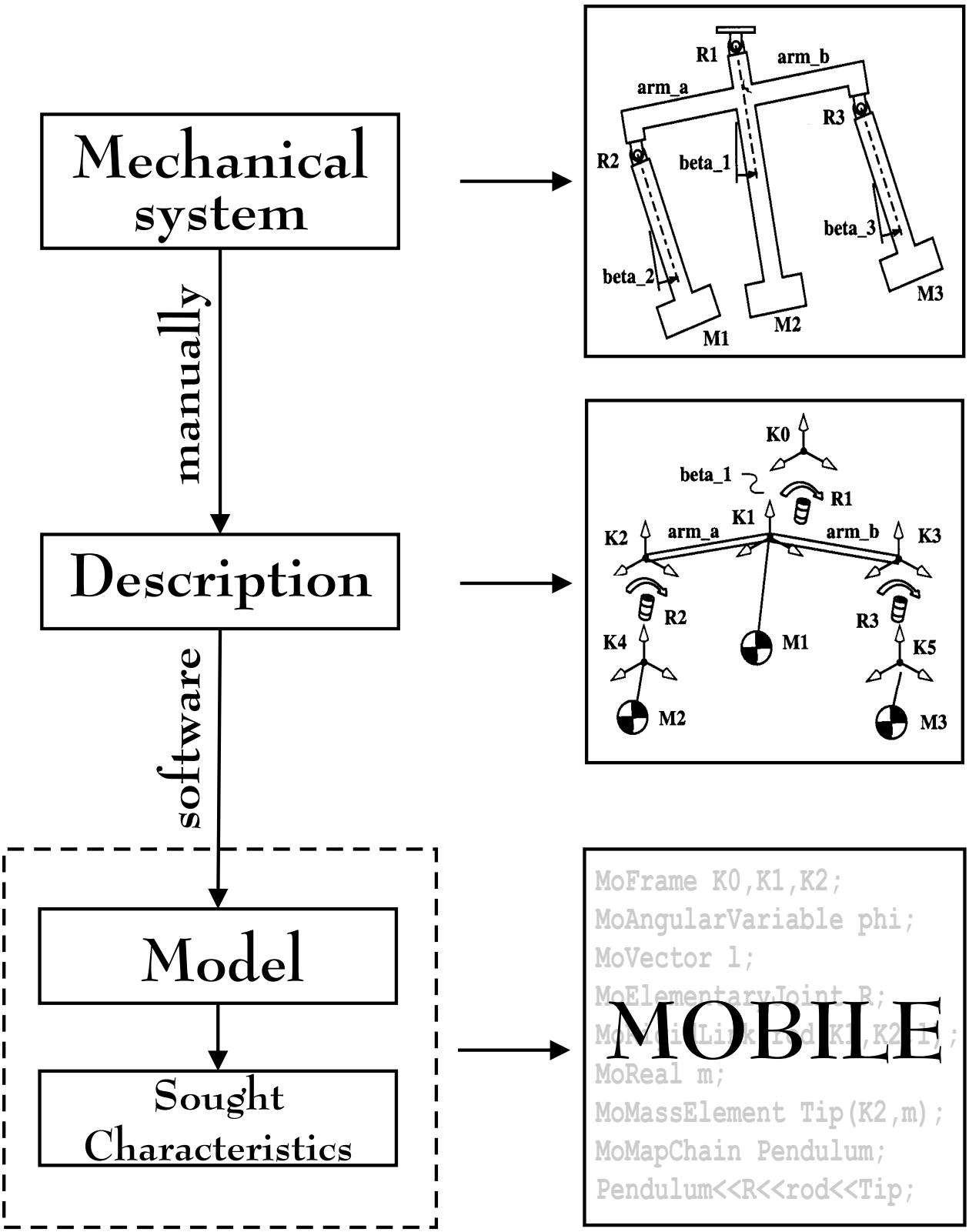
Taylor Models and Multibody Modeling in MOBILE

E. Auer

University of Duisburg-Essen, Germany

December 2004

1 The Task

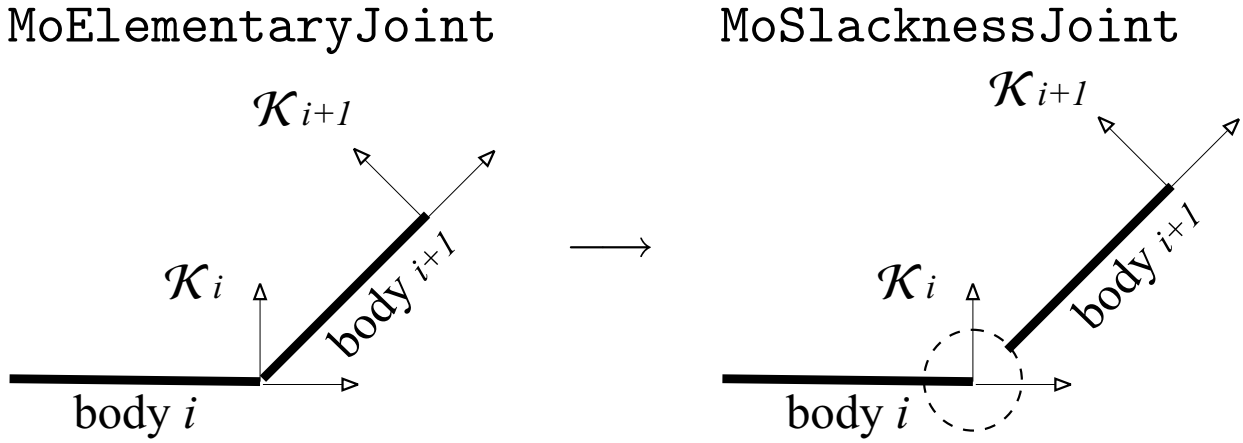

Mechanical system
manually
Description
software
Model
Sought Characteristics
R1
arm_b
arm_a
R3
R2
beta_1
beta_3
beta_2
M3
M1
M2
K0
beta_1
R1
K1
K2
arm_a
arm_b
K3
R2
R3
K4
M1
K5
M2
M3
MoFrame K0,K1,K2;
MoAngularVariable phi;
MoVector l;
MoElementaryJoint R;
MoRigidLink rod(K1,K2,l);
MoReal m;
MoMassElement Tip(K2,m);
MoMapChain Pendulum;
Pendulum<<R<<rod<<Tip;
MOBILE

1.1 Sought Characteristics

- kinematic behavior of a system
- dynamic behavior of a system
- equilibrium of a system, etc.

1.2 Main Reasons for Using Interval Arithmetic in MOBILE

- guaranteed correctness of results (verification)
- through allowing uncertainty in parameters:
  - uniform study of system's behavior for different parameters
  - more realistic models:



(see C. Hörsken, H. Traczinski, *Modeling of Multibody Systems with Interval Arithmetic*, 2001)

1.3 Goals and Functions of the ext. MOBILE

The goal:	Given a system's description in MOBILE, provide its validated characteristics
Achievements:	kinematics, dynamics, equilibrium validated
Drawbacks:	wrapping effect

1.4 The Task for This Talk

Study the possibilities of wrapping effect reduction through using COSY in MOBILE

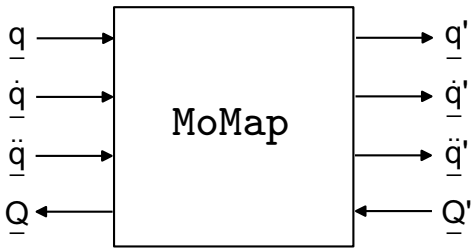
1.5 The Topics of This Talk

- Basics on MOBILE
- Basics on the extended MOBILE:
  - our approach to verified modeling
  - notes on implementation
  - some results
  - notes on wrapping effect
- Introduction of COSY into the ext. MOBILE:
  - implementation
  - kinematics: `TMoSlacknessJoint`
  - an outlook on dynamics

2 Basics on MOBILE

2.1 MOBILE's Structure (Kecskeméthy, 1993)

Transmission elements



`MoRigidLink`, ...

$$\begin{aligned} \underline{q}' &= \underline{\phi}(\underline{q}) \\ \dot{\underline{q}}' &= \mathbf{J}_\phi\, \dot{\underline{q}} \\ \ddot{\underline{q}}' &= \mathbf{J}_\phi\, \ddot{\underline{q}} + \dot{\mathbf{J}}_\phi\, \dot{\underline{q}} \\ \underline{Q} &= \mathbf{J}_\phi^{\mathrm{T}}\, \underline{Q}' \end{aligned}$$

↓

Global kinematics

`doMotion()`

$$\begin{aligned} \mathrm{Joint}_1 &: \mathcal{K}_0 \overset{f}{\mapsto} \mathcal{K}_1 \\ \mathrm{Joint}_2 &: \mathcal{K}_1 \overset{g}{\mapsto} \mathcal{K}_2 \\ \mathcal{K}_0 \overset{\varphi}{\mapsto} \mathcal{K}_2 &: \varphi = g \circ f \end{aligned}$$

↓

Equations of motion

`MoEqmBuilder`

$$\begin{aligned} M(\underline{q};t)\, \ddot{\underline{q}} + \underline{b}(\underline{q}, \dot{\underline{q}}; t) \\ = \underline{Q}(\underline{q}, \dot{\underline{q}}; t) \end{aligned}$$

↓

State-space form

`MoMechanicalSystem`

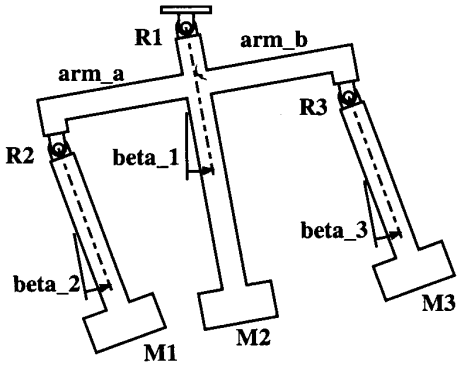
$$\begin{cases} \dot{y} = f(y;t) \\ y_0 = y(t_0) \end{cases}$$

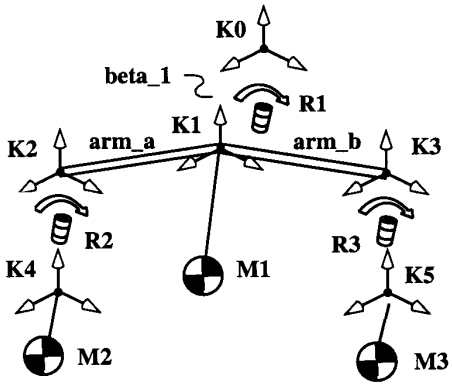
↓

Solution `MoIntegrator` $y(y_0;t)$

2.2 Example: a Triple Pendulum

The system and **its description**



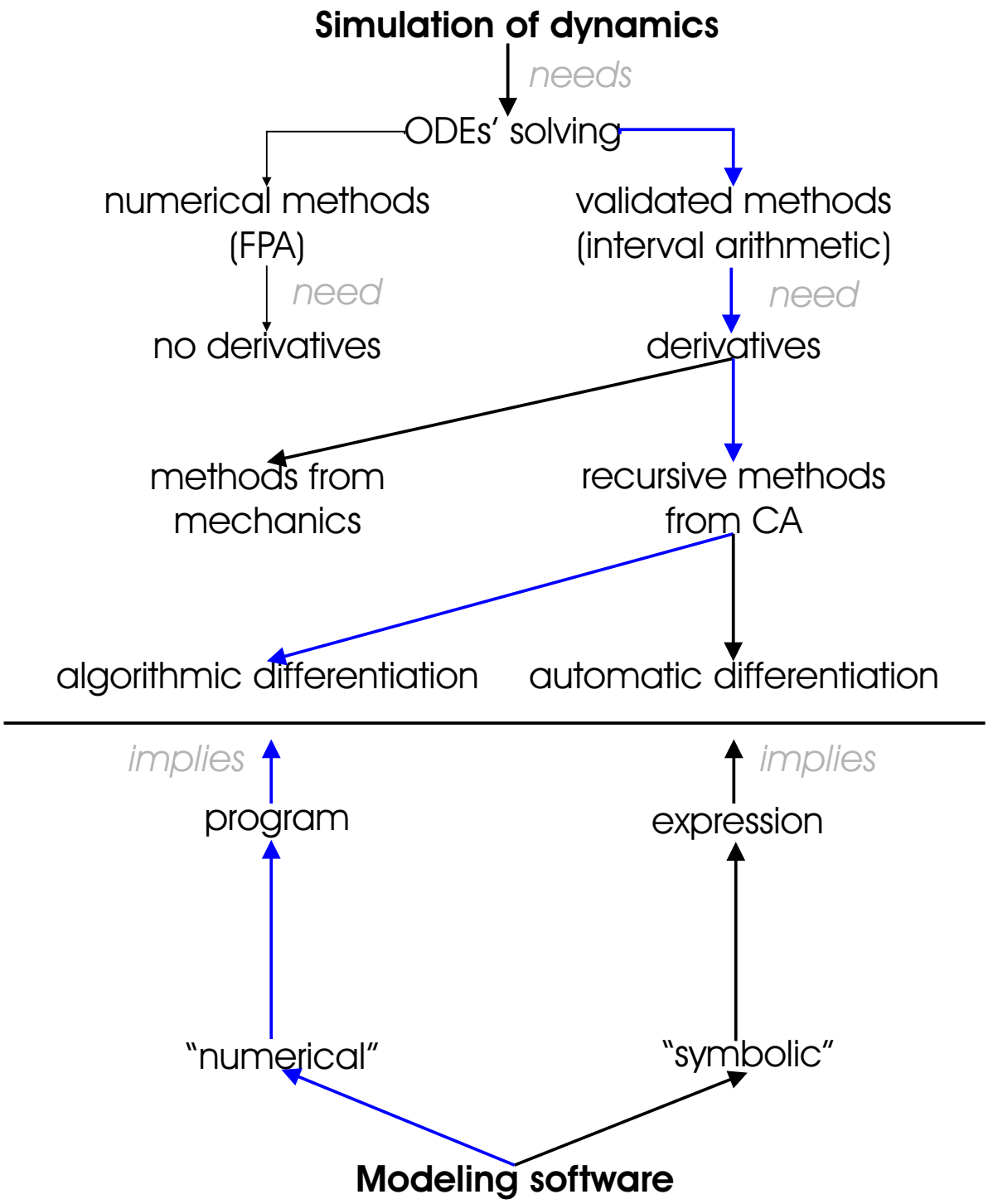


The MOBILE program

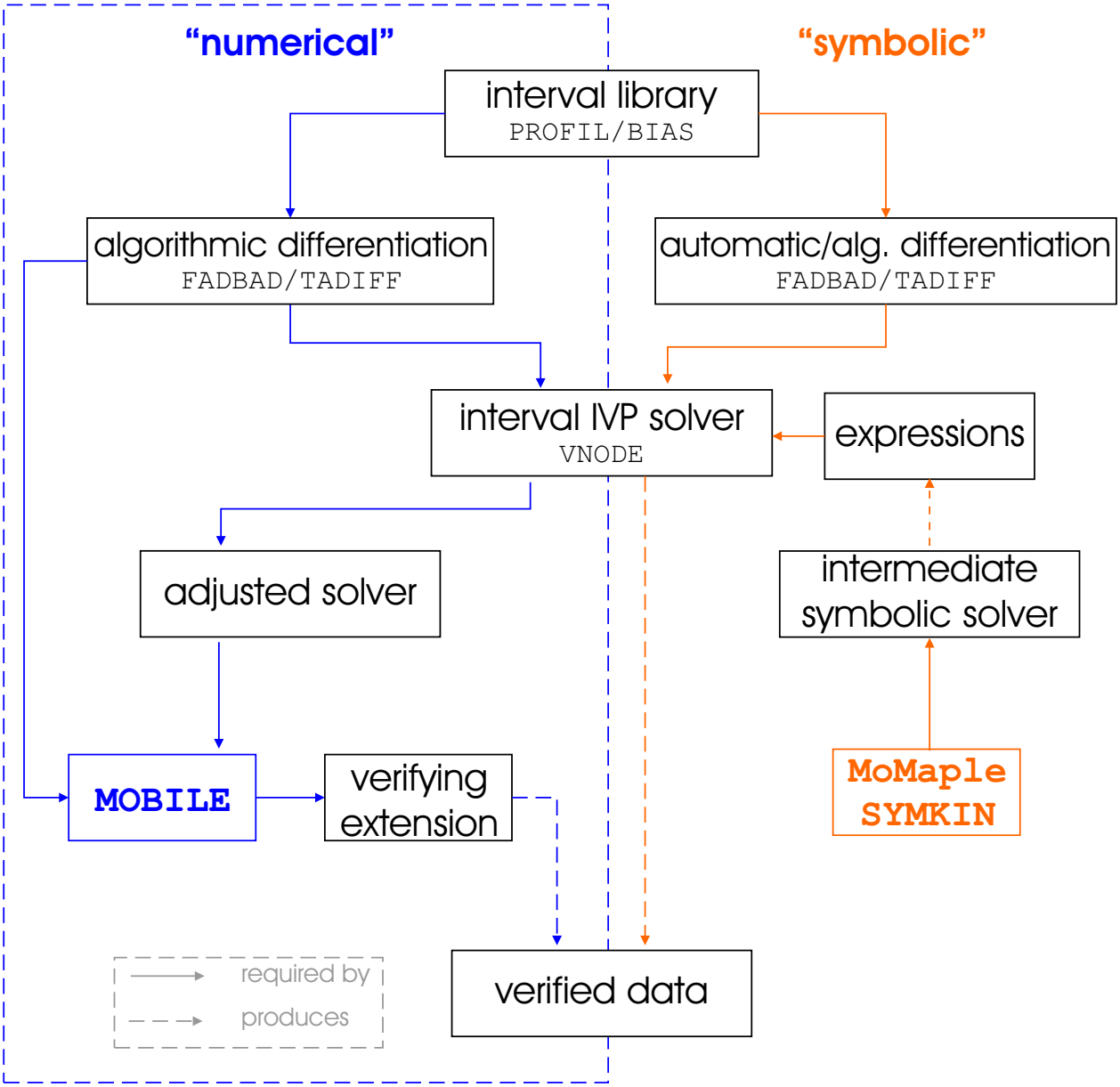
```
MoFrame K0, K1, K2, K3, K4, K5 ;
MoAngularVariable beta_1, beta_2, beta_3 ;
MoElementaryJoint R1 ( K0, K1, beta_1, x_axis ) ;
MoElementaryJoint R2 ( K2, K4, beta_2, x_axis ) ;
MoElementaryJoint R3 ( K3, K5, beta_3, x_axis ) ;
MoVector a_1, a_2, a_3, a_4 ;
MoRigidLink arm_a ( K1, K2, a_1 ) ;
MoRigidLink arm_b ( K1, K3, a_2 ) ;
MoReal m1, m2 ;
MoMassElement M1 ( K1, m1, a_3 ) ;
MoMassElement M2 ( K4, m2, a_4 ) ;
MoMassElement M3 ( K5, m2, a_4 ) ;

MoMapChain pendulum ;
pendulum << R1 << arm_a << arm_b << R2
         << R3 << M1 << M2 << M3 ;

MoVariableList q ; q << beta_1 << beta_2 << beta_3 ;
MoMechanicalSystem sys ( q , pendulum , K0 , zAxis ) ;
MoAdamsIntegrator SystemIntegrator(sys) ;
for (int i=0;i<1000;i++) SystemIntegrator.doMotion() ;
```

3 Extended MOBILE

3.1 Choices for Modeling of Dynamics



3.2 Verification Approach and Software Choices



3.3 Design: Basic Data Type

Required data types:

- for ordinary values: `INTERVAL`
- for Taylor coefficients (TC): `TINTERVAL`
- for TC of the variational equation: `TFINTERVAL`

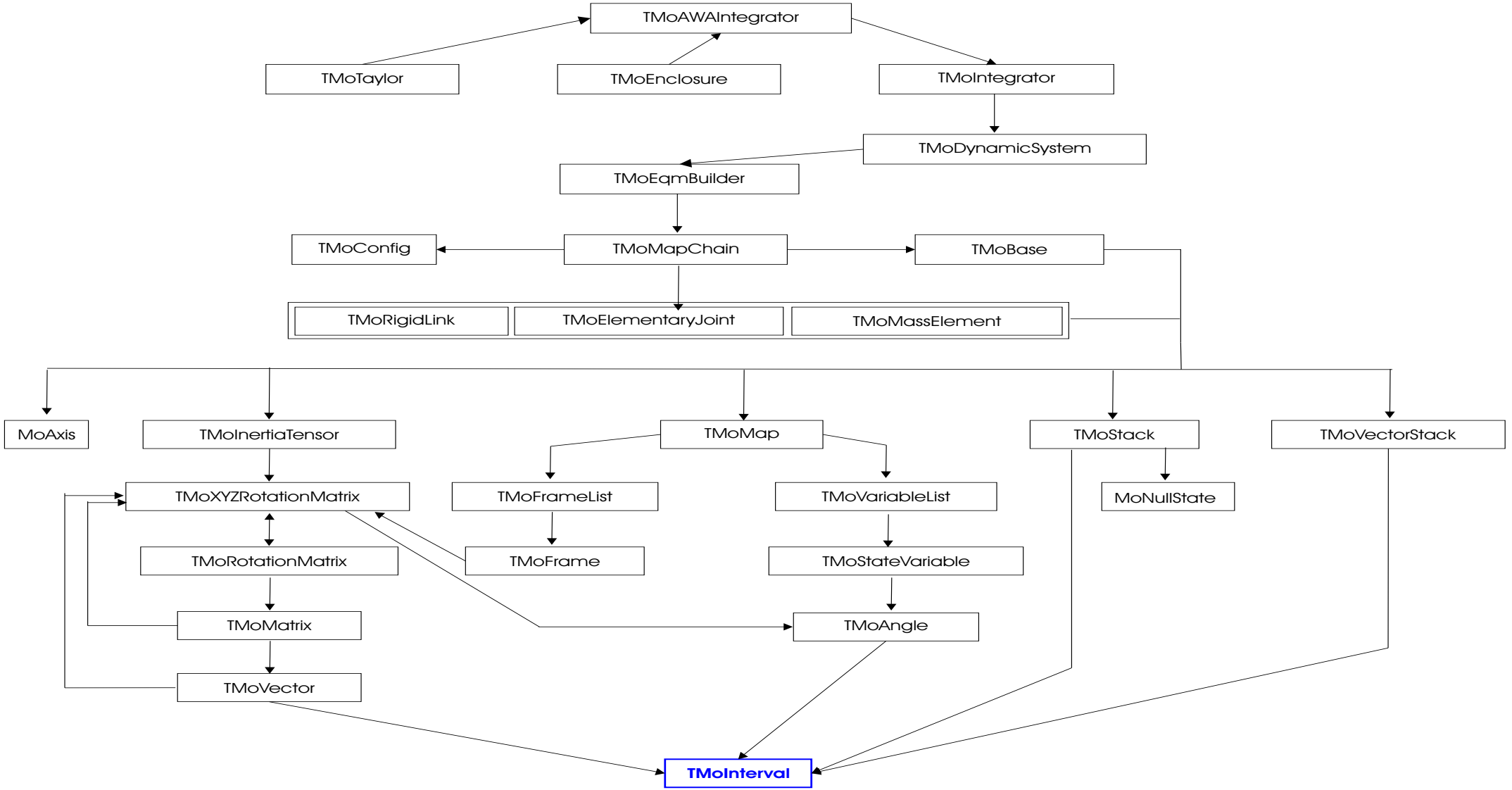
"The designing problem":

Given:	f_1 to compute the right side
Required:	2 types of DAGs
Extra code:	f_2, f_3 that differ in data types only

Our solution:

```
class TMoInterval{
    INTERVAL   Enclosure;
    TINTERVAL  TEnclosure;
    TFINTERVAL TFEnclosure;
}
```

3.4 Design Aspects: Universality



Goal: FP/Interval/... modeling in a single version

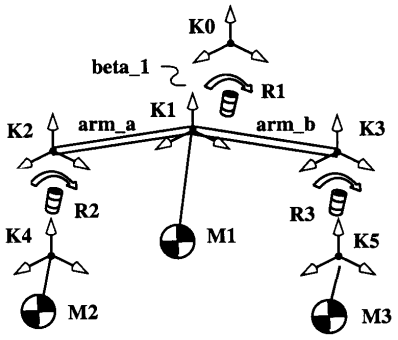
Our solution:

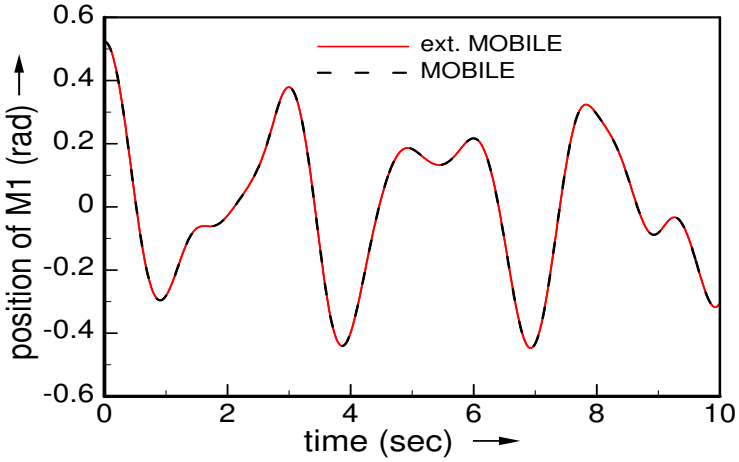
Replace MOBILE's classes with template classes

Example:

```
typedef TMoInterval mytype; //typedef MoReal mytype;

TMoFrame<mytype> K0, K1, K2, K3, K4, K5 ;
TMoAngularVariable<mytype> beta_1, beta_2, beta_3 ;
TMoElementaryJoint<mytype> R1 ( K0, K1, beta_1, x_axis ) ;
TMoElementaryJoint<mytype> R2 ( K2, K4, beta_2, x_axis ) ;
TMoElementaryJoint<mytype> R3 ( K3, K5, beta_3, x_axis ) ;
TMoVector<mytype> a_1, a_2, a_3, a_4 ;
TMoRigidLink<mytype> arm_a ( K1, K2, a_1 ) ;
TMoRigidLink<mytype> arm_b ( K1, K3, a_2 ) ;
mytype m1, m2 ;
TMoMassElement<mytype> M1 ( K1, m1, a_3 ) ;
TMoMassElement<mytype> M2 ( K4, m2, a_4 ) ;
TMoMassElement<mytype> M3 ( K5, m2, a_4 ) ;

TMoMapChain<mytype> pendulum ;
pendulum << R1 << arm_a << arm_b << R2 << R3 << M1 << M2 << M3;

TMoVariableList<mytype> q ; q << beta_1 << beta_2 << beta_3;
TMoMechanicalSystem<mytype> sys (q,pendulum,K0,zAxis) ;
TMoAWAIntegrator SystemIntegrator(sys,0.01,ITS_QR,15) ;
//MoAdamsIntegrator SystemIntegrator(sys) ;
SystemIntegrator.doMotion() ;
```

3.5 Example: the Triple Pendulum





Initial conditions:

$\beta_1 = 30°$,

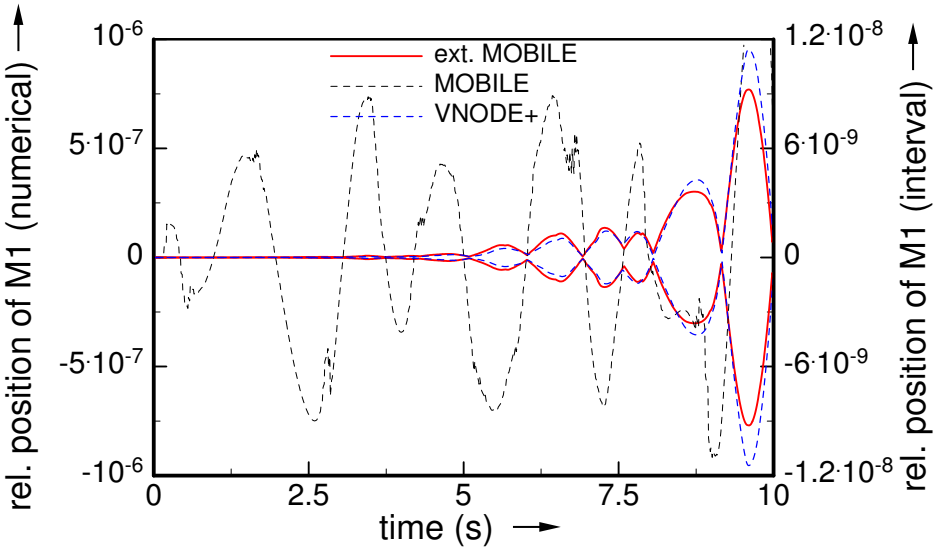
$\beta_2 = 10°$,

$\beta_3 = -10°$

Time: 646 s

Error: $1.61e-07$

Break-down: 33.6



- "symbolic" and "numerical" solutions intersect
- the trajectory is verified
- the standard MOBILE's solution is inaccurate

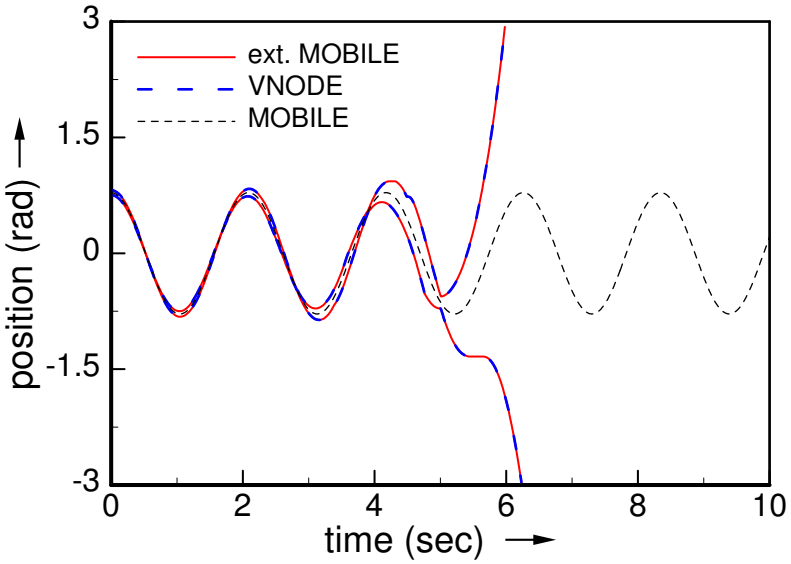
3.6 Achievements

- verified simulation of kinematics:
  - transmission functions
  - equilibrium
- verified simulation of dynamics:
  - (non-)autonomous systems
  - explicitly solvable closed-loop systems
- universality with respect to the basic data type

3.7 Drawbacks

- computing time
- wrapping effect:

  a simple pendulum ($\beta = [0.76969, 0.801106] \ni \frac{\pi}{4}$)

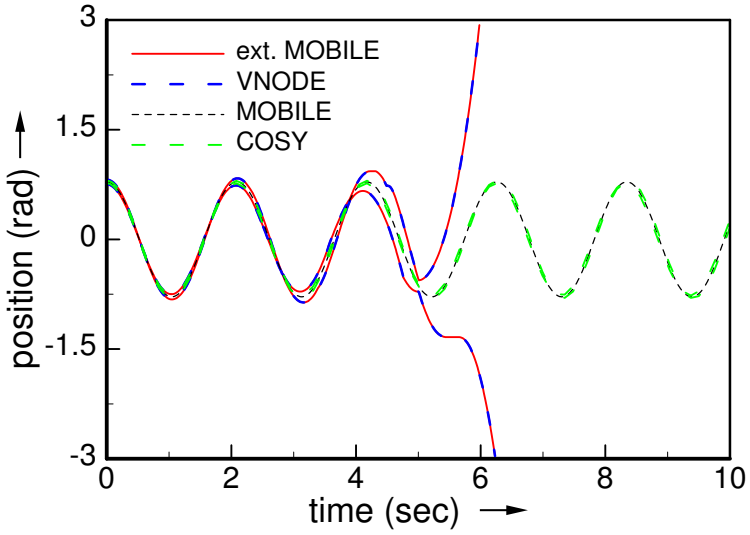


Solution of ext. MOBILE is comparable to VNODE's solution to $\ddot{y} + 9.81 \sin y = 0$!

3.8 Possible Solutions

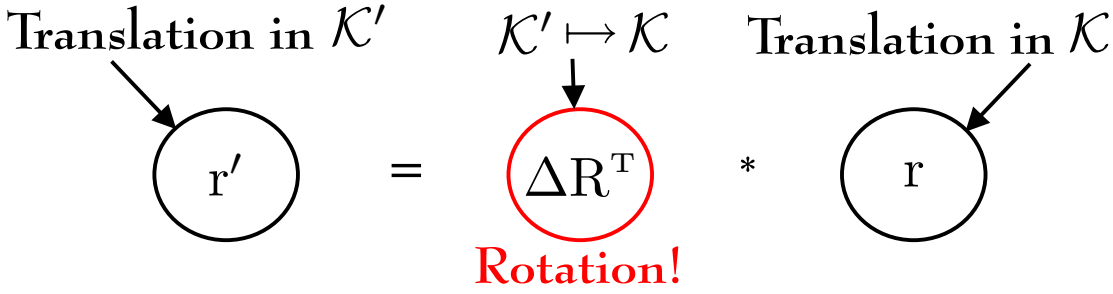
- **Use Taylor models** (kinematics and dynamics):

COSY-VI solution to $\ddot{y} + 9.81 \sin y = 0$



- **Eliminate rotation errors** (kinematics only):

Example of a revolute joint (`TMoElementaryJoint`)



$$[r] = \mathrm{mid}([r]) + [\varepsilon],$$

$$[r'] = [\Delta R^T]\mathrm{mid}([r]) + [\Delta R^T][\varepsilon] = [\Delta R^T]\mathrm{mid}([r]) + [\varepsilon]$$

4 Taylor Models in MOBILE – First Steps

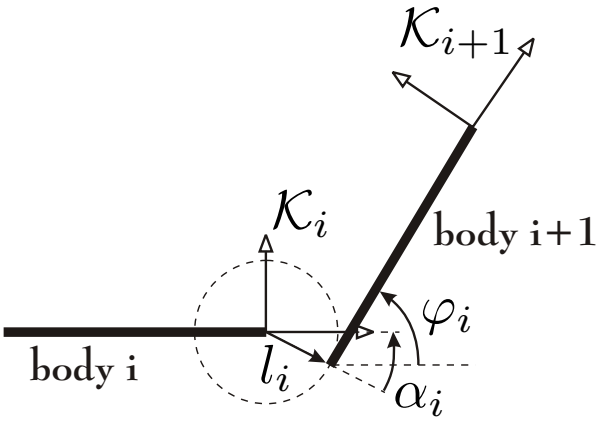
4.1 Basic Data Type – a Wrapper for RDA-Intervals

INTERVAL $\Longrightarrow$ RDAInterval

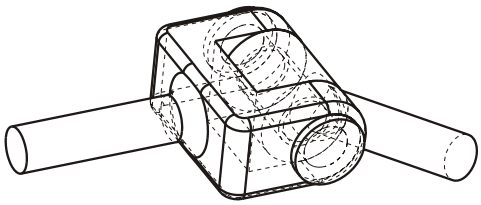
```
class RDAInterval {
private:
   Cosy *Enclosure;
public:
    Cosy& getEnclosure() {return *Enclosure;}
    RDAInterval();
    RDAInterval(const RDAInterval&);
    RDAInterval& operator=(const RDAInterval&);
    RDAInterval(const INTERVAL& mi){
       wid=Diam(mi); x0=Mid(mi);
       *Enclosure=x0+0.5*wid*rda(i,0,0,2);
    }
     RDAInterval(MoReal , MoReal );
    ...
    ~RDAInterval(){delete Enclosure;}
    RDAInterval& operator+=(const RDAInterval&);
    ...
    friend RDAInterval sqrt(const RDAInterval&);
};
```

4.2 Kinematics with COSY: `TMoSlacknessJoint`

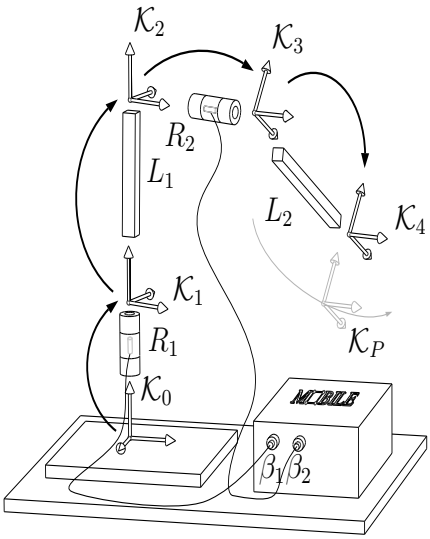
Scheme



3D-Drawing



A Simple Manipulator with Sloppy Joints



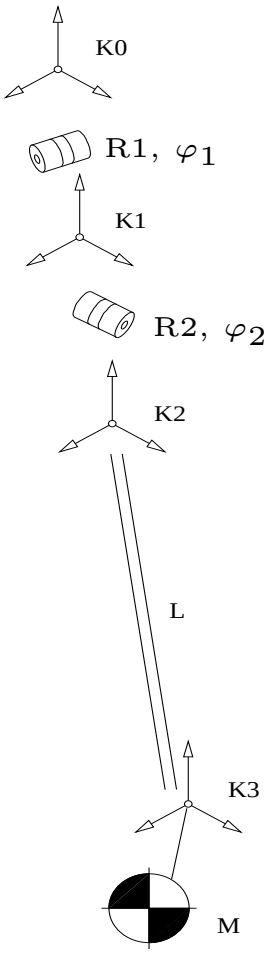
$l_{1,2} = [0; 0.02]$

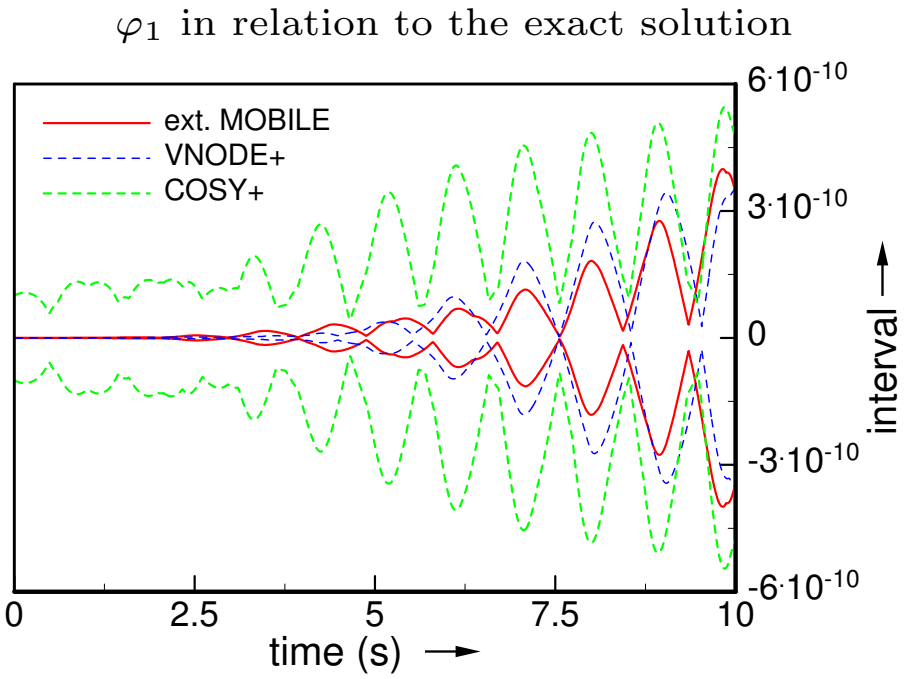
Strategy	Position (y, z)	%
`TMoInterval`	[5.6224..., 5.7406...] [1.8422..., 1.9604...]	–
`TMoInterval`*	[5.6311..., 5.7319...] [1.8509..., 1.9517...]	15% 15%
`RDAInterval`	[5.6563..., 5.7067...] [1.8602..., 1.9425...]	57% 30%

* with rotation error elimination

4.3 Dynamics with COSY: the Symbolic Case

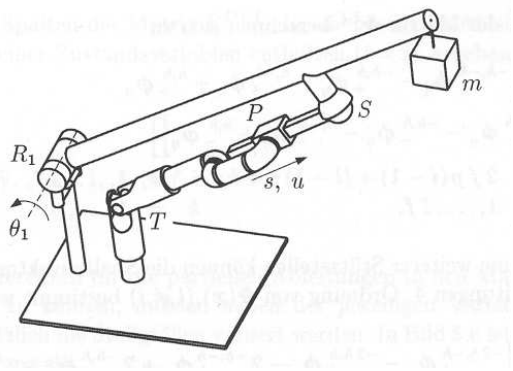
Examples: a Conical Pendulum

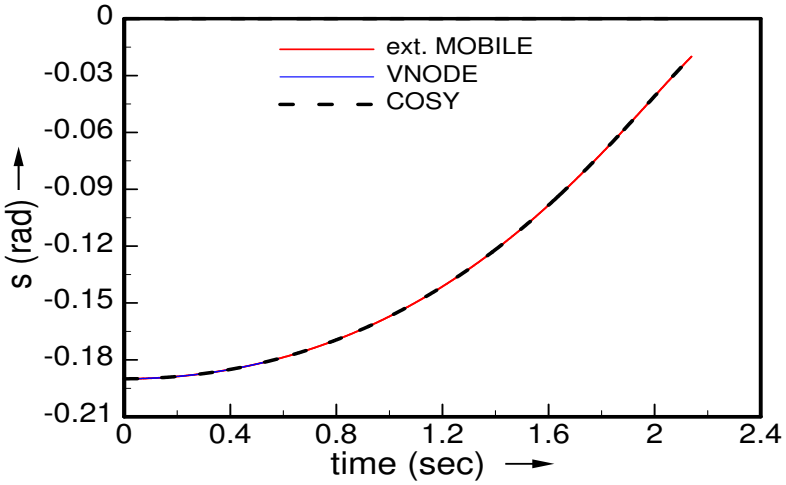




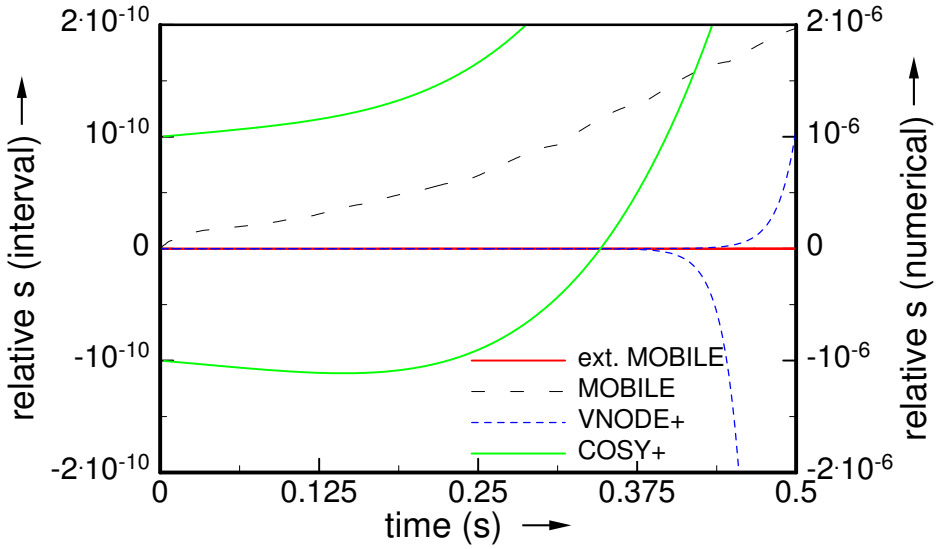
Data	VNODE+	E-MOBILE	COSY+
Time (s)	21.7+	235	1417.9+
Global error	$1.67e-09$	$2.28e-09$	$3.33e-09$
Break-down	110.78	105.06	158.4

Examples: A One-Arm Manipulator





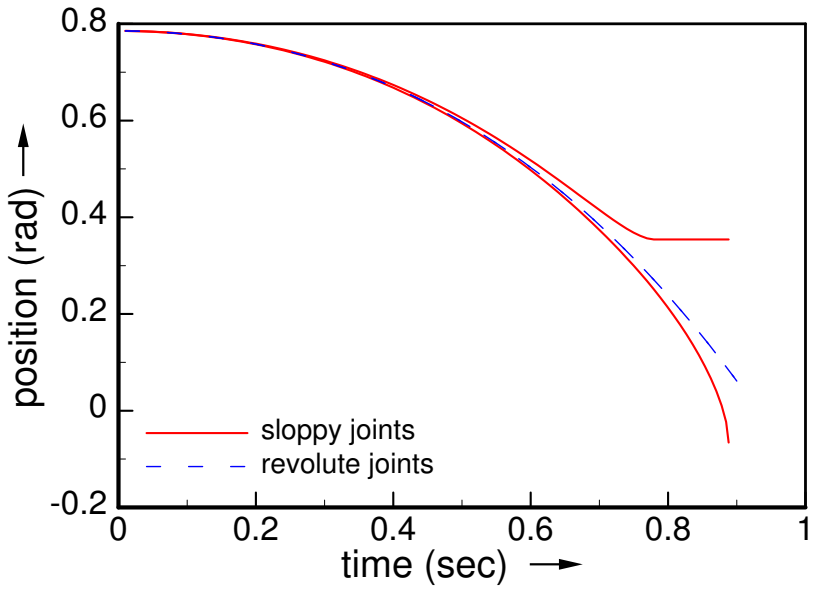
Stepsize: 0.001



Data	VNODE+	E-MOBILE	COSY+
Time (s)	117+	1758*	318+
Global error	$1.35e-08$	$1.0e-12$	$3.28e-10$
Break-down	0.582	2.139	2.108

4.4 Dynamics with COSY: an Outlook on the Numerical Case

Dynamic behavior with `TMoSlacknessJoint` (ext. MOBILE): unsatisfactory



Possible Solutions:

- VNODE with Taylor models
  - implementation necessary
  - computing time ?
- COSY-VI
  - C++ implementation necessary
  - computing time too high

5 Summary

Our Goal: Verified modeling in MOBILE

Our Solution:

- MOBILE + Intervals + AD
- universality through templates

Achievements:

- (Non-)autonomous systems simulated
- A closed-loop system verified

Problems: Computing time and wrapping

Solutions:

- kinematics: REE or TM
- dynamics: still open
  - TM + VNODE ?
  - COSY-VI in C++?