

**Using the Advantages  
of SIMPACK's  
Linear System Interface  
to MATLAB®**



SIMPACK User Meeting, Salzburg, May 2011  
by Peter Häse, Uster

**BOMBARDIER**

# Table of content

---

**1 Introduction**

**2 General on Linear Systems in SIMPACK and MATLAB**

**3 Examples**

**3.1 Drive Chain Investigation**

**3.2 Implementing an Observer**

**4 Summary**

---

# Introduction

---

- **Linear methods are commonly used in the field of vehicle dynamics**
- **History leads back to the famous formula found by Klingel in 1883**
- **SIMPACK offers a number of methods like**
  - **Eigen values and –vectors**
  - **Linear System Analyses**
  - **Critical Parameter Investigation**
- **However, in some special cases additional methods would be very helpful or simply the presentation of the results should be improved**
- **In these cases the Linear System Interface is very helpful, e.g. the export of Linear System Matrices to MATLAB**

# Linear Systems in MATLAB

---

## ■ LTI-Objects

„Linear Time Independent“ Systems

### – SISO

- „Single Input Single Output“ System
- Most commonly described by its Transfer Function (TF) or even Frequency Response Function (FRF)

$$G(s) = \frac{\sum b_{nn} * s^{nn}}{\sum a_{nd} * s^{nd}}$$

### – MIMO

- „Multiple Input Multiple Output“ System
- Either described by a set of Transfer Functions or a so-called State-Space-Description (SS)
- FRF may converted to SS and vice versa (tf2ss, ss2tf)

$$\frac{dx}{dy} = \underline{A}x + \underline{B}u$$

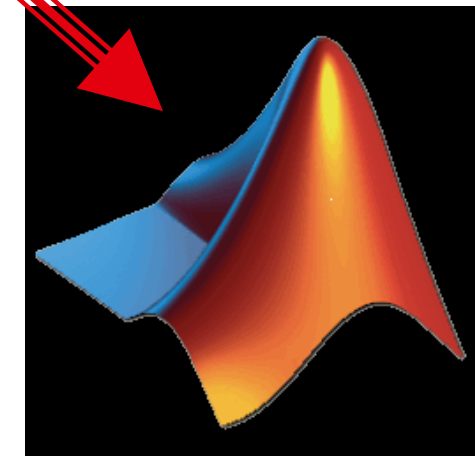
$$y = \underline{C}x + \underline{D}u$$

# Export a Model

Screenshot of SIMPACK 8900 software interface. The 'System Matrices' menu is open, showing options for exporting to SIMPACK, SIMPACK + MATRiX, and SIMPACK + MATLAB. The main window displays a 3D model of a rail vehicle and a large matrix of numerical values.

**Preparation:**  
**Define a input  $u(t)$  instead of a time excitation !**

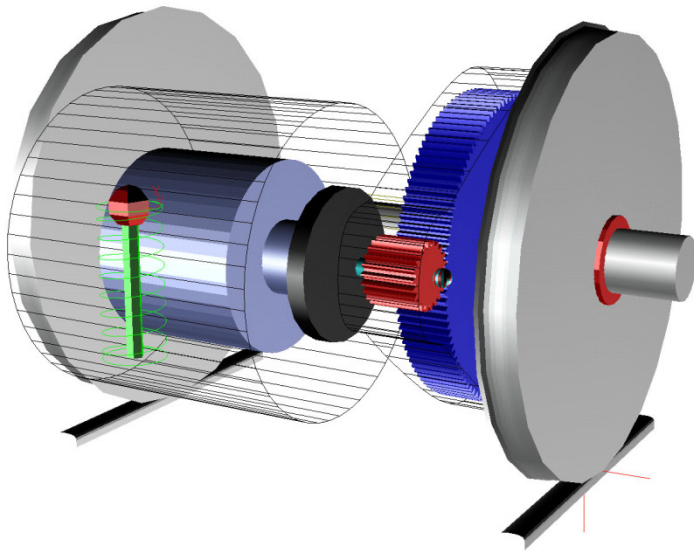
**Consider using the ParVariation to export the model**



**All names of states, inputs and outputs are available in MATLAB!**

# Example 1 – Drive Chain Oscillation

---



- **Aim**

- Establish countermeasures to torsional vibration of wheelset

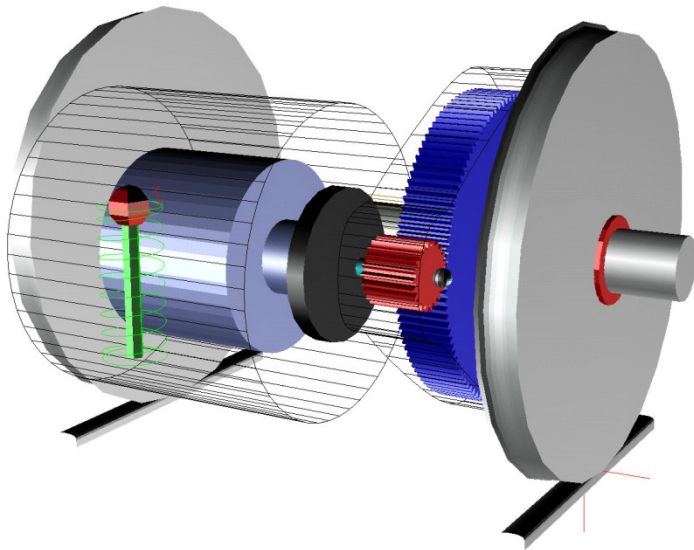
- This kind of vibration may occur if the traction control fails, e.g. an operating point behind the maximum of adhesion is stabilized
    - This kind of vibration covers a high risk for damaging the wheel nave or the wheelset axle itself
    - The speed sensor is the favorite base to implement a supervision

- **Task**

- Calculate the mechanical admittance or mobility

# Example 1 – Drive Chain Oscillation

---



- **Process**

- **Add the necessary outputs to the model**

- Angular velocity of the rotor
- Angular velocities of the wheels (E.g. in order to prepare tests with the laser-vibrometer)
- Torque at the wheel set axle
- Force in torque reaction rod

- **Check for equilibrium state**

- **Export/import the model**

# Example 1 – Drive Chain Oscillation

---

## ■ Observability

- Definition:
  - The system `sys` is (fully) observable, if its observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

has full rank

- MATLAB command
  - `rank(observf(sys.a,sys.b,sys.c))`
- Consider also to check for detectability

## ■ Controlability

- Definition:
  - The system `sys` is (fully) controlable, if its controlability matrix

$$R = [B \ AB \ A^2B \ \dots \ A^{n-1}B]$$

has full rank

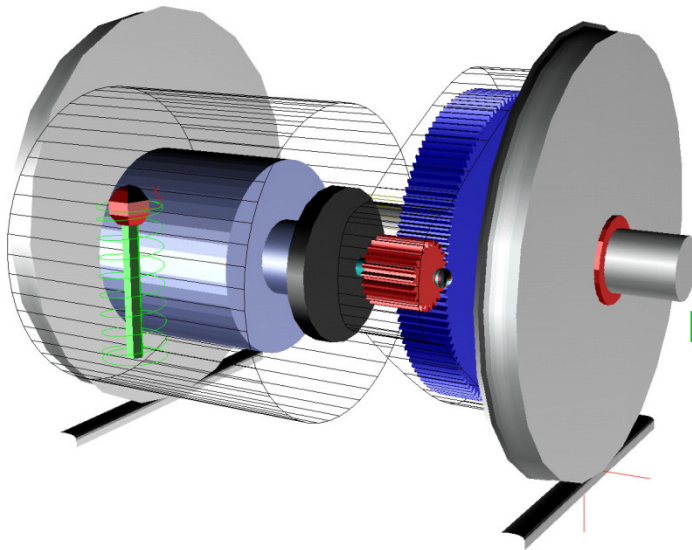
- MATLAB command
  - `rank(ctrbf(sys.a,sys.b,sys.c))`
- Consider also to check for stabilizability

# Example 1 – Drive Chain Oscillation

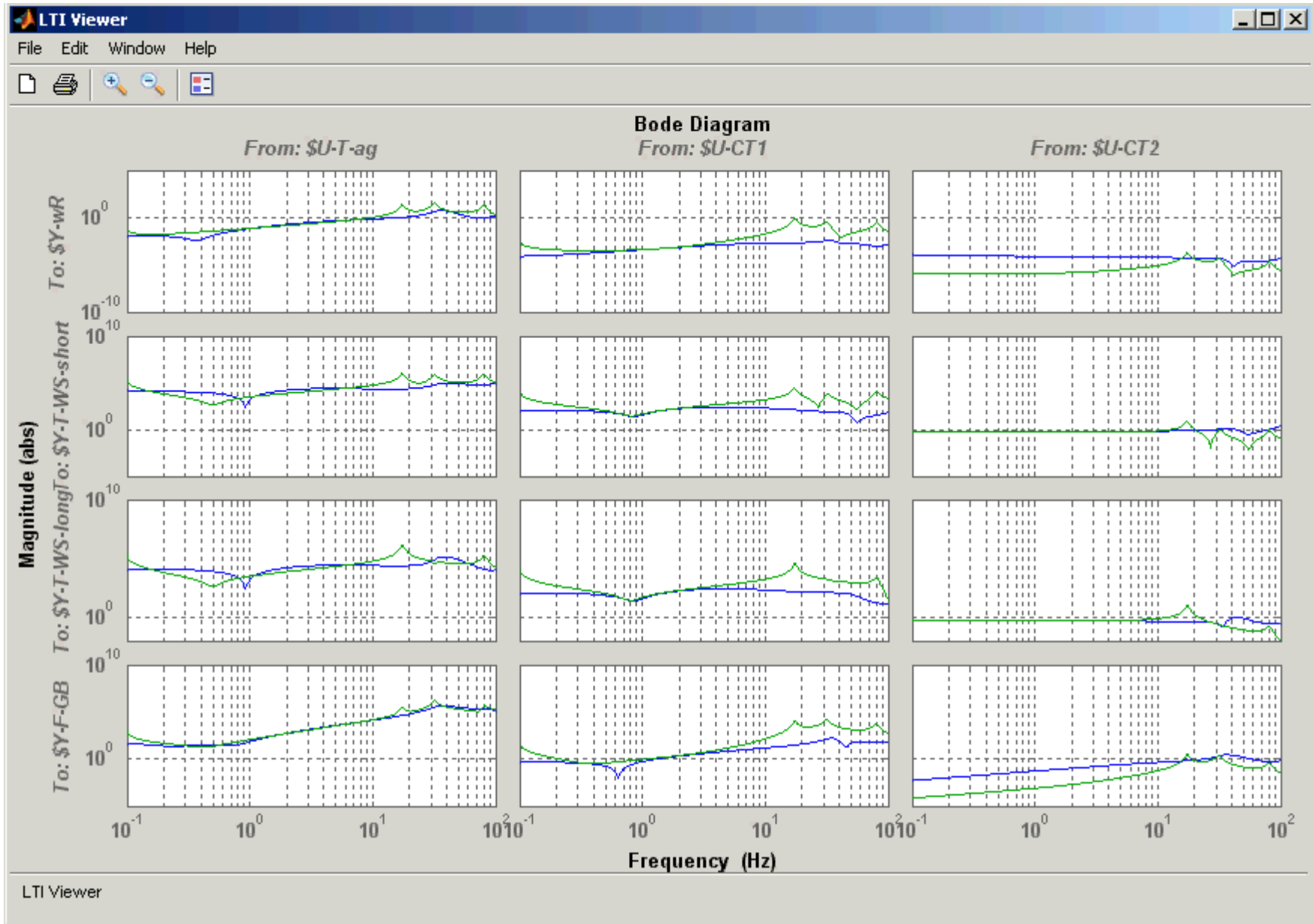
---

- **LTI-Viewer**

- Fast and detailed overview of general system behavior
- One, two or more systems may be handled at once !
- Invoke the LTI-Viewer interactively or e.g. by the MATLAB-command



```
ltiview('bodemag',drive_stick(:,1:3), drive_slip(:,1:3),  
        logspace(-1,2,1000)*2*pi);
```



# Example 1 – Drive Chain Oscillation

```

# -----
#
EV.No:      3 / 4          5 / 6
#
# Re      Im      f0 [Hz] | Re      Im      f0 [Hz] |
# -2.0979E+01 2.2014E+02 3.520E+01 -6.5083E+00 3.2993E+02
# 5.252E+01
# -----
# Amplitude Phase Energy | Amplitude Phase Energy
# |
# [deg] kin. / | [deg] kin. / |
# modal | modal |
# -----
.
.
.

body.cm: $B_Wheel_left
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.2983 -353.90 0.4120 1.0000 0.00 1.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000

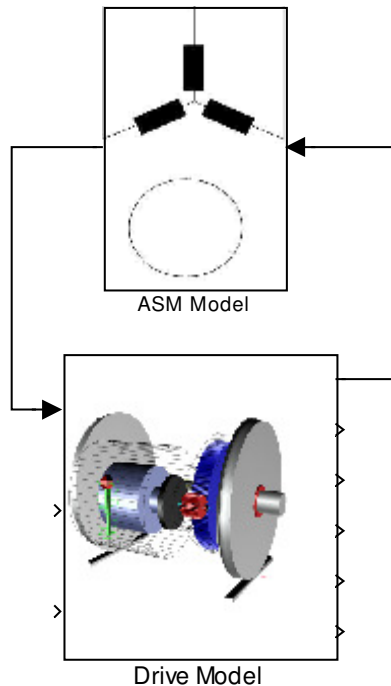
body.cm: $B_Wheel_right
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.0000 0.00 0.0000 0.0000 0.00 0.0000
0.1265 -11.80 0.0741 0.6329 -172.10 0.4005
0.0000 0.00 0.0000 0.0000 0.00 0.0000
.
.
.

```

- In **SIMPACK** the necessary figures must be read from the file <model>.eva and further processed by hand
- After exporting the SIMPACK model “drive” to **MATLAB** use the following commands:
  - $[V,D] = \text{eig}(\text{drive.a})$
  - $f = \text{imag}(\text{diag}(D))/2/\pi$
  - $y = \text{drive.c} * V$
  - Calculate and normalize the absolute values
- This process is more stable and may be integrated in the quality management

# Example 1 – Drive Chain Oscillation

---



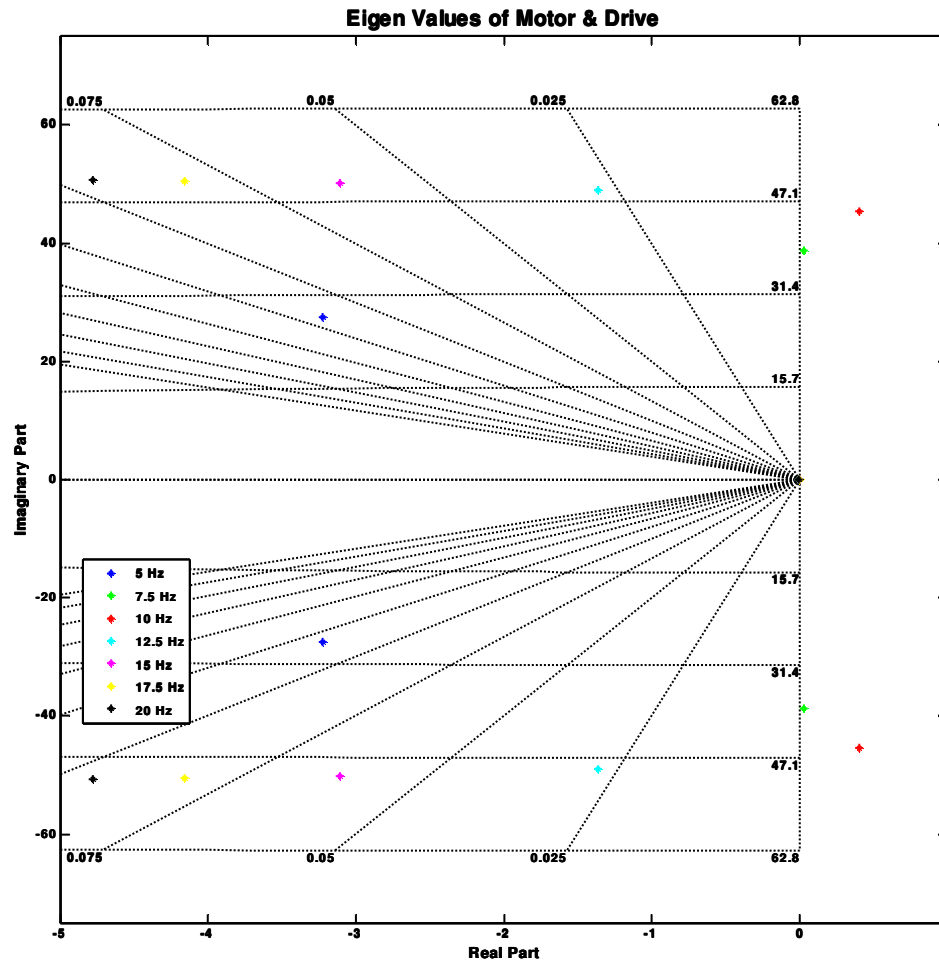
- Investigation may be continued e.g. by closing the loop with any control or other subsystem, e.g. the electrical part of the motor

- The asynchronous motor is non-linear, i.e. we have to linearize the model for each operating point separately

- MATLAB commands:

```
asm_drive_ff =  
    feedback(drive(1,1),asm_ff,1,1,+1);  
pzmap(asm_drive_ff, pzopts)  
sgrid([0.05:.05:.25],[0:2.5:10]*2*pi)
```

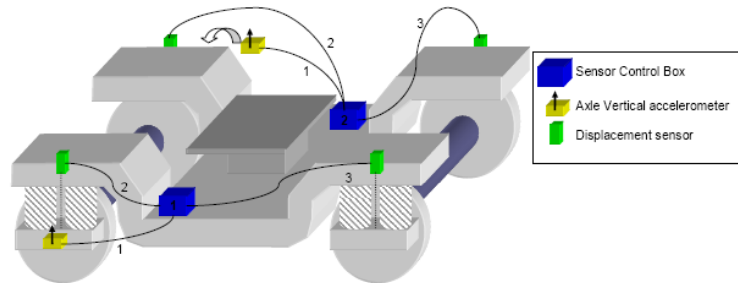
# Example 1 – Drive Chain Oscillation



## Result:

- The unit of motor & drive may oscillate for certain frequencies depending on
  - Control of the converter & motor
  - Design of the motor
  - Saturated adhesion
  - Low rotor temperature

# Example 2 – Implementing an Observer



6 Sensoren pro Drehgestell zur Messung von:

Gleisverwindung  
Stossanregungen  
Gleislageabweichungen

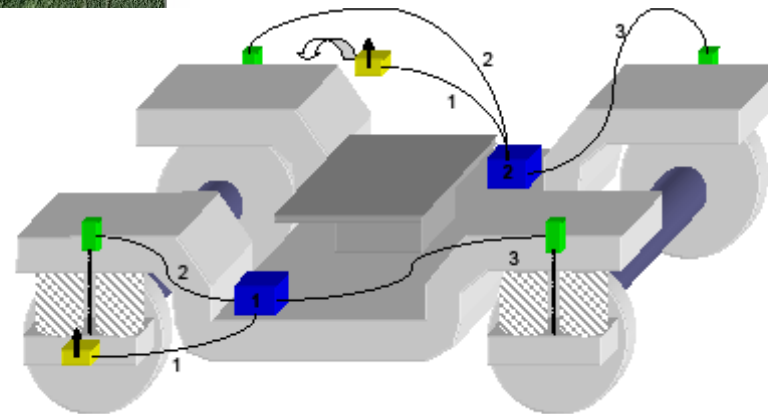
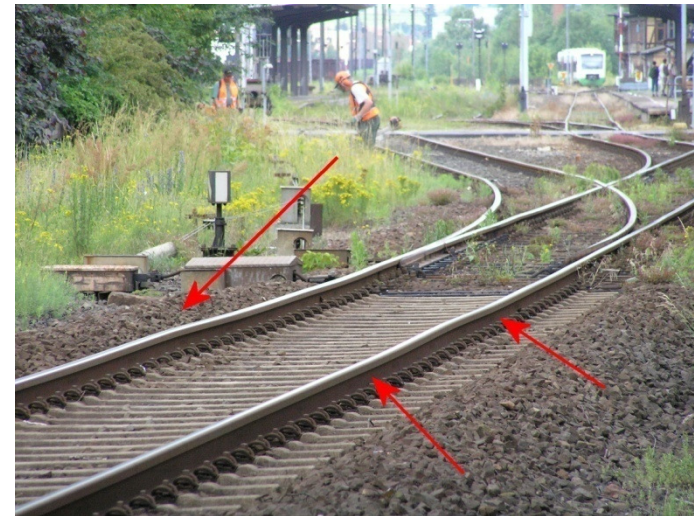
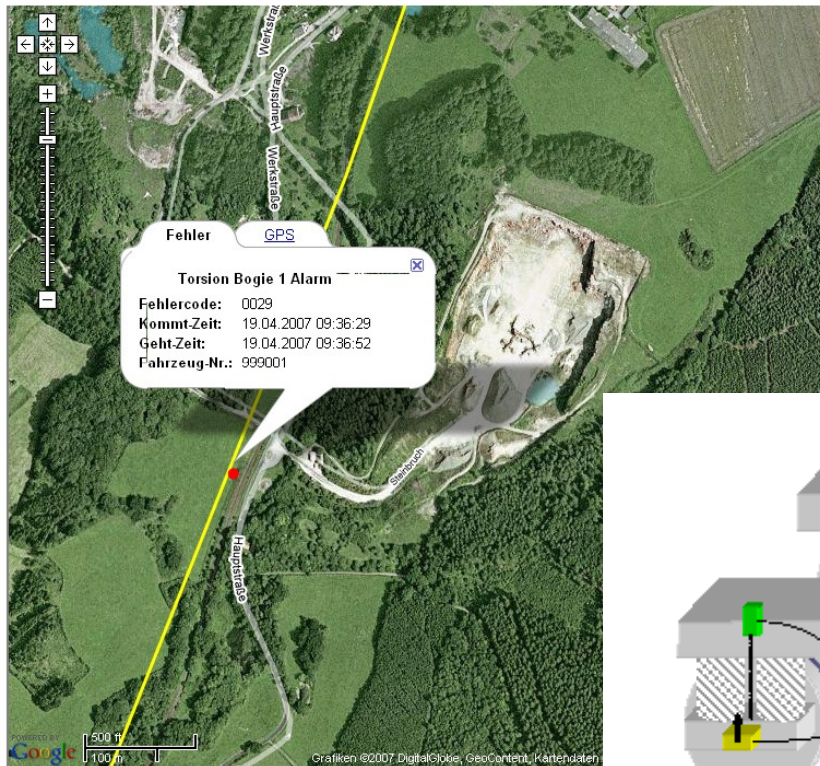
## ■ Aim

- Identify track disturbances during standard service, e.g. without specialised measuring car
- Sensors already defined
- Standard vehicle dynamics model available (Thanks to my colleague C. Bussmann!)

## ■ Task

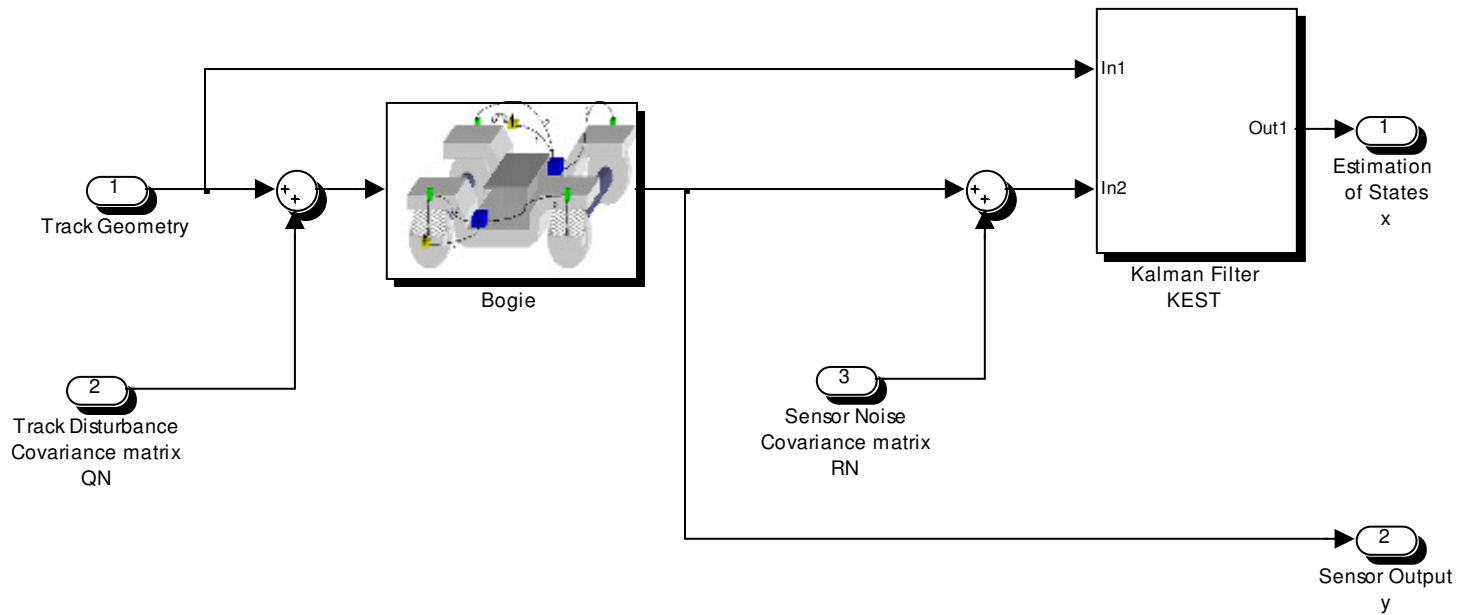
- Use an observer, e.g. a so-called Kalman Filter in order to estimate the states of the bogie and finally the track disturbances

# Example 2 – Implementing an Observer



# Example 2 – Implementing an Observer

- Basics of Kalman Filtering



# Example 2 – Implementing an Observer

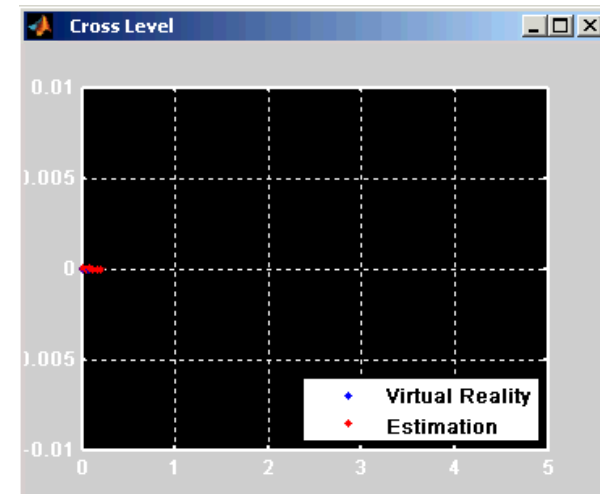
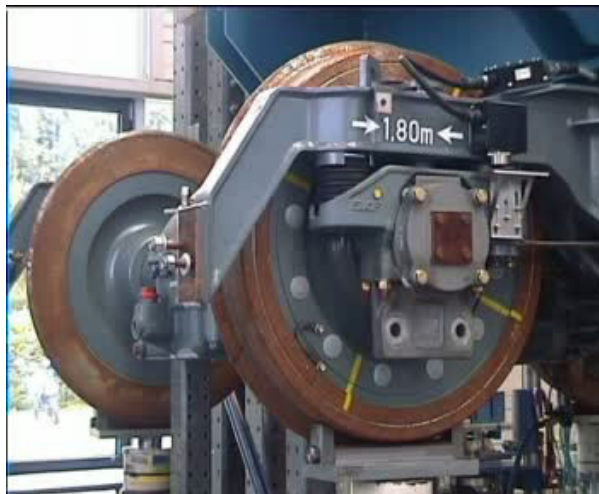
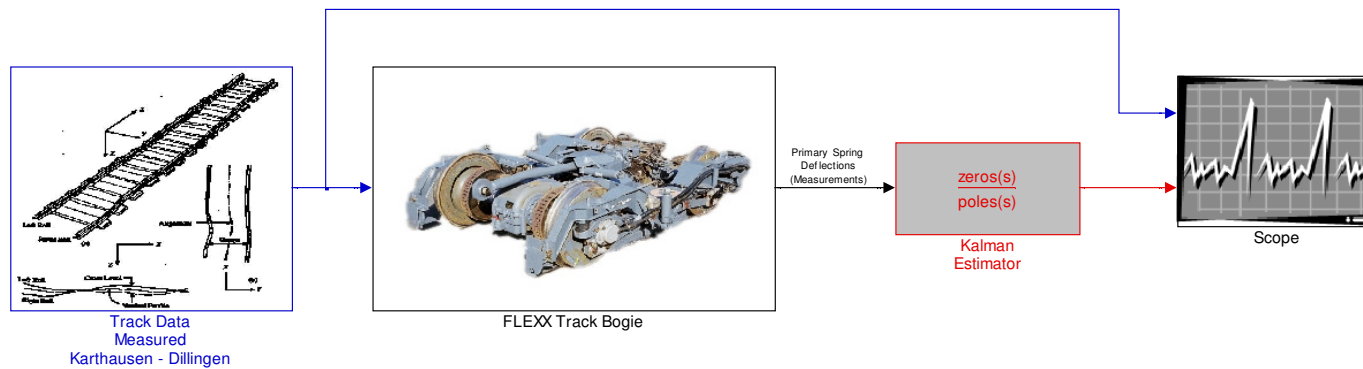
---

## ■ Process

- Simplify the SIMPACK model in order to reduce the number of DOF
- Select the appropriate WRC model
  - “elastic” -> D-matrix is empty
  - “constraint” -> D-matrix may be not empty
- Define the necessary inputs and outputs
- Export the state-space model to MATLAB
- Create the observer using the Control System Toolbox using the covariance matrices of disturbances QN, RN, NN
  - `[KEST,L,P] = kalman(bogie,QN,RN,NN,sensors,known_inputs)`
- Implement the observer to SIMPACK,  
e.g. using the control element FE 142: AD-Filter: A,B,C,D > File
- Validate the observer
  - Versus the non-linear simulation by SIMPACK
  - Versus the measurements at the test stand

# Example 2 – Implementing an Observer

## FLEXX Track Cross Level Estimation using advanced Kalman Filter



# Summary & Outlook

## ■ Summary

### – SIMPACK's Linear System Interface to MATLAB

- Increases the capability to
  - Analyze complex systems like railway vehicles by advanced linear methods
  - Synthesize advanced (e.g. model-based) control systems
- Allows to share the workload between specialists

## ■ Outlook

- Investigation/prediction of Structure born Noise in the range (10 ... 1000) Hz

