Project Reports

NUMN26 / FMNN05, Simulation Tools

Salvador Castagnino, Theo Koppenhöfer

Lund

March 6, 2023

Project 1

Introduction

In the following report we will discuss our implementation of the projects to the course NUMN26 / FMNN05, Simulation Tools. During the project we try out solvers from the `Assimulo` package which wraps the SUNDIALS ODE solvers and in project 3 briefly the `dune` package for discretising PDEs. In project 1 we will use a variant of the pendulum as a toy problem to test various explicit solvers of Assimulo. In project 2 we will then use a mechanical model of a seven body mechanism in various formulations as a benchmark to test Assimulo's implicit solvers. In project 3 we then test an implementation of the explicit Newmark method on the pendulum and an implementation of the HHT-α and the implicit Newmark solvers on a discretised PDE given by an elastic beam. This report and the code belonging to it can be found online under [3].

The Benchmark

In the following we use the model of a pendulum attached to a rod which is elastic in the radial direction. 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.

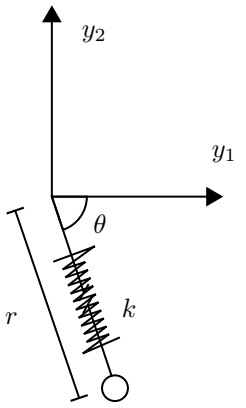

Figure 1: The pendulum

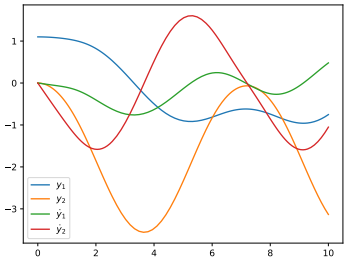

Figure 2: State in dependence of time.

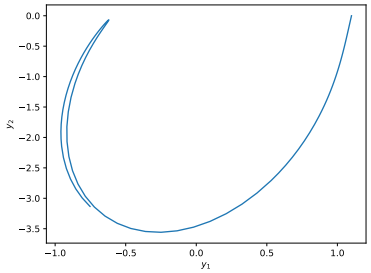

Figure 3: Path traced out by pendulum.

We can calculate the potential, kinetic and approximate elastic energies 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 almost constant which indeed can be seen in figure 5 for that previously calculated numerical solution. Because of this property we can use the variance of the approximate total energy as an index to measure the stability of a solver. In the ideal world this index almost vanishes.

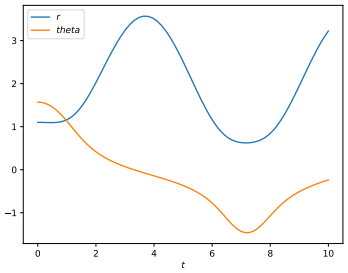

Figure 4: Polar coordinates.

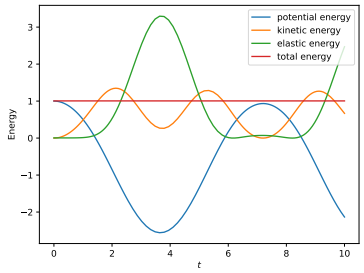

Figure 5: Energy plot

Testing Explicit Methods

With increasing k the elastic pendulum problem behaves increasingly stiff. As explicit methods have a relatively small stability region when compared with

the left-half plane this means the step size h has to be reduced with increasing k for the numerical solution to remain bounded.

The problem was simulated using Explicit Euler and RK4. All the experiments in this section are simulated on the domain $[0, 20]$ and have initial the initial value

$$y_0 = \begin{bmatrix} 1.1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The graphs are presented in polar coordinates where r refers to the length of the spring and θ refers to the angle between the pendulum and the vertical axis.

It can be observed that for the fixed step size $h = 0.01$ Explicit Euler (figure 6) already shows instability for values of $k = 50$ while RK4 (figure 7) remains stable for values up to $k = 3000$.

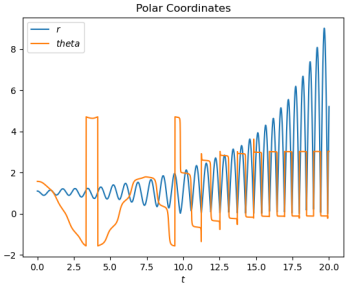

Figure 6: Explicit Euler with parameters $h = 0.01$ and $k = 50$.

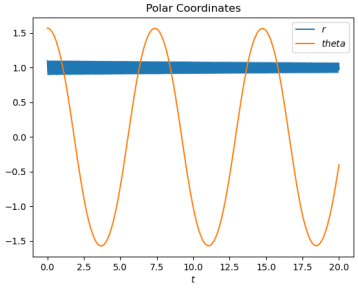

Figure 7: RK4 with parameters $h = 0.01$ and $k = 3000$.

By keeping the value of k constant and reducing the step size by an order of magnitude with Explicit Euler (c.f. figure 8) we see how the instability is attenuated and it thus presents a similar amplitude over time. It takes a much larger step size and spring constant for RK4 to become unstable as can be seen in figure 9. Furthermore its growth is much more rapid than Explicit Euler's when unstable and occurs without oscillating.

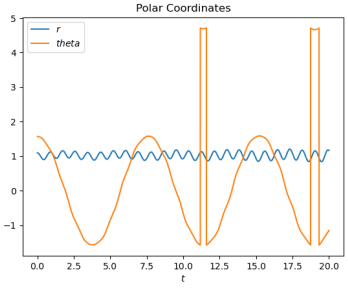

Figure 8: Explicit Euler with parameters $h = 0.001$ and $k = 50$.

It is interesting to observe that the oscillation of the spring is rapidly damped with RK4 (figure 10), a behavior similar to that presented by implicit methods. This behavior was not observed in the other explicit methods.

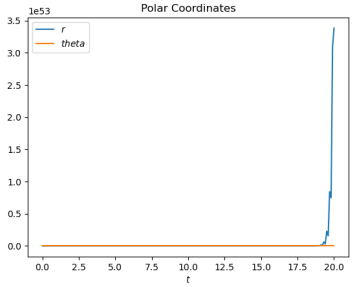

Figure 9: RK4 with parameters $h = 0.1$ and $k = 975$.

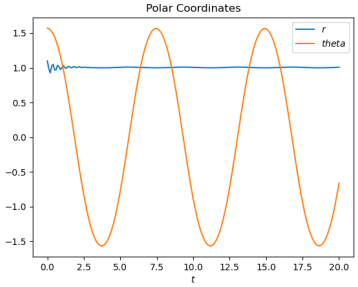

Figure 10: RK4 with parameters $h = 0.1$ and $k = 300$.

Testing Implicit Methods

In contrast to explicit methods for linear problems implicit methods have an extensive stability region. Hence their stability does not depend 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. Unless otherwise stated the following experiments have $[0, 20]$ as domain and initial value

$$y_0 = \begin{bmatrix} 1.1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

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.

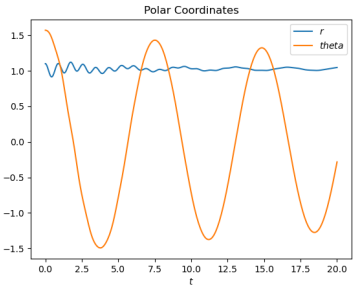

Figure 11: Implicit Euler with parameters $h = 0.01$ and $k = 50$.

The BDFk method with Newton presents a decay in the spring oscillation as the other methods do. However, it also shows decay of the pendulum oscillation

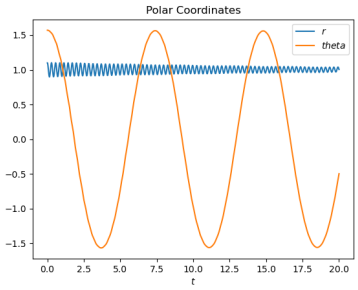

Figure 12: BDF2 with parameters $h = 0.001$ and $k = 500$.

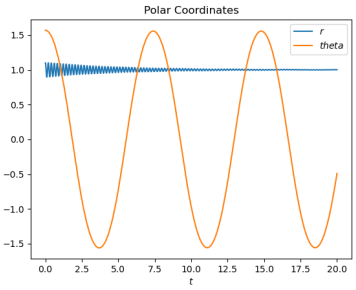

Figure 13: BDF2 with parameters $h = 0.01$ and $k = 1000$.

which cannot be observed in the other implicit methods. To better observe this decay (figure 14 and figure 15) the domain is increased to $[0, 100]$.

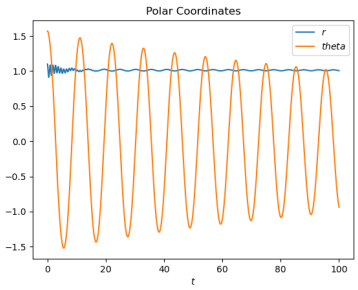

Figure 14: BDF2-Newton $h = 0.01$ $k = 100$

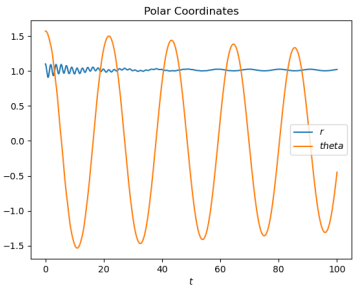

Figure 15: BDF4-Newton $h = 0.01$ $k = 100$

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

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-Moulton's 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 step size, `nfcns/nsteps`, hovers slightly above 1 for all methods (c.f. figure 18) the number of function evaluations increase proportionally 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.

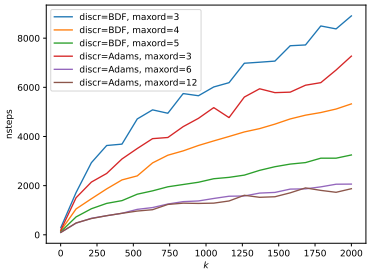

Figure 16: `nsteps` in relation to the parameter k.

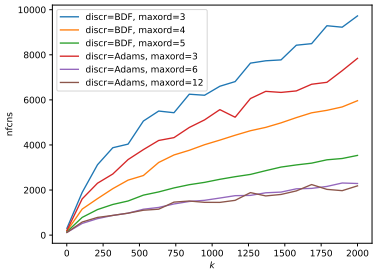

Figure 17: `nfuncs` in relation to the parameter k.

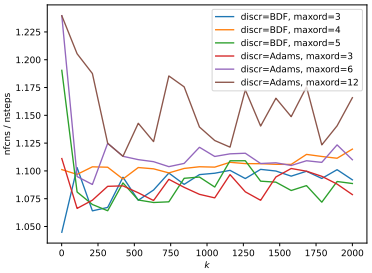

Figure 18: `nfcns/nsteps` in relation to the parameter k.

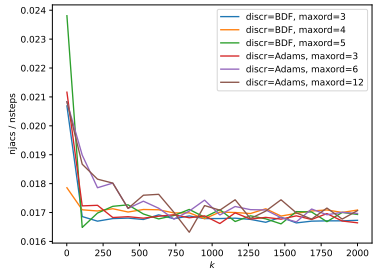

Figure 19: `njacs/nsteps` in relation to the parameter k.

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

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.

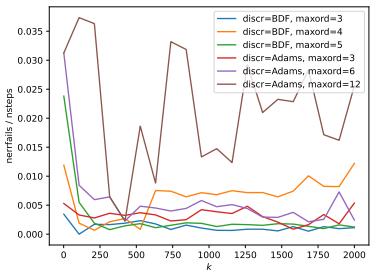

Figure 20: `nerrfails/nsteps` in relation to the parameter k.

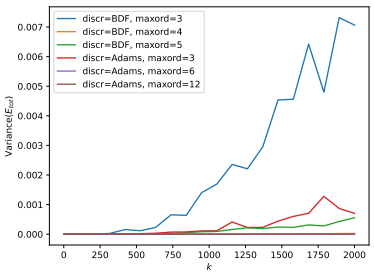

Figure 21: Variance of the energy in relation to the parameter k.

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 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). We see in 26 that for a particular choice of `rtol` the simulation with Adams-Moulton has a large energy sink or energy source which indicates that this method becomes unstable for this particular value. It is unsurprising that this occurs for a relatively large `rtol`.

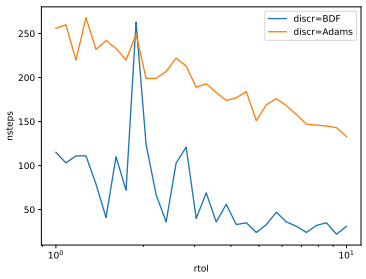

Figure 22: `nsteps` in relation to the parameter `rtol`.

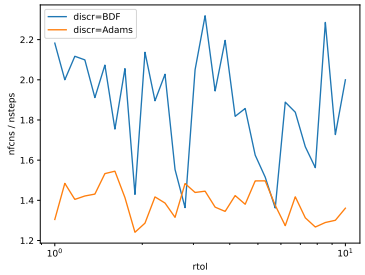

Figure 23: `nfcns/nsteps` in relation to the parameter `rtol`.

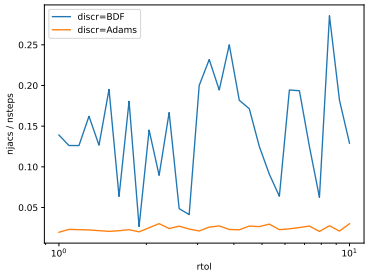

Figure 24: `njacs/nsteps` in relation to the parameter `rtol`.

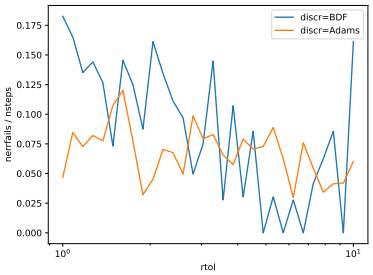

Figure 25: `nerrfails/nsteps` in relation to the parameter `rtol`.

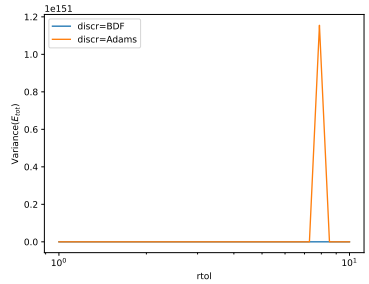

Figure 26: Variance of the energy in relation to the parameter `rtol`.

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 can see that for the Adams-Moulton method the variance of the energy (c.f. figure 27) is for certain `rtol` significantly above the value for the default parameter which indicates some instability. In either case we observe that increasing the tolerance seems to come at the cost of energy conversation as is dramatically visualised in figures 28 and 29.

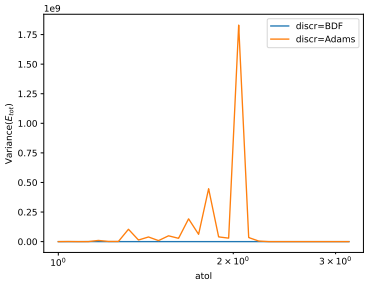

Figure 27: Variance of the energy in relation to the parameter `atol`.

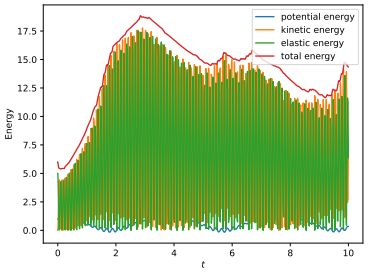

Figure 28: Energy plot for $k = 10^3$ with `atol=1E-2`.

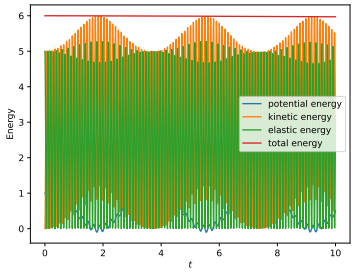

Figure 29: Energy plot for $k = 10^3$.

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

Project 2

In this project we used the implementation of the seven body mechanism as described in [2] 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 condition

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

and differentiating this again we obtain the index 1 formulation

$$0 = \partial_q^2 g(q)\,(q',q') + G(q)\,q''\,. \tag{3}$$

Note that condition (3) 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

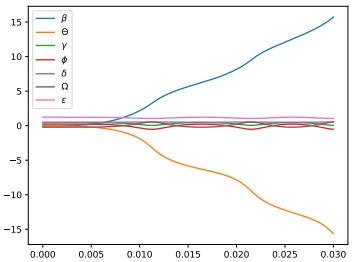

Figure 30: The angles of the solution to the index 2 problem.

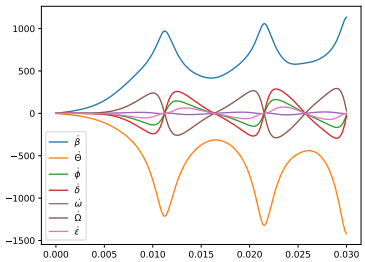

Figure 31: The angle speed of the solution to the index 2 problem.

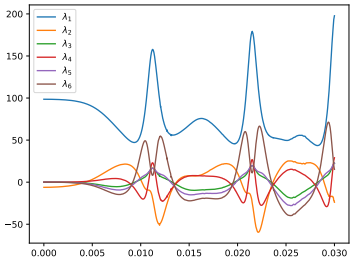

Figure 32: The Lagrange parameter of the solution to the index 2 problem.

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

Due to the restrictions imposed on the system the generation of initial values is no trivial task. To do this we follow the steps presented in [2]. We start with q and v.

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
$\ddot{\delta}$	1.53229e-13
$\ddot{\omega}$	-1.71547e-14
$\ddot{\Omega}$	-1.53229e-13
$\ddot{\epsilon}$	5.79407e-14

Figure 34: Consistent initial accelerations

λ_1	98.5669
λ_2	-6.12269
λ_3	2.2899e-17
λ_4	-1.87294e-17
λ_5	3.37745e-17
λ_6	4.83113e-17

Figure 35: Consistent initial lambdas

First we take $\theta = 0$ which can be done given that the system is underdetermined under the assumption that a solution exists (this makes sense as a physical model of the 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 [2]. Doing this we get the values given in tables 33, 34 and 35. Many of the values are minute but non-zero. This is due to rounding errors, but in theory these small values equal zero.

A comparison of the index 1, 2 and 3 formulations

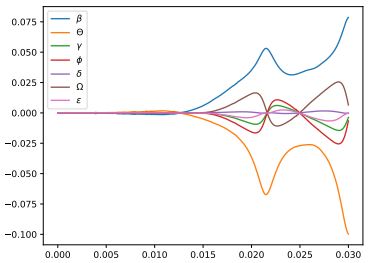

Figure 36: The difference of angles of the index 1 and the index 3 solution.

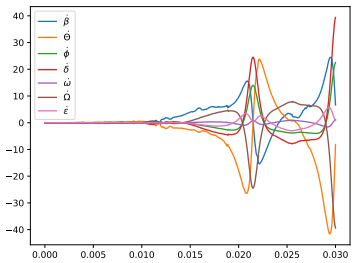

Figure 37: The difference in angle speeds of the index 1 and the index 3 solution.

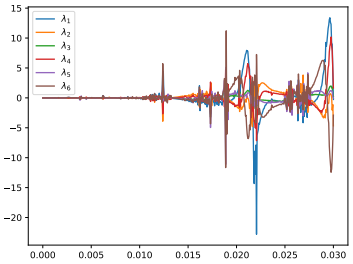

Figure 38: The difference of lambdas of the index 1 and the index 3 solution.

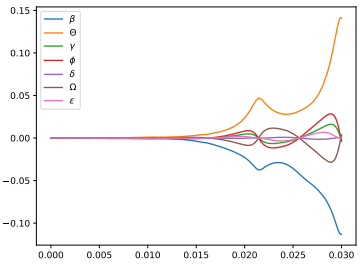

Figure 39: The difference of angles of the index 2 and the index 3 solution.

We now would like to compare the solutions of the various formulations. To calculate the solutions we used the IDA solver. 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 unchanged and in the following we only vary the index of the problem. One can see in the figures 36 to 38 the index 1 solution subtracted 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.

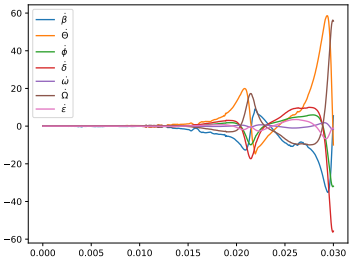

Figure 40: The difference in angle speeds of the index 2 and the index 3 solution.

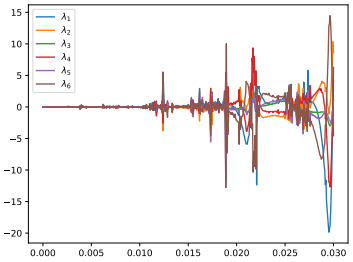

Figure 41: The difference of lambdas of the index 2 and the index 3 solution.

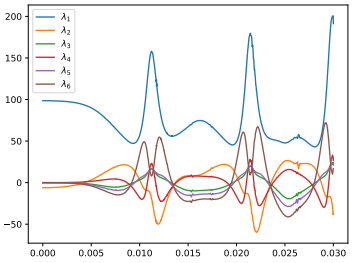

Figure 42: The Lagrange parameter of the index 2 problem.

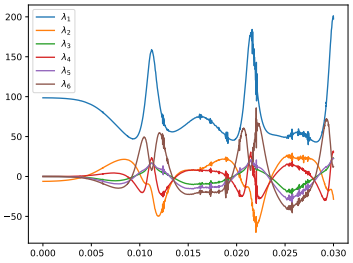

Figure 43: The Lagrange parameter of the index 3 problem.

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 noticeable 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 in 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.

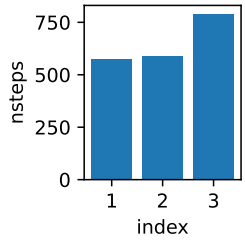

Figure 44: `nsteps` in relation to the index.

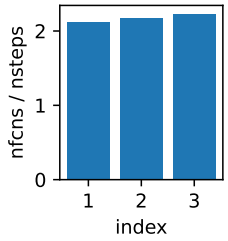

Figure 45: `nfcns/nsteps` in relation to the index.

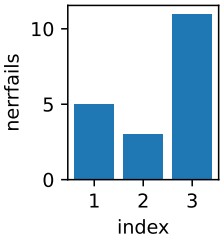

Figure 46: `nerrfails` in relation to the index.

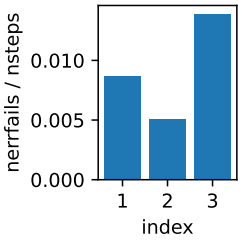

Figure 47: `nerrfails/nsteps` in relation to the index.

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.'
```

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 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

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

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 comparatively many Jacobian evaluations and having relatively few error test failures.

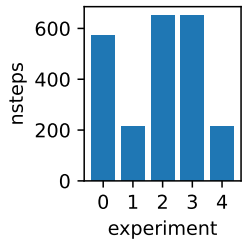

Figure 49: `nsteps` of the experiments.

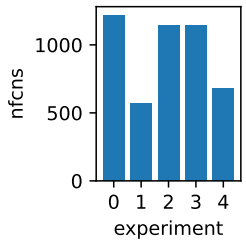

Figure 50: `nfcns` of the experiments.

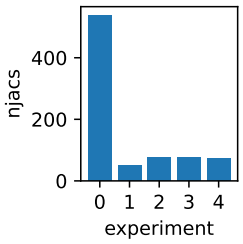

Figure 51: `njacs` of the experiments.

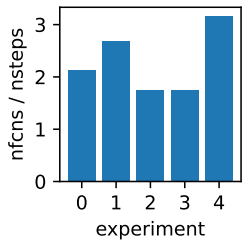

Figure 52: `nfcns/nsteps` of the experiments.

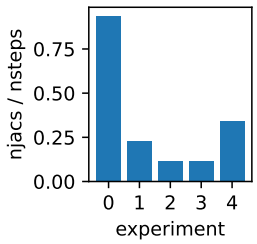

Figure 53: `njacs/nsteps` of the experiments.

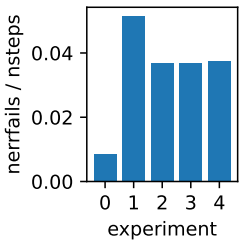

Figure 54: `nerrfails/nsteps` of the experiments.

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 figure 55 the L_2 norm of each angle over time is plotted with respect to $h \in [0.001, 0.002)$ with $\Delta h = 5\text{e}{-6}$.

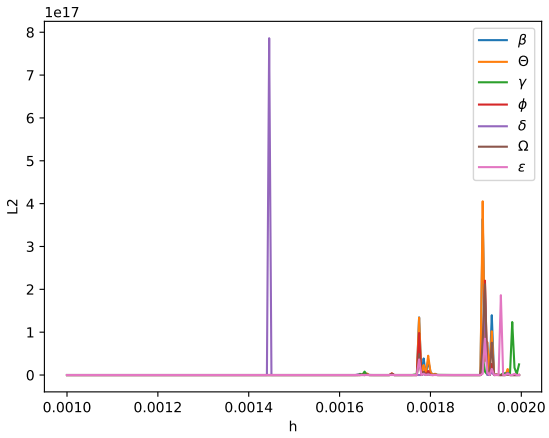

Figure 55: Value of L_2 norms depending on h

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

In figures 56 and 57 the approximation using the explicit RK4 with the stable step size $h = 10^{-4}$ can be observed.

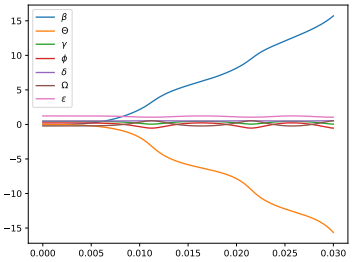

Figure 56: Approximation of angles using explicit method

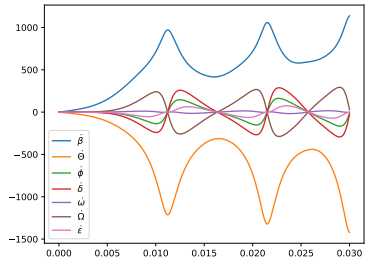

Figure 57: Approximation of angles derivatives using explicit method

Project 3

In this project we consider an initial value problem of the form

$$\begin{aligned} M\ddot{u} + C\dot{u} + Ku &= f(t) \\ u(0) &= u_0 \\ \dot{u}(0) &= v_0 \end{aligned} \tag{4}$$

where $M, C, K \in \mathbb{R}^{n\times n}$ are sparse matrices and $u_0, v_0 \in \mathbb{R}^n$ initial values. For this we implemented an Assimulo problem class, the HHT solver and the Newmark implicit and explicit solvers. The explicit solver was tested on the pendulum. The implicit solvers were tested on a discretised PDE obtained from an elastic beam which yields a system of the form (4).

The elastic pendulum revisited

In this section we will use the aforementioned methods to solve the problem of the elastic spring from Project 1. In particular we will compare the performance of different explicit methods for solving the problem. We will compare Newmark Explicit with Explicit Euler and Newmark Explicit with RK4.

It is common to both methods that the simulations starts at the same point and rapidly drifts away. This drift will continue in different ways depending on the method tested.

Newmark - RK4

This is probably the most interesting of both cases given that RK4 does not lose stability as quickly as Explicit Euler does. The distance between the approximations will dilate in an oscillating manner with increasing amplitude. This can be observed in figure 58 for y and in figure 59 for $\dot{y}$.

The amplitude of the oscillations will eventually converge to a value which appears to depend on the value of k. For bigger values of k the amplitude appears to converge faster. Convergence of the amplitudes can be observed in figure 60 for y and in figure 61 for $\dot{y}$.

With regards to performance, Newmark's method is faster than RK4 being for different values of h and t_f approximately c times faster having c order of magnitude 0.

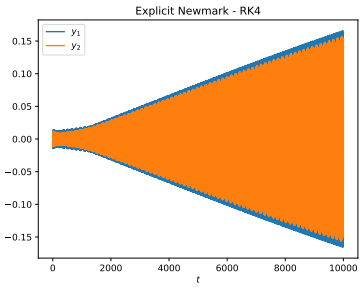

Figure 58: Difference of y with $h = 0.01$ and $k = 10$.

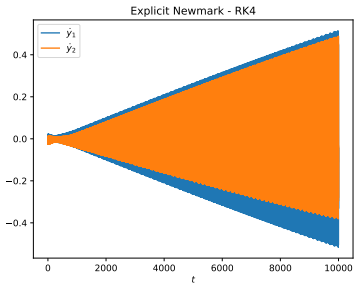

Figure 59: Difference of $\dot{y}$ with $h = 0.01$ and $k = 10$.

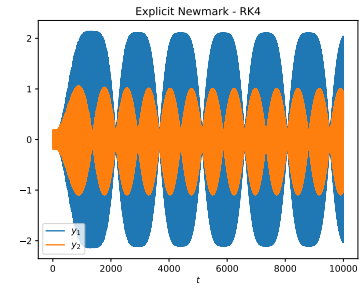

Figure 60: Difference of y with $h = 0.01$ and $k = 1e3$.

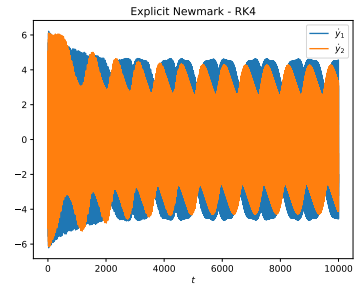

Figure 61: Difference of $\dot{y}$ with $h = 0.01$ and $k = 1e3$.

Newmark - Euler

Not much can be said about the relation between Newmark's and Euler's methods given the unstable nature of Explicit Euler. Euler's method explodes to infinity while Newmark's remains stable. This can be observed in figure 62.

With regard to performance, Euler's method is faster than Newmark's for different values of h and t_f approximately c times faster having c order of magnitude 0.

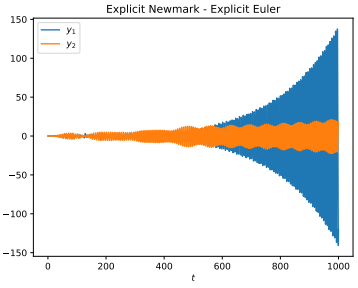

Figure 62: Difference of y with $h = 0.01$ and $k = 10$.

An elastic beam

In the second part of the project we tested the implicit Newmark and the HHT method on a discretised beam plotted in

figure 63. The beam is displaced by a force until it is deformed as in figure 64. With time it then swings back and forth between the positions in figures 63 to 66. Figure 67 shows the displacement of the tip of the beam in dependence of time.

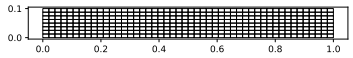

Figure 63: Beam position at $t \approx 0$.

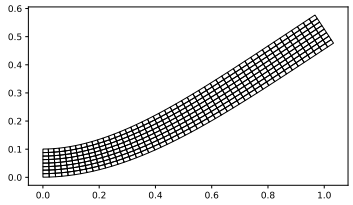

Figure 64: Beam position at $t \approx 1.7$.

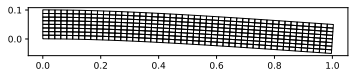

Figure 65: Beam position at $t \approx 2.4$.

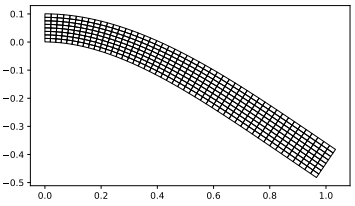

Figure 66: Beam position at $t \approx 2.7$.

We can calculate the elastic and kinetic energies according to the formulas

$$E_{\text{kin}} = \frac{1}{2} v^\top M v \qquad E_{\text{elast}} = \frac{1}{2} u^\top C u$$

which add up to the total energy

$$E_{\text{tot}} = E_{\text{kin}} + E_{\text{elast}}$$

The development of the energy of the system can be seen in figure 68. One can see in particular that after the initial application of an external force to the system the energy remains almost constant. As in project 1 the variance of the total energy serves as a measure of the instability of the solver. Here we calculate this variance only for the latter 4/5 of the simulation because the applied force changes the total energy in the first part. With an ideal solver this quantity vanishes.

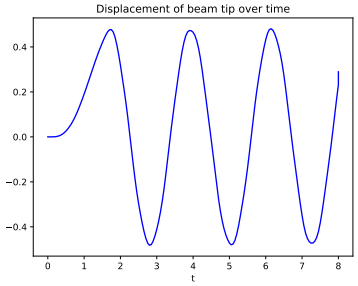

Figure 67: Displacement of the tip of the beam of the solution.

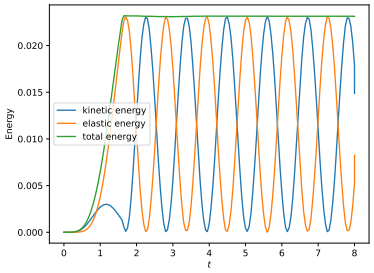

Figure 68: Energy in dependence of time of the solution.

A brief comparison of solvers

In a first experiment we compare the performance of our implementation of the HHT solver and the implicit Euler solver from Assimulo. The HHT method was applied with the parameter $\alpha = 0$ and the step size $h = 0.05$ was identical for both methods. The results are plotted in figure 69. One sees that for all solvers the variance of the total energy is small. There is however a big difference in the performance of the methods. On my computer the implicit Euler solver takes roughly two orders of magnitude longer than the HHT method.

solver	HHT	ImplicitEuler
Variance(E_{tot})	2.3e-10	2.2e-13
Elapsed simulation time [s]	1.4	216.9

Figure 69: Performance of various solvers for the beam problem.

Testing the implicit Newmark solver

In a second experiment we test the dependence of the implicit Newmark method on the parameters β and γ while keeping the step size $h = 0.05$ constant. The variance of the total energy can be seen in figure 70. It should be noted that we cut off the value of the variance of the total energy at 10^2 because any greater value shows that the solution is unstable for the specific choice of β and γ. One sees that for $1/2 \leq \gamma \leq 2\beta$ the solver is stable. Also observe that the solver is most stable for $\gamma \approx 1/2$ and for $\beta \approx 1/4$ which is precisely the value at which the method achieves second order accuracy. Figure 71 shows what happens if we leave the region of stability.

Testing the HHT solver

In a final experiment we test the dependence of the HHT solver on the parameter α. The variance of the total energy can be seen in figure 72. It is noticeable

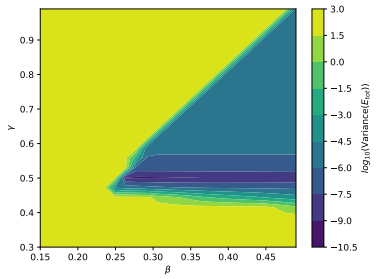

Figure 70: Dependence of the variance of the total energy on the parameters β and γ.

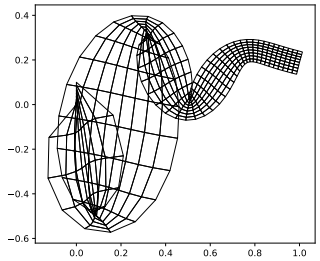

Figure 71: The parameter choice $\beta = 0.25$ and $\gamma = 0.7$ yields rather peculiar beam configurations.

that this value decreases as α increases albeit from a low level. To make it more visible what is happening we set the step size to $h = 1$ and plotted the energies of the solutions for the HHT solver as seen in figures 73 and 74. Here the parameter $\alpha = -1/3$ acts in a dampening manner in comparison to the plot for $\alpha = 0$. Despite the very rough step size the energy plot for $\alpha = 0$ shares many features of the solution with a more refined step size. For example the total energy is almost constant and the kinetic and elastic energies are eventually periodic.

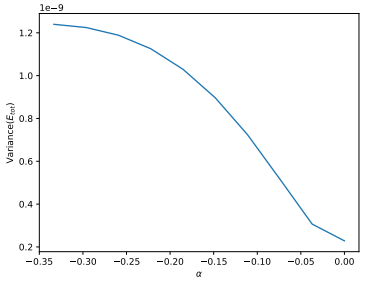

Figure 72: Dependence of the variance of the total energy on the parameter α.

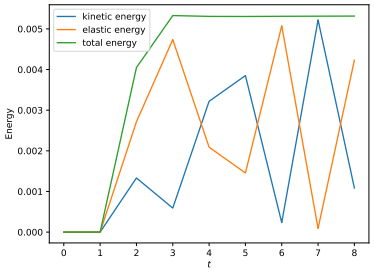

Figure 73: Energy f1or the HHT method with $\alpha = 0$ and step size $h = 1$.

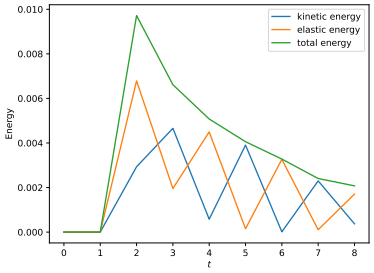

Figure 74: Energy for the HHT method with $\alpha = -1/3$ and step size $h = 1$.

Appendix

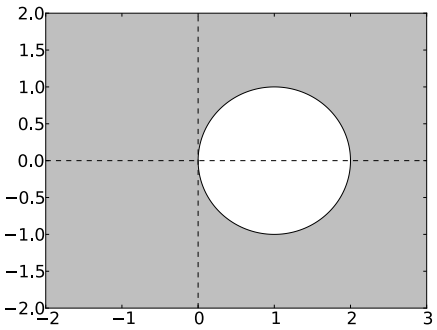

Figure 75: Stability region for BDF1, taken from [1]

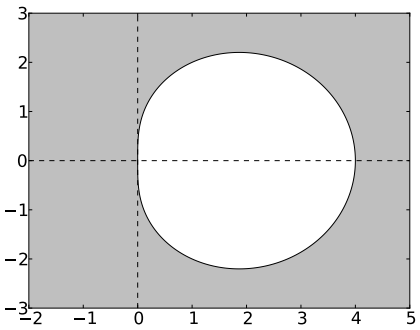

Figure 76: Stability region for BDF2, taken from [1]

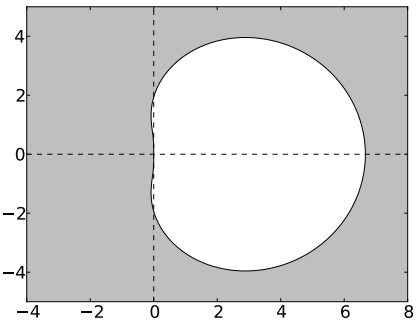

Figure 77: Stability region for BDF3, taken from [1]

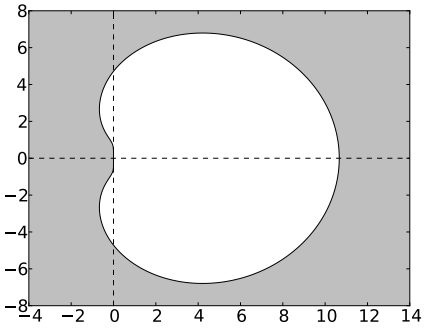

Figure 78: Stability region for BDF4, taken from [1]

Bibliography

[1] Backward differetiation formula. *Estimation lemma — Wikipedia, The Free Encyclopedia*. Online; accessed 27-January-2023. 2022. URL: https://en.wikipedia.org/wiki/Backward_differentiation_formula.

[2] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*. Vol. 14. Springer Series in Computational Mathematics. Stiff and differential-algebraic problems, Second revised edition, paperback. Springer-Verlag, Berlin, 2010, pp. xvi+614. ISBN: 978-3-642-05220-0. DOI: 10.1007/978-3-642-05221-7. URL: https://doi-org.ludwig.lub.lu.se/10.1007/978-3-642-05221-7.

[3] simulation-tools-VT23. *Github repository to the project.* Online. 2023. URL: https://github.com/TheoKoppenhoefer/simulation-tools-VT23.