Project Reports

Salvador Castagnino, Theo Koppenhöfer

February 18, 2023

Project 1

Introduction

The Benchmark

In the following we use the model of a pendulum attached to a rod which is elastic in the radial direction as described in Task 1. The situation is depicted in figure 1.

This problem leads to the formulation as an ODE

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}' = \begin{bmatrix} y_3 \\ y_4 \\ -y_1\lambda(y_1, y_2) \\ -y_2\lambda(y_1, y_2) - 1 \end{bmatrix}$$

with

$$\lambda(y_1, y_2) = k\frac{\|(y_1, y_2)\| - 1}{\|(y_1, y_2)\|}\,.$$

The plot of a numerical solution to this problem for $k = 1$ can be seen in figures 2 3 and 4.

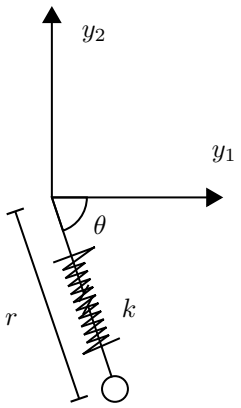


Figure 1: The pendulum

We can calculate the potential, kinetic and approximate the elastic energy with the formulas

$$E_{\text{pot}} = 1 + y_2 \qquad E_{\text{kin}} = \frac{\|(y_3, y_4)\|^2}{2} \qquad E_{\text{elast}} = k\frac{(\|(y_1, y_2)\| - 1)^2}{2}\,.$$

Adding these up we get the approximate total energy

$$E_{\text{tot}} = E_{\text{pot}} + E_{\text{kin}} + E_{\text{elast}}\,.$$

We expect the approximate total energy to be constant which indeed can be seen in Figure 5 for that previously calculated numerical solution. Because of this property we can use the relative variation of the approximate total energy as an index to measure the stability of the method. We specifically implement

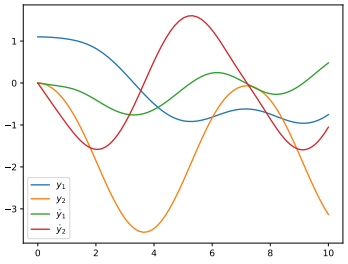


Figure 2: State in dependence of time.

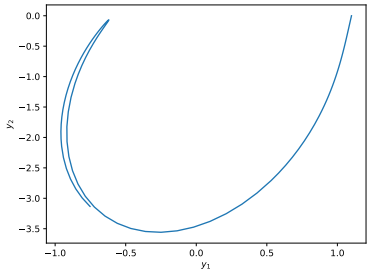


Figure 3: Path traced out by pendulum.

```
import numpy as np
instability_index = (np.max(total_energy)-np.min(total_energy)) \
                    /np.mean(total_energy)
```

In the ideal world this index almost vanishes.

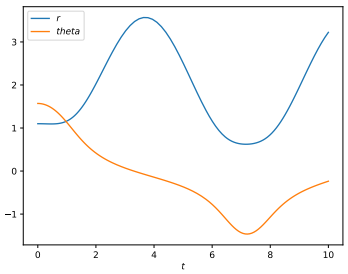


Figure 4: Polar coordinates.

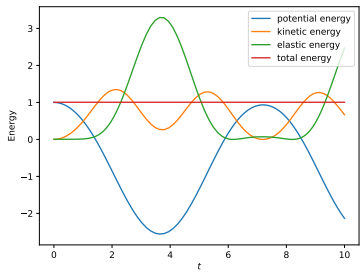


Figure 5: Energy plot

Testing Explicit Methods

For linear problems, explicit methods present a much reduced stability region which dictates the possible step sizes for that specific method. For the problem of the elastic pendulum, approximated by explicit methods, when the value of k is increased we are expected to see the approximation blow up showing oscillations of unbounded amplitude. This unstable behavior will be attenuated by reducing the value of the step h.

The problem was simulated using Explicit Euler and RK4. All the experiments in this section, if not otherwise stated, take as initial value $y_0 = [1.1, 0, 0, 0]$ and have $[0, 20]$ for domain. The graphs are presented in polar coordinates where r

refers to the length of the spring and θ refers to the angle conformed between the pendulum and the vertical axis.

It can be observed that for a step size of $h = 0.01$ Explicit Eurler (Figure 6) already shows instability for values of $k = 50$ while RK4 (Figure 7) with that same step size remains stable for values up to $k = 3000$.

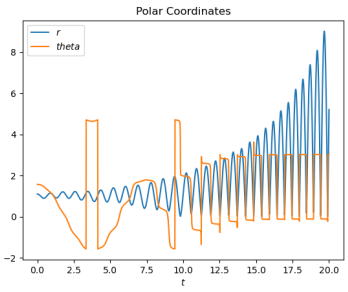


Figure 6: Explicit Euler $h = 0.01$ $k = 50$

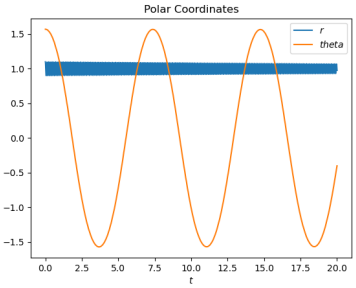


Figure 7: RK4 $h = 0.01$ $k = 3000$

For Explicit Euler (Figure 8), by keeping the value of k constant and reducing the step size by a decimal place we can see how the instability is attenuated presenting a similar amplitude over time. It takes a much larger step size and spring constant for RK4 to become unstable (Figure 9), once unstabilized it's growth is much more rapid than Explicit Euler's and it does so without oscillating.

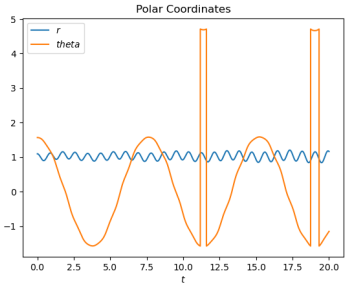


Figure 8: Explicit Euler $h = 0.001$ $k = 50$

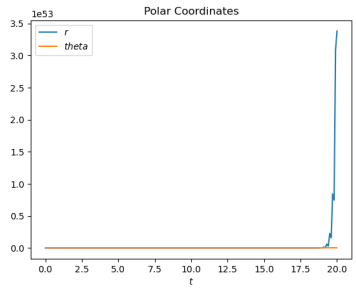


Figure 9: RK4 $h = 0.1$ $k = 975$

It is interesting to observe that the oscillation of the spring is rapidly dumped when using RK4 (Figure 10), a behavior similar to that presented by implicit methods. This behavior cannot be observed in the other explicit methods.

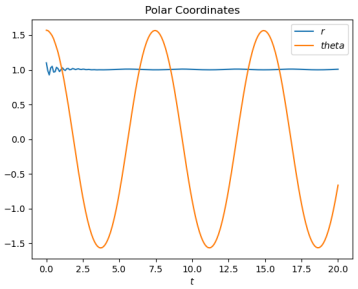


Figure 10: RK4 $h = 0.1$ $k = 300$

Testing Implicit Methods

Opposite to the case of explicit methods, for linear problems implicit methods count with an extensive stability region which doesn't make their stability dependent on the value of the step k. The problem was simulated using Implicit Euler, BDF2 with Fixed Point as corrector and BDFk with Newton as corrector for k between 1 and 4. All the following experiments take as initial value $y_0 = [1.1, 0, 0, 0]$ and have $[0, 20]$ for time domain. It is interesting to see how the oscillation of the spring decays for implicit methods. This decay can be attenuated by reducing the step size or accelerated by increasing the value of the spring constant.

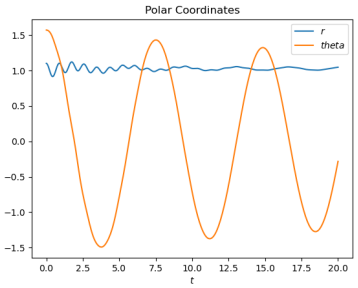


Figure 11: Implicit Euler $h = 0.01$ $k = 50$

The method BDFk with Newton presents a decay in the spring oscillation as the other methods do. However, it also shows decay of the pendulum oscillation which cannot be observed in the other implicit methods. To better observe this decay (Figure 14 and Figure 15) the domain is increased to $[0, 100]$.

It is interesting to see how the relation between the speed of decay of the spring oscillation in inversely proportional to the size of the stability region of the methods tested.

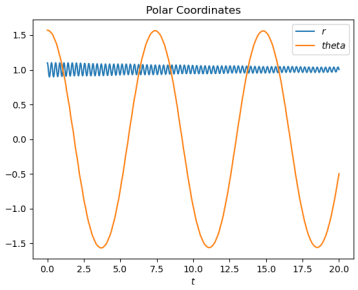


Figure 12: BDF2-Fixed Point $h = 0.001$ $k = 500$

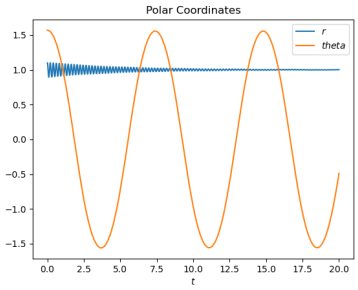


Figure 13: BDF2-Fixed Point $h = 0.01$ $k = 1000$

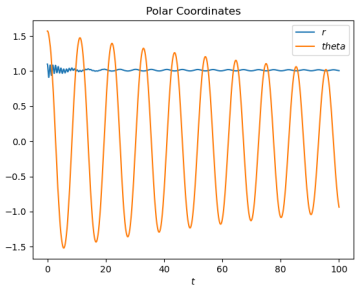


Figure 14: BDF2-Newton $h = 0.01$ $k = 100$

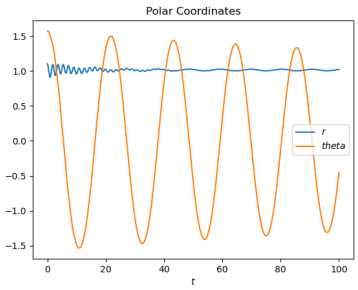


Figure 15: BDF4-Newton $h = 0.01$ $k = 100$

Testing CVODE

A first test series

In the first specific test of CVODE we solve our toy problem for increasing k. Here we switch between the BDF and Adam-Moultons discretisation method. We also vary the `maxorder` parameter for both methods. A higher k reflects a problem which is more stiff. As a stiff problem requires smaller steps the number of steps `nsteps` increases as k increases which can be seen in figure 16. As the number of function evaluations per stepsize `nfcns/nsteps` hovers slightly above 1 for all methods (c.f. figure 18) the number of function evaluations increase analogously to `nsteps` with k as can be seen in figure 17. There is however a difference in how many steps each method needs. The BDF-method requires in general more steps than the Adams-Moulton method. And the general trend is that the number of steps increases as `maxord` is reduced.

From figure 19 it can be seen that the number of jacobian evaluations stays roughly constant and happens roughly every 5th step. The number `nerrfails/steps` stays roughly constant in dependence of k though the general tendency

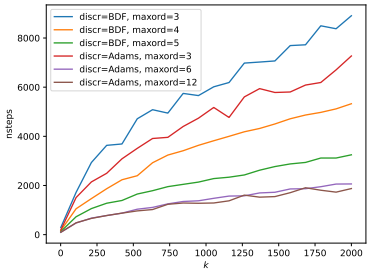


Figure 16: `nsteps` in relation to the parameter k.

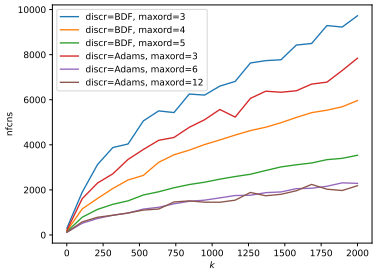


Figure 17: `nfuncs` in relation to the parameter k.

is that it is smaller the lower `maxord` is set. This makes sense because a lower `maxord` means there are fewer possibilities for the method order and hence fewer changes of order. In figure 21 we see a difference in how much the methods obey the principles of energy conservation. One can see that for growing k the result tends to be further away from physical reality. Once again the methods with higher `maxord` do better with the exception of the BDF method where for some reason a `maxord` of 4 performs best.

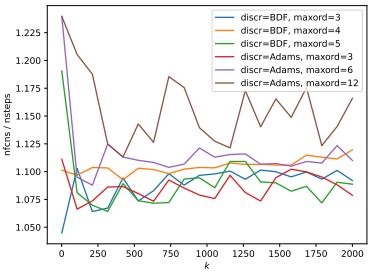


Figure 18: `nfcns/nsteps` in relation to the parameter k.

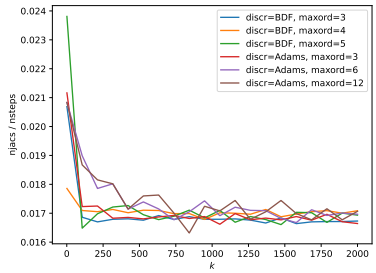


Figure 19: `njacs/nsteps` in relation to the parameter k.

This test confirms once again that a stiffer Problem needs more function evaluations in CVODE. Perhaps surprisingly the Adams-Moulton-method seems to perform better on this problem. This experiment also highlights that a lower `maxord` parameter tends to be more computationally expensive though it reduces the number of error test failures `nerrfails`.

Testing the parameter rtol

We now test the influence of the parameter `rtol` on the methods BDF and Adams-Moulton. For this we set $k = 10^3$ and keep all other parameters on their

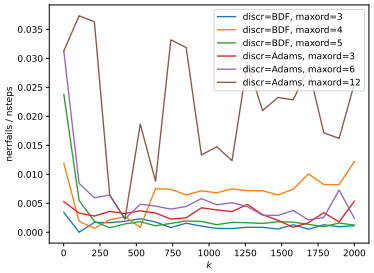


Figure 20: `nerrfails/nsteps` in relation to the parameter k.

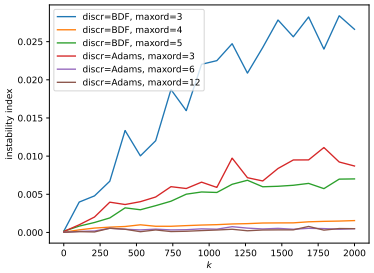


Figure 21: instability index in relation to the parameter k.

default values. The results can be seen in figures 22 to 26. We note that as `rtol` increases the number of steps decreases (c.f. figure 22). If one compares figures the instability index for $k \approx 10^3$ in figure 21 with the instability index in figure 26 one sees that changing the `rtol` parameter from the default makes the result significantly worse in terms of energy conservation.

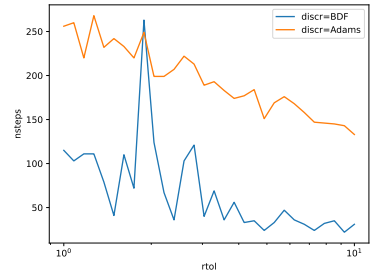


Figure 22: `nsteps` in relation to the parameter `rtol`.

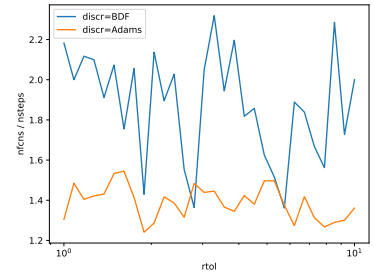


Figure 23: `nfcns/nsteps` in relation to the parameter `rtol`.

Testing the parameter atol

If we test the `atol` parameter on the Adams and Newton method analogously to the test of the `rtol` parameter we once again get an instability index that is significantly above the value for the method in which we did not specify this value as can be seen in Figure 27. In either case we observe that fixing the tolerance seems to come at the cost of energy conversation as is dramatically visualised in Figures 28 and 29.

All in all we see that none of the (admittedly crude) tweaking of the parameters improved the performance of CVODE. To the contrary, most changes worsened the performance. The choice of the discretisation method on the other

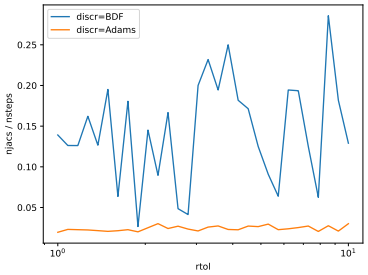


Figure 24: `njacs/nsteps` in relation to the parameter `rtol`.

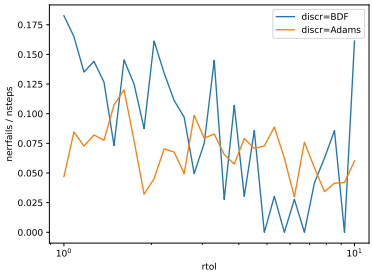


Figure 25: `nerrfails/nsteps` in relation to the parameter `rtol`.

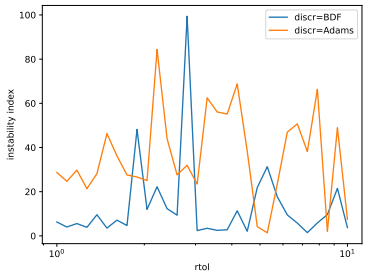


Figure 26: instability index in relation to the parameter `rtol`.

hand did make a big difference and the performance for solving the toy problem could be improved by switching from the default BDF method.

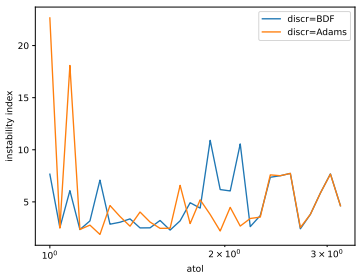


Figure 27: instability index in relation to the parameter `atol`.

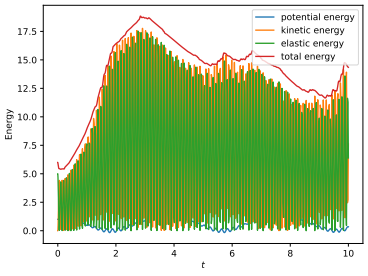


Figure 28: Energy plot for $k = 10^3$ with `atol=1E-2`.

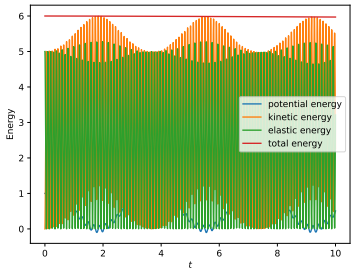


Figure 29: Energy plot for $k = 10^3$.

Project 2

In this project we used the implementation of the seven body mechanism as described in [**HW˙SolvingODEs**] to test Assimulo's implicit solvers. The problem formulation leads to an index 3 problem of the form

$$M(q)\,q'' = f(q,q') - G(q)^\top \lambda \tag{1}$$
$$0 = g(q) \tag{2}$$

where $q \in \mathbb{R}^7$, $\lambda \in \mathbb{R}^6$ and $G = Dg$. If we differentiate condition (2) we obtain the index 2 condtition

$$0 = G(q)\,q'$$

and differentiating this again we obtain the index 1 formulation

$$0 = \partial_q^2 g(q)\,(q',q') + G(q)\,q''\,.$$

Note that condition (**??**) and (1) can be uniquely solved for q'' and λ and we then obtain an ODE in the explicit formulation. For the implementation we rewrite this second order system as a first order system by the usual trick of introducing the variable $v = q'$ so that in the implicit formulation the problem depends on the variable

$$y = \begin{bmatrix} q \\ v \\ \lambda \end{bmatrix}$$

A plot of the solution of the index 1 formulation can be seen in Figures 30 to 32.

Generation of consistent initial values

Given the restrictions imposed over the system the generation of initial values is not a trivial task. To do this we follow the steps presented in the literature, we start with q and v.

First we take $\theta = 0$ which can be done given that the system is undetermined and we assume a solution exists (this makes sense as a physical model of the

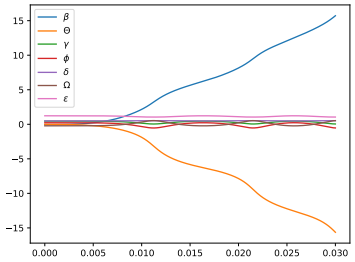


Figure 30: The angles of the solution to the index 2 problem.

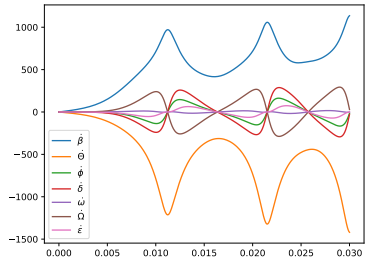


Figure 31: The angle speed of the solution to the index 2 problem.

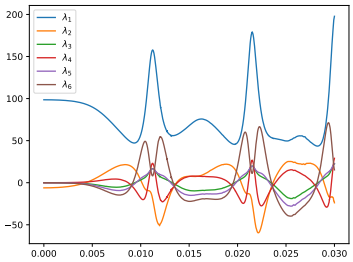


Figure 32: The Lagrange parameter of the solution to the index 2 problem.

system was presented in class). We then use *Newton Iteration* to solve the equations obtaining values for the remaining angles

We also take the initial value of v to be 0 as we assume the system starts at rest. Now for w and λ using the Index 1 formulation we have to solve a linear system which is presented in the literature. Doing this we get the values

Many of the values are minute bu non-zero. This is given because of a rounding error, but in theory these small values equal zero.

A comparison of the index 1, 2 and 3 formulations

We now would like to compare the solutions of the various formulations. To calculate the solutions we used the IDA solver (TODO: hyperlink). However, to get the problem to run we set the `atol` parameter to the large number `1E5` and the `algvar` parameter to `False` for the algebraic variable λ and for v. These settings remain unchainged and in the following we only vary the index of the problem. We can see in the figures 36 to 38 the difference of the index 1 solution

β	-0.0617139
Θ	0
γ	0.45528
ϕ	0.222668
δ	0.487365
Ω	-0.222668
ϵ	1.23055

Figure 33: Consistent Initial Angles

$\ddot{\beta}$	14222.4
$\ddot{\Theta}$	-10666.8
$\ddot{\phi}$	7.58763e-14
$\ddot{\delta}$	3.22329e-13
$\ddot{\omega}$	7.02222e-13
$\ddot{\Omega}$	-3.22329e-13
$\ddot{\epsilon}$	8.6019e-13

Figure 34: Consistent Initial Accelerations

subtraced from the index 3 solution. In the figures 39 to 41 we see the index 2 solution subtracted from the index 3 solution. As expected we see that the difference the solutions grows as time progresses.

Rather unexpectedly however the difference of the index 1 to the index 3 solution is in general greater than the difference of the index 2 to the index 3 solutions. Also unexpectedly these differences are noticable in the plots of the Lagrange parameter λ as shown in figures 42, 43 and 32. Here we see that the solution becomes increasingly rough as the index increases.

The performance of the IDA solver for the various indexes can be seen in figures 44 to 47. We see figure 44 that the number of steps of the solver increases with the index. As the number of function evaluations per step stays roughly constant by figure 45 this means that the number of function evaluations increases with the index. One other other notable statistic regards the number of error test failures for the different problems which can be seen in figure 47 where we see a larger difference between the problems though this is probably not statistically significant as the total number of error test failures is approximately a dozen.

Dependence on the parameters `algvar` and `atol`

As previously indicated the IDA solver will throw an error

```
assimulo.solvers.sundials.IDAError: 'Convergence test failures
                                    occurred too many times during
                                    one internal time step or minimum
                                     step size was reached. At time 0
                                    .000000.'
```

λ_1	98.5669
λ_2	-6.12269
λ_3	-7.37165e-16
λ_4	9.40165e-16
λ_5	4.35256e-16
λ_6	7.59352e-16

Figure 35: Consistent Initial Lambdas

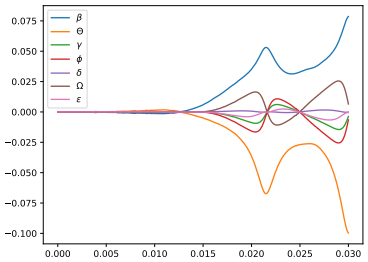


Figure 36: The difference of angles of the index 1 and the index 3 solution.

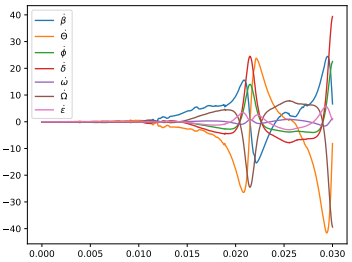


Figure 37: The difference in angle speeds of the index 1 and the index 3 solution.

This can be resolved by declaring the entries of y corresponding to v and or λ to be algebraic variables with the parameter `algvar` and to set the parameter `atol` to a large variable. In the following we would like to check how these parameters impact the performance of the solver in the case of the index 1 formulation. For this we denote by `algvar_v` and `algvar_lambda` the value of the `algvar` parameter for v and λ. The default value of `algvar` is set to `True`. Analogously we denote the components of `atol` corresponding to v and λ with `atol_v` and `atol_lambda` and set the default value to `1E-6`. We now run a series of 5 experiments as depicted in table 48.

As all experiments deliver more or less the same result we will be comparing the statistics of IDA as depicted in figures 49 to 54. Once again we observe in figure 52 that the total number of function evaluations is roughly proportional to the number of steps needed. Here experiments 1 and 4 stick out for requiring comparatively more function evaluations per step. However in figure 49 we see that these are also precisely the experiments in which the total number of steps taken is by far the least. Experiments 1 and 4 are also precisely those experiments that have the most stringent requirements on the v part of y. Both have set `atol_v=1E-6` and declare v to not be an algebraic variable. We see in figures 51 and 54 that experiment 0 is an outlier in requiring comparitively many jacobian evaluations and having relatively few error test failures.

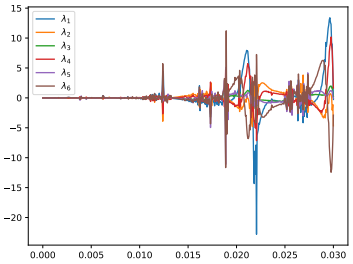


Figure 38: The difference of lambdas of the index 1 and the index 3 solution.

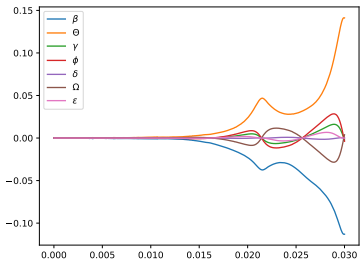


Figure 39: The difference of angles of the index 2 and the index 3 solution.

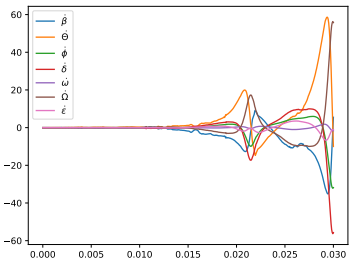


Figure 40: The difference in angle speeds of the index 2 and the index 3 solution.

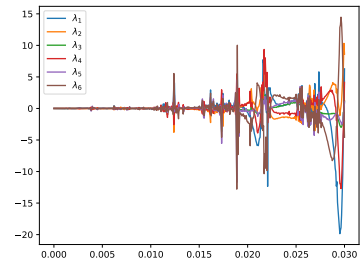


Figure 41: The difference of lambdas of the index 2 and the index 3 solution.

Using an explicit method

As part of the final task we used an explicit RK4 method to solve the index 1 problem. As a result of the method exploding for $h = 0.01$, the default step value, the method was tested for various values of h.

In particular in Figure ... the L_2 norm of each angle over time is plotted with respect to $h \in [0.001, 0.002)$ with $\Delta h = 5\text{e}{-6}$.

The explicit method can then be tested with individual step sizes and as expected, the method explodes for $h \in \{0.001446, 0.0018, 0.00195\}$ and is stable for $h \in \{0.00185, 0.0012, 0.0016\}$.

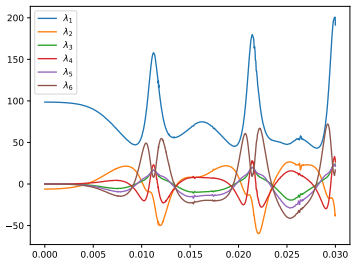


Figure 42: The Lagrange parameter of the index 2 problem.

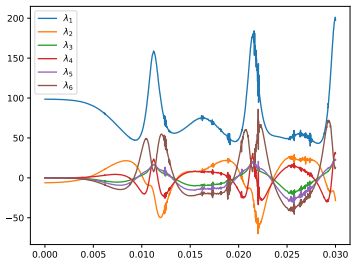


Figure 43: The Lagrange parameter of the index 3 problem.

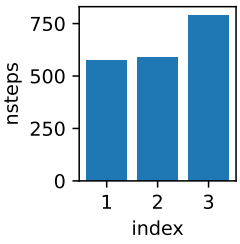


Figure 44: `nsteps` in relation to the index.

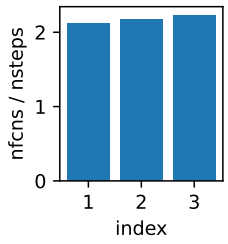


Figure 45: `nfcns/nsteps` in relation to the index.

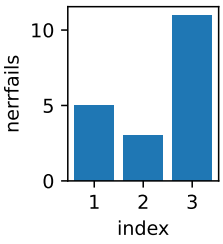


Figure 46: `nerrfails` in relation to the index.

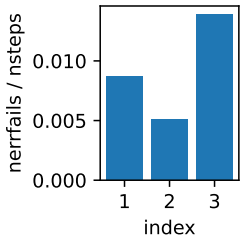


Figure 47: `nerrfails/nsteps` in relation to the index.

experiment	index	atol_v	atol_lambda	algvar_v	algvar_lambda	suppress_alg
0	1	100000	100000	False	False	True
1	1	1e-06	100000	False	True	True
2	1	1e-06	100000	True	False	True
3	1	1e-06	100000	True	True	False
4	1	1e-06	1e-06	False	False	True

Figure 48: Parameters in the experiments

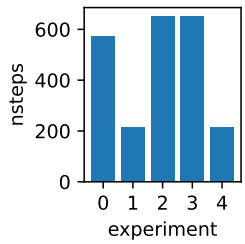


Figure 49: `nsteps` of the experiments.

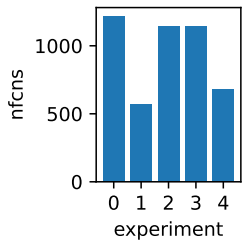


Figure 50: `nfcns` of the experiments.

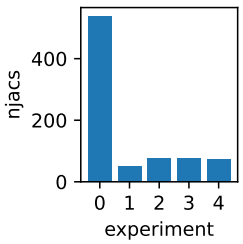


Figure 51: `njacs` of the experiments.

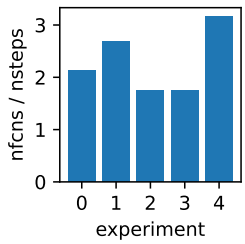


Figure 52: `nfcns/nsteps` of the experiments.

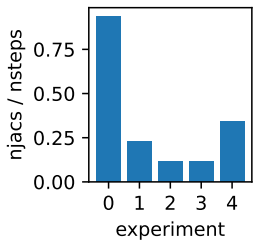


Figure 53: `njacs/nsteps` of the experiments.

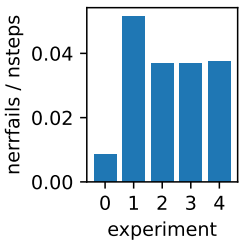
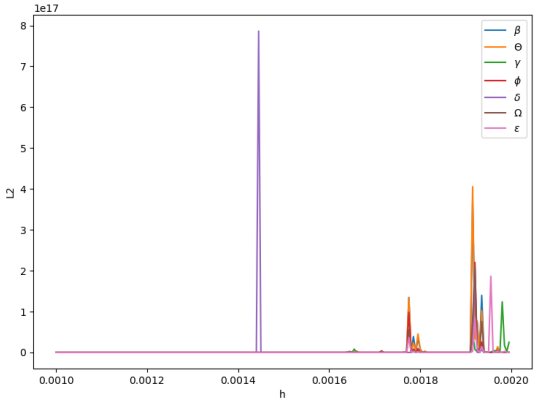


Figure 54: `nerrfails/nsteps` of the experiments.


1e17
8
7
6
5
4
3
2
1
0
L2
0.0010
0.0012
0.0014
0.0016
0.0018
0.0020
h
β
Θ
γ
ϕ
δ
Ω
ε

Appendix

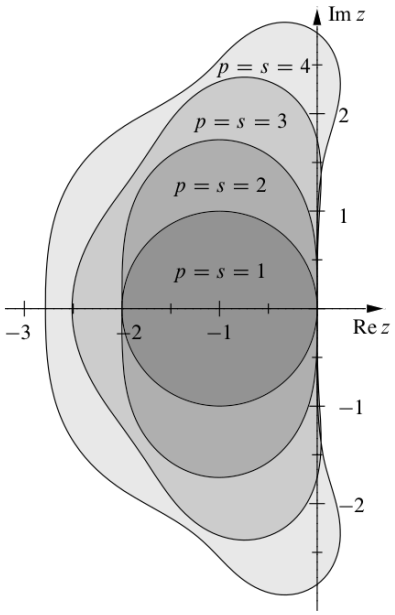


Figure 55: Stability regions of the Runge-Kutta-methods, taken from [**Stab˙RK**][p.238]

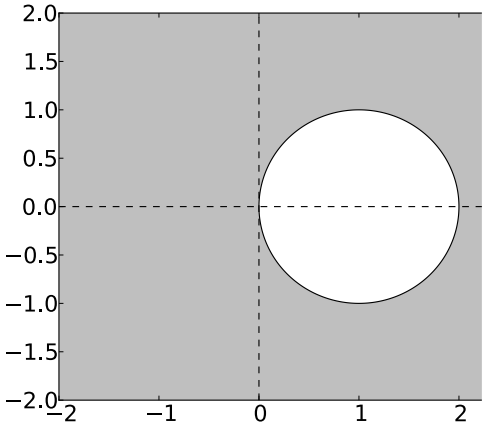


Figure 56: Stability region for BDF1, taken from [**Stab˙BDF**]

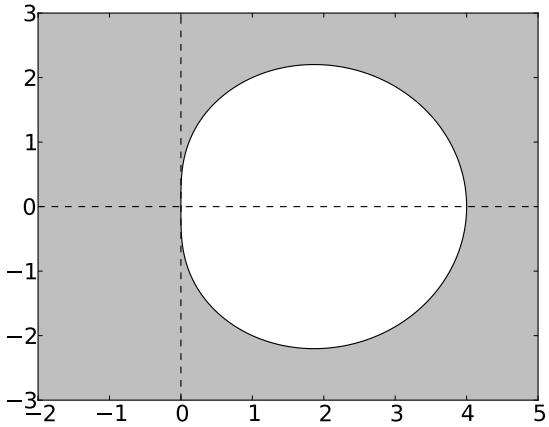


Figure 57: Stability region for BDF2, taken from [**Stab˙BDF**]

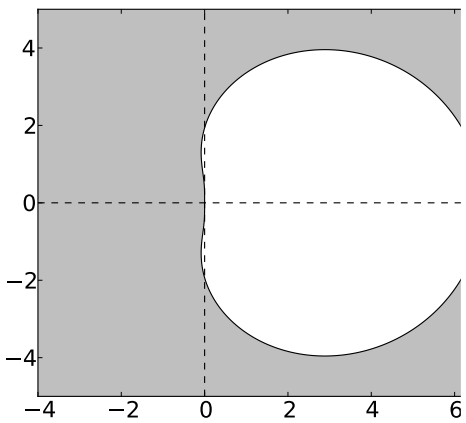


Figure 58: Stability region for BDF3, taken from [**Stab˙BDF**]

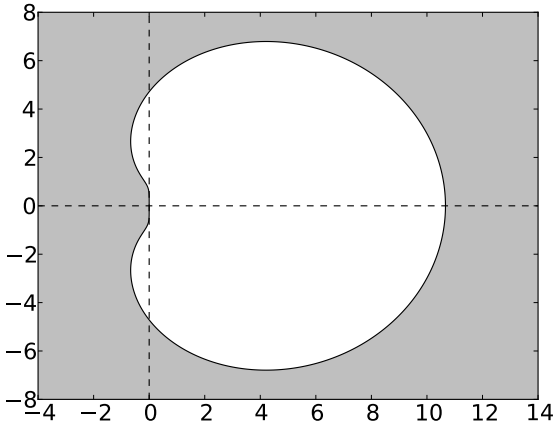


Figure 59: Stability region for BDF4, taken from [**Stab˙BDF**]

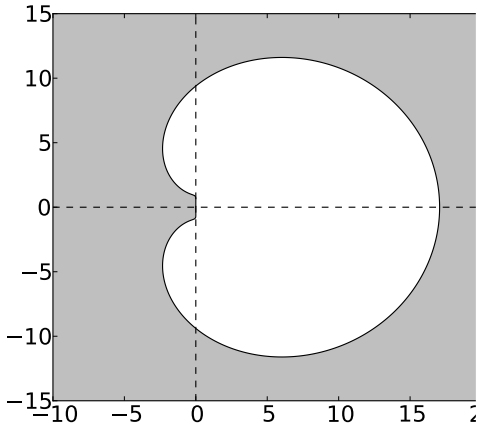


Figure 60: Stability region for BDF5, taken from [**Stab˙BDF**]

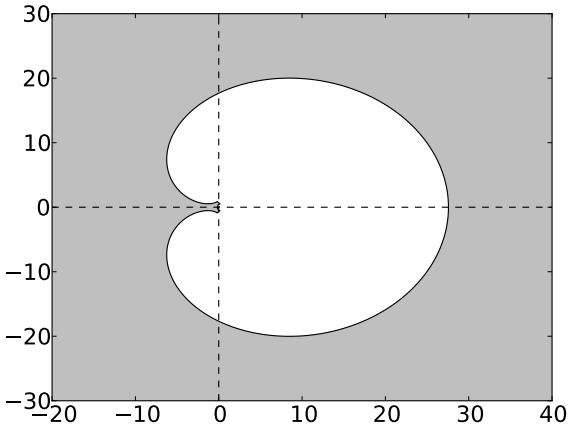


Figure 61: Stability region for BDF6, taken from [**Stab˙BDF**]