Simulation Tools, VT23

Report on Project 1

Salvador Castagnino, Theo Koppenhöfer

January 27, 2023

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

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

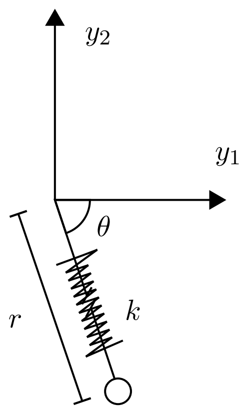

Figure 1: The pendulum

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

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

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

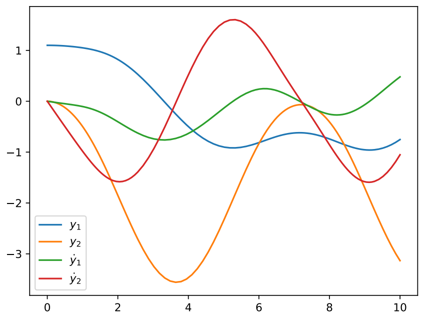

Figure 2: State in dependence of time.

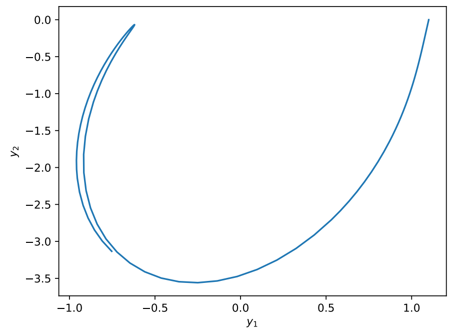

Figure 3: Path traced out by pendulum.

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

```
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.

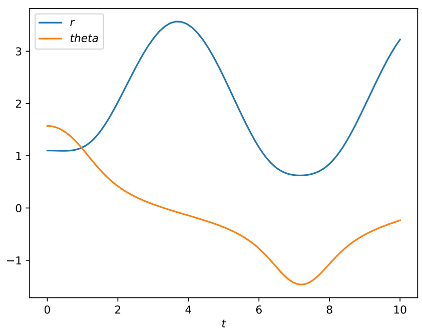

Figure 4: Polar coordinates.

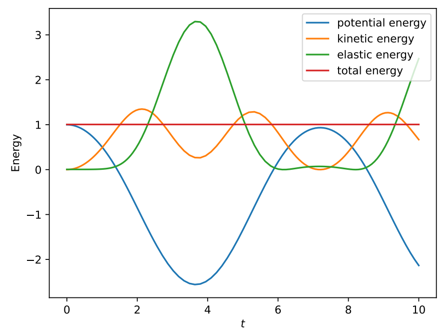

Figure 5: Energy plot

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 $y_0 = 1.1, 0, 0, 0 and have$

$$0, 20$$

$fordomain.Thegraphsarepresentedinpolarcoordinateswhererreferstothelengthofthespringand$thetaref
It can be observed that for a step size of $h = 0.01$ Explicit Eurler $Figure$**??** already shows instability for values of $k = 50$ while RK4 $Figure$**??** with that same step size remains stable for values up to $k = 3000$.

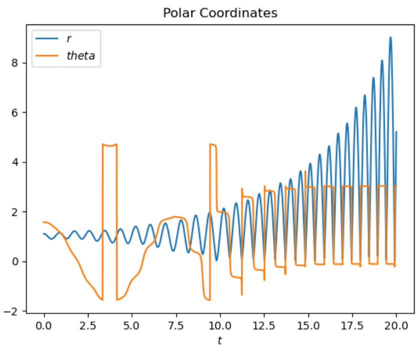

Figure 6: `nfcns/nsteps` Explicit Euler $h = 0.01$ $k = 50$

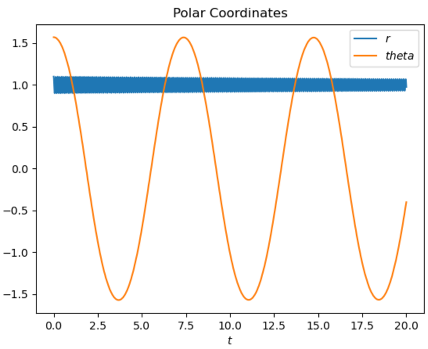

Figure 7: `njacs/nsteps` RK4 $h = 0.01$ $k = 3000$

For Explicit Euler $Figure$**??**, 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 $Figure$**??**, once unstabilized it's growth is much more rapid than Explicit Euler's and it does so without oscillating.

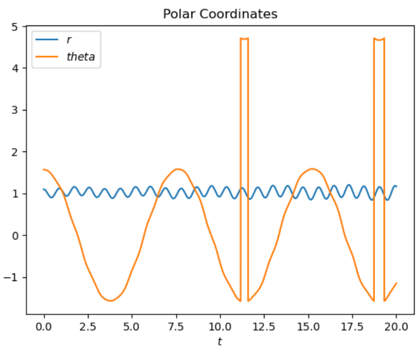

Figure 8: `nfcns/nsteps` Explicit Euler $h = 0.001$ $k = 50$

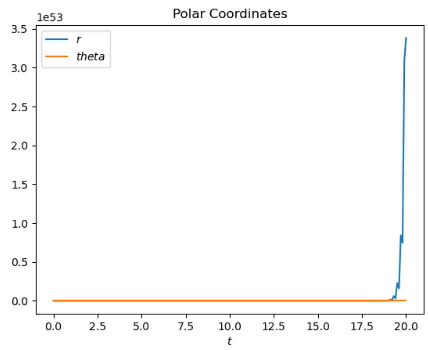

Figure 9: `njacs/nsteps` RK4 $h = 0.1$ $k = 975$

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

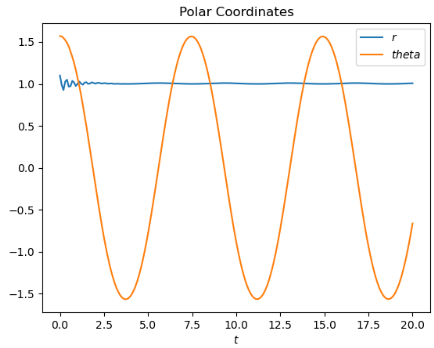

Figure 10: `njacs/nsteps` RK4 $h = 0.1$ $k = 300$

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 $y_0 = 1.1, 0, 0, 0andhave$

$$0, 20$$

$fortimedomain.Itisinterestingtoseehowtheoscillationofthespringdecaysforimplicitmethods.Thisdecayc$

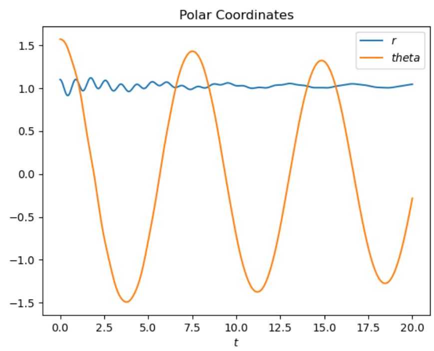

Figure 11: `njacs/nsteps` Implicit Euler $h = 0.01$ $k = 50$

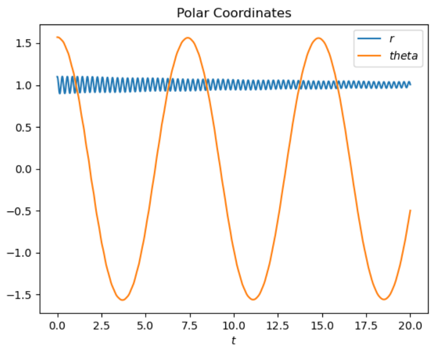

Figure 12: `nfcns/nsteps` BDF2-Fixed Point $h = 0.001$ $k = 500$

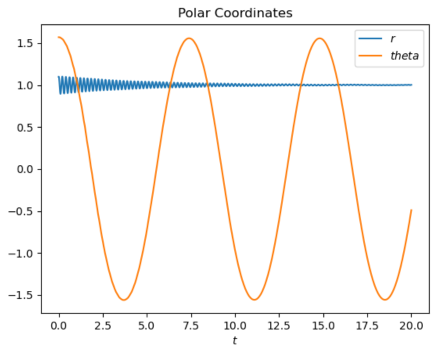

Figure 13: `njacs/nsteps` BDF2-Fixed Point $h = 0.01$ $k = 1000$

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 $Figure$**??**$Figure$**??** the domain is increased to $0, 100$.

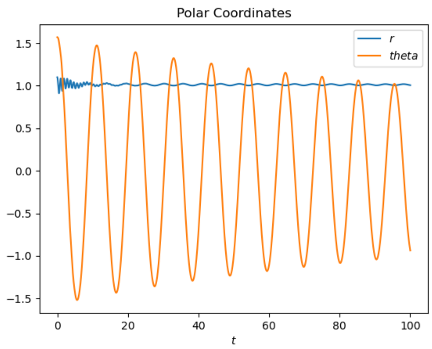

Figure 14: `nfcns/nsteps` BDF2-Newton $h = 0.01$ $k = 100$

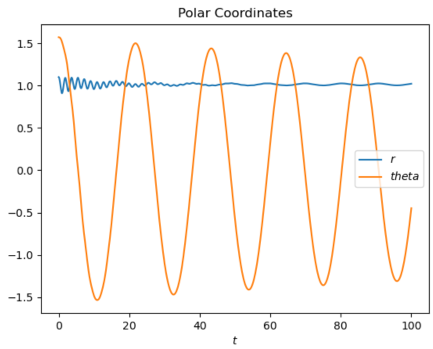

Figure 15: `njacs/nsteps` BDF4-Newton $h = 0.01$ $k = 100$

It is interesting to see how the relation between the speed of decay of the spring oscillation in inversely proportional 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-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 6. As the number of function evaluations per stepsize nfcns/nsteps hovers slightly above 1 for all methods (c.f. figure 8) the number of function evaluations increase analogously to nsteps with k as can be seen in figure 7. 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.

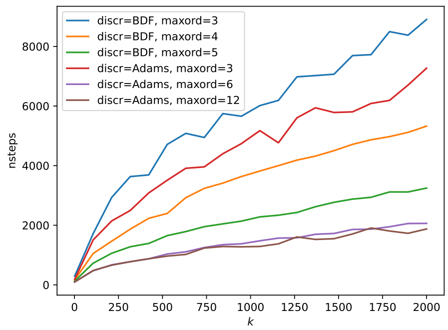

Figure 16: nsteps in relation to the parameter k.

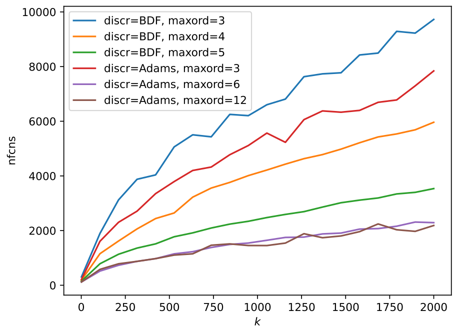

Figure 17: nfuncs in relation to the parameter k.

From figure 9 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 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 11 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.

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 12 to 16. We note that as rtol increases the number of steps decreases (c.f. figure 12). If one compares figures the instability index for $k \approx 10^3$ in

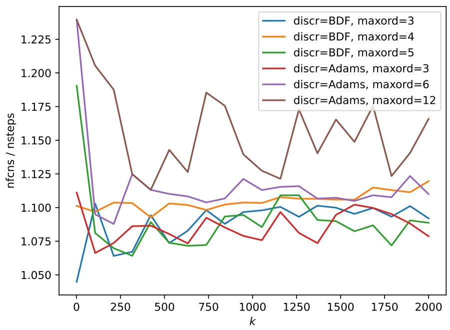

Figure 18: `nfcns/nsteps` in relation to the parameter k.

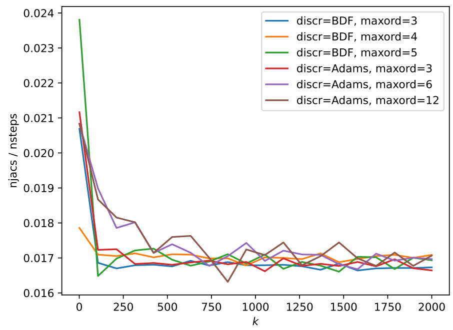

Figure 19: `njacs/nsteps` in relation to the parameter k.

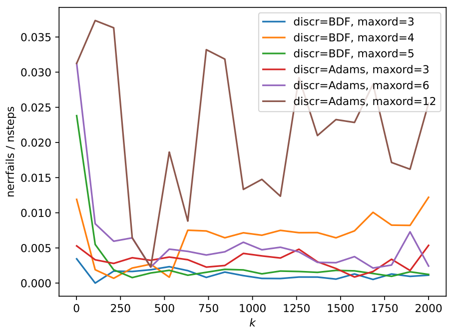

Figure 20: `nerrfails/nsteps` in relation to the parameter k.

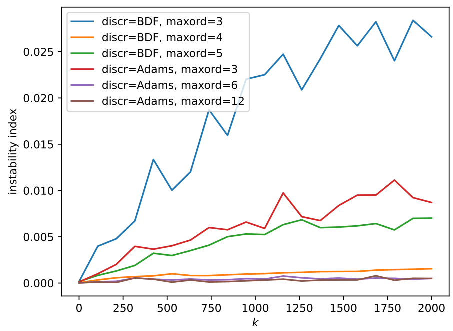

Figure 21: instability index in relation to the parameter k.

figure 11 with the instability index in figure 16 one sees that changing the `rtol` parameter from the default makes the result significantly worse in terms of energy conservation.

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 17. In either case we observe that fixing the tolerance seems to come at the cost of energy conversation as is dramatically visualised in Figures 18 and 19.

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.

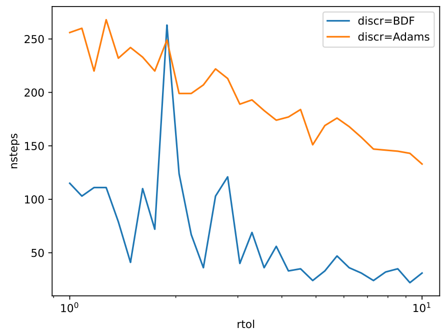

Figure 22: `nsteps` in relation to the parameter `rtol`.

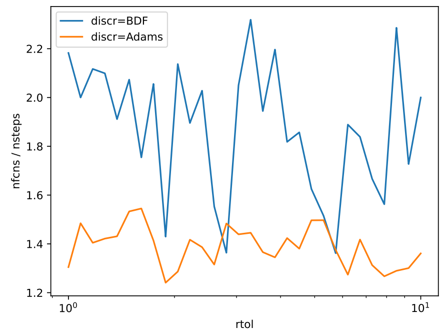

Figure 23: `nfcns/nsteps` in relation to the parameter `rtol`.

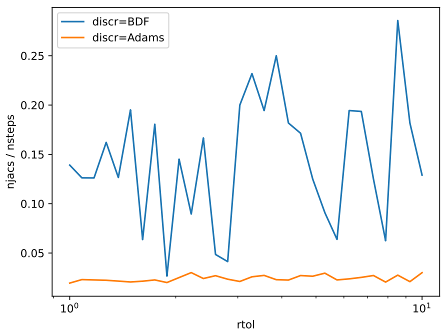

Figure 24: `njacs/nsteps` in relation to the parameter `rtol`.

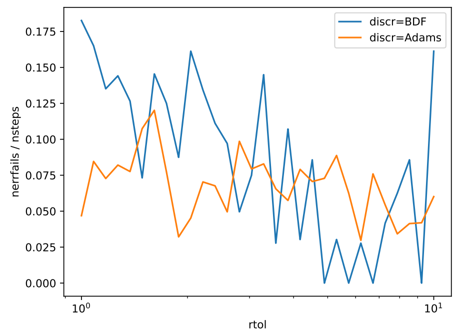

Figure 25: `nerrfails/nsteps` in relation to the parameter `rtol`.

References

[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] Peter Deuflhard and Folkmar Bornemann. *Numerische Mathematik 2*. revised. de Gruyter Lehrbuch. [de Gruyter Textbook]. Gewöhnliche Differentialgleichungen. [Ordinary differential equations]. Walter de Gruyter & Co., Berlin, 2008, pp. xii+499. ISBN: 978-3-11-020356-1.

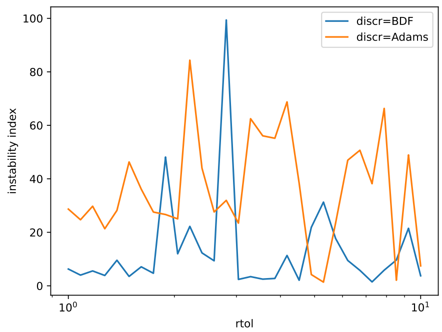

Figure 26: instability index in relation to the parameter `rtol`.

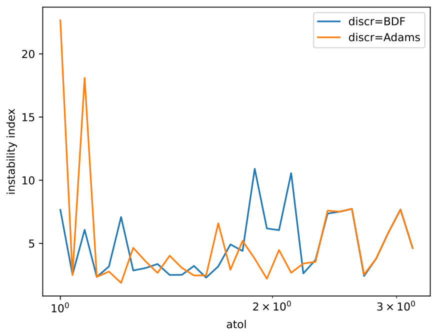

Figure 27: instability index in relation to the parameter `atol`.

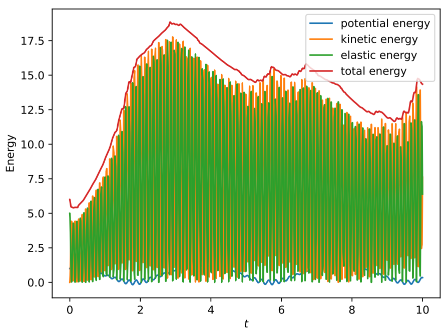

Figure 28: Energy plot for $k = 10^3$ with `atol=1E-2`.

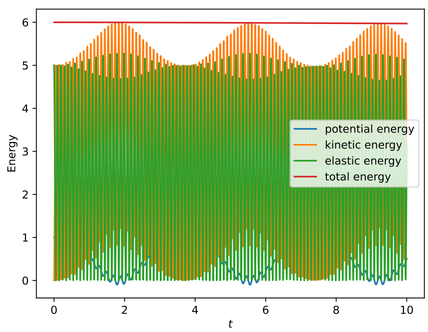

Figure 29: Energy plot for $k = 10^3$.

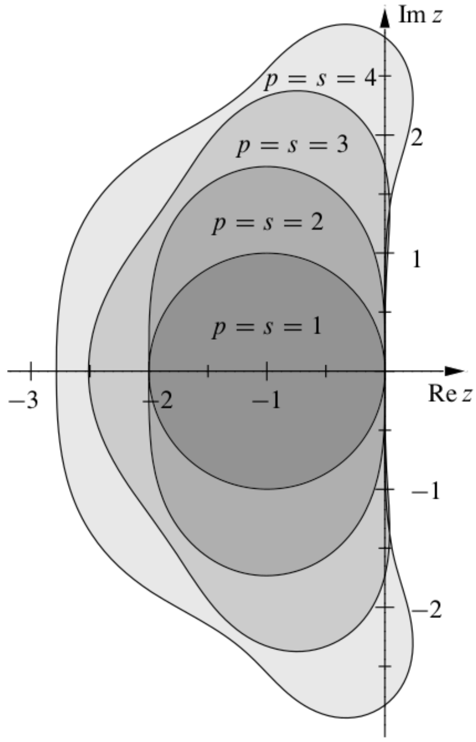

Figure 30: Stability regions of the Runge-Kutta-methods, taken from [2][p.238]

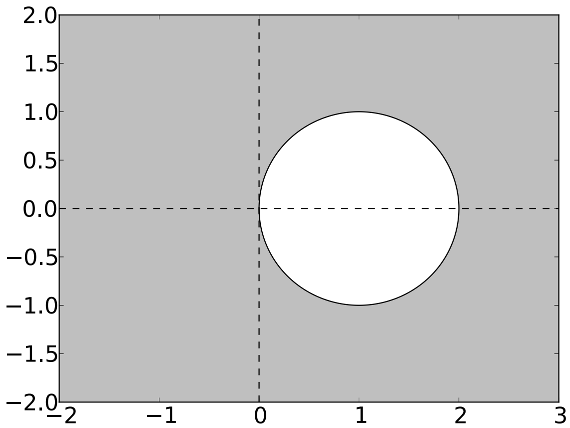

Figure 31: Stability region for BDF1, taken from [1]

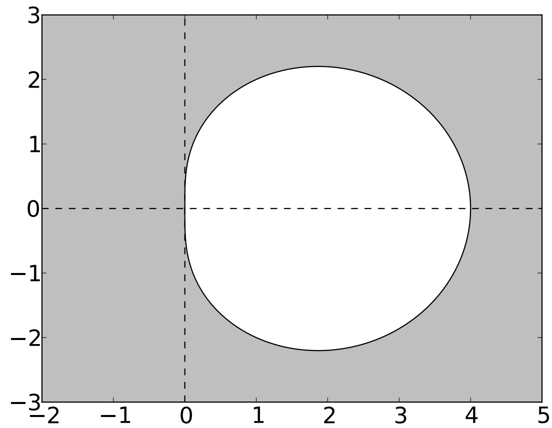

Figure 32: Stability region for BDF2, taken from [1]

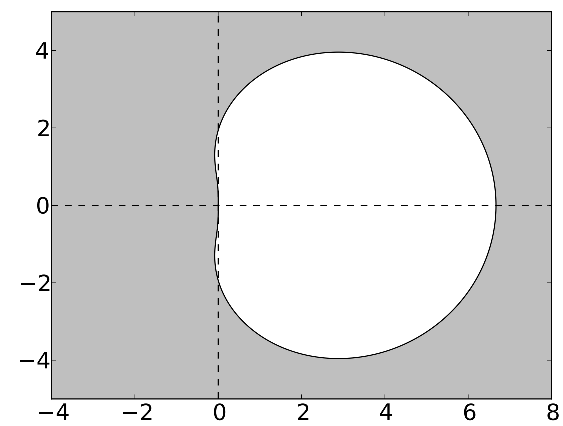

Figure 33: Stability region for BDF3, taken from [1]

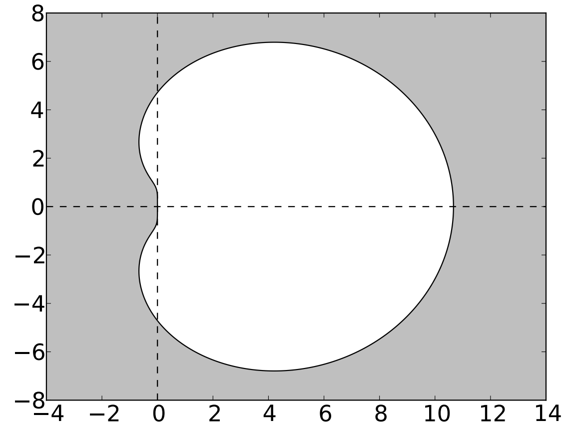

Figure 34: Stability region for BDF4, taken from [1]

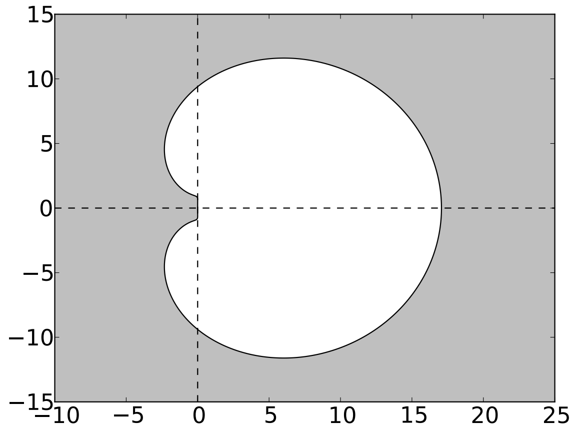

Figure 35: Stability region for BDF5, taken from [1]

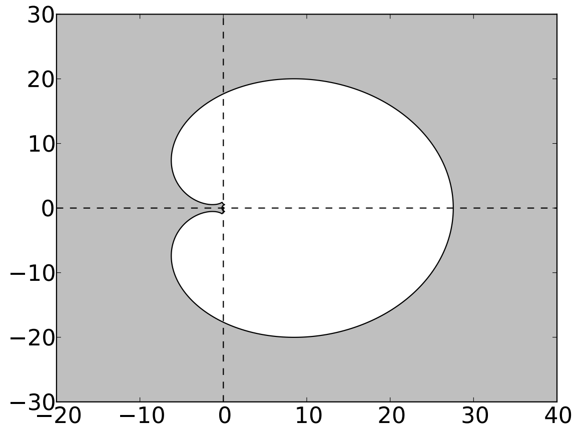

Figure 36: Stability region for BDF6, taken from [1]