Simulation Tools FMNN05
Project 2

Nils Fagerberg, Daniel Odenbrand, Felicia Seemann and Linus Jangland December 3, 2015

1 Background

As a part of the course Simulation Tools, FMNN05, at LTH there are three projects related to simulating solutions of differential equations using the tools Assimulo/Sundials, Dymola and the programming languages Python and Modelica.

Every project is divided into smaller tasks. For this project we have separated the tasks concerning Python (Assimulo) and Dymola. This is the report for project 2.

1.1 Andrew's Squeezer Mechanism

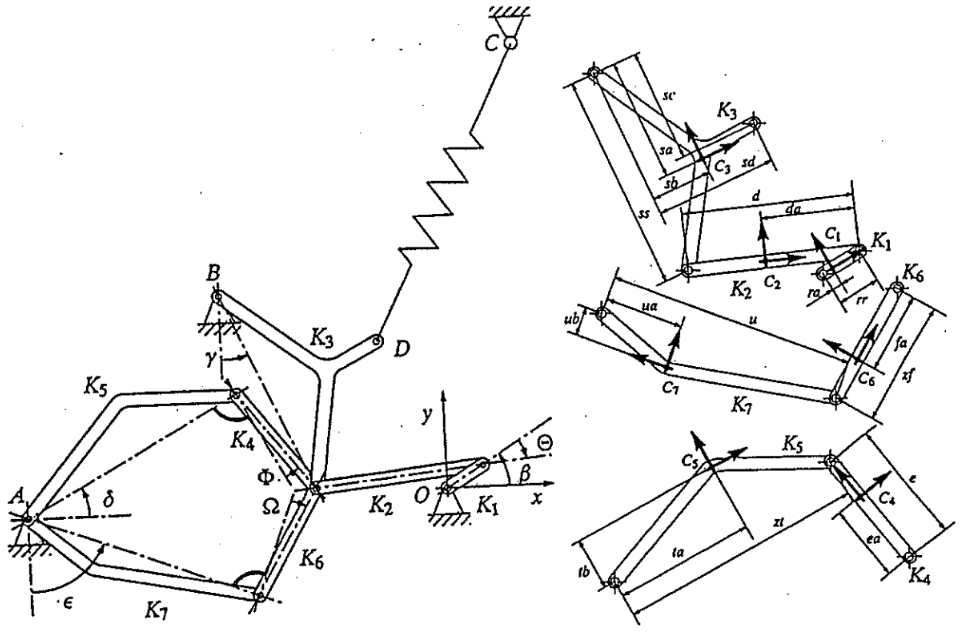

Figure 1: Representation of a squeezer from reference [1]

Andrew's squeezer mechanism is a planar multibody mechanism consisting of 7 rigid bodies, who are all connected via joints without any friction. The squeezer is assumed to be driven by a motor with constant drive torque. The equations of motion are derived using the angles in the system $q = [\beta, \Theta, \gamma, \phi, \delta, \Omega, \epsilon]$. The Index of a system of differential algebraic equations (DAE) is the number of derivatives necessary to find consistent initial values. We get the Index

3 Formulation with v and w as newly introduced variables:

$$\dot{q} = v \tag{1}$$
$$\dot{v} = w \tag{2}$$
$$0 = M(q) \cdot w - f(q, v) + G^T(q) \cdot \lambda \tag{3}$$
$$0 = g(q) \tag{4}$$

To obtain the Index 2 Formulation we differentiate equation 4 with respect to q and get:

$$0 = G(q)v, \tag{5}$$

where $G(q)$ is the Jacobian of $g(q)$. We can get the Index 1 Formulation with an additional differentiation:

$$0 = g_{qq}(q)(v, v) + G(q)w, \tag{6}$$

where $g_{qq}(q)(v, v) = v^T \cdot g_{qq}(q) \cdot v$.

In this project, we set up a model of this squeezer mechanism and simulate it using first Assimulo, then Dymola. For both softwares we implement the model with parameters and constraints as reported by Hairer and Wanner in [1].

2 Assimulo

2.1 Initial values

Hairer and Wanner provide the initial conditions for solving the DAE describing Andrew's squeezer mechanism in [1, p.535]. By fixing the parameter $\theta = 0$ and then computing the solution to $g(q) = 0$ using Newton iteration, the initial values can be derived. In Table 1 the given and calculated initial values are reported, together with the difference between them. We observe that the calculated and the given initial differ very little, so any of the values can be used.

Angle	Given values	Calculated values	Difference
β	-0.061	-0.061	$2.9 \cdot 10^{-12}$
θ	Fixed to 0	Fixed to 0	-
γ	0.455	0.455	$2.4 \cdot 10^{-12}$
ϕ	0.223	0.223	$1.8 \cdot 10^{-11}$
δ	0.487	0.487	$-3.4 \cdot 10^{-12}$
ω	-0.223	-0.223	$1.1 \cdot 10^{-12}$
ϵ	1.231	1.231	$-1.8 \cdot 10^{-11}$

Table 1: Difference between the given initial values and the ones calculated using Newton iteration.

2.2 Simulation

The DAE model with index 3 and index 2 formulation was implemented in python and simulated using the IDA solver in Assimulo. The time interval $[0, 0.03]$ seconds was used for all simulations, with the number of communication points set to 500. In order to obtain a convergent solution the solver might require some variables to be suppressed due to the introduction of a local error. This error does not affect the solution globally but the error might still be larger than the selected tolerance which leads to the solver complaining. This can be done either by changing the tolerances for the different variables $[v, \dot{v}, \lambda]$, or by using the python command `suppress_alg` which controls the error estimator of the solver. The different tolerance and suppression settings for the index 3 formulation are presented in Table 2, and for the index 2 formulation in Table 3.

Variables	Tolerance	Suppressed variables	Convergence?
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{-6}]$	-	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{5}]$	-	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{5}, 10^{5}]$	-	Yes
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{-6}]$	λ	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{-6}]$	$\dot{v}, \lambda$	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{5}]$	λ	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{5}, 10^{5}]$	$\dot{v}, \lambda$	Yes

Table 2: Simulation settings for the index 3 formulation.

Variables	Tolerance	Suppressed variables	Convergence?
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{-6}]$	-	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{5}]$	-	Yes
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{5}, 10^{5}]$	-	Yes
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{-6}]$	λ	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{-6}]$	$\dot{v}, \lambda$	No
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{-6}, 10^{5}]$	λ	Yes
$[v, \dot{v}, \lambda]$	$[10^{-6}, 10^{5}, 10^{5}]$	$\dot{v}, \lambda$	Yes

Table 3: Simulation settings for the index 2 formulation.

We note that we need to set the tolerance of λ to 10^5 in order to obtain convergence for the index 2 formulation. For index 3 both $\dot{v}$ and λ need the tolerance 10^5. This is due to the fact that the Newton iteration matrix becomes ill-conditioned for some index formulations. The higher the index formulation, the worse conditioning we get, and therefore convergence issues.

In figures 2 and 3 the simulation results are presented for index 3 and index 2 respectively. Both plots shows a similair result, which also corresponds to the simulation results presented by Hairer and Wanner [1].

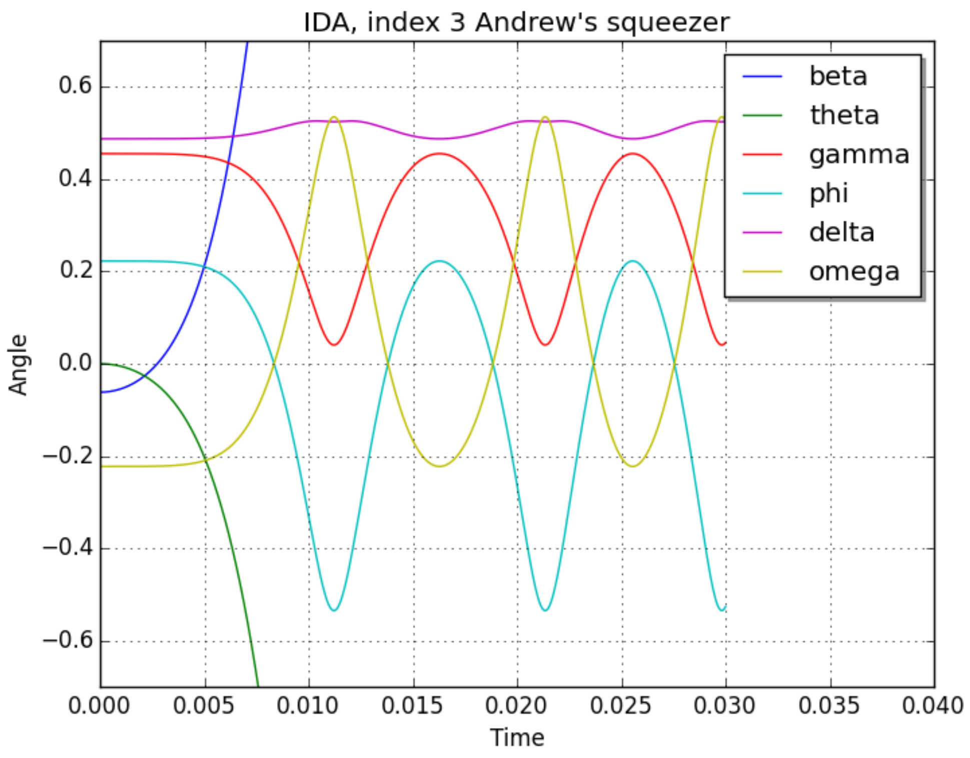

Figure 2: Simulation results for index 3 with high tolerances and suppressed $\dot{v}$ and λ.

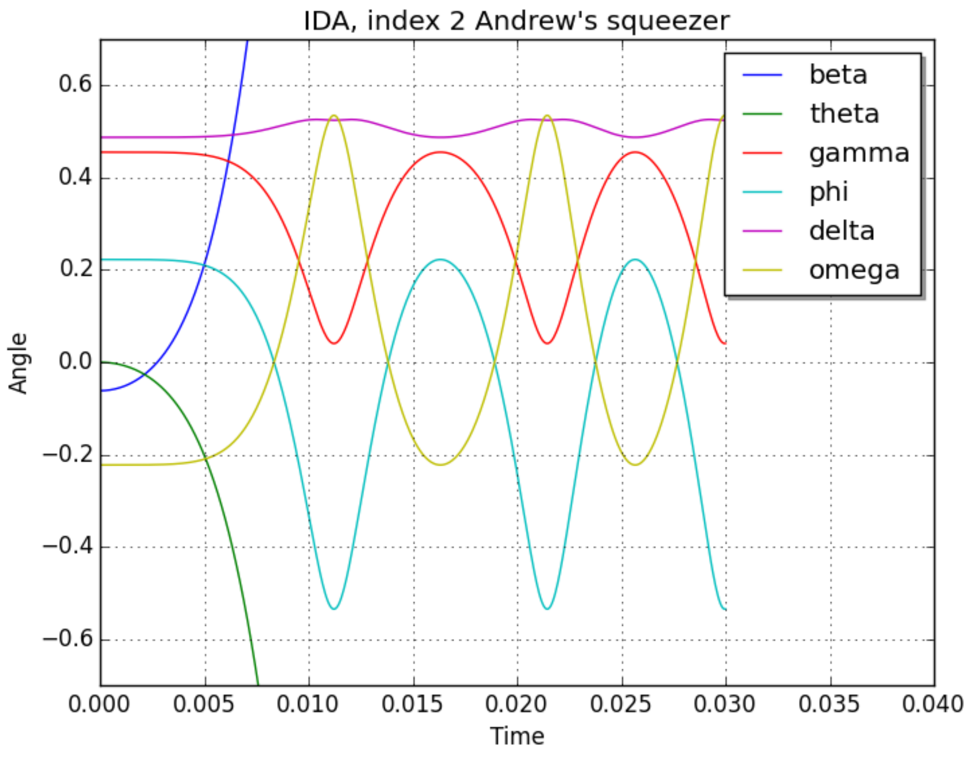

Figure 3: Simulation results for index 2 with high tolerance and suppressed λ.

2.3 Lagrange multipliers

When observing the simulation results for the Lagrange multipliers λ in figures 4 and 5, we note that the solution has some local instabilities for the index 3 formulation, while the index 2 formulation looks smooth and stable. In order to be able to study the step sizes used, shown in figures 6 and 7, no number of communication points where specified in this simulation.

For the index 3 formulation, we note that the step sizes changes quickly between two values during several parts of the simulation. The timing of these changes coincide with the local instabilities in the Lagrange multipliers, where the solver seem to have issues in controlling the error. We do not observe the same step size variations for the index 2 formulation, and the solution looks stable. Hence, we draw the conclusion that the suppressed velocities $\dot{v}$ in the index 3 formulation influence the error control of the solver, but if we would have allowed the controller to use the velocities we would have issues with the newton iteration instead, as discussed above.

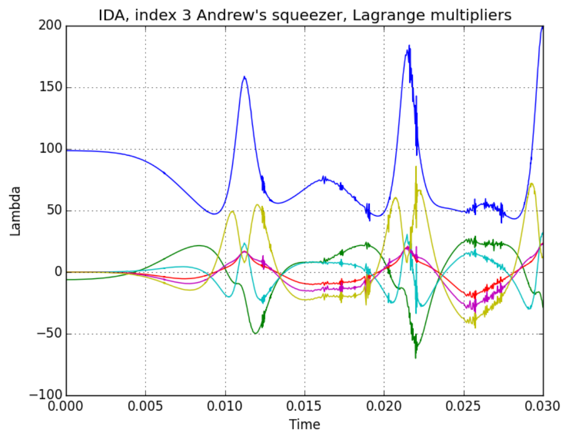

Figure 4: Simulation results of the Lagrange multipliers λ for index 3, with high tolerances and suppressed $\dot{v}$ and λ.

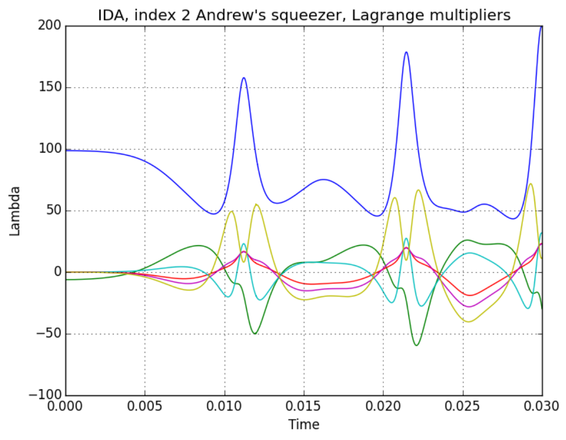

Figure 5: Simulation results of the Lagrange multipliers λ for index 2, with high tolerance and suppressed λ.

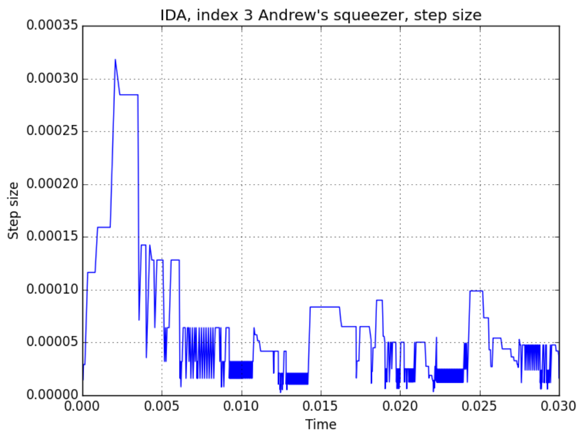

Figure 6: Step sizes for the index 3 simulation.

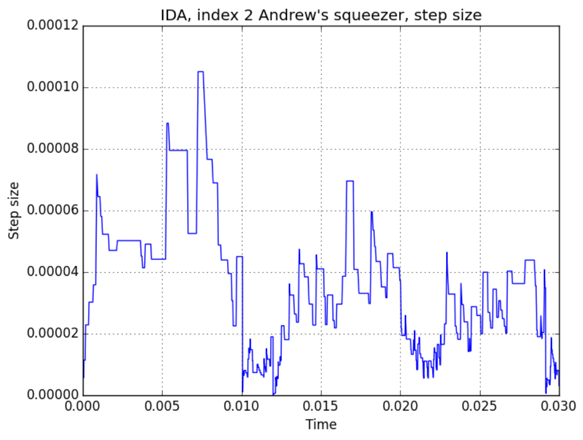

Figure 7: Step sizes for the index 2 simulation.

2.4 Runtime statistics

In table 4 the runtime statistics for solving the index 2 and index 3 formulation using IDA with 500 communication points can be seen. We see that we have more error test faliures for the index 2 formulation, which is to be expected since the velocities $\dot{v}$ are not suppressed as in the index 3 case. This yield more steps and function evaluations for the index 2 case. For the index 3 case we have more nonlinear convergence faliures, and therfore more of Jacobian evaluations and nonlinear function evaluations due to Jacobian evaluations. This is also to be expected, since the Newton iteration matrix is more ill-conditioned for the index 3 formulation than for index 2.

Index	2	3
Steps	1603	843
Function evaluations	2692	1811
Jacobian evaluations	227	834
Function eval. due to Jacobian eval.	4540	16680
Error test failures	81	9
Nonlinear iterations	2692	1811
Nonlinear convergence failures	1	261

Table 4: Runtime statistics for index 2 and index 3 formulation when solving with IDA.

2.5 Index 1 formulation

Next we investigate the index 1 formulation of the problem and try to solve it using an explicit Runge-Kutta method, Dormand-Prince method (DOPRI5). To do this we first need to reformulate the residual form as an explicit problem of the form:

$$\dot{y} = f(t, y) \tag{7}$$

$$y(t_0) = y_0 \tag{8}$$

From (3) and (6) we get the linear system:

$$\begin{pmatrix} M & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} w \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ -g_{qq}(v, v) \end{pmatrix} \tag{9}$$

Solving this and inserting the resulting w into (1) and (2) we get an explicit problem that can be solved using DOPRI5. The result of the simulation with DOPRI5 can be seen in figure 8. We see that there are significant drift-off errors, which is a well-known effect of reducing the index of a problem.

The runtime statistics can be seen in table 5.

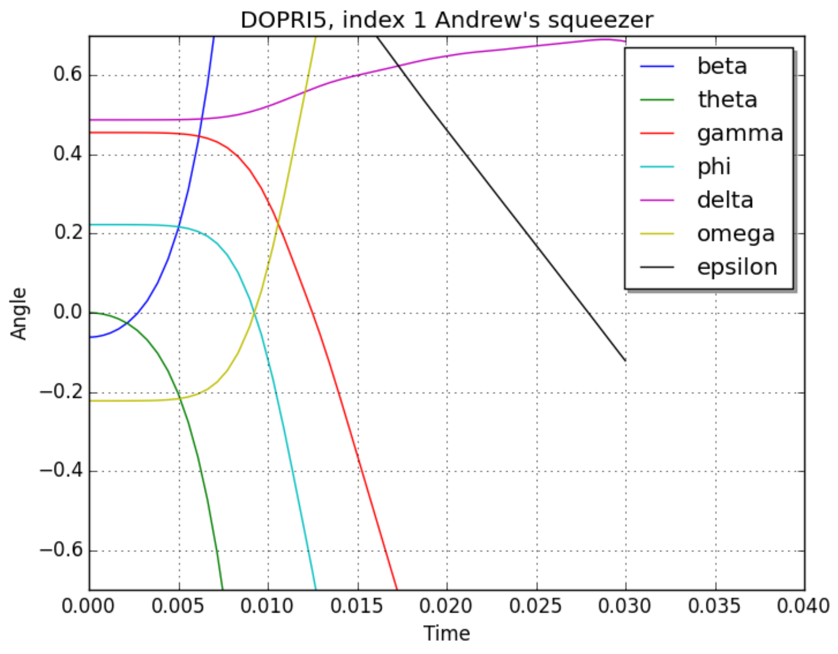

Figure 8: Simulation results for the index 1 formulation using DOPRI5.

Steps	85
Function evaluations	704
Error test failures	32

Table 5: Runtime statistics for the index 1 formulation when solving with DOPRI5.

3 Dymola

Dymola supports Modelica standard library which lets us "drag and drop" parts to build our (mechanical) systems with. Thus, using mainly the multibody library (see [2]) we were able to construct the squeezer with a combination of springs, revolutes, masses and a torque. Dymola then converts the connections between the different objects into a system of DAEs which it solves and simulates over time.

3.1 Degrees of freedom and constraints

Since the multibody library supports mechanical constructs in three dimensions and our squeezer mechanism only moves in two dimensions the underlying system of DAEs will become overconstrained if we use too many three-dimensional revolutes. A body in three dimensions has 6 degrees of freedom (x, y and z positions, rotation around the x, y and z axis). An ordinary revolute has 5 constraints and 1 degree of freedom (rotation around an axis). So connecting a revolute to a body will reduce its degrees of freedom by 5, thus giving it only one. Now consider a loop with 4 bodies, connected via 4 revolutes. That gives:

- 4 bodies = 24 degrees of freedom

- 4 revolutes = 20 constraints

So we still have $24 - 20 = 4$ degrees of freedom (rotation around 4 revolutes). If we fix one of the revolutes at a point on the other hand, we lose 6 degrees of freedom resulting in an overconstrained system. In reality, the system should still be able to rotate around the point which is fixed and this is solved by introducing a planar revolute which only has 2 constraints, assuming that the system is planar and not spatial (which is the case for the squeezer). Since the system is now combined in such a way that we no longer have the possibility to move in three dimensions, constraining the dimension in which we already cannot move makes no sense.

3.2 Setup of the model in Dymola

To set up the model in Dymola we need to calculate each mass centre relative to its respective frame which is easily done using the values provided by [1, Tables 7.1, 7.2] together with figure 1. Slight modification is needed since Hairer and Wanner calculates mass centra relative to the origin. Since all mass centra except C_3, C_5 and C_7 are located inside the mass objects we usually get mass centre vectors with only x-components. Even C_3, C_5 and C_7 are easily calculated since the lengths were all given in the correct coordinate system as seen in the image 1. Note that all crooked parts are modelled in Dymola as straight lines with mass centre outside of the body which behaves the same physically.

The one interesting thing of the squeezer figure, 1, is the part K_3 which is connected at 3 points rather than 2. To model this in Dymola we used a fixed translation which basically moves the connection point of K_4, K_6, K_2 and K_3 to the point D for the spring only. The fixed translation vector is also trivial to calculate since lengths still are given in the correct coordinate system.

The Dymola model can be seen in figure 9. As one can see some of the components are rotated which also implies that their internal coordinate systems are rotated. This is to be considered when setting the internal vector for mass centre - the positive x-axis of a body shape is always in the direction a to b.

In table 6 the mass, length and relative mass centre vectors used in our Dymola model can be viewed. Remember that each part was modelled as a straight body shape in Dymola, thus the length always has a component in the x-axis only.

C	x	y	mass	length	inertia
$C1$	0.00092	0	0.04325	0.007	$2.194 \cdot 10^{-6}$
$C2$	0.0165	0	0.00365	0.028	$4.410 \cdot 10^{-7}$
$C3$	0.01626	0.01043	0.02373	0.035	$5.255 \cdot 10^{-6}$
$C4$	0.01421	0	0.00706	0.02	$5.667 \cdot 10^{-7}$
$C5$	0.02308	0.00916	0.07050	0.04	$1.169 \cdot 10^{-5}$
$C6$	0.01421	0	0.00706	0.02	$5.667 \cdot 10^{-7}$
$C7$	0.01228	-0.00449	0.05498	0.04	$1.912 \cdot 10^{-5}$

Table 6: Information concerning mass centre of the bodies used in the model

The coordinates for the fixed points can be seen in figure 7. The spring constant used was

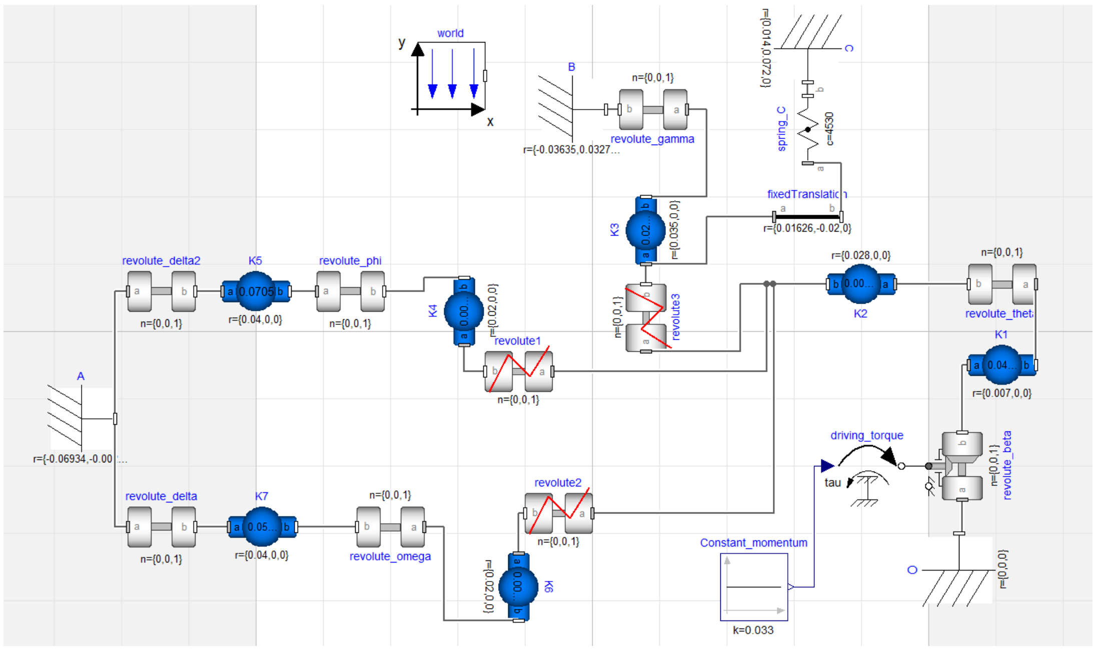
Figure 9: Dymola representation of a squeezer

$c = 4530$ and its unstretched length was $l = 0.07785$. The fixed translation for the body K_3 resulted in $(0.01626, -0.02, 0)$ which is relative to the point a of K_3. Remember the rotation of K_3, see 9, so the translation is located above (positive x-direction) and to the right (negative y-direction) of the body K_3 - which is expected according to figure 1. The constant momentum used to drive the revolute in the origin is set to $k = 0.033$.

	x	y
O	0	0
A	-0.06934	-0.00227
B	-0.03635	0.03273
C	0.014	0.072

Table 7: Planar coordinates for the fixed points

3.3 Results

Running the simulation in Dymola gives us a result like figure 10 (many objects such as revolutes, spring forces and fixed points are hidden). The angles are plotted in figure 11. We can see that the angle plot is similar to the one we got from simulating in Assimulo.

3.3.1 Runtime statistics

Running our model in Dymola, simulating the time interval $[0, 0.3]$ with 500 communication points gives the runtime statistics in table 8:

If compared with the runtime statistics when running the model in Assimulo (table 4) we notice that much fewer computations are needed. This also shows in the CPU time, which is more than 10 times faster for Dymola.

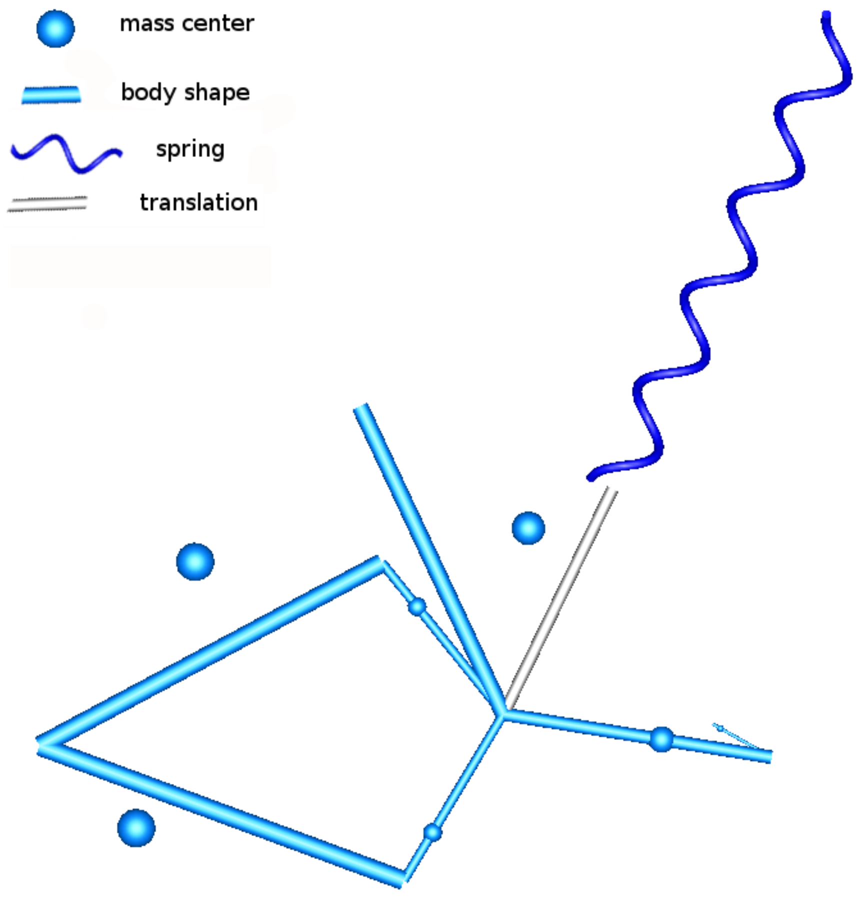

Figure 10: Simulation of the squeezer model in Dymola

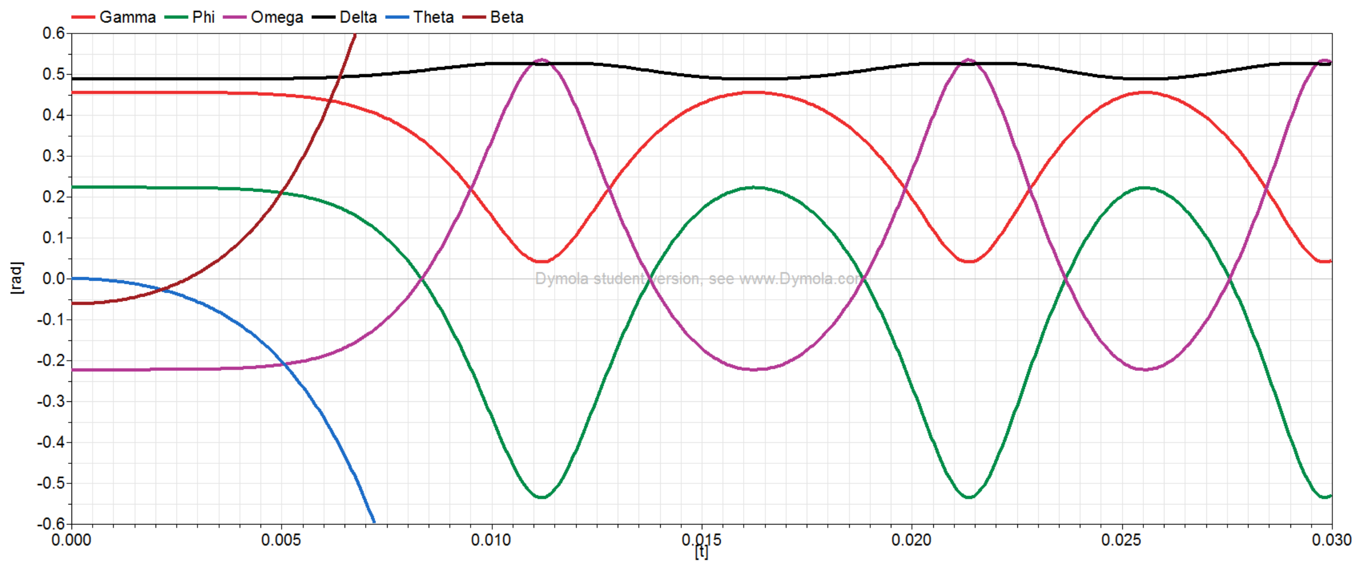

Figure 11: Simulation of the squeezer model in Dymola

Successful steps	169
Function evaluations	421
Jacobian evaluations	30

Table 8: Runtime statistics for simulating the squeezer mechanism in Dymola.

4 Discussion

In this project we have simulated the Andrew's squeezer mechanism in two different softwares. When simulating in Assimulo the main focus is directed towards the differential equations and how to adapt the problem to the solver in order to obtain convergent and accurate solutions. One example of this is that we needed to suppress different combinations of variables in order to get convergence of the Newton iteration, thus requiring a deeper understanding for the underlying equations and how they are solved. For Dymola on the other hand, we obtained accurate results by focusing on the physics and mechanics of the system, focusing a lot on the definition of different coordinate systems and mass centra. The DAEs were formulated and evaluated by Dymola, and worked well if the drag and drop model was correctly implemented. Here there was a need for a deeper understanding of how the components of Modelica standard library worked as well as some understanding regarding degrees of freedom and constraints.

References

[1] E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II*, Springer, 2nd revised edition, 1996.

[2] https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.html, ®Modelica Association.