Chair of Structural Analysis

EMPIRE: A N-Code Coupling Tool for Multiphysic Co-Simulations

EMPIRE (Enhanced MultiPhysics Interface Research Engine)

Author: S. Sicklinger, T. Wang

Introduction: What is N-Code Coupling?

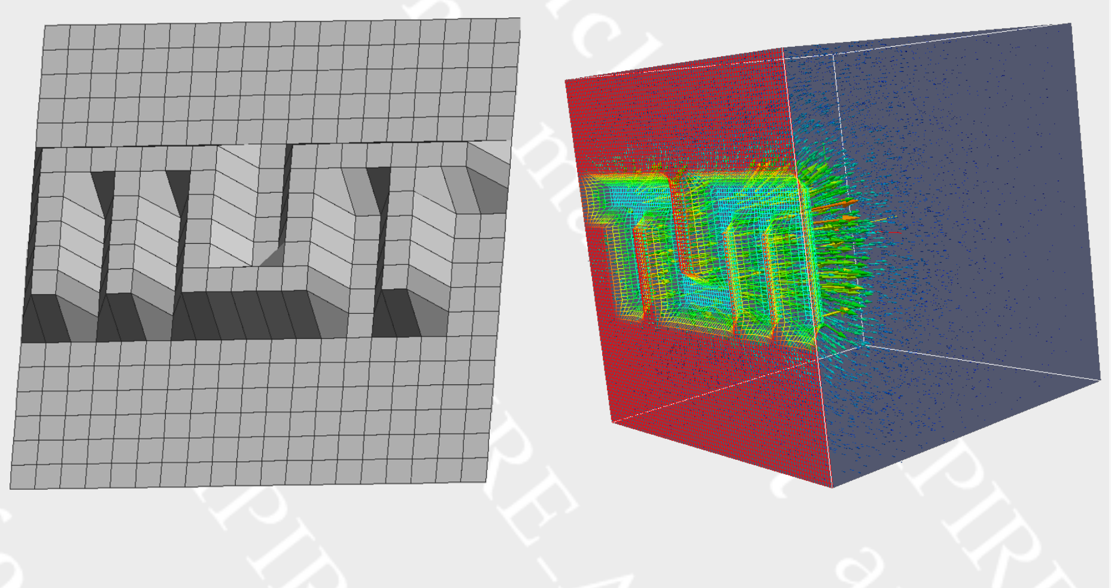


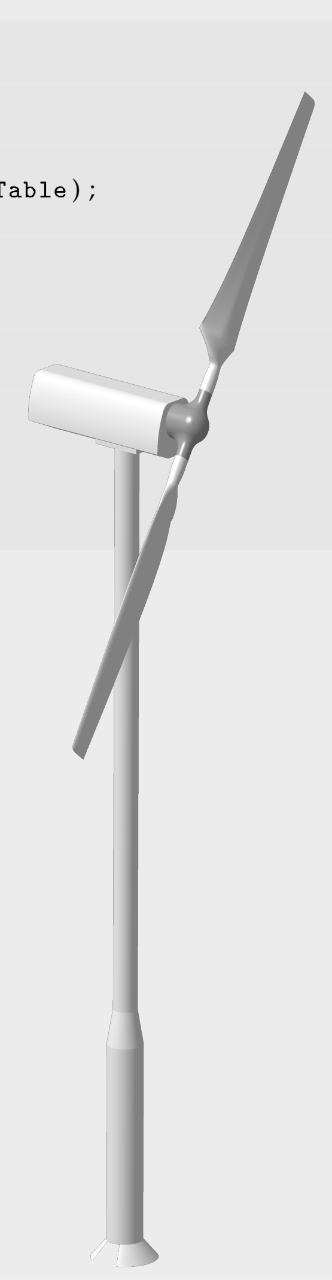
Figure 1: TUM example CSM - OpenFOAM coupling

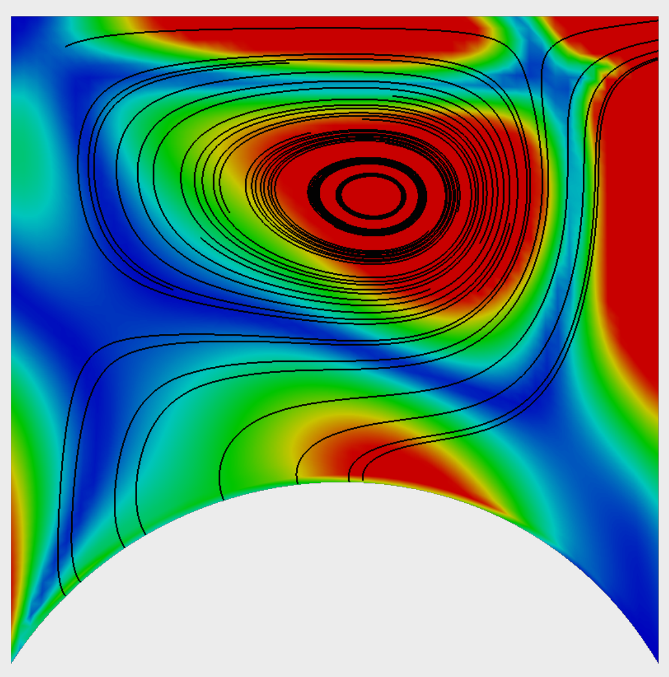
Most technical products are an assembly of different systems. In product design, simulation is a well established tool in order to accelerate the development-to-market time. For a lot of physical phenomena sophisticated simulation tools exist, e.g. for fluids Computational Fluid Dynamics (CFD) or for structures Computational Solid Mechanics (CSM). A product usually consists of systems which may be modeled in different dimensions i.e. the coupling of ordinary differential equations (ODEs) and partial differential equations (PDEs) is required. For instance the simulation of a car requires a CSM solver to reproduce the structural behavior. Moreover, the simulation of the flow around the car necessitates the use of a CFD solver. The CSM and CFD solver need to resolve all four dimensions, i.e. three spatial and time dimensions to accurately model the physics. Other components like sensor or actors (e.g. hydraulic system of the brakes) may be modeled with two dimensions, i.e. one spatial and one time. Thus the CSM and CFD solver need to be coupled to the solvers which represent the sensors and actors.
A concept for this kind of Co-Simulation is presented where in contrast to Fluid-Structure Interaction the number of simulation tools is larger than two. Furthermore, the presented concept accounts for the coupling of PDE and ODE systems. The motivation for Co-Simulation (also called partitioned coupling) is the high flexibility and reusability of existing simulation tools. The presented framework is based on a client-server approach where MPI-2 functionality is used to implement socket-like communication between the client and the server. The dataflow handling inside the server program is done dynamically, hence the dataflow is determined at runtime. The presented approach can also handle non-matching discretizations and is suited for very large simulations involving a large number of unknowns. The software concept is illustrated by interfacing OpenFOAM® with the in-house coupling tool called EMPIRE (Enhanced MultiPhysics Interface Research Engine) to various other simulation tools. By this OpenFOAM® capabilities can be extended to very complex simulation scenarios involving a large number of different physical phenomena e.g. nonlinear CSM with control.

Interface to OpenFOAM®

The EMPIRE API is very easy to use. It is simple to incorporate the API to OpenFOAM®. One may start with a standard OpenFOAM® solver like pimpleDyMFoam and add the following calls to the solver:

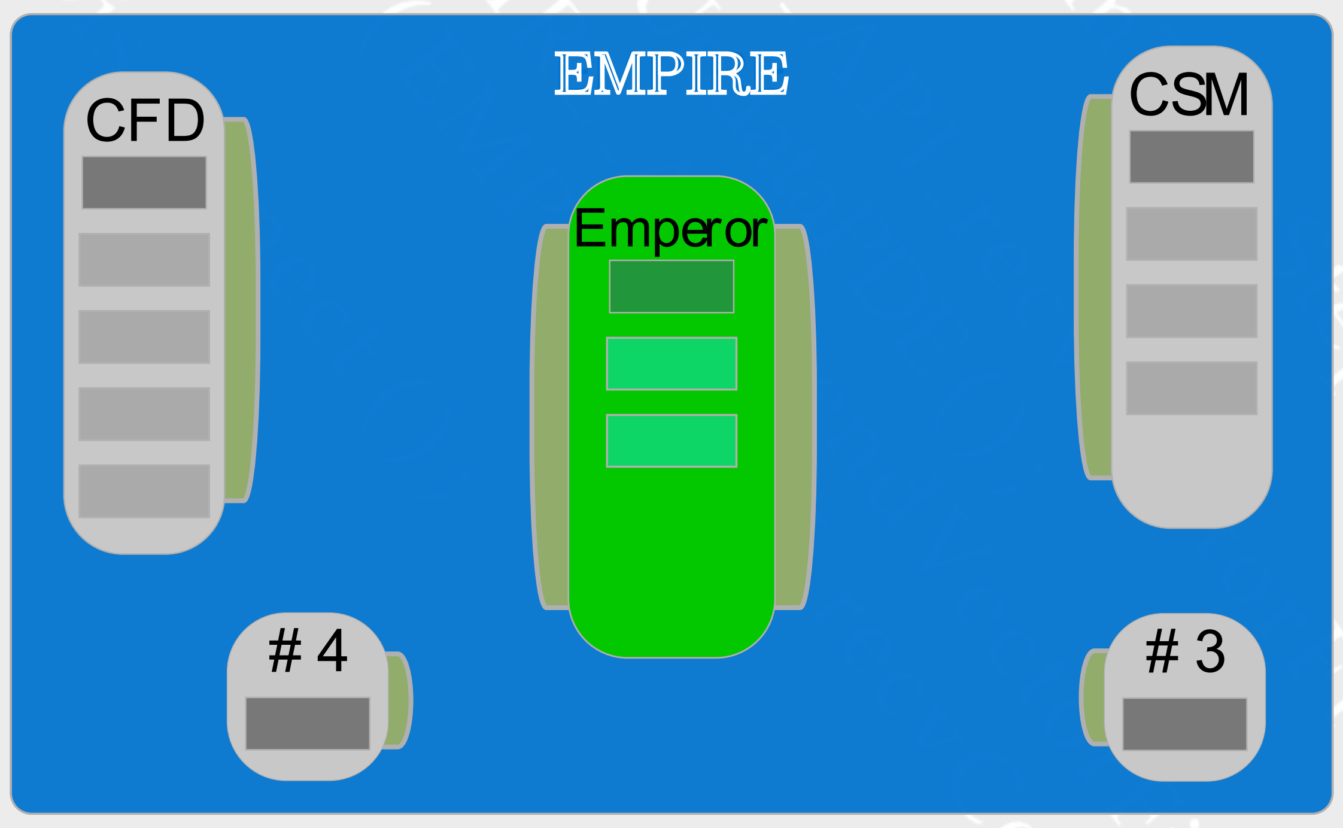
```
#include "EMPIRE_API.h"
int main(int argc, char **argv) {

   EMPIRE_API_Connect(argv[1]);

   EMPIRE_API_sendMesh(BNumNodes, BNumElems, BNodesPerElem, BNodeCoors, BNodeIDs, BElemTable);

   for (int i = 1; i <= numTimeSteps; i++) {
        do {
                EMPIRE_API_recvVectorField(BNumNodes * 3, dofsRecv);
                dummyUpdatePDE();
                EMPIRE_API_sendVectorField(BNumNodes * 3, dofsSend);
           } while (EMPIRE_API_recvConvergenceSignal() == 0);
   }
   EMPIRE_API_Disconnect();
   return (0);
}
```





Basic Concept and Communication Hub

EMPIRE is based on a client server approach. This gives very high flexibility within the context of Co-Simulation. Thus EMPIRE consists of two parts. One is the server called EMPEROR the other is the EMPIRE API which interacts with the client code. The setup of the coupling algorithm is entirely defined via a xml file which EMPEROR reads at the beginning of a Co-Simulation. The clients can be connected to EMPEROR at any-time during the simulation.



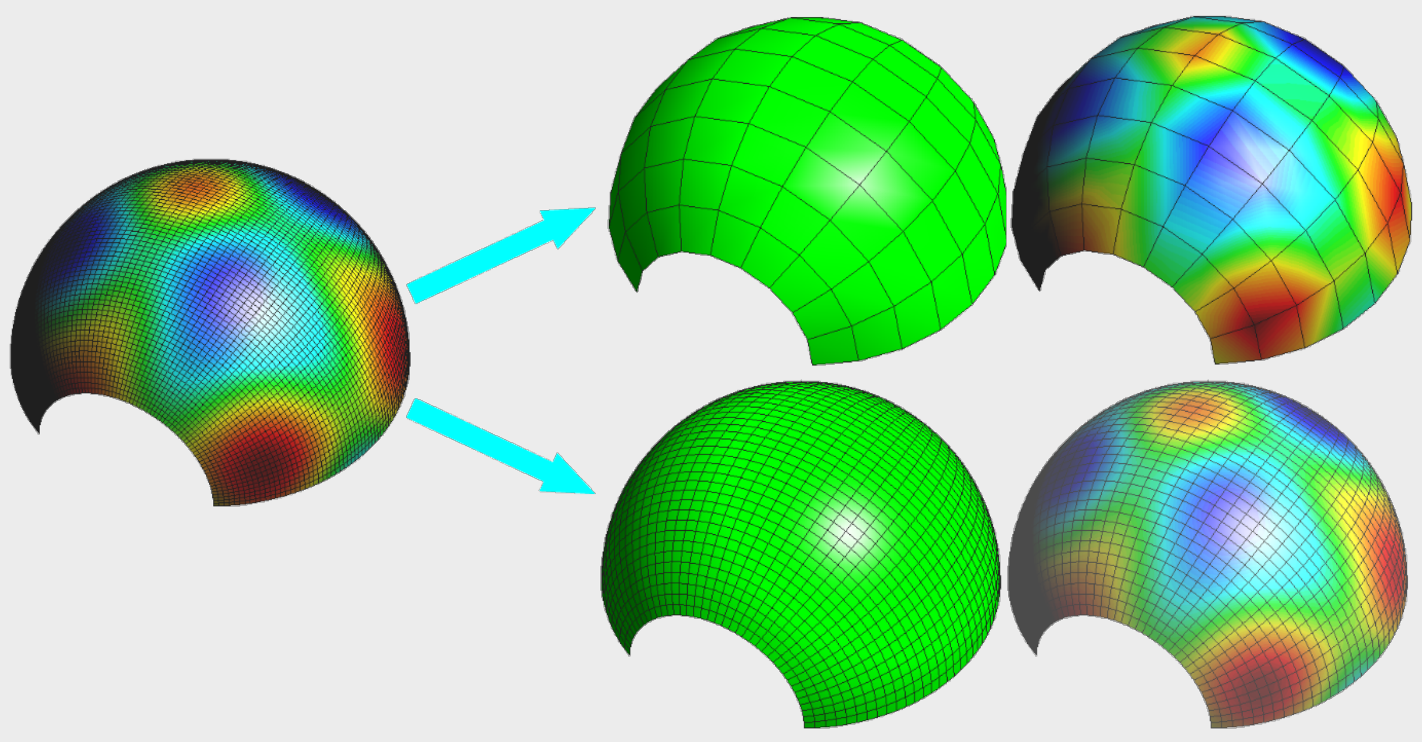
EMPIRE API - Client API

- Input based on XML
- Clients connect via Socket-like MPI-2 communication
- Written in C++
- Wrapped in C for better interoperability (Tested with C++, C, FOTRAN and JAVA)
- Handles all the MPI calls internally (not visible to the user)

EMPEROR - Server

- Input based on XML
- Asynchronous listening (OpenMP)
- Clients connect via Socket-like MPI-2 communication
- Meta-database data structure
- Very open data structure
- Very efficient and accurate mapping algorithms

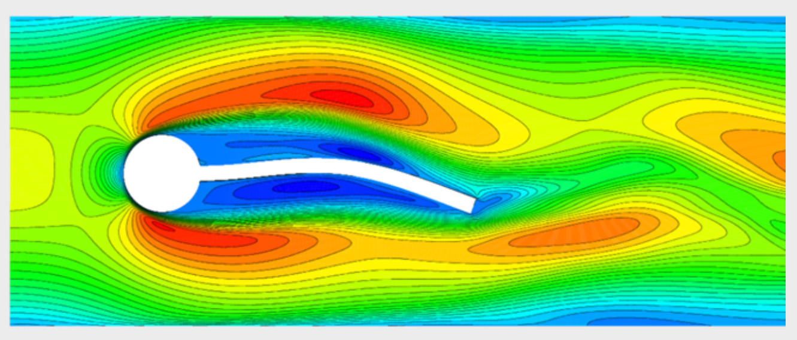
Spatial Mapping



EMPEROR handles non-matching grid problems. The user can choose between conservative and consistent mapping methods. Highly efficient mapping methods based on dual mortar are also available. The consistent condition of mapping the data field $u(x,y,z)$ from surface mesh A to surface mesh B is:

$$u(x,y,z)_B = u(x,y,z)_A$$

The mortar method can be derived for mapping by incorporating a weighted residual method for the interface. From discretization we have



$$u(x,y,z)_A = N_A^T \cdot u_A$$
$$u(x,y,z)_B = N_B^T \cdot u_B$$

where N represents the shape function vector.
The weighted residual method may be written as

$$\int_B N_B N_B^T \cdot u_B \, dB = \int_B N_B N_A^T \cdot u_A \, dB$$

Note that the integration is carried out over boundary B on both sides.
Written in matrix form

$$C_{BB} \cdot u_B = C_{BA} \cdot u_A$$

Since the matrix C_{BB} is a sparse matrix, solving the equation system is more efficient than a full matrix.

Co-Simulation for N Codes

One major challenge is the design of Co-Simulation algorithms for a large number of different simulation tools. The main focus of our research is to develop robust, efficient and accurate Co-Simulation methods.

Coupling Algorithms for N Codes

The coupling algorithm for a large number of clients should have no flow dependency in order to avoid a computational load imbalance of the Co-Simulation. Therefore Jacobi like communication patterns are well suited for these kind of simulations. However these loose coupling schemes may suffer from stability and accuracy issues. EMPIRE has a new approach for tackling these problems while preserving flexibility and reusability of client codes.

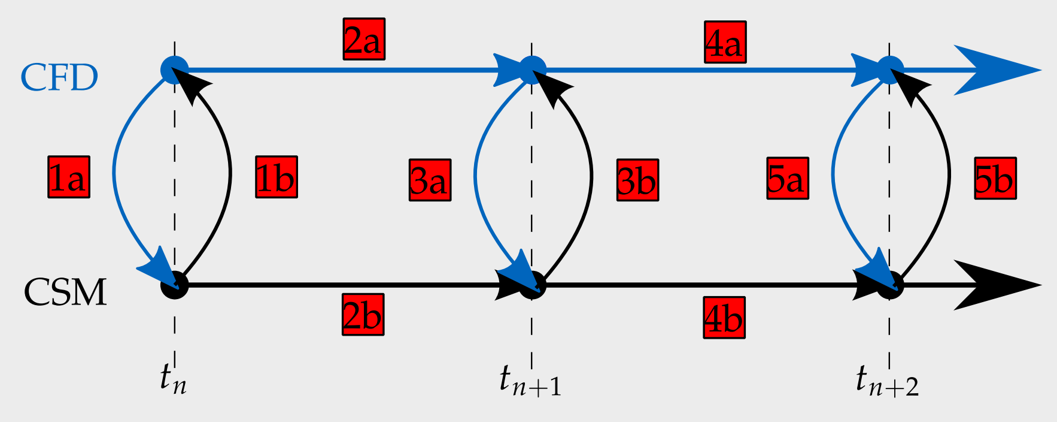


Figure 5: Communication pattern for loose Jacobi scheme

Demonstration Example - Multiphysics Segway

The Multiphysics Segway is a model problem to test the coupling algorithms which we developed. The idea is to build an inverted pendulum like problem which is evolutionary. The first evolution should be the classical inverted pendulum (segway). In a second stage the rigid truss should be replaced by a very flexible beam which adds the need of a structural nonlinear solver. The last stage should activate the surrounding fluid by attaching a sail like structure to the top of the flexible beam which adds the necessity for a fluid solver to the Co-Simulation scenario. The model should be built in hardware in order to compare simulations to reality. For the last stage the following codes need to participate in the Co-Simulation.

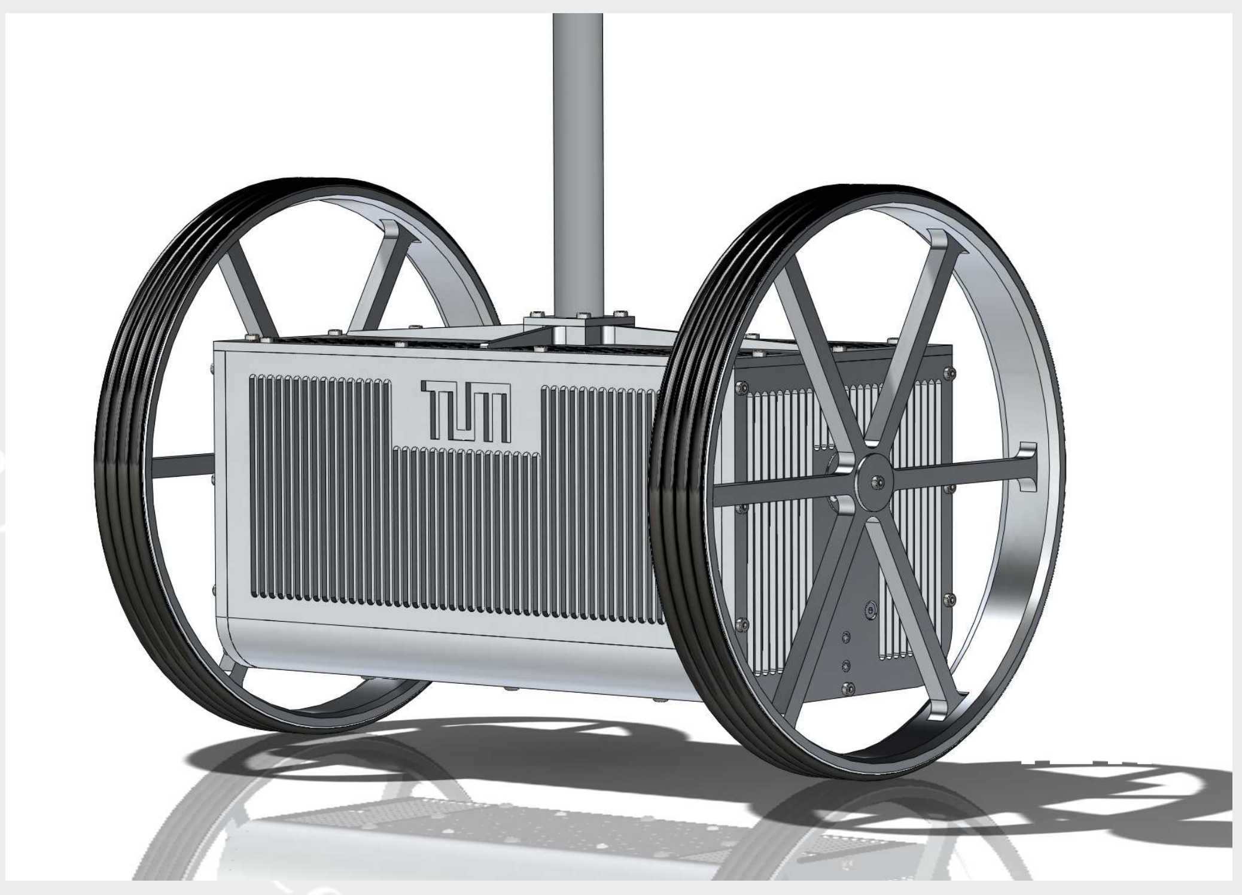


Figure 6: CAD model of the Multiphysics Segway

- Controller
- Rigid Body Dynamics
- Sensor Dynamics
- Actor Dynamics
- Structural Dynamics
- Computational Fluid Dynamics

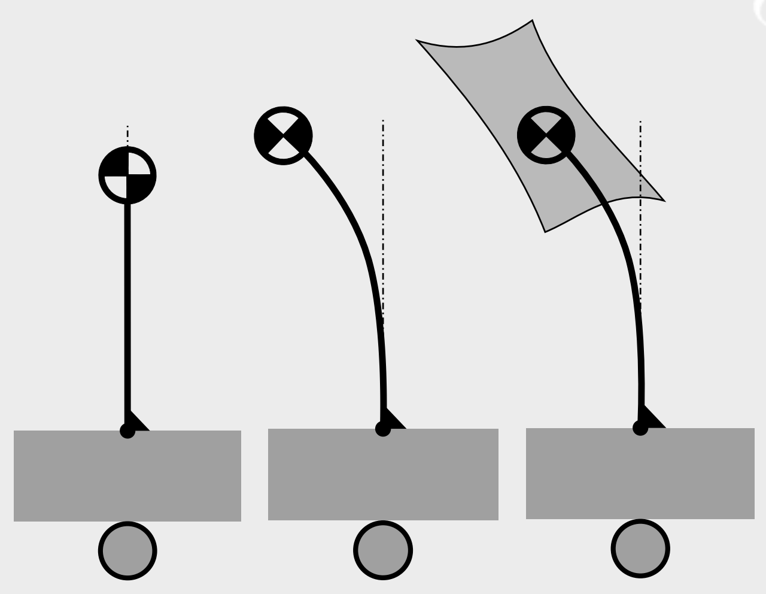


Figure 7: Stages of Multiphysics Segway

Supervisor: K.-U. Bletzinger
Advisor: R. Wüchner