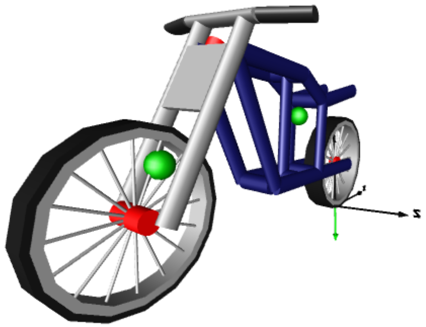
A Virtual Motorcycle Rider
Based on Automatic Controller Design

… a new and freely available Modelica Library for the purpose of simulation, analysis and control of bicycles and motorcycles

MODELICA



by
Thomas Schmitt, Dirk Zimmer and
François E. Cellier

FHV
University of Applied Sciences

Abstract

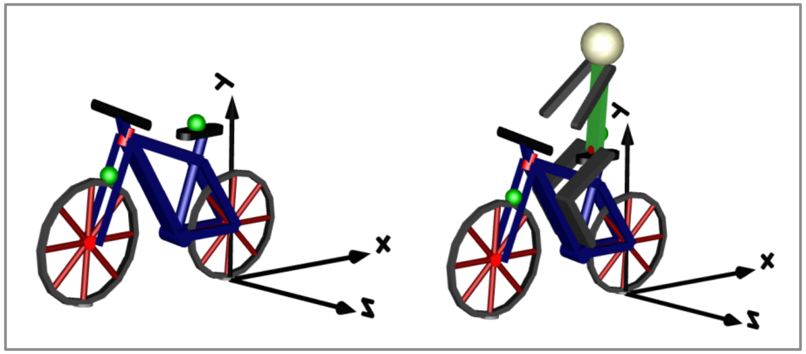
- In this presentation a new and freely available Modelica library for the purpose of simulation, analysis and control of bicycles and motorcycles (single-track vehicles) is introduced: The MotorcycleLib
- The focus of the library lies on the modeling of virtual riders based on automatic controller design
- For the vehicles, several models of different complexity have been developed
- To validate these models virtual riders are included in the library
- To this end, several test tracks are included in the library

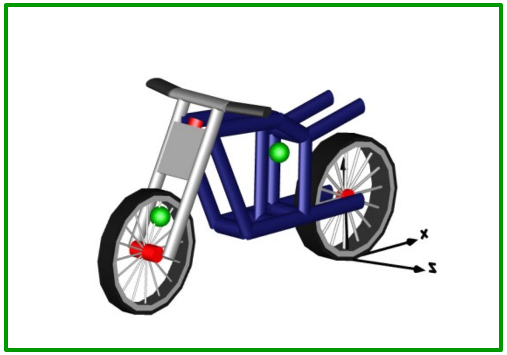
Content

1. Provided Single-Track Vehicles
2. Eigenvalue (Stability) Analysis
3. State-Space Controller Design
4. Development of a Virtual Rider
5. Conclusion

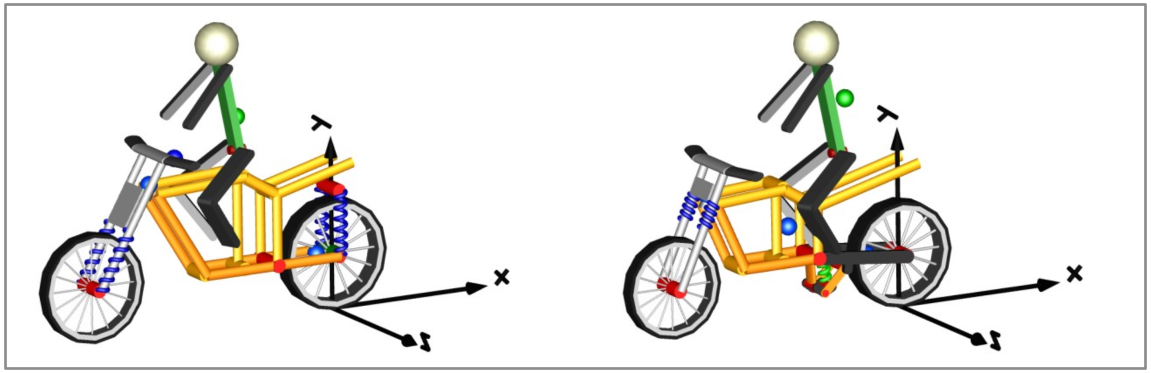
1. Provided Single-Track Vehicle Models

Basic Models (3 or 4 degrees of freedom)





Advanced Models (up to 11 degrees of freedom)

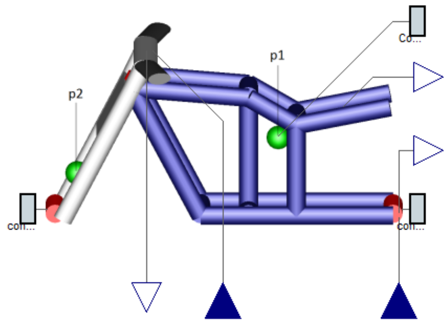


- ... the basic motorcycle model is used in this presentation in order to develop a virtual rider

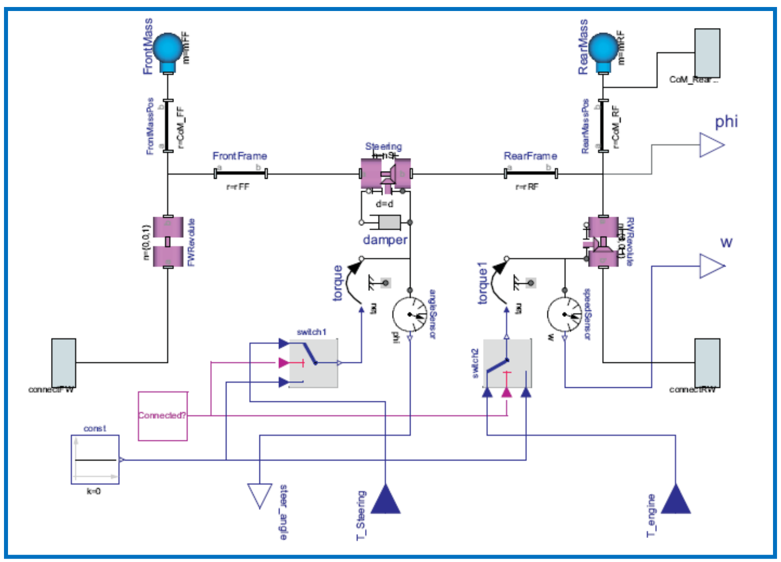
1. Provided Single-Track Vehicle Models

Basic Motorcycle Model

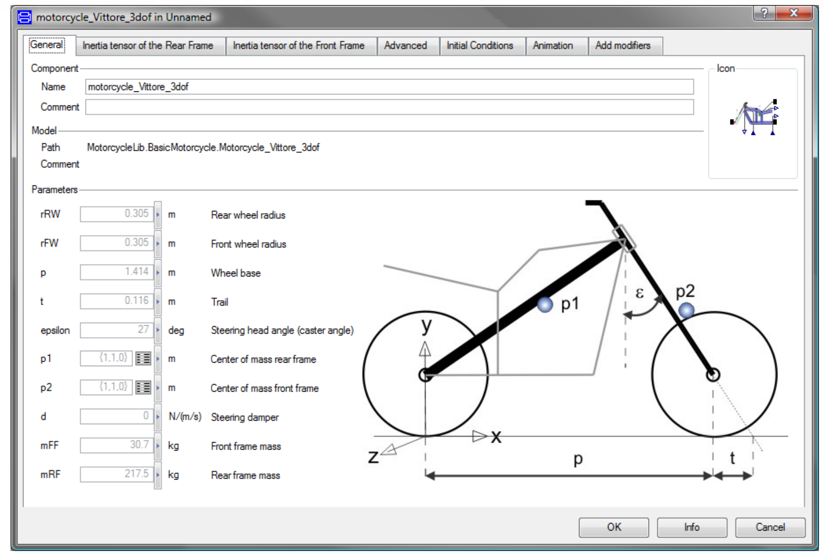
Top Layer (Wrapped Model)



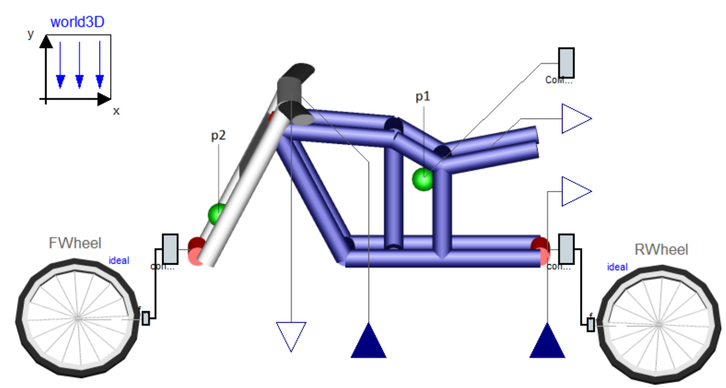
Sub Layer (Multi Bond Graphs)



Parameter Window



Complete Model (with Wheels provided by the *WheelsAndTires* Library)



2. Eigenvalue (Stability) Analysis

An eigenvalue analysis is performed in order to determine the self-stabilizing region of an uncontrolled vehicle.

➲ For this purpose the state variables of the vehicle that are responsible for the stability are of interest (see next slide).

➲ Afterwards, the corresponding eigenvalues are calculated as a function of the vehicle‘s velocity $\lambda_i = f(v_i)$

Example: 3 degrees of freedom (basic) motorcycle

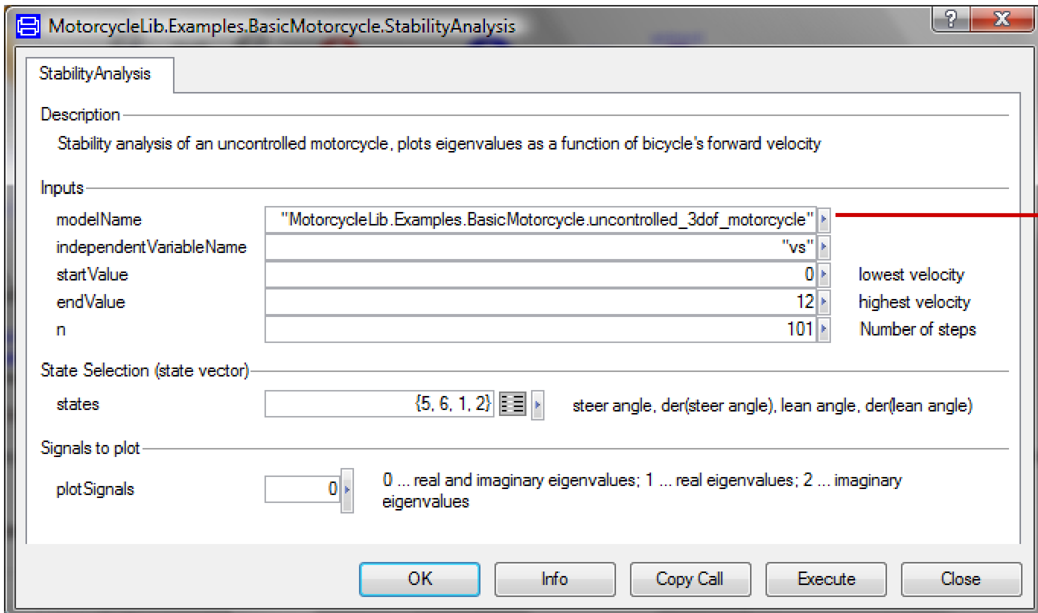
$$x = \begin{pmatrix} RWheel.xA \\ RWheel.xB \\ leanAngle \\ leanRate \\ FWRevolute.phi \\ FWRevolute.w \\ Steering.phi \\ Steering.w \\ RWRevolute.phi \\ dynamic\ state \end{pmatrix} = \begin{pmatrix} x_{long} \\ x_{lat} \\ \phi \\ \dot{\phi} \\ \varphi_{FW} \\ \dot{\varphi}_{FW} \\ \delta \\ \dot{\delta} \\ \varphi_{RW} \\ dynamic\ state \end{pmatrix}$$

$$\rightarrow \quad x_{sub} = \begin{pmatrix} \delta \\ \dot{\delta} \\ \phi \\ \dot{\phi} \end{pmatrix}$$

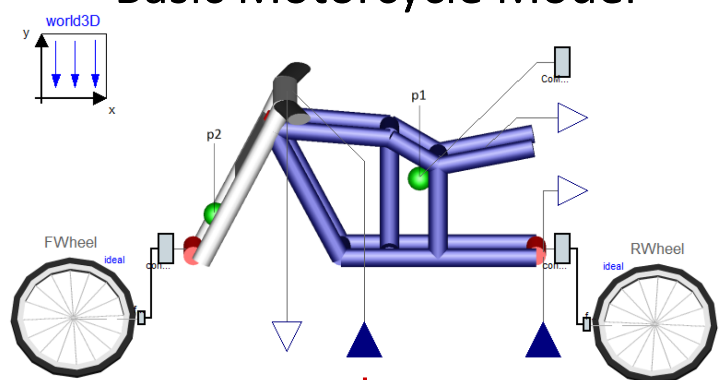
... all the other state variables have no influence on the stability and are thus irrelevant for the eigenvalue analysis.

Example: 3 degrees of freedom (basic) motorcycle

Eigenvalue Analysis Function



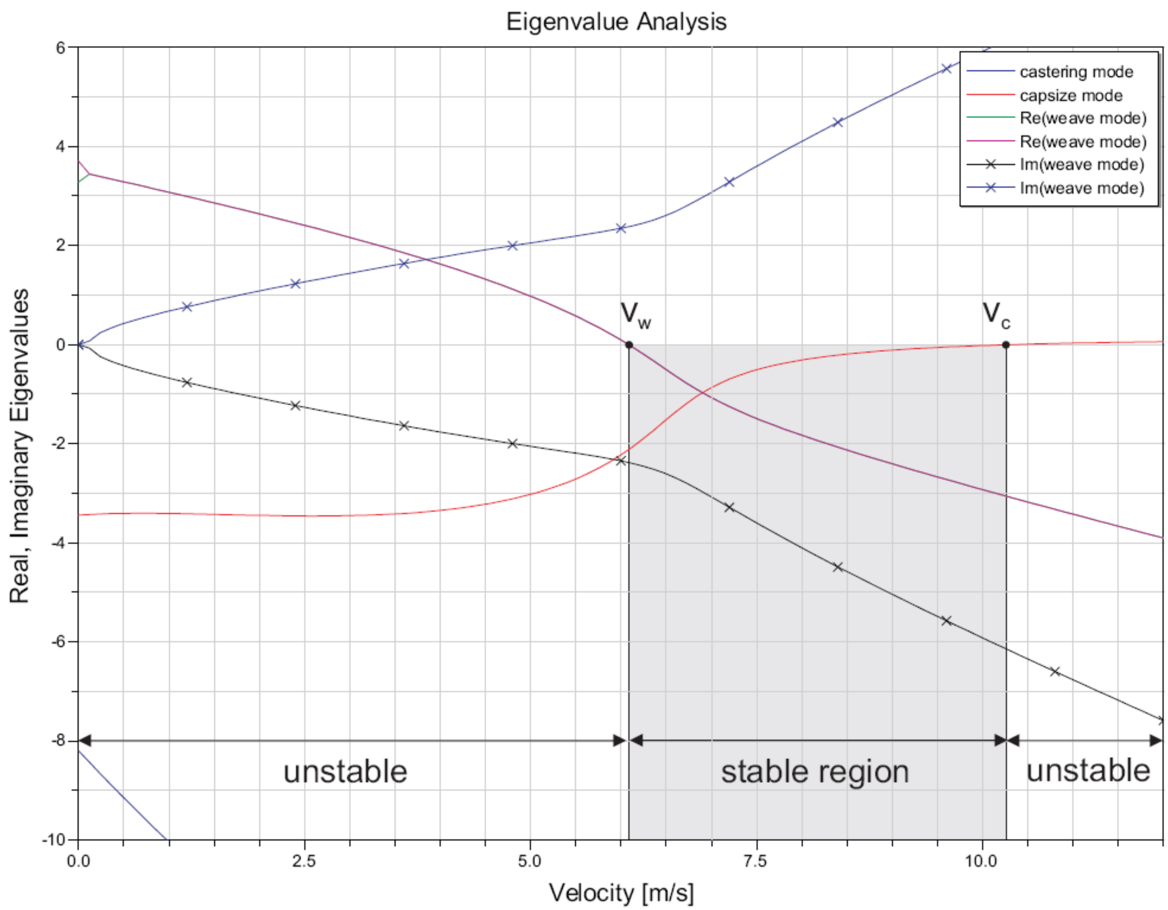
Basic Motorcycle Model




```
Arelevant := (ABCD.A)[states, states];
EigenValuesi :=
Modelica.Math.Matrices.eigenValues(Arelevant);
```

```
ABCD :=
LinearSystems.linearize(modelName);
```

Example: 3 degrees of freedom (basic) motorcycle



Result

3 different velocity ranges at which the motion of the vehicle changes qualitatively

3. State-Space Controller Design

A single-track vehicle does not remain stable on its own. For this reason, the stabilization of such a vehicle, a control issue, requires special attention.

➲ A key task of a **virtual rider** is to stabilize the vehicle

➲ To this end, a controller which is the core of the virtual rider has to generate a suitable steering torque based on the feedback of **appropriate state variables** of the vehicle

➲ One major problem in controlling single-track vehicles is that the coefficients of the controller are strongly velocity dependent.

This makes the manual configuration of a controller laborious and error-prone.

To overcome this problem, an automatic calculation of the controller's coefficients is desired.

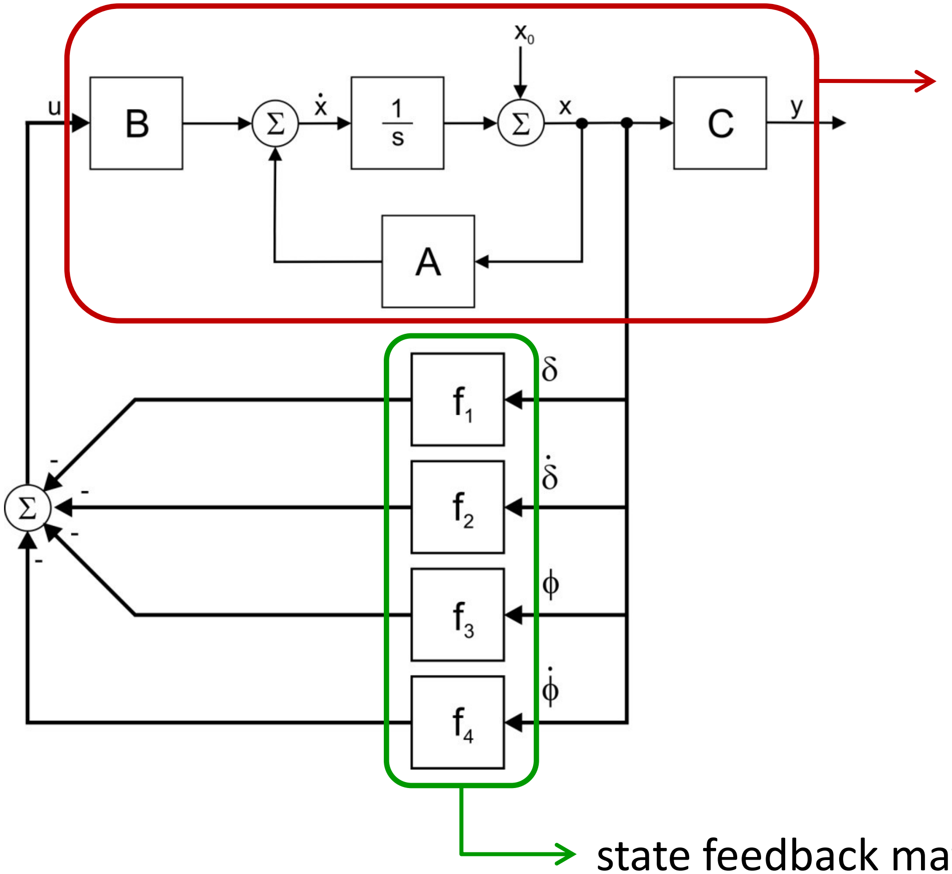
How can we do that ?

➲ Since we already performed an eigenvalue analysis we thus perfectly know how the dynamics of the vehicle depends on the velocity.

➲ Hence, the controller can be conveniently designed with reference to a preceding eigenvalue analysis

Controller Design Based on a Preceding Eigenvalue Analysis

Example: 3 degrees of freedom motorcycle



The plant (vehicle)

$$\dot{x} = A \cdot x + B \cdot u, \qquad x(0) = x_0$$

$$y = C \cdot x + D \cdot u$$

Control Law

$$u(t) = -F \cdot x_{sub}(t)$$

where $x_{sub} = \begin{pmatrix} \delta \\ \dot{\delta} \\ \phi \\ \dot{\phi} \end{pmatrix}$

state feedback matrix **F**

Controller Design Based on a Preceding Eigenvalue Analysis

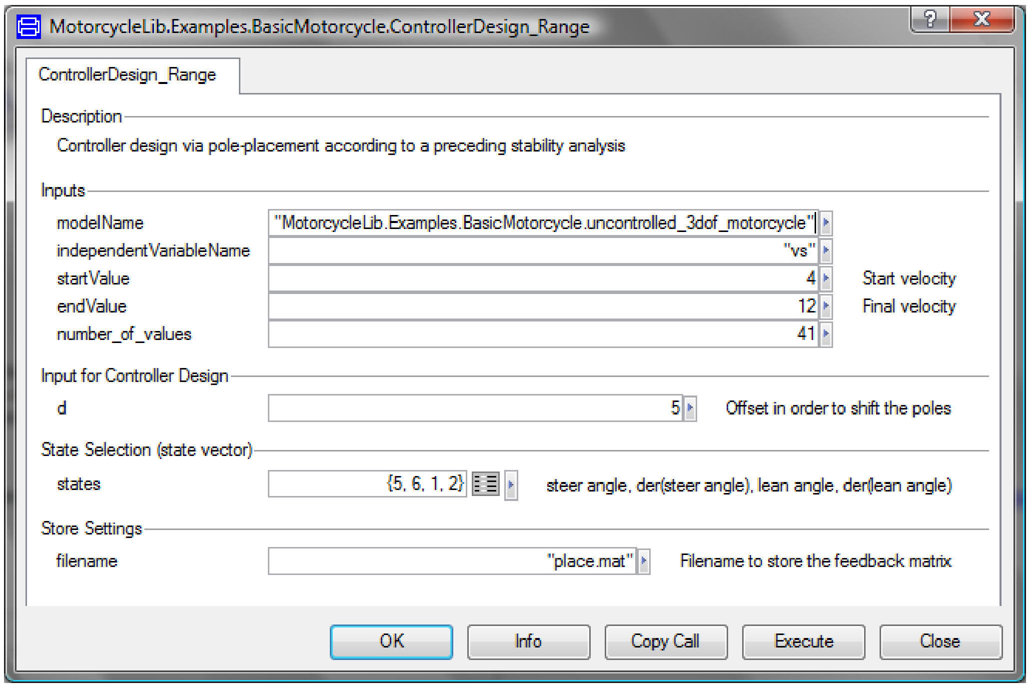
Basic Procedure

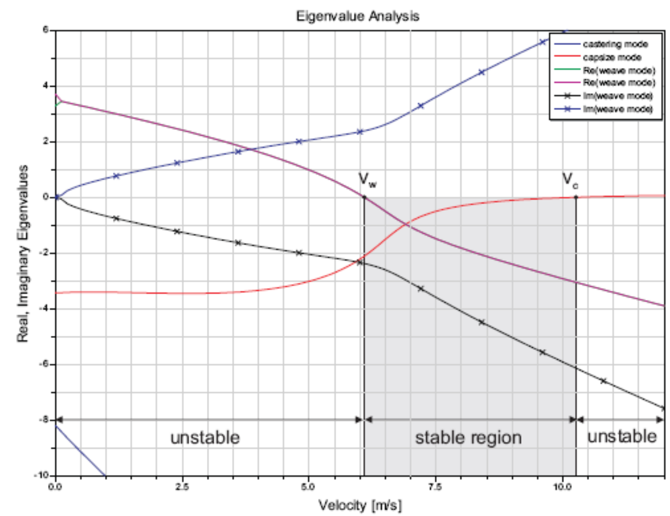
- Define a velocity range to stabilize the vehicle
- For each velocity v_i
  - Simulate and linearize the model → linear state-space representation of the vehicle
  - Compute a reduced state-space representation of the system
  - Calculate the corresponding eigenvalues λ_i and store them into a matrix
  - Calculate the state feedback matrix F (Ackermann's formula)
- Plot the eigenvalues as a function of the velocity → $\lambda = f(v)$

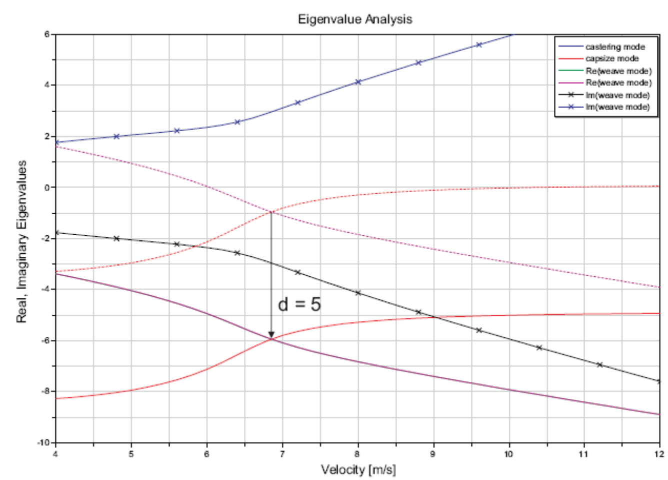
Controller Design Based on a Preceding Eigenvalue Analysis

Approach: Shift all real parts of the eigenvalues (poles) towards the left-half plane

Pole Placement Function

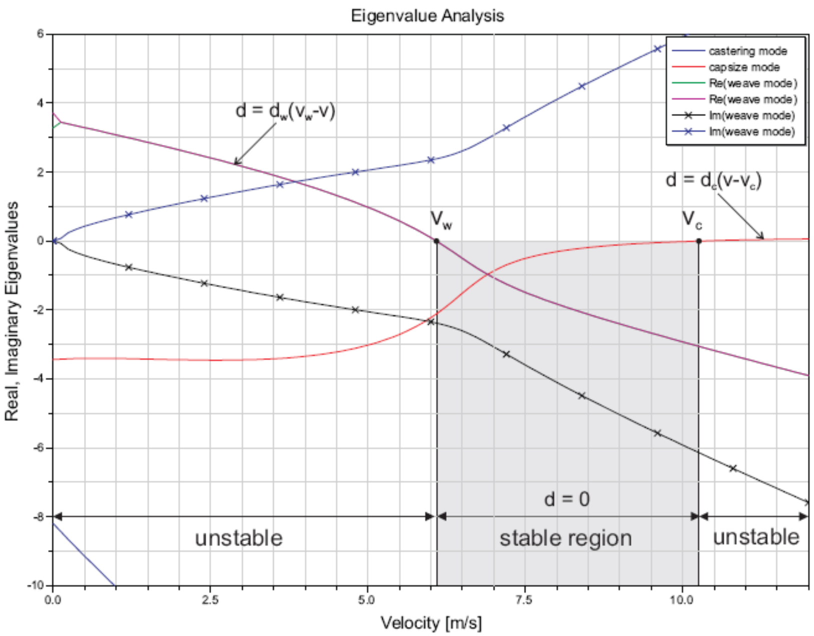


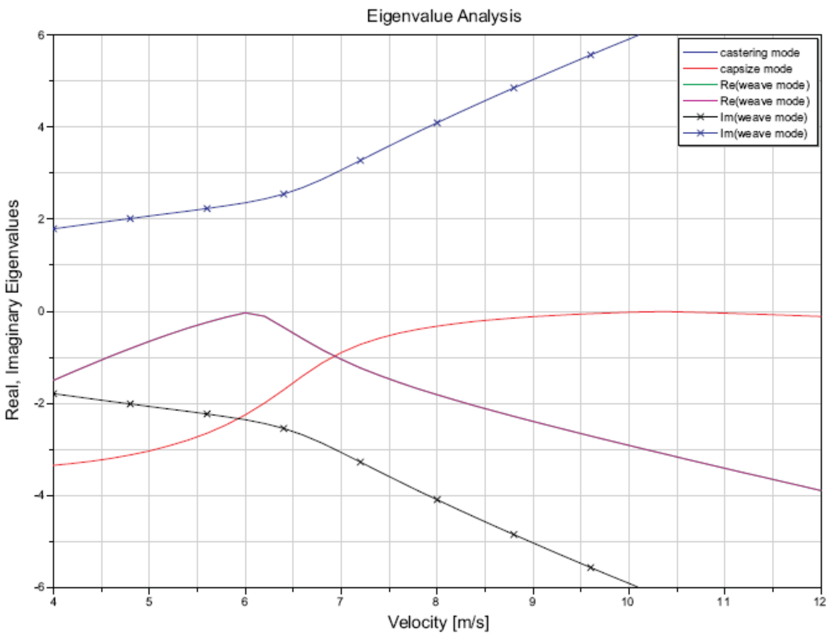




Controller Design Based on a Preceding Eigenvalue Analysis

Improved Approach: Modify solely those Eigenvalues that are unstable

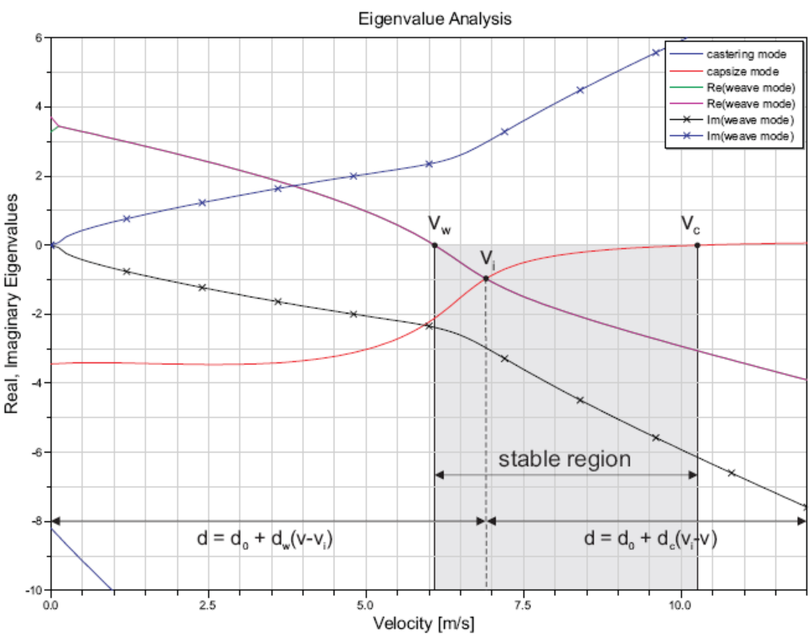


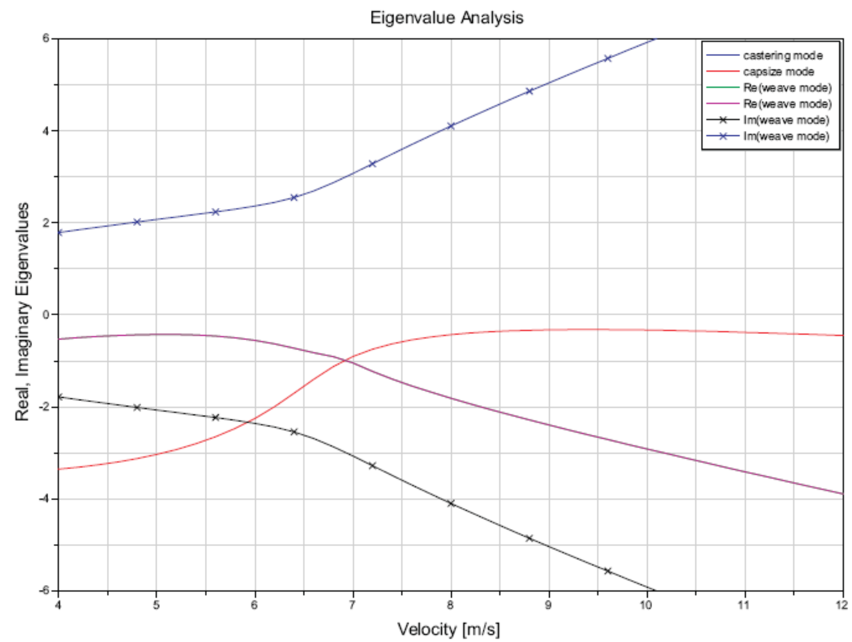


$$\text{control law} \begin{cases} v < v_w: & d = d_w \cdot (v_w - v) \\ v_w < v < v_c: & d = 0 \\ v_c < v: & d = d_c \cdot (v - v_c) \end{cases}$$

Controller Design Based on a Preceding Eigenvalue Analysis

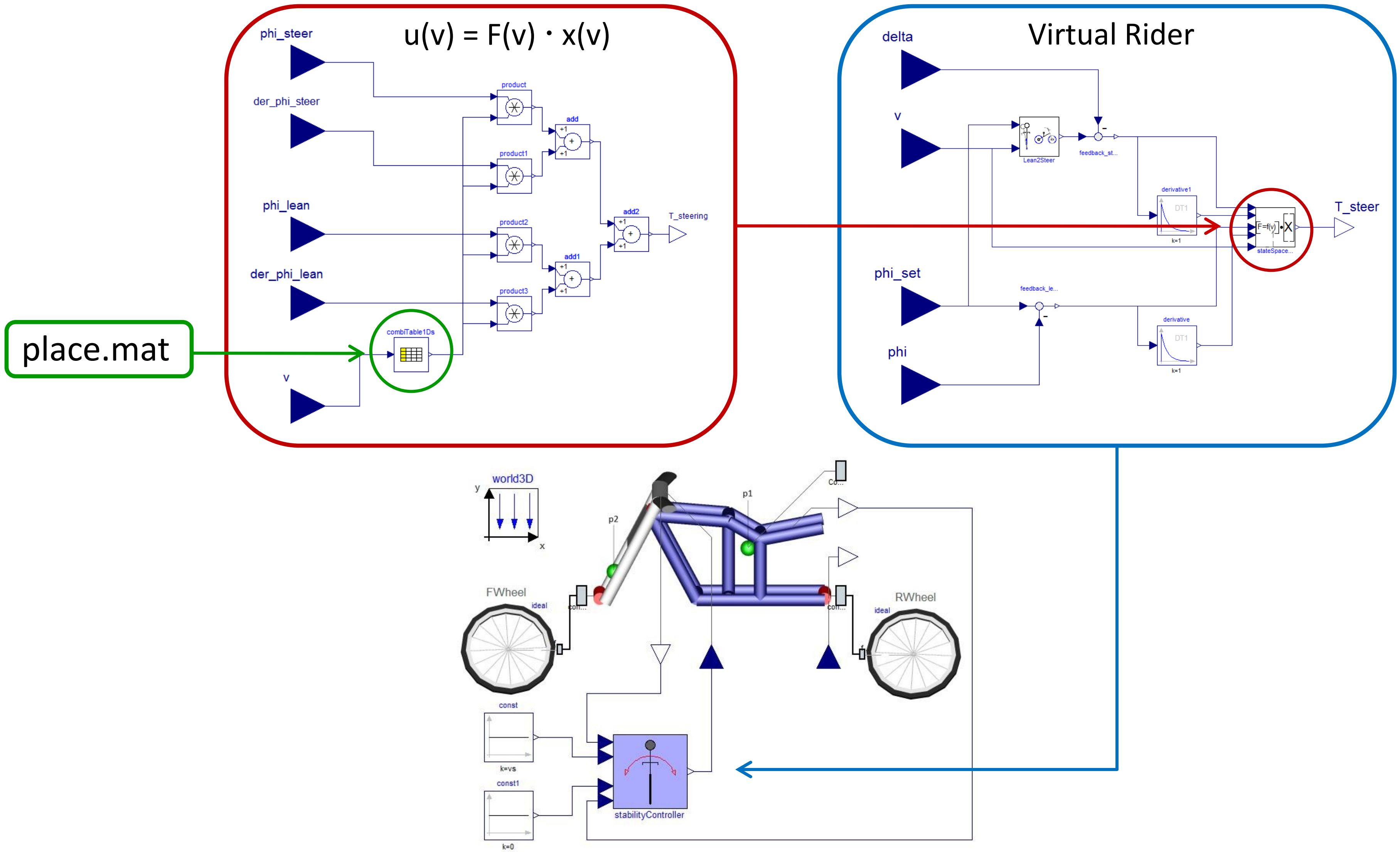
Improved Approach 2: Modify solely those Eigenvalues that are unstable



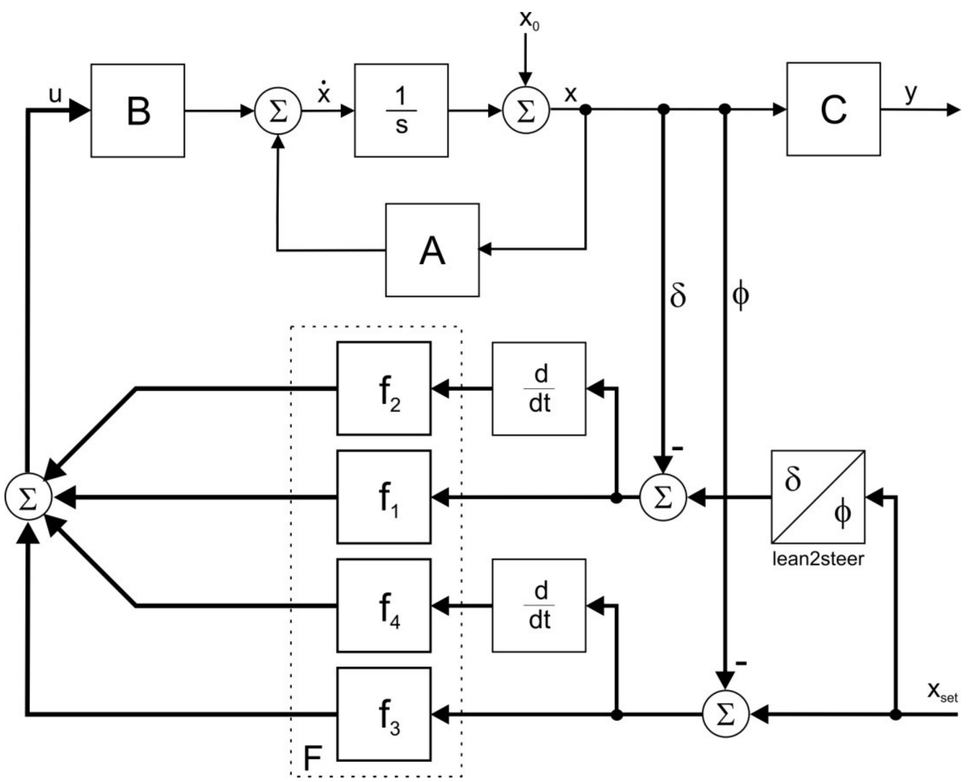


$$\text{control law} \begin{cases} v < v_i : & d = d_0 + d_w \cdot (v_i - v) \\ v_i < v : & d = d_0 + d_c \cdot (v - v_i) \end{cases}$$

4. Development of a Virtual Rider

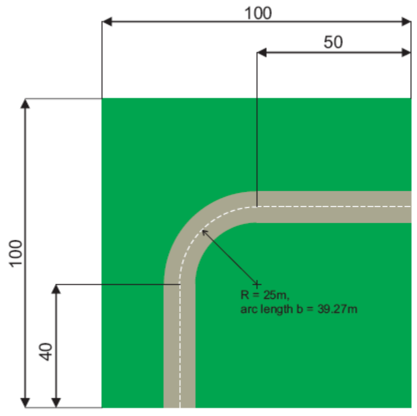


4.1 Roll Angle Tracking

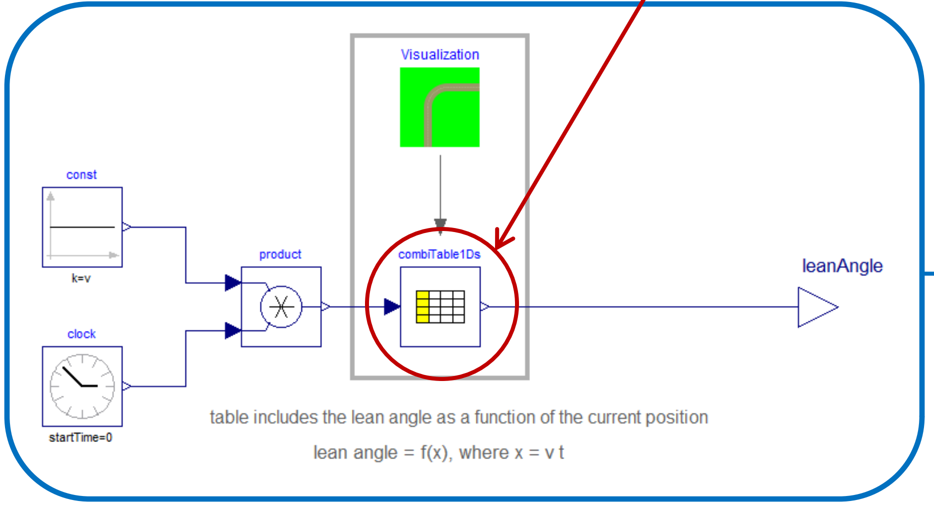


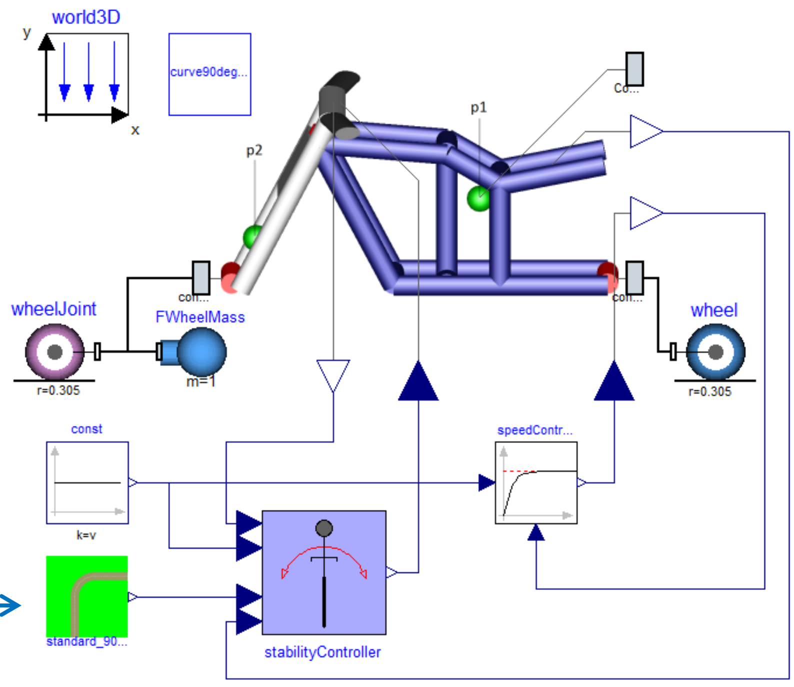
… Instead of a set-value equal to zero, the roll angle profile (e.g. of a 90-curve) is fed into the virtual rider.

4.1 Roll Angle Tracking: *Example*



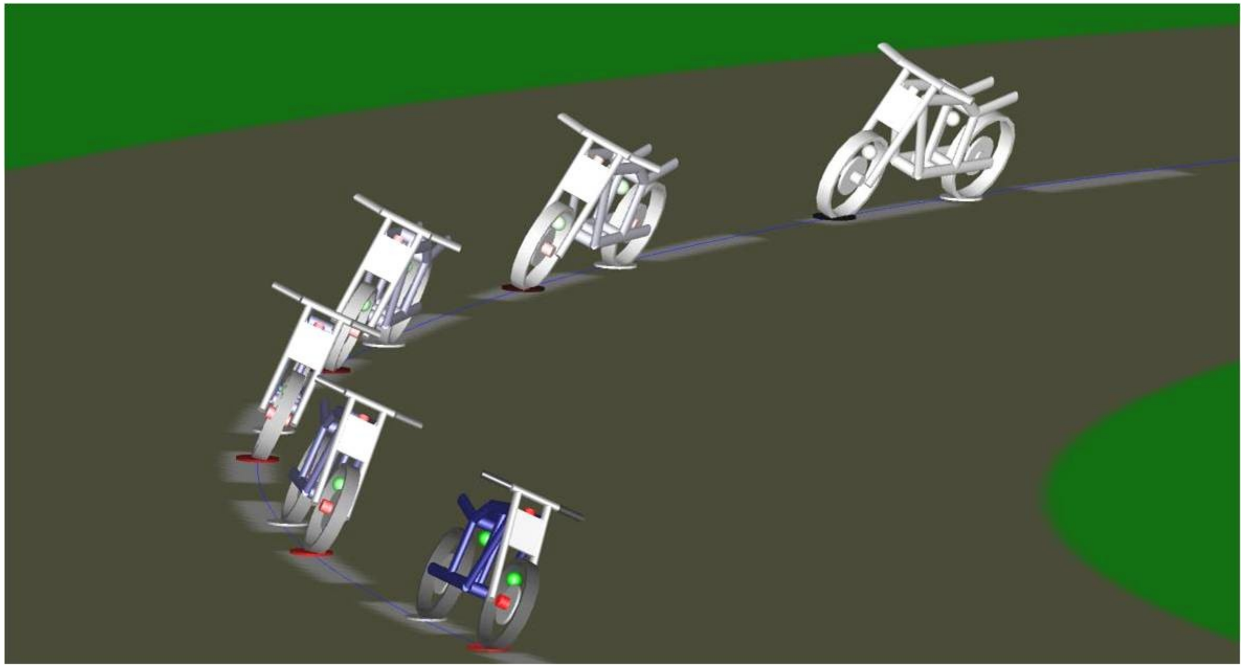
Path $x = v \cdot t$ [m]	Lean Angle ϕ [°]
0	0
50 - 5	0
50 + 1	$atan\left(\frac{v^2}{R*g}\right)$
51 + 39.27 - 5	$atan\left(\frac{v^2}{R*g}\right)$
50 + 39.27 + 1	0
100	0



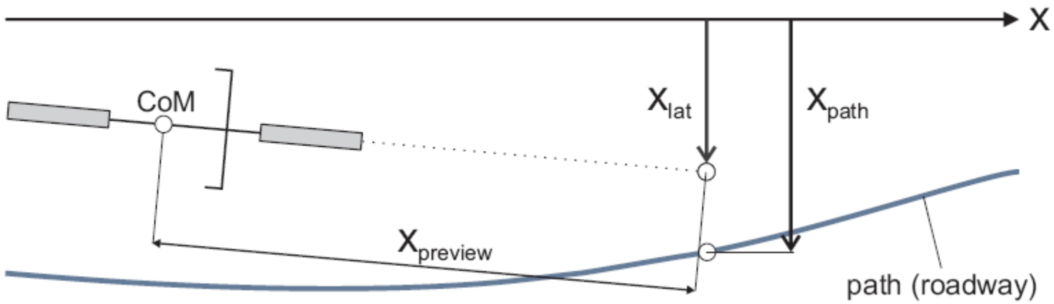


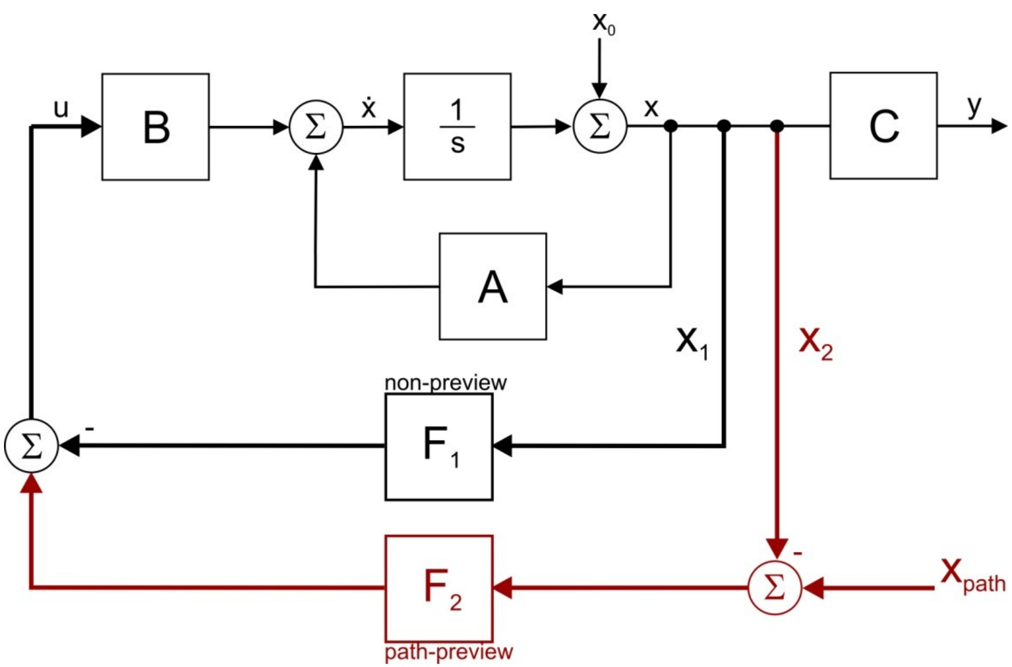
4.1 Roll Angle Tracking: *Example*

Simulation Result



4.2 PathTracking





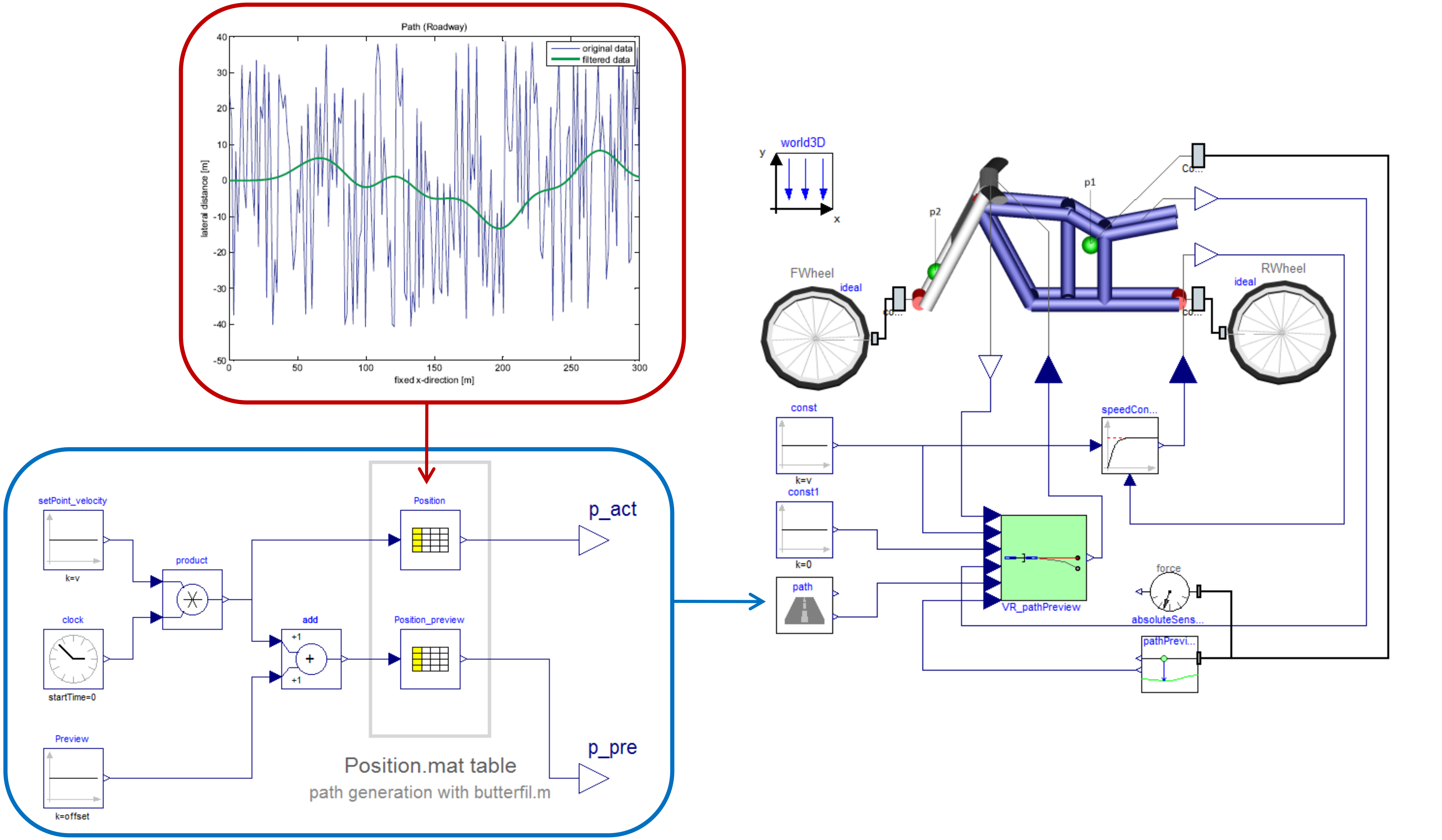
Control Law

$$u = -\begin{pmatrix} F_1 & F_2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

where

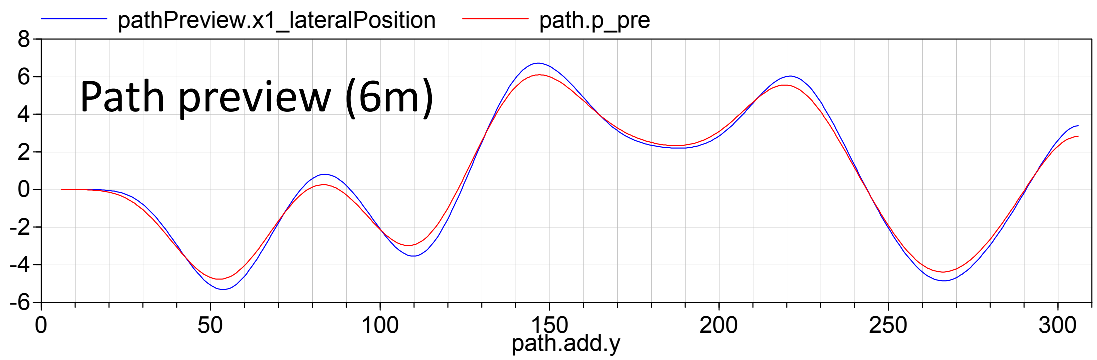
$$x_1 = \begin{pmatrix} \delta \\ \dot{\delta} \\ \phi \\ \dot{\phi} \end{pmatrix} \text{ and } x_2 = \begin{pmatrix} x_{lat} \\ \dot{x}_{lat} \end{pmatrix}$$

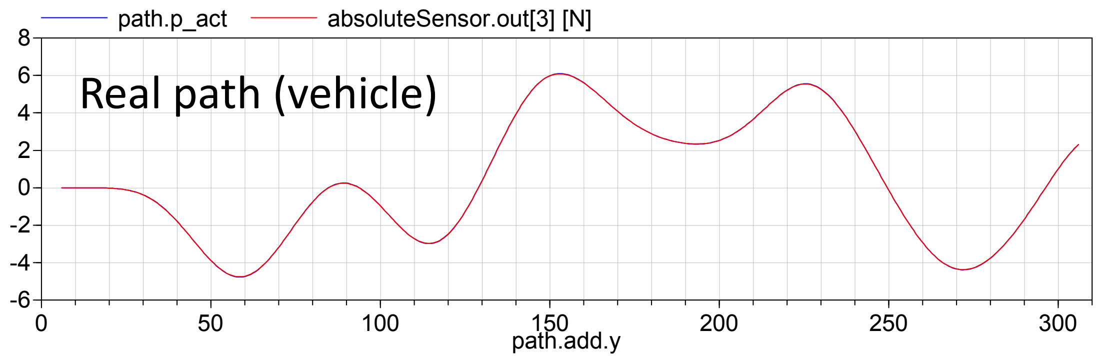
4.2 PathTracking: *Example*



4.2 PathTracking: *Example*

Simulation Result





5. Conclusion

- The library provides appropriate eigenvalue functions for each vehicle. Beside the controller design such an analysis is beneficial for the optimization of the vehicle's geometry. By changing the geometry or the center of mass' locations of a vehicle, the eigenvalues of the system are changing as well. It is thus possible to optimize the design of a vehicle regarding self-stability.
- Due to the results of the eigenvalue analysis it is now possible to conveniently design a state-space controller valid for a specific velocity range of the vehicle. Thus, for the calculation of the state feedback matrix coefficients, a pole placement function was developed.
- To test the performance of the vehicles, the virtual riders are capable of tracking both, a roll angle profile and a pre-defined path. Therefore, several test tracks are included in the library.

Thanks for your Attention!