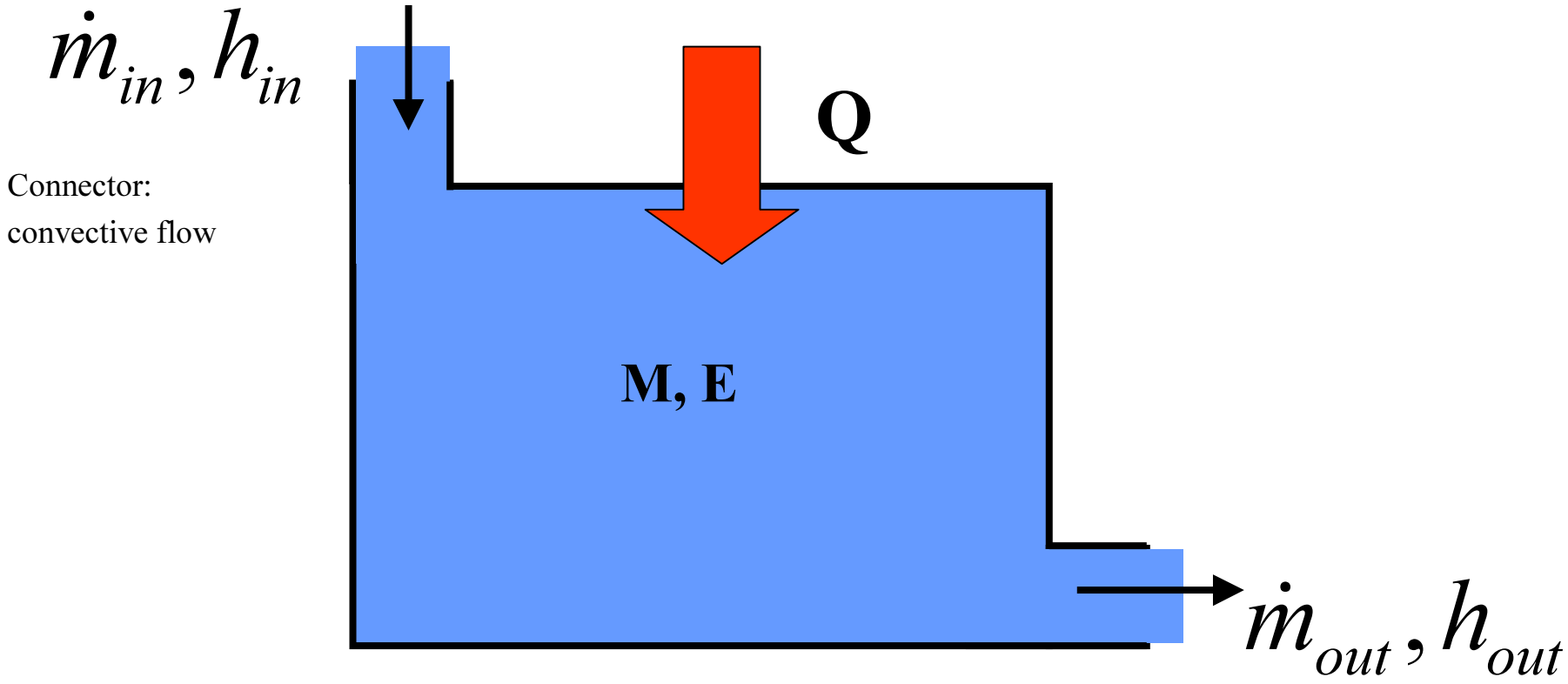
Exercise 3: Modeling of **automatic gearbox** and design of simple gear shift strategy (dynamically coupled friction elements, planetary gears)

Exercise 1: Steam Superheater

- Selection of states
- Initial conditions
- Use of derivative functions
- Structural properties for different states and initial conditions

Thermodynamic Control Volume

Control volume: First Law for Open Systems



Mass- and Energy Balances

Steam Volume Equations

$$\frac{dM}{dt} = \dot{m}_{in} + \dot{m}_{out} = dM$$

$$\frac{dE}{dt} = h_{in}\dot{m}_{in} + h_{out}\dot{m}_{out} + Q = dE$$

$$E = V\rho e,\ e = h - p/\rho$$

$$M = V\rho$$

$$\rho = g(p,h)$$

Simple model for, e.g., a steam superheater

Given model parts:

- A simple version of $\rho = g(p,h)$ in file SteamProperties.mo
- Model Equations

Exercise Step by Step

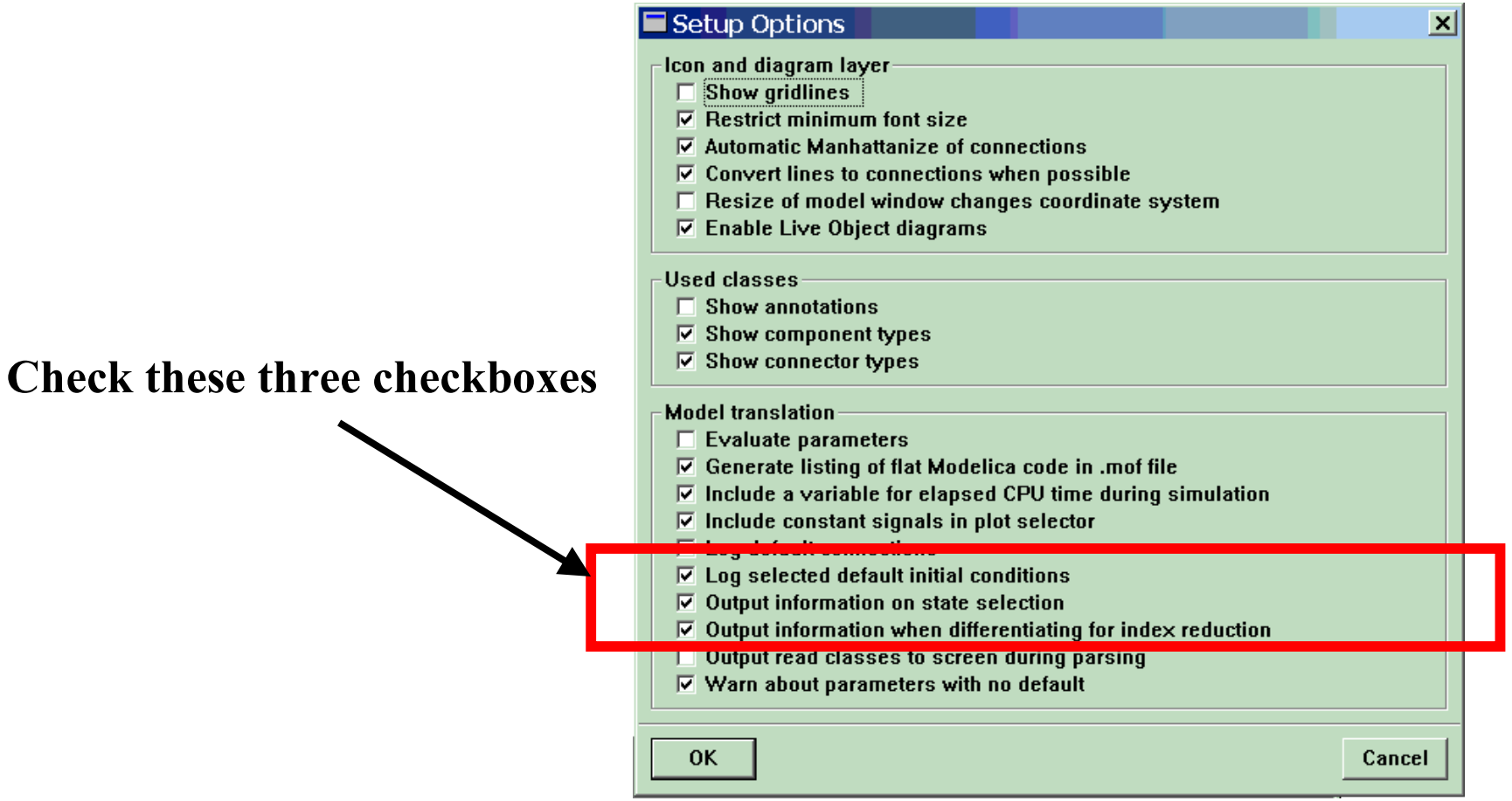
- Write a Modelica model for the given equations, use the given property function for ρ
- Find simple boundary conditions for $\dot{m}_{in}, \dot{m}_{out}, h_{in}, h_{out}, Q$ (0.0, Q0*sin(time), ...)
- Translate and run the model in the basic form.
- Use stateSelect attribute to look at 2 cases:
  - Use M and E as dynamic states (mass and energy)
  - Use p and h as dynamic states (pressure and enthalpy)
- Use of derivative functions:
  - annotation(derivative = nameOfDerFunction)
- When **`stateSelect=StateSelect.prefer`** for p, a derivative function for ρ = g(p,h) is needed. Write it and use the annotation. If the derivative function is not found, Dymola will complain that it cannot do index reduction.
- Translate and run the model when p and h are used as states.
- For steady state initialization of both cases, use **`initial equation`** section
- Use Dymolas logging features to get feedback about the equation manipulation

Dymola Advanced Logging Tips

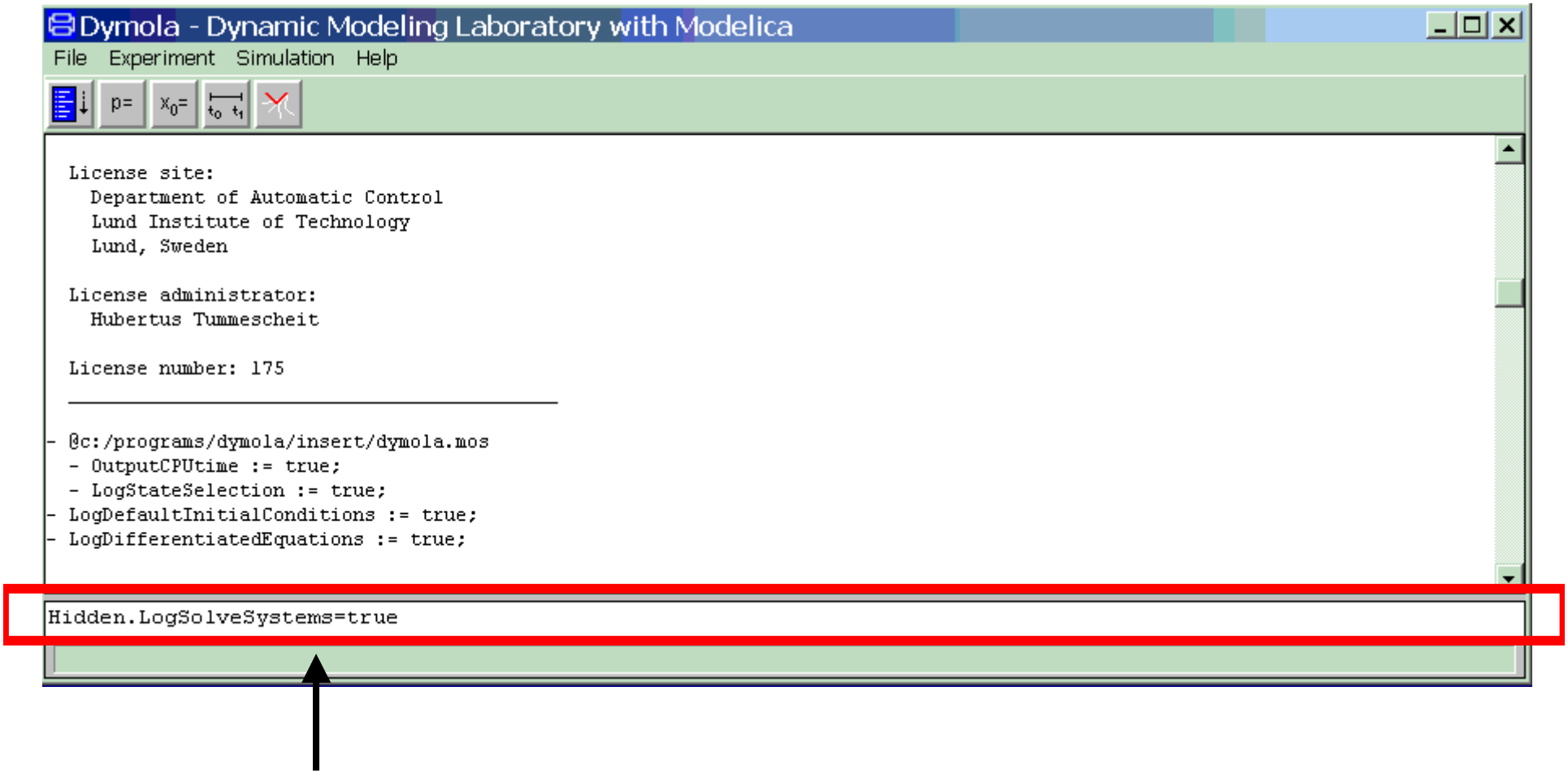
- Open Setup -Options and choose
  - Output Information on State Selection
  - Output Information when differentiating for Index reduction
- In Simulation Shell, type
  Hidden.LogSolveSystems=true
  for Verifying which variables occur in equation systems, open `'dsmodel.c'` with an editor (Notepad) and search for Non-Linear

Dymola Advanced Logging Tips (Setup-Options)

Check these three checkboxes



Dymola Logging of Equation Systems:



In the Dymola shell, set the above option

Initial conditions

- By default, the system is initialized with start values for the states
- Other cases can be specified using initial equations, e.g. for steady state for x and y:

```
initial equation
  der(x) = 0.0;
  der(y) = 0.0;
```

Derivative Functions

- Rules for derivative functions:
  - for each argument x of type Real,
    add another input der(x)
  - for each output of type Real,
    add the gradient in time direction

Derivative Functions, Example:

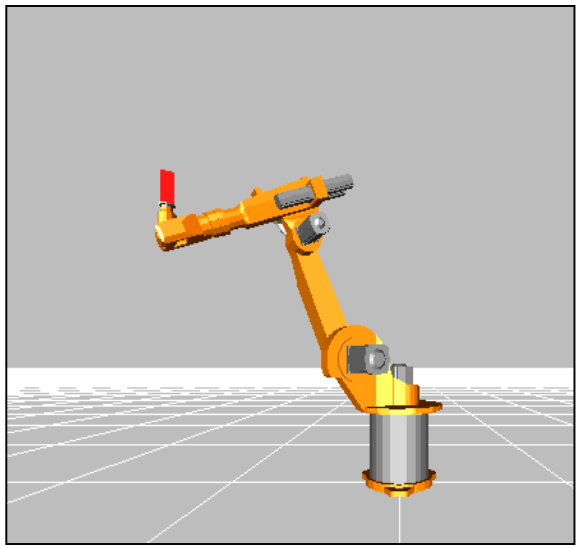
```
function foo
  input Real x, y;
  output Real z;
  annotation(derivative=foo_der);
algorithm
  z := f(x,y);
end foo;


function foo_der
  input Real x, y;
  input Real dx, dy
  output Real dz;
algorithm
  dz := foo_x*dx + foo_y*dy;
end foo_der;
```

Initialization Data:

- Data for superheated steam at 6.0 MPa (60 bar) , for initial conditions
  - Density: $\rho \approx 30 kg/m^3$
  - Specific Enthalpy: $h \approx 2.8e6 J/kg$

Exercise 2: Robot with Gear Elasticity

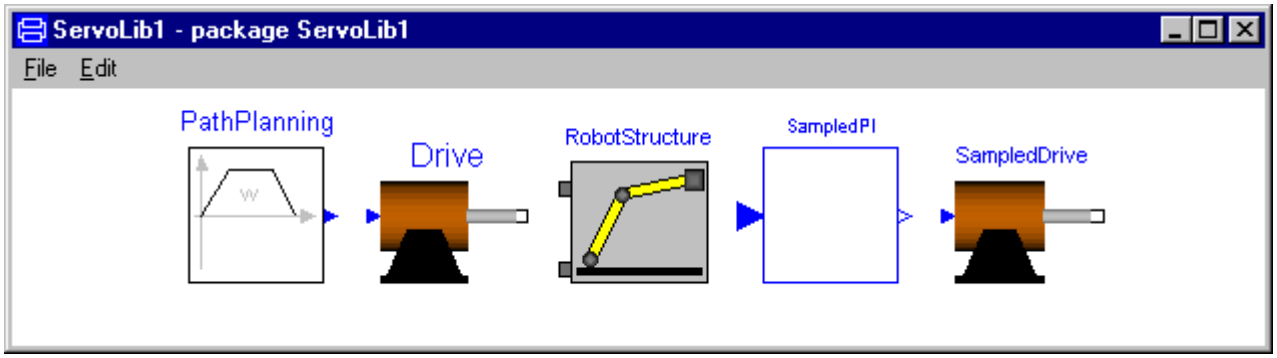
The goal of this exercise is to learn how to initialize complicated, mechanical systems in a reliable way



analyse a simplified version of an industrial robot

2.1 Problem Description

On the Tutorial CD, the library **ServoLib1.mo** is provided containing the most important components to build up robots but **without initialization**:



PathPlanning : Provide reference angles as function of time, given start and end angles and drive limitations:

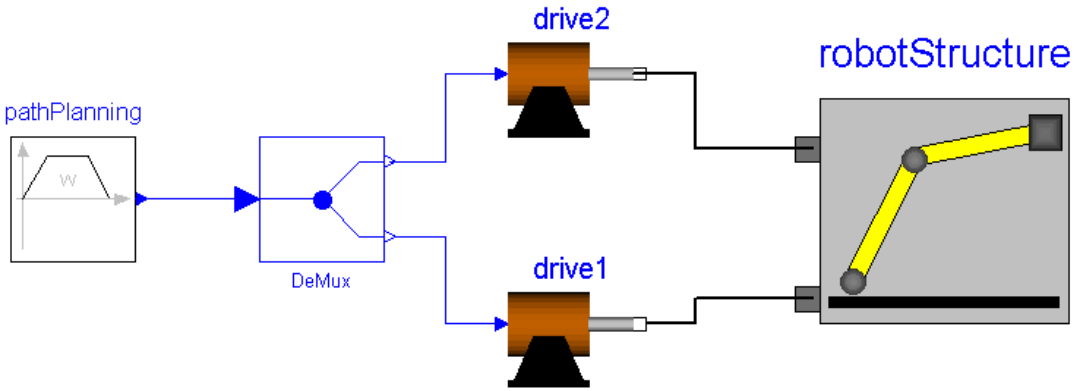
Drive : Simplified model of a drive train consisting of controller, motor and gearbox with a reference angle as input and the driven flange of the gearbox as output.

RobotStructure: 3-dim. mechanical structure of two robot arms (double pendulum) including animation information. The joint axis flanges of the mechanical structure can be accessed from the outside.

sampledPI : PI controller as sampled data system

sampledDrive : Drive with samplePI controller

In this **exercise**, you should build up a simple robot (double pendulum type) with this library



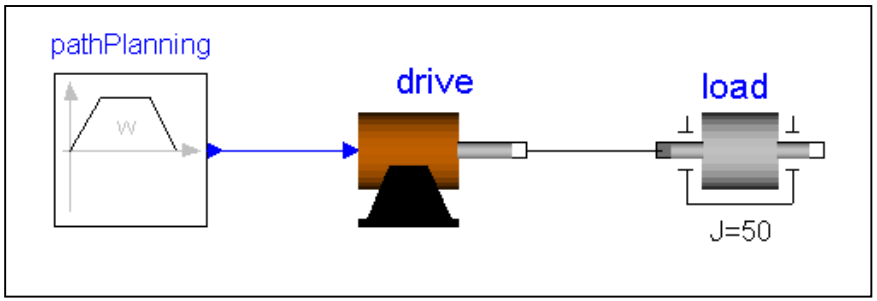
experiment with it and find a useful initialization scheme. This scheme should then be inserted into the library in order to get simple to use library components with built-in initialization.

2.2 Component Tests

Real systems are never built together and afterwards tested. Instead every used component is first tested extensively by itself, before it is used in a bigger unit. The same should also be performed for simulation models.

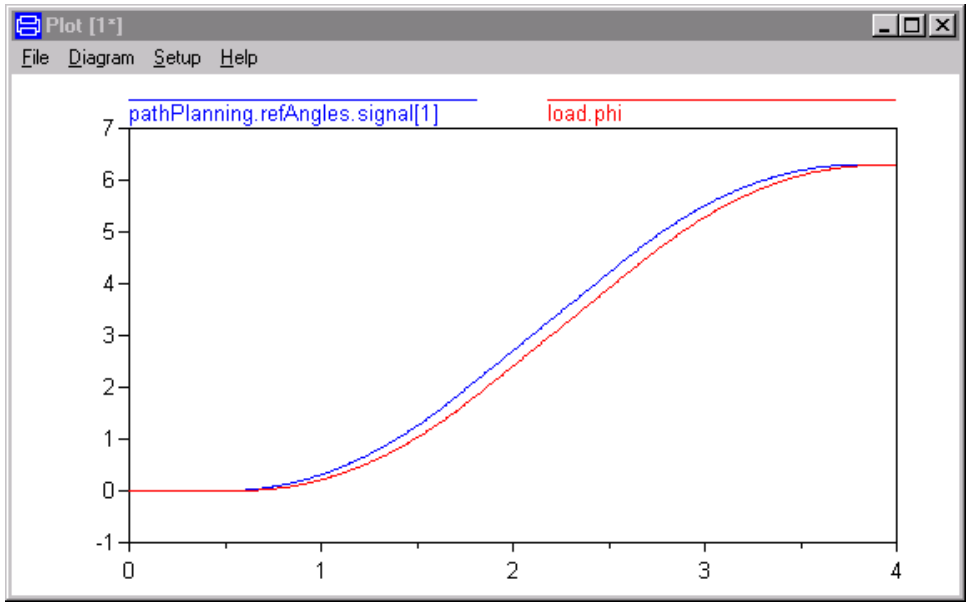
Since we do not have much time, just perform one **simple test** with the **drive** train to get used to this and the PathPlanning element:

test model:

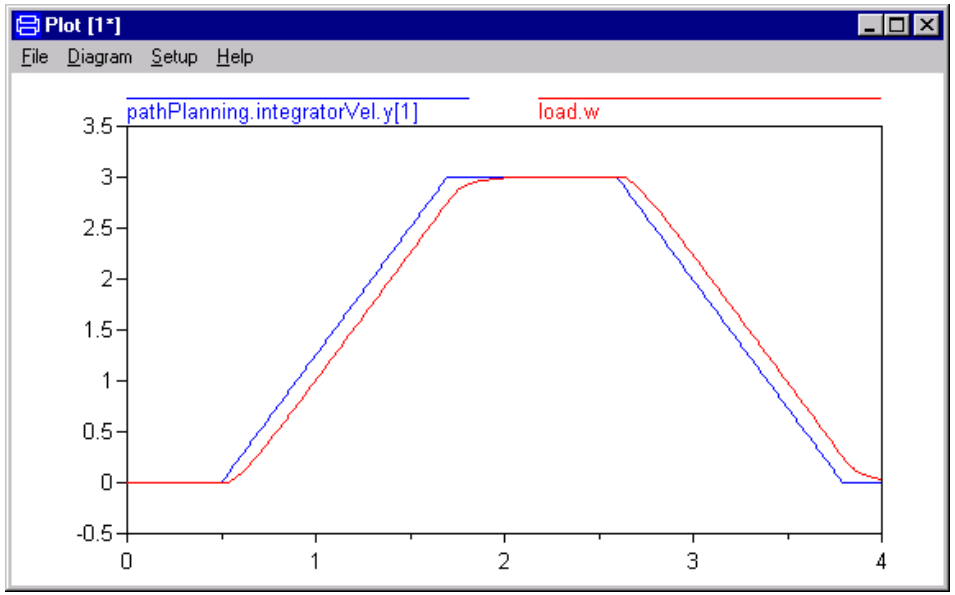


move load from 0 to 360 degree using default initialization
(set this range as parameter of "pathPlanning" and use J=50 for the load; for all other parameters use default seetings). Simulate for 4 s.

The result should look like:



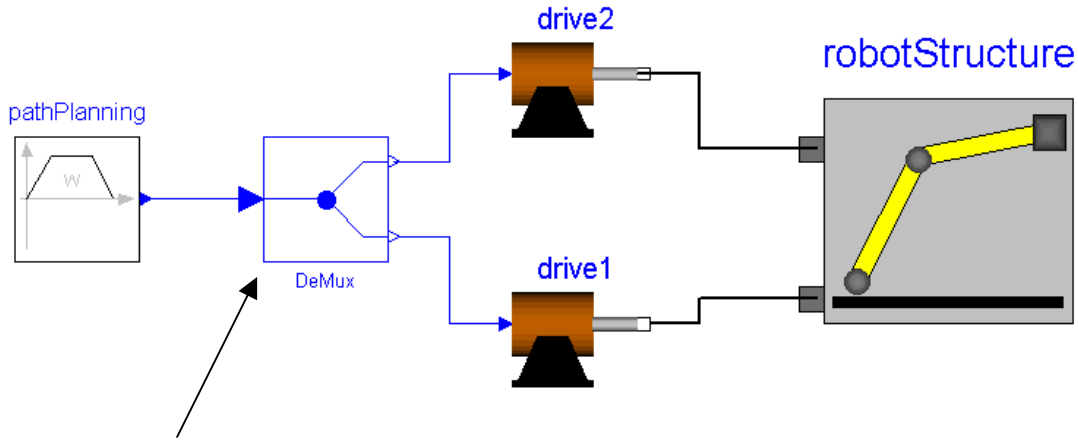
reference and actual load angle



reference and actual load speed

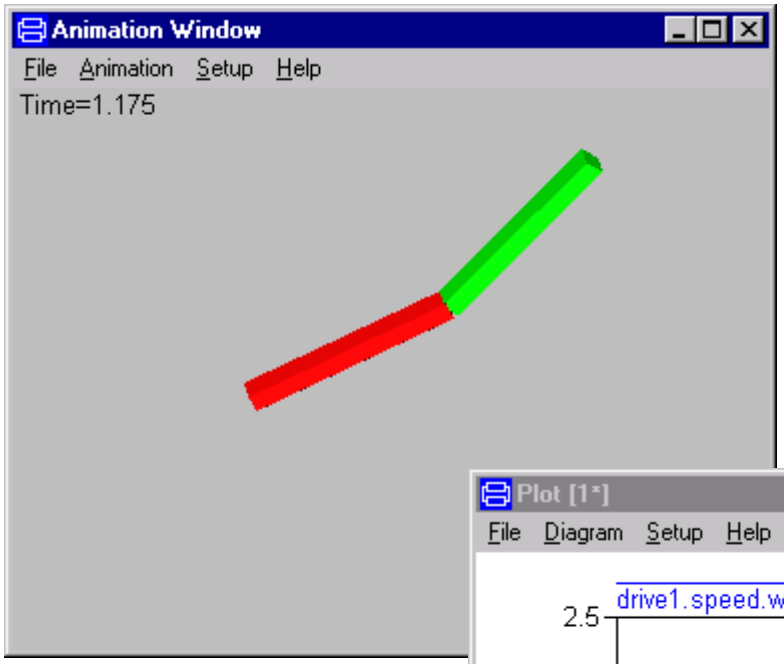
2.3 Robot with default initialization

Build robot model



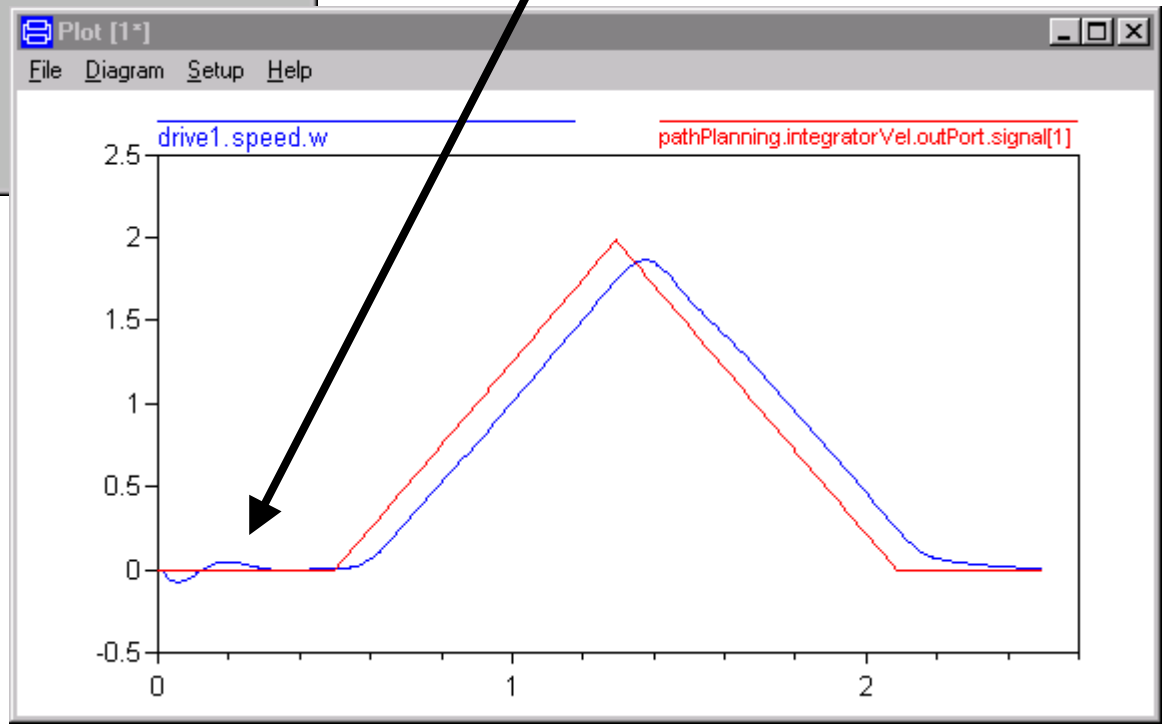
Move axes from {0,0} to {90,90} degrees using default initialization

Check **animation:**



seems to be allright

However, at the beginning when the robot should be in rest for 0.5 s, oscillations occur. Why?



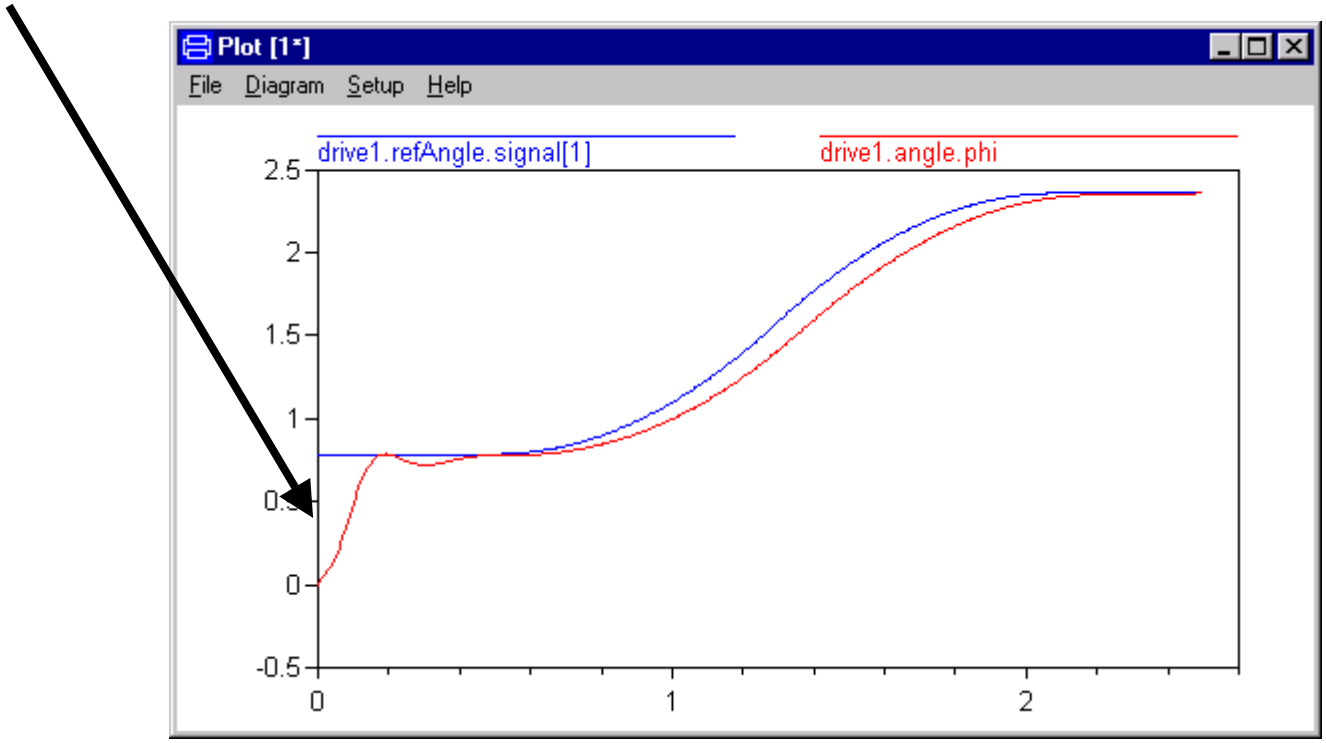
Another experiment:

Move robot from {45,45} to {135,90} degree (again using default initialization)

This time, the movement in the animation and in the plot looks odd!

What is wrong?

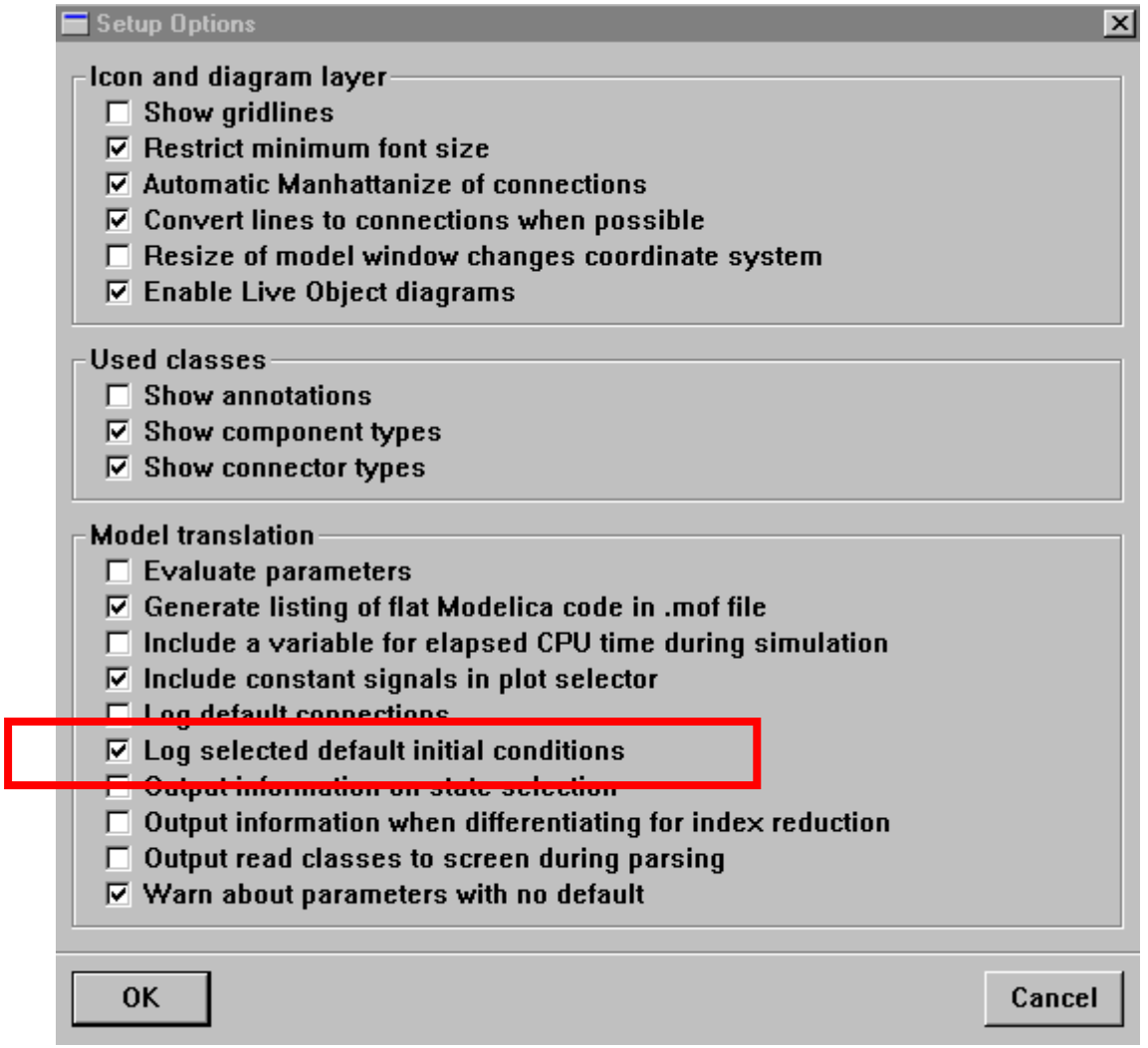
Comparing **reference** and **actual angle** of joint 1 shows, that the robot starts far away from the reference motion.



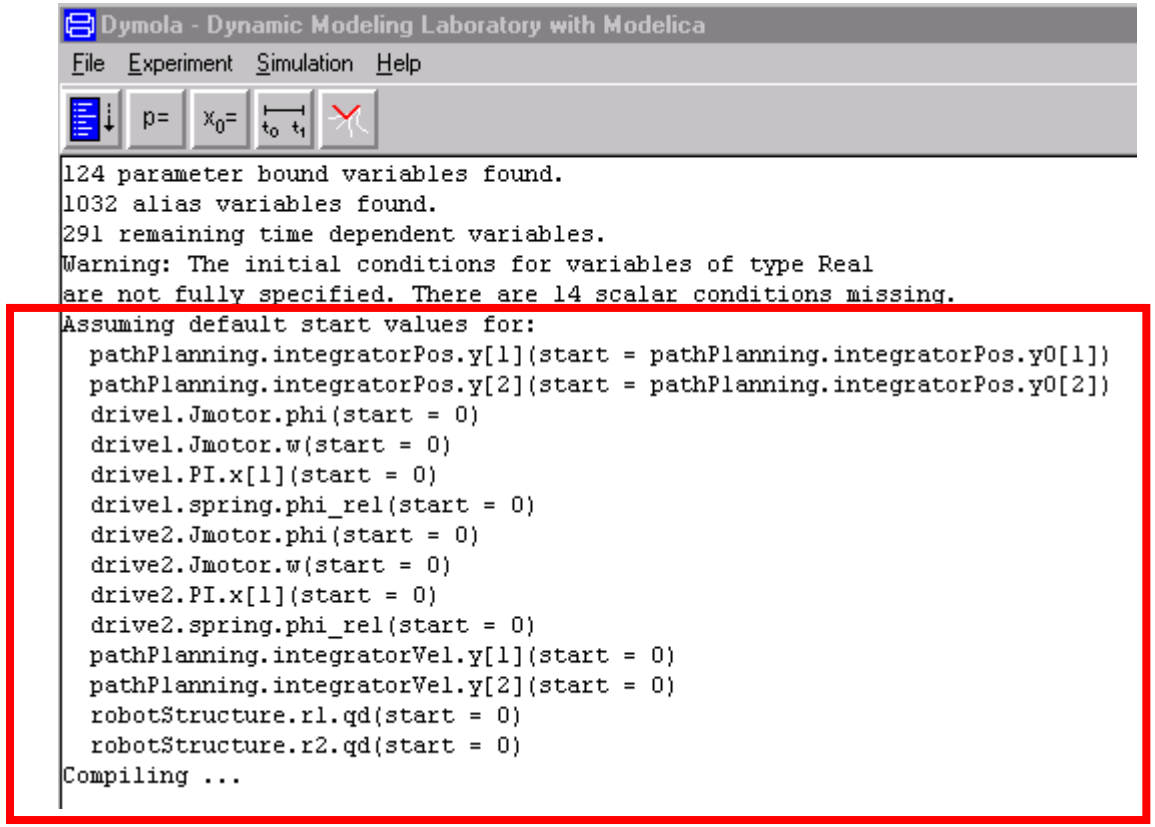
It seems natural to require that the robot should start at the initial value of the reference motion in order to avoid completely unrealistic transients.

2.4 Robot with steady state initialization

Use option "**Log selected default initial condition**" to show missing initializations:

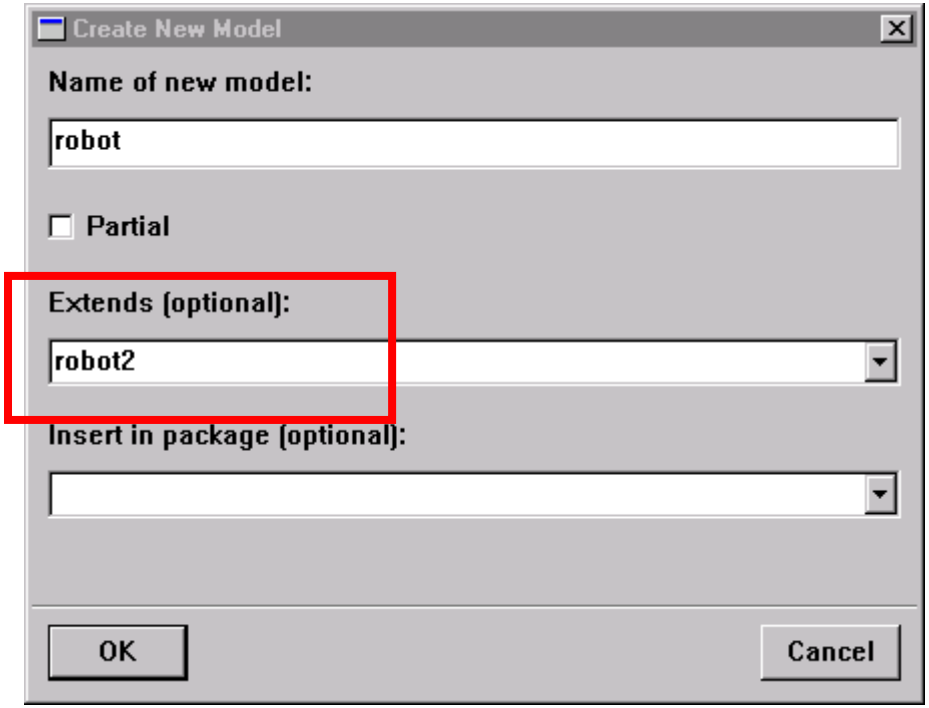


Translate robot model again with the mentioned option:

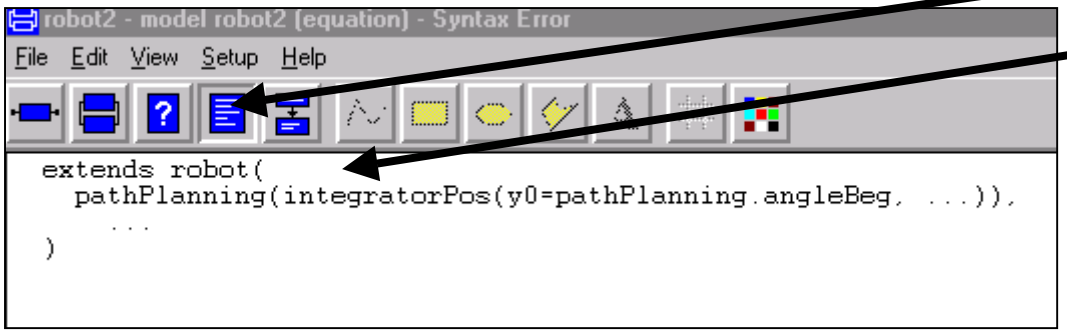


14 missing initial conditions!

In order to experiment easily with different initial conditions, the following idiom is useful:



Extend a **new model** from the robot model with default initialization



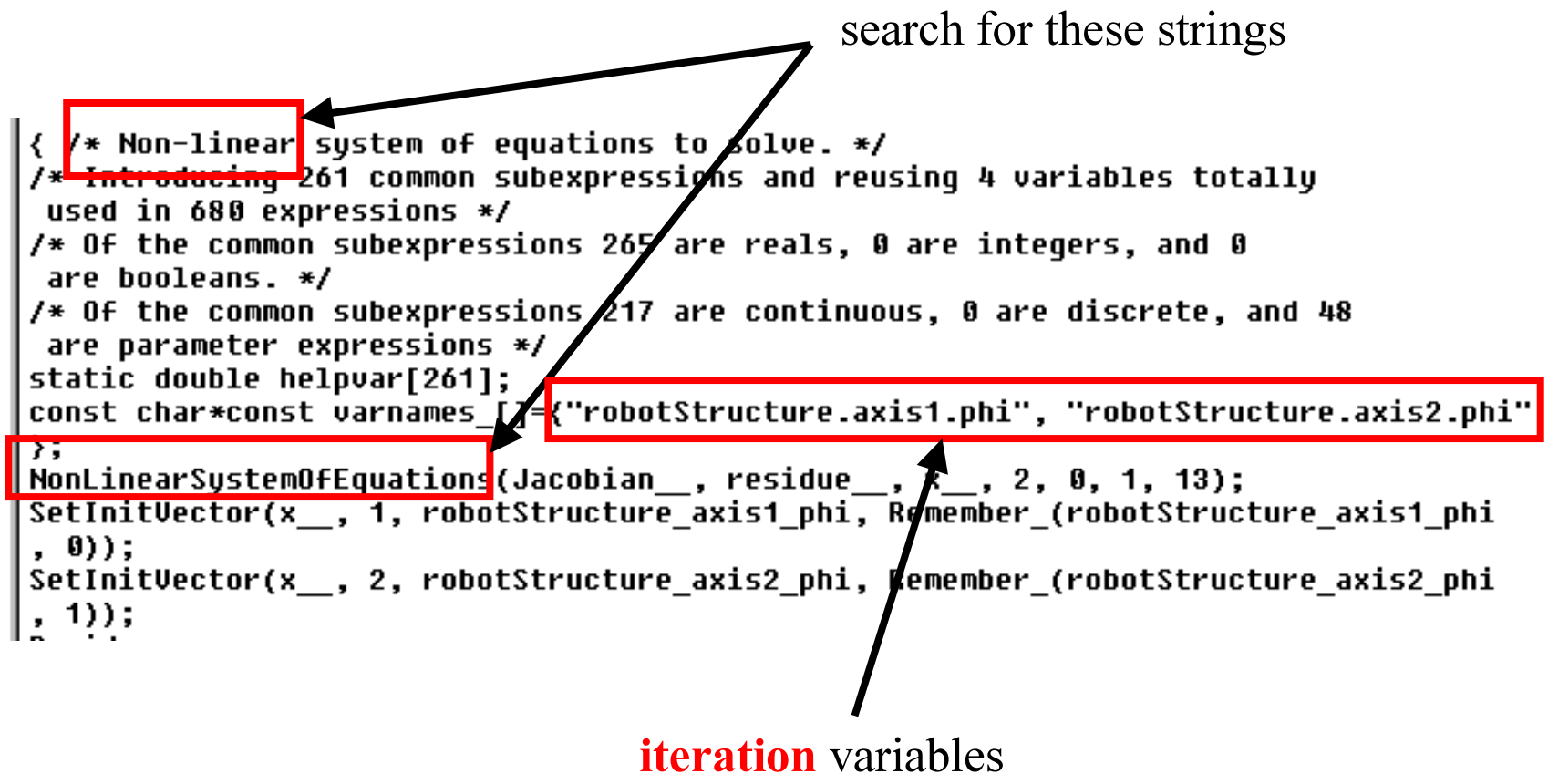
In **equation layer**, add a modification in which the initial values are defind (14). If a modification is not possible introduce an "**initial equation**"

The advantage is that the original model remains unchanged and it is clearly seen at one place how the model is modified.

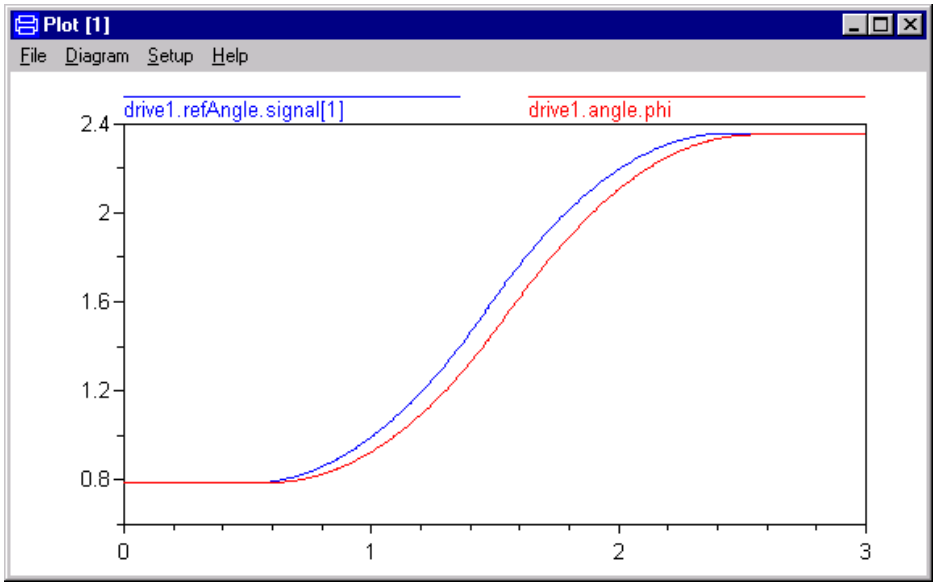
Here are some **hints** for selecting **appropriate initial conditions**:

- The **pathPlanning** component has two integrators which must be initialized. The states should be fixed at initial time, because the reference motion should start at the define values. The integrators are vectorized, therefore vectorized modifications should be given (e.g., "**each** fixed = **true**")
- All other components should be initialized in **steady state**.
- Steady state initialization leads in nearly all non-trivial cases to nonlinear systems of equations. Find out whether this is also the case here by activating the corresponding option in Dymolas command window:
  Hidden.LogSolveSystems = true
- If **nonlinear equations** appear, figure out what **iteration** variables Dymola selects by inspection of the C-code (dsmodel.c) and provide meaningful **guess** values for them, i.e., provide a start value with fixed=**false**. Otherwise, no or the wrong solution may be found.

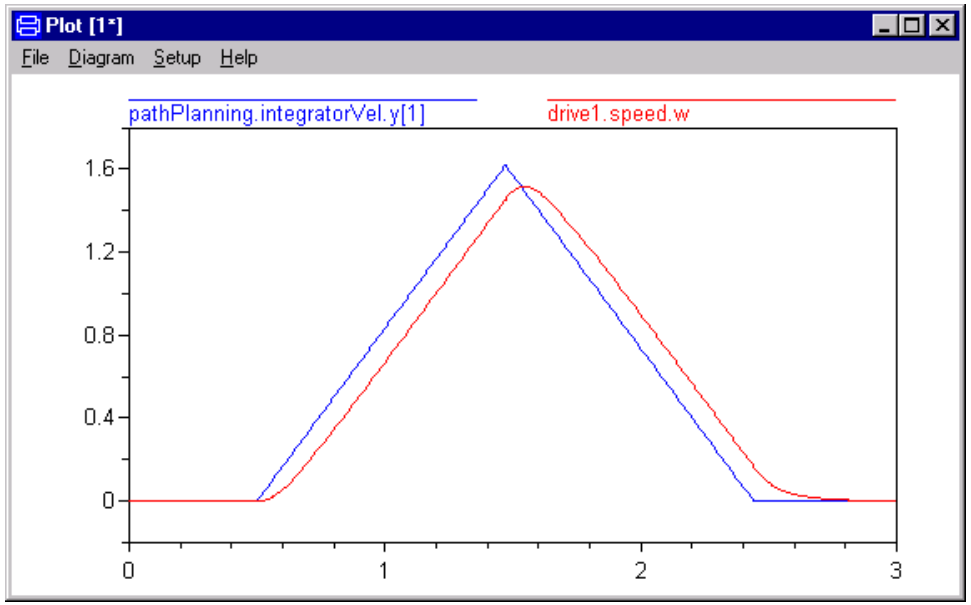
dsmodel.c for **robot2.mo**



For {45,45} to {135, -90} the solution should look as:



i.e., the model actually starts from steady state from the reference angles.



Note, the axes angles of the robot do not start at the reference angle, since the spring needs to be deflected in steady state in order to produce a torque which compensates for the gravity torque in the robot structure.

2.5 Robot with approximate steady state initialization

Mechanical systems in **tree-structure** have the following property:

When the **relative** coordinates on **position**, **velocity** and **acceleration** level for all n joints are given, the needed driving torques/forces in the joints can be computed recursively **without solving** any **system of equation** in O(n) operations.

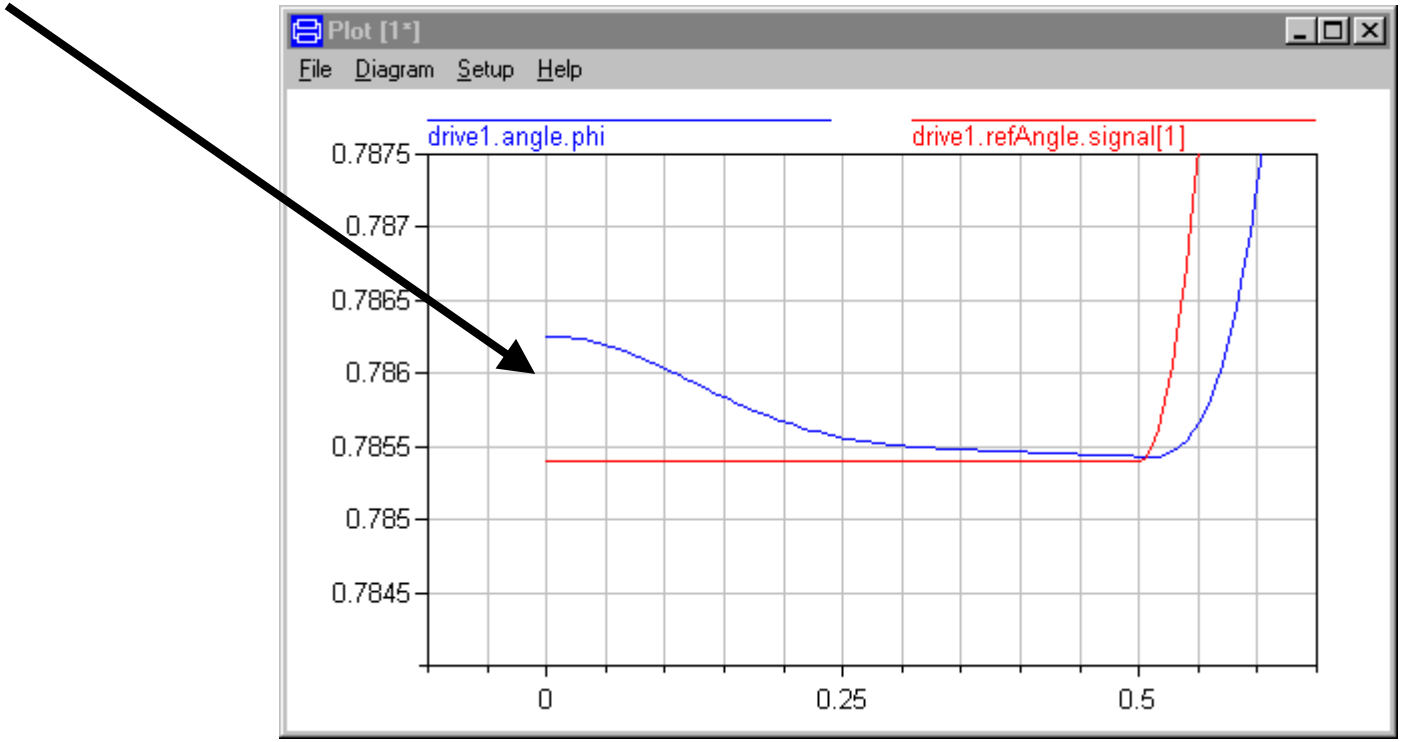
This property gives a hint how to get rid of the nonlinear system of equations during initialization, since the solution of it becomes inefficient und unreliable for bigger mechanical systems:

Initialize **approximately** in **steady state** by providing initial values for the n joint angles (and vanishing joint speed + joint acceleration for steady state). This requires to relax the steady state condition on some other variables. As a result, no nonlinear system of equations occurs and the system is initialized near steady state.

Which steady state condition could be relaxed for the robot exercise?

Remove the steady state condition on the PI-controller and add an initial condition for the joint angles of the robotStructure:

A nonlinear system of equations does no longer occur!
Still, the system is **nearly** in **steady state**:



Note, the **initialization** with a modification was only performed in order to experiment with it. Since it is now clear how initialization should be performed, it can be **built** into the **library** components. Possibly with an option to switch off all default initialization and give the user the freedom to use other initial conditions.

2.6 Robot with discrete controller and approximate steady state initialization

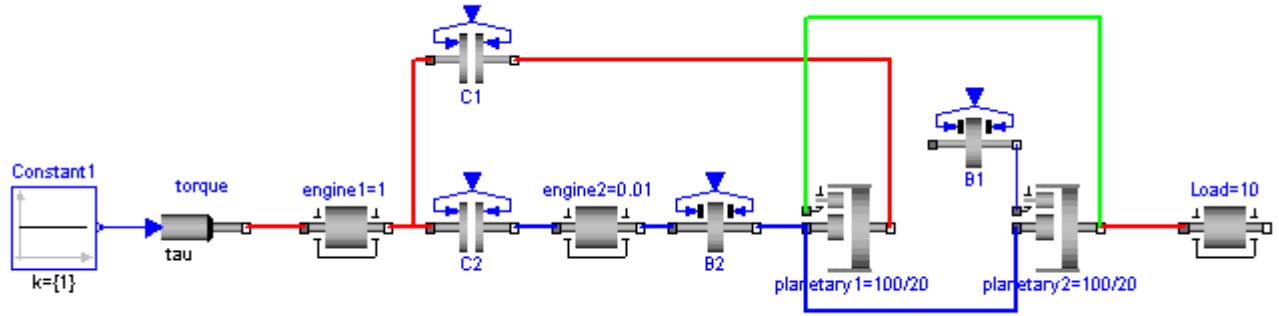
Replace the PI-controller by a **sampled PI-controller**
(= ServoLib1.sampledPI) and perform the **same initialization** as in **2.5**.

Contrary to other simulation systems, it is possible in Modelica to also initialize discrete components. E.g. here, the discrete state of the PI-controller is computed, such that the robot angles start with the reference angle.

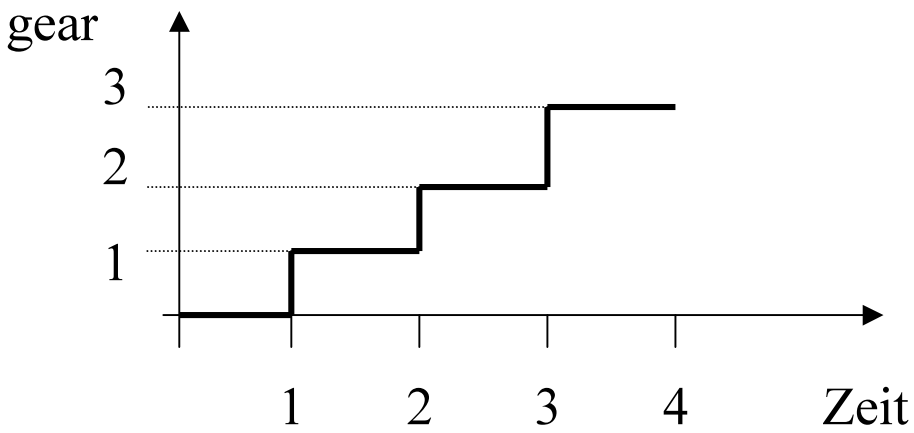
Exercise 3: Simple Automatic Gearbox

Task 1:

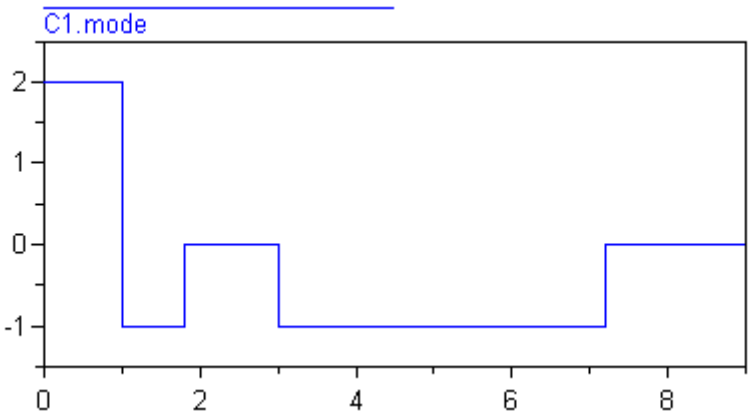
Build up the following model of a simple automatic gearbox and design a simple control for the clutches C1, C2 and brakes B1, B2, and simulate the gear shift from the figure at the lower, right part (shift from gear 1 to gear 3). Use the default value for the friction characteristics for all clutches and brakes.

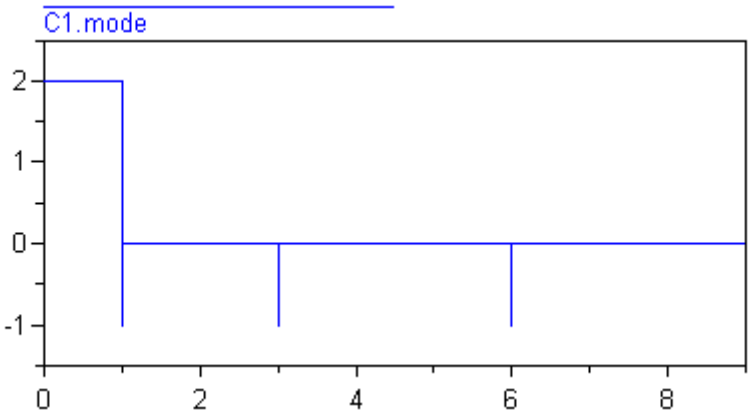


gear	C1	C2	B1	B2
0				
1	on		on	
2	on			on
3	on	on		



Task 2: Determine an appropriate contact pressure of the Clutch.





Clutch.**mode**

= **2**: clutch is **not active**

= **1**: **forward sliding**

= **0**: **stuck** (no relative motion)

= **-1**: **backward sliding**

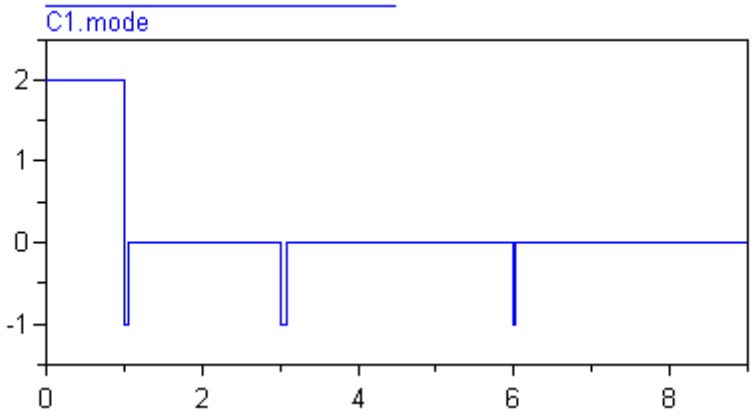
k=200

Clutch slips a lot

(heavy losses and wastage)

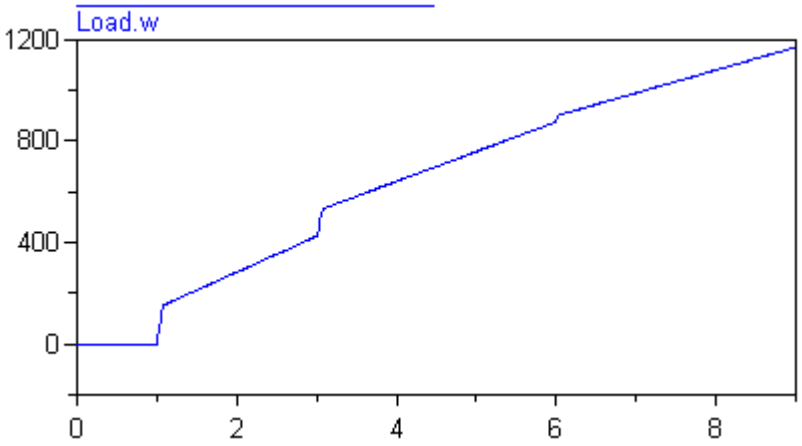
k=10000

Clutch does not slip at all
(jerky changing gears)



k = 2000

good compromise



speed of load