thesis of me

some terrible student

November 2020

Contents

Chapter 1

Introduction and Motivation

1.1 Earthquake Simulations

1.1.1 Friction Laws

1.1.2 Sequences of Eartquakes and Aseismic Slip

1.2 Differential Algebraic Equations (DAE)

1.3 Time-adaptive Time Integration Methods

general concept: compute in addition to the solution another, higher order approximation. The difference between the two solutions is the error approximation. With the estimate of the local truncation error, a new timestep size can be defined to match a set tolerance.

1.3.1 Explicit RK methods

Runge Kutta methods 4th order (Fehlberg and Dornand-Prince) -> reuse already computed coefficients for the higher order error evaluation + Bogacki-Shampine

1.3.2 Implicit BDF methods

Implicit methods are well-suited to solve stiff problems and allow for higher timesteps than explicit methods. Instead of evaluating the time-derivative in the right-hand side of an ODE for a known slution ψ_n, it is evaluated at the next timestep with ψ_{n+1}, which is not known. This requires solving an algebraic equation at each timestep without an analytic expression at hand for it. BDF (Backward Differentiation Formula) methods offer a convenient framework for implicit methods up to the order $p = 6$. They are multi-step methods, where a method of order p requires the solutions at the p previous timesteps.
The first order BDF-scheme corresponds to the backward Euler method in Equation 1.1.

$$\psi_{n+1} = \psi_n + h_n f(\psi_{n+1}, V_{n+1}) \tag{1.1}$$

Next, we try to derive the second order BDF-scheme. Because of the adaptive time-stepping, the traditional coefficients of the BDF2 scheme cannot be used, but will be dependent of the current and previous

timestep sizes h_{n+1} and h_n. To find these coefficients, the Taylor polynomials of ψ_n and ψ_{n+1} are evaluated with respect to the unknown ψ_{n+2}.

$$\psi_n = \psi_{n+2} - (h_n + h_{n+1})\frac{d\psi_{n+2}}{dt} + \frac{(h_n + h_{n+1})^2}{2}\frac{d^2\psi_{n+2}}{dt^2} + \mathcal{O}\left((h_n + h_{n+1})^3\right) \tag{1.2}$$

$$\psi_{n+1} = \psi_{n+2} - h_{n+1}\frac{d\psi_{n+2}}{dt} + \frac{h_{n+1}^2}{2}\frac{d^2\psi_{n+2}}{dt^2} + \mathcal{O}\left(h_{n+1}^3\right) \tag{1.3}$$

The idea is to add equations (1.2) and (1.3), where the latter is multiplied by a factor α in a way that the second-derivative term drops. The addition of the two Taylor-expansions yields:

$$\psi_n + \alpha\psi_{n+1} = (1+\alpha)\psi_{n+2} - (h_n + (1+\alpha)h_{n+1})\frac{d\psi_{n+2}}{dt} + \frac{(h_n + h_{n+1})^2 + \alpha h_{n+1}^2}{2}\frac{d^2\psi_{n+2}}{dt^2} + \mathcal{O}\left(h^3\right) \tag{1.4}$$

By the choice of α in Equation 1.5, the coefficient in front of the second time derivative of ψ_{n+2} vanishes and the second order time adaptive BDF method is given by: Equation 1.6.

$$\alpha = -\left(\frac{h_n}{h_{n+1}}\right)^2 - 2\frac{h_n}{h_{n+1}} - 1 \tag{1.5}$$

$$\psi_n + \alpha\psi_{n+1} - (1+\alpha)\psi_{n+2} = -(h_n + (1+\alpha)h_{n+1})\, f(\psi_{n+2}, V_{n+2}) \tag{1.6}$$

Analogously, the time-adapative thirdd-order BDF3 scheme is given with the coefficients:

$$\alpha = -\frac{(h_n + h_{n+1})\,(h_n + h_{n+1} + h_{n+2})^2}{h_{n+1}\,(h_{n+1} + h_{n+2})^2} \tag{1.7}$$

$$\beta = \frac{h_n\,(h_n + h_{n+1} + h_{n+2})^2}{h_n^2 h_{n+1}} \tag{1.8}$$

$$\psi_n + \alpha\psi_{n+1} + \beta\psi_{n+2} - (1+\alpha+\beta)\psi_{n+3} = (h_n + (1+\alpha)h_{n+1} + (1+\alpha+\beta)h_{n+2})\, f(\psi_{n+3}, V_{n+3}) \tag{1.9}$$

A more general description of the time-adaptive BDF coefficients can be obtained with the derivatives of the Lagrange interpolation polynomial. If one has a BDF method of order k, the coefficients α_{n+i} in front of the previous k solutions are needed to calculate the new solution ψ_{n+k}.

$$\sum_{i=0}^{k} \alpha_{n+i}\psi_{n+i} = f(\psi_{n+k}, V_{n+k}) \approx \dot{\psi}_{n+k} \tag{1.10}$$

To approximate the time derivative $\dot{\psi}_{n+k}$ at the new solution, one could find the polynomial $L(t)$ that interpolates all points (t_{n+i}, ψ_{n+i}) and calculate its derivative at the last point t_{n+k}. This polynomial $L(t)$ is exactly the Lagrangian interpolation polynomial, calculated as:

$$L(t) = \sum_{i=0}^{k} \psi_{n+i}\ell_i(t) \qquad \text{with} \qquad \ell_i(t) = \prod_{\substack{0\le j\le k \\ j\neq i}} \frac{t - t_{n+j}}{t_{n+i} - t_{n+j}} \tag{1.11}$$

We want to express the derivative $\dot{L}(t)$ at the specific time $t = t_{n+k}$, so that we can approximate $\dot{\psi}_{n+k} \approx \dot{L}(t_{n+k})$ For that, we first need the derivatives of the Lagrange basis polynomials $\dot{\ell}_i(t)$. Because of $\dot{L}(t_{n+k}) = \sum_{i=0}^{k} \dot{\ell}_i(t_{n+k})\psi_{n+k}$, it can be seen that if we evaluate $\dot{\ell}_i(t)$ at the time $t = t_{n+k}$, we obtain exactly the wanted coefficients α_{n+i}. The basis polynomials can be calculated with the product rule:

$$\alpha_{n+i} := \dot{\ell}_i(t_{n+k}) = \sum_{\substack{0\le m\le k \\ m\neq i}} \left[\frac{1}{t_{n+i} - t_{n+m}} \prod_{\substack{0\le j\le k \\ j\neq i,m}} \frac{t_{n+k} - t_{n+j}}{t_{n+i} - t_{n+j}} \right] \tag{1.12}$$

It can be verified that this general definition of α_{n+i} matches the previous expressions for the BDF coefficients of 1st, 2nd and 3rd order derived from Taylor expansions in Equations (1.1), (1.6) and (1.9).

1.3.2.1 Error estimate with a higher-order BDF scheme

To evaluate the local truncation error at a given timestep, a similar approach as for the time-adaptive Runge-Kutta can be used. If a scheme of order k is used, it involves solving the system a second time with order $k+1$. The error estimate is then the norm of the difference between the two calculated solutions. An obvious drawback of this method are the high computational costs, since the evaluation of the error estimate is as expensive as calculating the solution at the next timestep and requires to calculate the right-hand side of the ODE several times, which might be an expensive operation. Another limitation of this method is that the 6th order BDF scheme cannot be used for the time integration, since it implies to calculate a 7th order solution for the error estimate, which is not possible because the BDF method is only stable up to the order $k = 6$.

1.3.2.2 Error estimate with Lagrange polynomials

With the derivatives of the Lagrangian polynomial, a new possibility to estimate the error appears. Once the solution ψ_{n+k} is found, the derivative $\dot{\psi}_{n+k}$ is first approximated with the last k solutions, which gives the coefficients α_i, and then with the last $k+1$ solutions, which results in the set of coefficients β_i. Now, assume that the kth-order approximate results in a perturbed $\dot{\psi}'_{n+k}$ and the $(k+1)$th-order approximates the exact $\dot{\psi}_{n+k}$. We have:

$$\sum_{i=0}^{k} \alpha_{n+i}\psi_{n+i} = \dot{\psi}'_{n+k} \qquad \text{and} \qquad \sum_{i=-1}^{k} \beta_{n+i}\psi_{n+i} = \dot{\psi}_{n+k} \tag{1.13}$$

We now want to find the perturbation ϵ in ψ_{n+k} which is the reason for the different approximations $\dot{\psi}_{n+k}$ and $\dot{\psi}'_{n+k}$. We set:

$$\sum_{i=0}^{k-1} \alpha_{n+i}\psi_{n+i} + \alpha_{n+k}(\psi_{n+k} + \epsilon) = \sum_{i=-1}^{k} \beta_{n+i}\psi_{n+i} \tag{1.14}$$

$$\Leftrightarrow \epsilon = \frac{1}{\alpha_{n+k}} \left(\beta_{n-1}\psi_{n-1} + \sum_{i=0}^{k} (\beta_{n+i} - \alpha_{n+i})\psi_{n+i} \right) \tag{1.15}$$

This expression for ϵ can be used as an estimate for the local truncation error instead of calculating the difference to a higher order BDF method. The main advantage is that the nonlinear system does not need to be solved twice in one timestep.

1.3.2.3 Adaptive BDF order

Which one of the six BDF methods is best appropriate to solve the problem is not an easy task to determine in ahead, as the ideal order of the scheme might change throughout the simulation. Generally speaking, a high order gives the best results when the timestep size remains approximately constant between the timestes, whereas low orders are more appropriate if the timestep size currently increases or decreases a lot. The idea is to find at the end of a timestep an optimal BDF order for the next step.
If k_n denotes the current order of the scheme, one can evaluate the error estimate if a scheme of order $k_n - 1$, k_n or $k_n + 1$ was used. We then obtain three values for the error estimate ϵ_-, ϵ and ϵ_+. This is easy to calculate with the Lagrangian polynomials, since it only requires to calculate the coefficients α_i for the considered order and sum up the weighted solution vectors. For the embedded higher-order BDF scheme it is also possible, but requires then in total four executions of the Newton iteration in one step, which is very likely not worth it.
Next, one calculates the factors f_-, f and f_+ to change the timestep size h_n from the current step to the next step $h_{n+1} = f h_n$. For the most basic timestep adapters with a safety factor C, this gives:

$$f_- = C{\epsilon_-}^{-1/k_n} \quad , \quad f = C\epsilon^{-1/(k_n+1)} \quad \text{and} \quad f_+ = C{\epsilon_+}^{-1/(k_n+2)} \tag{1.16}$$

One then finds the largest of the three factors, and decreases the order by one if it is f_-, remains at the same order if it is f and increases the order by one if it is f_+. Of course, one has to ensure that the new order k_{n+1} remains in the range $[1, 6]$.

Chapter 2

First Experiments

In a first step, different adaptive time-stepping schemes are evaluated for accuracy and stability on a single state variable. For that, we consider the DAE of the rate and state problem on a single node.

$$\frac{d\psi}{dt} = f(\psi, V) = 1 - \frac{V\psi}{L} \tag{2.1}$$

$$0 = g(\psi, V) = \tau - \sigma_n a \sinh^{-1}\left(\frac{V}{2V_0} e^{\frac{f_0 + b\log\left(\frac{V_0\psi}{L}\right)}{a}}\right) - \eta V \tag{2.2}$$

2.1 Method of Manufactured Solutions

To be able to evaluate different adaptive timestepping schemes, an analytical solution of the problem is required. For any given combination of functions f and g in the DAE, this is an almost impossible task. One approach is to solve the problem backwards [Roa01], thus starting from a possible solution of the problem and then to adapt the functions f and g according to it. For the two problems described above, we can start from the evolution of the slip rate $V^*(t)$. In Equation 2.3, the slip rate increases from 0 to 1 over a time span t_w at the time t_e.

$$V^*(t) = \frac{1}{\pi}\tan^{-1}\left(\frac{t - t_e}{t_w} + \frac{\pi}{2}\right) \tag{2.3}$$

The manufactured evolution of the state variable $\psi^*(t)$ can be calculated by solving the algebraic equations (2.2) and (??). The time derivatives $\frac{dV^*(t)}{dt}$ and $\frac{d\psi^*(t)}{dt}$ of the manufactured solutions can be easily evaluated and the new DAE is defined in Equation 2.4.

$$\frac{d\psi}{dt} = f(\psi, V) - f(\psi^*, V^*) + \frac{d\psi*}{dt} \tag{2.4}$$

$$0 = g(\psi, V) \tag{2.5}$$

For any initial conditions $V_0 = V^*(0)$ and $\psi_0 = \psi^*(0)$, the solution of the DAE exists and with the expression $V(t) = V^*(t)$ $\psi(t) = \psi^*(t)$. Therefore, we know an analytical solution and the results of the numerical simulations can be directly compared to it.

2.2 Time integration

In this chapter, the Runge-Kutta-Fehlberg method (RKF4) of 4th order with an embedded 5th order error estimate is used as explicit method.

For the implicit methods, we consider for now only the BDF schemes of first and second order with an error estimate of respectively 2nd and 3rd order. Both described methods to estimate the error, with an embedded higher-order evaluation of the scheme as described in subsubsection 1.3.2.1 and with the derivatives of the Lagrangian polynomials in subsubsection 1.3.2.2 are used and compared.
Because of the DAE form, the values of $\psi n + 1$ in Equation 1.1 and $\psi n + 2$ in Equation 1.6 cannot be calculated analytically easily. To solve the equation, it is transformed into an algebraic equation, in which the right hand side is 0, to obtain the form:

$$F(\Psi) = 0 \tag{2.6}$$

where Ψ stands respectively for ψ_{n+1} and ψ_{n+2}. It is solved iteratively with the Newton-Raphson method [Rap97] in Equation 2.7 and the secant method to approximate the derivative $F'(\Psi) = \frac{dF(\Psi)}{d\psi}$ in Equation 2.8.

$$\Psi_{k+1} = \Psi_k + \frac{F(\Psi_k)}{F'(\Psi_k)} \tag{2.7}$$

$$F'(\Psi_k) = \frac{F(\Psi_k) - F(\Psi_{k-1})}{\Psi_k - \Psi_{k-1}} \tag{2.8}$$

The Newton-Raphson method converges in theory with second order, however the approximate of the derivative with the secant method, which bases on the first order finite differences, reduces the overall convergence of the iterative scheme to first order. The iteration is stopped as soon as the difference between two consecutive terms remains below a tolerance value. As initial value, one explicit Euler step is taken, and the Newton-Raphson method converges usually after less than three iterations. With the initial step, the BDF scheme needs about four evaluations of the DAE, and since it has to be executed twice for each considered timestep size (once for the solution and once for the error estimate) with a common initial step, it requires in total seven evaluations of the DAE, which is only one more than in the previously presented Runge-Kutta-Fehlberg method.

2.3 Timestep Update

At each timestep, the goal is to maximise the size of the timestep h_{n+1} under the condition that the local error estimate r_{n+1} remains inferior to an allowed tolerance ϵ. The controller C is a function

$$h_{n+1} = C(\epsilon, r_{n+1}, h_n) \tag{2.9}$$

At each timestep, the controller is iteratively called until the step size allows a local error that fulfills the tolerance. In the ideal case, it only requires one iteration to find a new suitable timestep which is still as large as possible.

2.3.1 Elementary Local Error Control

The simplest realisation of the timestep size controller is the method of the elementary local error control. For a numerical scheme of order $k - 1$, it assumes that at each timestep, the local error is directly proportional to the k-th power of the step size by a factor Φ.

$$r_{n+1} = \Phi h_n^k \tag{2.10}$$

To maximise the timestep size, h_{n+1} is chosen in a way that the induced error matches exactly the allowed tolerance $\Phi h_{n+1}^k = \epsilon$. It can be rewritten as:

$$h_{n+1} = \left(\frac{\epsilon}{\Phi}\right)^{1/k} \Leftrightarrow h_{n+1} = \left(\frac{\epsilon}{\Phi h_n^k}\right)^{1/k} h_n \Leftrightarrow h_{n+1} = \left(\frac{\epsilon}{r_{n+1}}\right)^{1/k} h_n \tag{2.11}$$

In practice, there is never such a constant error factor Φ, but it will take a different value ϕ_n at each timestep. The approach is still valid if only small variations occur from one timestep to the following,

thus $\phi_{n+1} \approx \phi_n$. To cover those small variations, a security factor $\theta < 1$ is usually multiplied to the tolerance such that the method does not aim exactly the tolerance ϵ for the next local error, but some value below it. A factor $\theta = 0.98$ is sufficient to significantly reduce the number of iterations until a fitting local error has been reached. The controller can hence be expressed as:

$$h_{n+1} = \left(\frac{\theta\epsilon}{r_{n+1}}\right)^{1/k} h_n \tag{2.12}$$

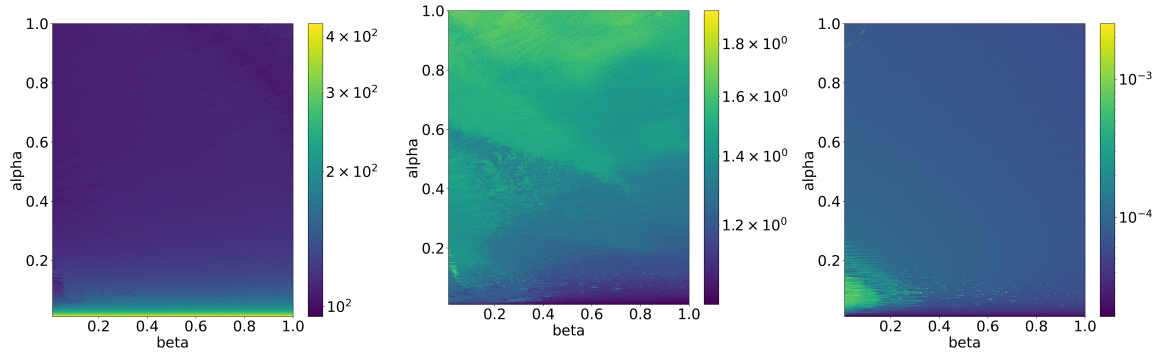
The initial assumption in Equation 2.10 that the error exposes an asymptotic behaviour is not always given. Especially for stiff problems, such a relationship between the timestep size and the local error is only true for very small timesteps.

2.3.1.1 PI controller

The proportional integral (PI) control is an extension of the previous method by taking into account the trend of the error evolution [SÖ2]. An additional term is added to the controller which depends on the ratio between the previous error estimate and the current one. thus, if the local error is decreasing compared to the previous timestep, it is likely that the timestep can be further increased, and reversely, an increase of the local error should imply a decrease of the timestep size. The controller is given by:

$$h_{n+1} = \left(\frac{\theta\epsilon}{r_{n+1}}\right)^{k_I} \left(\frac{r_n}{r_{n+1}}\right)^{k_P} h_n \tag{2.13}$$

There are now two design parameters k_I and k_P that have to be determined. Their ideal values depend on the considered problem and the picked numerical solver, therefore they have to be empirically found. The parameters can be expressed in function of the order of the numerical solver $k-1$ by $k_I = \alpha/k$ and $k_P = \beta/k$. For the three considered solvers (RKF45, BDF12 and BDF23), the total amount of timesteps and the average number of iterations until a suitable timestep size has been found are measured for varying values of α and β. The results are shown in Figures 2.1, 2.2 and 2.3. The tolerance for the local truncation error estimate has been set to $\epsilon = 1 \cdot 10^{-6}$ to ensure convergence of all numerical schemes and thus obtain comparable results.

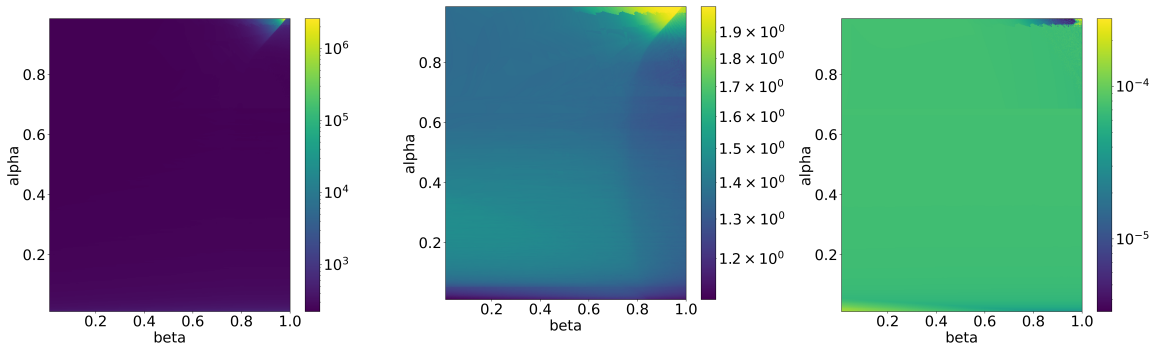


(a) Number of required timesteps

(b) Average number of iterations per timestep

(c) Absolute error of the variable ψ

Figure 2.1: Impact of the parameters α and β of the PI controller on the number of required iterations to solve the model problem from Equation 2.1 using the explicit Runge-Kutta-Fehlberg method (**RKF45**) for a simulation time of $t = 5.0s$

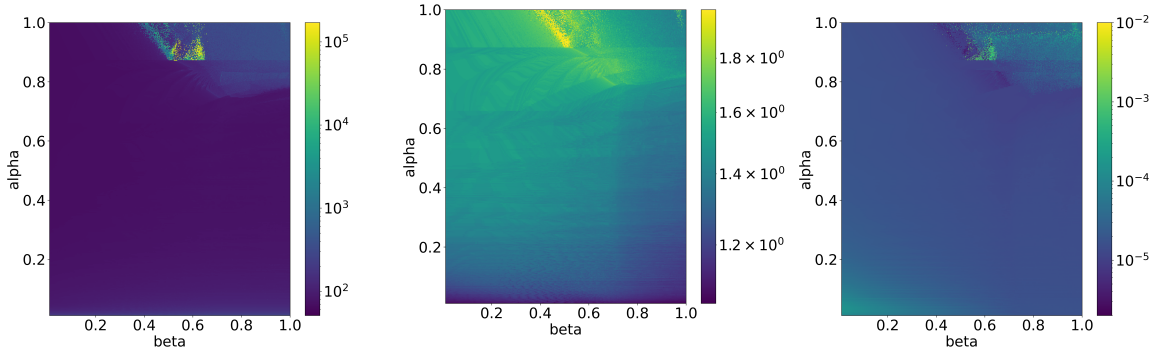


(a) Number of required timesteps

(b) Average number of iterations per timestep

(c) Absolute error of the variable ψ

Figure 2.2: Impact of the parameters α and β of the PI controller on the number of required iterations to solve the model problem from Equation 2.1 using the implicit BDF method of first order (**BDF12**) for a simulation time of $t = 5.0s$



(a) Number of required timesteps

(b) Average number of iterations per timestep

(c) Absolute error of the variable ψ

Figure 2.3: Impact of the parameters α and β of the PI controller on the number of required iterations to solve the model problem from Equation 2.1 using the implicit BDF method of second order (**BDF23**) for a simulation time of $t = 5.0s$

We chose as parameters:

- for **RKF45**: $\alpha = 0.4$ and $\beta = 0.6$
- for **BDF12**: $\alpha = 0.9$ and $\beta = 0.4$
- for **BDF23**: $\alpha = 0.9$ and $\beta = 0.4$

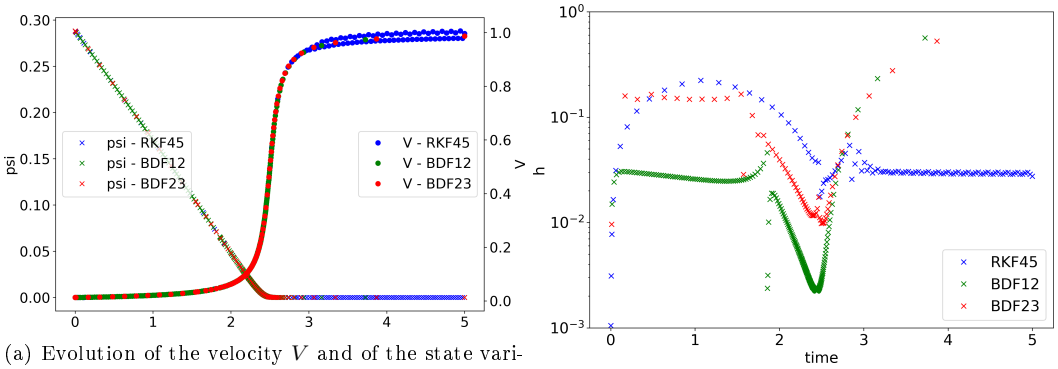
—Put some more explanations here —
—Find the balance between number of iterations and accuracy —
—Numerical instabilities for the implicit methods for high values of α and β —
failure of the error estimate? too small timesteps?

2.4 Comparison of the Numerical Schemes

So far, one explicit (RKF45) and two implicit (BDF12, BDF23) numerical solvers have been implemented for the initial DAE in Equation 2.2 and their ideal parameter choice for their use with a PI controller has been discussed. Now, the quality of the solutions is compared.

2.4.1 Numerical solutions

Again, the allowed tolerance of the PI controller is set to $\epsilon = 1 \cdot 10^{-6}$. The method of the manufactured solutions is chosen to have an analytical solution against which numerical computations can be compared. The parameters t_e and t_w are chosen in a way that the velocity is initially close to zero and increases to 1 over the time span of one second at the middle of the simulation time. Figure 2.4a depicts the solution variables ψ and V over time for all three solvers. The expected form of the atan function centered around 2.5 for the velocity is obtained with all methods and the results for ψ match too, as ψ decreases linearly before it stabilizes around zero as the velocity increases.
If adaptive timestepping methods are used, the actual size of the timestep h_n is of particular interest. Since at each timestep, the right-hand side of the ODE and the algebraic equation have to be solved a fixed number of times, a method that allows larger timesteps is considered more efficient. As shown in Figure 2.4b, this metric varies greatly with the chosen numerical solver. The simulation time can be roughly split into three sections: The first phase corresponds to low velocity and decreasing ψ until the time $t = 2.0s$. In it, the RKF45 and BDF23 methods allow for rather similarly large timestep sizes, whereas the BDF12 method is restricted to timestep sizes which are about fives times smaller. The second phase is marked by the fast increase in velocity and the stabilization of ψ. Such rapid changes in the solution should imply shorter timesteps to obtain accurate solutions, especially for RKF45, because explicit schemes face stability issues for stiff problems. Indeed, a decrease in the timestep size can be observed for all three methods, however the explicit RKF45 performs much better than expected with timesteps that are about twice as long as using BDF23. As previously, BDF12 has the worst performance of all three methods which much smaller timesteps. In the last phase, when the velocity approaches 1 and ψ remains close to 0, the two implicit BDF methods reach very large timesteps with an increasing trend, whereas the the timesteps generated by the controller RKF45 stagnate at a rather low value. Overall, BDF23 performs best with 69 timesteps, followed by RKF45 with 120 and finally BDF12 needs 230 timesteps to execute the whole simulation.



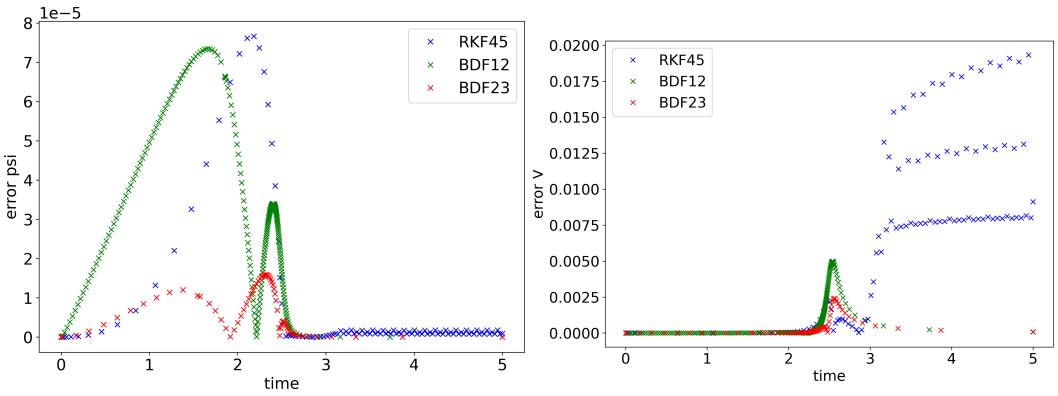
(a) Evolution of the velocity V and of the state variable ψ

(b) Evolution of the timestep size h_n

Figure 2.4: Evolution of the solution and the timestep sizes of the single state problem defined in Equation 2.2 with the implemented numerical schemes

Now, the accuracy of the three numerical methods is compared. Therefore, Figure 2.5a depicts the evolution of the absolute difference between the numerical solution and the analytical solution of the state variable ψ. It is of interest to compare this error with the absolute error in velocity, shown in Figure 2.5b. As ψ decreases, the error in ψ increases steadily for all three numerical solvers. The implicit BDF methods produce two peaks in the evolution of the error, the first shortly before the end of the decrease and the second at the stiff transition of the velocity from 0 to 1. The norm of the error is much higher for BDF12 than for BDF23, despite the lower number of timesteps of the latter. The explicit RKF45 method produces a similar error norm as BDF12, but only in one peak at the beginning of the transition phase. In the end of the simulation, when ψ stays around 0, all solvers match closely with the analytical solution and the error remains very low. On the other hand, the error in the velocity is very low for all solvers at the beginning and a peak in the error appears at the transition from 0 to 1. As usual, the highest error here appears for BDF12. Between the two remaining methods, RKF45 has the lowest error, which is astonishing for an explicit method applied to a stiff problem, especially considering the fact that in this section of the simulations, it also allows larger timesteps. Towards the end of the simulation, the

error in velocity obtained by the two implicit methods vanishes again, however RKF45 produces suddenly very high errors which alternate between three different values. In Figure 2.4a, it can be seen that the velocity oscillates around the expected solution without getting closer to it.
It is important to realise that the error in ψ and in the velocity are not directly correlated, thus a small error in ψ does not necessarily lead to a small error in the velocity. It might thus be wise to reconsider the way the timestep size is controlled. So far, it only depends on the ratio between the local truncation error and a predefined tolerance value. This error is estimated by applying another numerical scheme with higher order, by taking the difference in ψ of the two solutions as error estimate. Therefore, the velocity is not involved in the step size controller and the controller cannot ensure that the chosen timestep size guarantees sufficiently accurate results for the velocity. To ensure correct physical results, the controller needs to be extended in a way to restrict the timestep size with respect to some error estimate of the velocity.



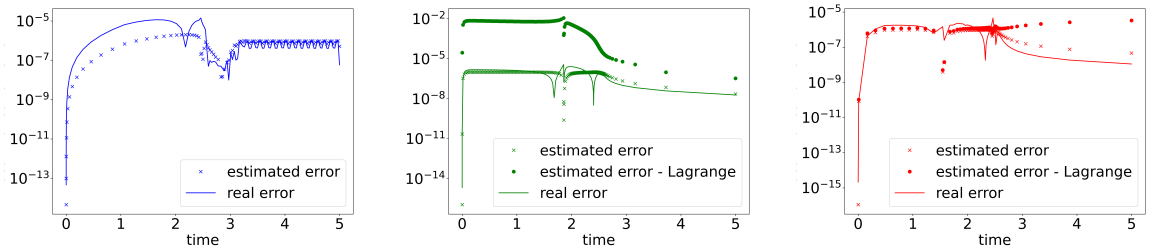
(a) Evolution of the absolute error of the state variable ψ

(b) Evolution of the absolute error of the velocity V

Figure 2.5: Evolution of the error for the single state problem defined in Equation 2.2 with the implemented numerical schemes

2.4.2 Quality of the error estimates

The whole theory of PI controller bases on an accurate error estimate, which is obtained in our case by calculating the solution with a higher order method. It is interesting to analyze whether the error estimate calculated in this way matches with the actual error to the analytical solution. The absolute error cannot be used as in the previous graphs, since the error estimate is calculated from the lower-order solution at the previous timestep. On the other hand, the local truncation error, which measures by how much the total error increases at each timestep, is much better suited to evaluate the accuracy of the error estimate.
In Figure 2.6, the difference between the two solutions calculated at each timestep by any of the schemes, being the error estimate, plotted against the real local truncation error. In the initial phase, the two implicit methods estimate the error very closely to its real value. Even though the BDF12 method approximates it better than the BDF23 method, the high amount of executed timesteps in this phase accumulate the total error which turns out to be much worse. The estimate of the explicit RKF45 method roughly follows the real evolution of the local truncation error, but underestimates it by a large factor up to 5. During the transition phase, the situation is reversed, because the implicit methods fail at estimating correctly peaks in the evolution of the real error and remain instead around a same value. On the other hand, the RKF45 follows much better the evolution of the local truncation error, which explains its good performance in the transition phase with respect to the allowed timestep size and to the total error. In the final phase, the two implicit methods match again with the expected error values and the RKF45 method seems to fit exactly the real error, but it shows nonphysical oscillations which seem to correspond to the larges oscillations in the total error of the velocity.
Overall, the implicit methods yield the better error estimates except for the stiff transition which seems to be better handled by the explicit scheme.



(a) **RKF45** (b) **BDF12** (c) **BDF23**

Figure 2.6: Evolution of the local truncation error and of the error estimate for the single state problem defined in Equation 2.2 with the implemented numerical schemes

In conclusion, the implicit BDF23 method gives the best results because overall, the induced total error remains low, it allows for the highest timestep sizes and the local truncation error is generally well estimated. In contrast, the BDF12 method is restricted to much smaller timesteps which makes it unattractive for most simulations. Another negative side effect of the small timesteps is the large difference to the analytical solution since the local errors accumulate steadily. The explicit RKF45 fails if the velocity is too high, so it should not be used in such cases. However, it has a strong potential against the BDF23 method for very small velocities because it allows for larger timesteps and in the transition phase from low to high velocities because of the better estimate of the local truncation error.

Chapter 3

Two-Dimensional SEAS model

In a second step we consider a trivial SEAS model with only two dimensions and square, symmetric tectonic plates.

3.1 Physical Description

— write down Poisson/elasticity equation for the domain —
— rate and state laws for the fault —
describe what all variables mean and stuff... Ageing law:

$$\dot{\psi} = g(\psi, V) = \frac{bV_0}{L} e^{\frac{f_0 - \psi}{b}} - \frac{V}{V_0} \tag{3.1}$$

Friction law:

$$0 = \tau(U) - a\sigma_n(U)\text{arsinh}\left(\frac{V}{2V_0} e^{\frac{\psi}{a}}\right) - \eta V \tag{3.2}$$

3.2 BP1 problem

— read more about BP1 and find good citations —
The displacement is applied orthogonal to the represented plane, thus, if the mesh is located in the X-Y plane, each element has one traction, velocity and displacement component acting in the Z direction. In this model problem, the represented tectonic plates have a symmetric layout and move in opposite direction, as one moves into the plane and the other one out of the plane. Therefore, it is enough to consider only one half of the domain, as the results in the other half will be identical, but with opposite sign. Figure 3.1 depicts the half-domain on which the solution is calculated. The fault here is located on the left side of the domain.

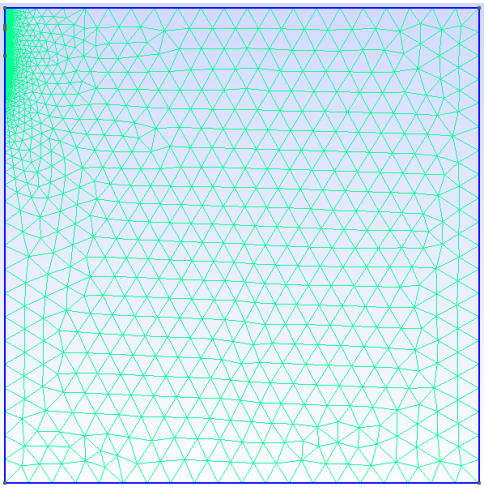


Figure 3.1: Space discretization of the BP1 problem with 200 elements on the fault

In the considered example, we choose the mesh such that there are 200 elements at the fault. The lower end of the plates is pulled with a constant velocity $V_0 = 10^{-6} m \cdot s^{-1}$.

3.3 Formulation of the Discontinous Galerkin

— write something about DG —

The numerical integration is achieved with the Gaussian quadrature. For that, Gauss-Legendre polynomials up to order p are chosen to interpolate relevant values on each element. Dependent on the chosen order p, the solution has to be calculated at different points x_i within the element and, associated with the correct weights to interpolate the data over the entire element, the integration is exact with respect to the interpolation polynomials.
— write more about GQ —

To solve the Poisson equation, it is necessary to evaluate the integral over the entire element and to handle boundary conditions and the fault, the integral over the edges of the element is needed.

3.4 First Order Differential Equation

The problem stated in section 3.1 is a first order DAE, as the involving terms only depend on the unknown quantities and at most there first time derivatives:

$$\dot{S}_i = V_i \tag{3.3}$$

$$\dot{\psi}_i = g(\psi_i, V_i) = \frac{bV_0}{L}\left(e^{\frac{f_0-\psi_i}{b}} - \frac{V_i}{V_0}\right) \tag{3.4}$$

$$0 = f_i(U, S, \psi, V) = \tau_i(U, S) - a\sigma_n \text{arsinh}\left(\frac{V_i}{2V_0} e^{\frac{\psi_i}{a}}\right) - \eta V_i \tag{3.5}$$

$$0 = AU - b(S) \tag{3.6}$$

The index $i \in [0, n]$ refers to all interpolation points of the elements located on the fault, meaning that S_i, V_i and ψ_i are respectively the slip, the velocity and the value of the state variable on the fault. The matrix-vector system in Equation 3.6 stems from the DG formulation of the Poisson problem and needs to be solved for all nodes with index $j \in [0, N]$ in the DG domain. Usually, there are much more nodes in

the full domain then purely on the fault, so $n \ll N$. The slip S_i at the fault is related to the displacement U_j by $[\![U_j]\!] = U_j^+ - U_j^- = -S_i$, where j is the index of the fault node i in the discretization of the full DG domain. The displacements U_j^+ and U_j^- correspond to the displacements on the two sides of the fault, for the current symmetric problem, they have the same magnitude with opposite signs. Outside of the fault, the term S is used to enforce boundary conditions on the domain, and typically increases with time, simulating constant tectonic movements of the plates. Nodes on the fault will first resist to this environment slip and when the forces reach a certain level, an earthquake is triggered.
To use implicit time solvers, a nonlinear equation has to be solved at each time step. For this purpose, Newton iterations are often used, however they require the Jacobian matrices of the right-hand side functions in (3.4) and (3.3). In general, if it is unknown, it is approximated along with the solution during the Newton iteration using for instance a Broyden iteration [cite Broyden]. However, this impairs the convergence speed of the Newton iteration compared to the analytical expression. It is especially relevant if a DAE formulation of the problem is chosen, for which an iterative solver is indispensable. An analytic expression of the Jacobian is provided for both the ODE and the DAE formulations of the problem, so that the Newton iterations can be solved efficiently when using implicit methods.

3.4.1 Formulation as a first order ODE

3.4.1.1 Problem

The first approach to solve the problem is to formulate the system as a classical ODE. The solution vector contains the values for the slip and for the state variable, and at each timestep, the the slip rate V is calculated from Equation 3.5 from the state with an iterative solver.

$$x = \begin{pmatrix} S \\ \psi \end{pmatrix} \qquad \dot{x} = \begin{pmatrix} \dot{S} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} V(S,\psi) \\ g(\psi, V) \end{pmatrix} = F \tag{3.7}$$

Such a formulation allows us to use many well-known and proved numerical schemes, both implicit and explicit.

3.4.1.2 Jacobian matrix

For the ODE formulation in Equation 3.7, it is not straightforward to calculate the Jacobian matrix because of the implicit evaluation of the slip rate within the right-hand side function. The matrix $\mathbf{J}_{\psi,S}(F)$ consists of four block matrices, which each contains the Jacobians related to the two quantities:

$$\mathbf{J}_{\psi_j,S_j}\left(F_i\left(\psi, V(\psi,S)\right)\right) = \begin{pmatrix} \frac{\partial V_i}{\partial S_j} & \frac{\partial V_i}{\partial \psi_j} \\ \frac{\partial g(\psi_i,V_i)}{\partial S_j} & \frac{\partial g(\psi_i,V_i)}{\partial \psi_j} \end{pmatrix} \tag{3.8}$$

In a first step, the partial derivatives of the slip rate V will be calculated. As this quantity is obtained implicitly from the friction law in Equation 3.5, the implicit function theorem has to be applied. The function $f()$ is continuously differentiable and its Jacobian matrix with respect to the slip rate $\frac{\partial f}{\partial V}$ is invertible. Indeed, from the analytic expression in Equation 3.12, it appears that the matrix is diagonal and its entries are all non-zero. All necessary conditions to apply the implicit function are fulfilled and it states that the partial derivatives of V are given by the product:

$$\frac{\partial V}{\partial S} = -\left(\frac{\partial f}{\partial V}\right)^{-1} \frac{\partial f}{\partial S} \tag{3.9}$$

$$\frac{\partial V}{\partial \psi} = -\left(\frac{\partial f}{\partial V}\right)^{-1} \frac{\partial f}{\partial \psi} \tag{3.10}$$

Two of the occurring Jacobian matrices can be easily obtained from the respective derivatives of the friction law.

$$\frac{\partial f_i(S,\psi,V)}{\partial \psi_j} = \left(-\frac{\sigma_n}{2V_0} \frac{V e^{\frac{\psi_i}{a}}}{\sqrt{e^{\frac{2\psi_i}{a}} \frac{V_i^2}{4V_0^2} + 1}} \right) \delta_{ij} \tag{3.11}$$

$$\frac{\partial f_i(S,\psi,V)}{\partial V_j} = \left(-\frac{\sigma_n a}{2V_0} \frac{e^{\frac{\psi_i}{a}}}{\sqrt{\frac{e^{\frac{2\psi_i}{a}} V_i^2}{4V_0^2} + 1}} - \eta \right) \delta_{ij} \tag{3.12}$$

Since $\frac{\partial f}{\partial V}$ is a diagonal matrix, its inverse is very easy to calculate and does not require to solve a linear system of equations. The evaluation of the last missing partial derivative is more complex to obtain and will be one reason for a partial filling of the Jacobian matrix.

$$\frac{\partial f_i}{\partial S_j} = \frac{d\tau_i(U)}{dS_j} = \frac{\partial \tau_i(U)}{\partial U_k}\frac{\partial U_k}{\partial S_j} + \frac{\partial \tau_i(U)}{\partial S_j} \tag{3.13}$$

The problem shifted to calculate the partial derivative $\frac{\partial U}{\partial S}$. Since $U = A^{-1}b(S)$ and the matrix A does not depend on S, we just need to find the derivative of $b(S)$. We get, for elements on the fault:

$$\frac{\partial b_i(S)}{\partial S_j} = \frac{\partial}{\partial S_j}\int_e \eta\{\{c_{mnkl}\epsilon_{kl}(w)\eta_n^e\}\}[\![U_m]\!] + \frac{\delta_e}{|e|^\beta}[\![U_m]\!][\![w_m]\!]dx \tag{3.14}$$

As already stated earlier, we have $[\![U_i]\!] = -S_i$, therefore we can straight eliminate this term. On all other interpolation points not located on the fault, the right hand side b does not depend on the fault displacement S and the derivatives at these points consequently vanish. We then obtain:

$$\frac{\partial b_i(S)}{\partial S_j} = -\int_e \eta\{\{c_{jnkl}\epsilon_{kl}(w)\eta_n^e\}\} + \frac{\delta_e}{|e|^\beta}[\![w_j]\!]dx \tag{3.15}$$

This expression can be calculated by plugging the unit vector e^i as argument of the right-hand side vector b. The Jacobian term $\frac{\partial U_i}{\partial S_j}$ is therefore evaluated by applying the solver method of the Poisson problem to the unit vectors as slips. We get:

$$\frac{\partial U_i}{\partial S_j} = A^{-1}_{ik} b_k(e^i) \tag{3.16}$$

The traction term $\tau(U,S)$ is calculated as $\tau = \mu\frac{\partial u}{\partial x_i}n_i$, and is numerically approximated on the nodal basis as $\tau_p = M^{-1}_{rp}e^T_{qr}w_q(\nabla u)_{kq}n_{kq}$, where M is the mass matrix of the fault basis, e^T maps from fault to quadrature points, w are the quadrature weights and n is the normal at the quadrature points. The gradient of u is approximated by $(\nabla u)_{pq} = \frac{1}{2}\left(D^0_{lpq}u^0_l + D^1_{lpq}u^1_l\right) + c_0\left(E^0_{lq}u^0_l - E^1_{lq}u^1_l - f_q\right)n_{pq}$. The tensor E maps from the quadrature points to the element basis and D is its gradient and the superscripts 0 and 1 refer to the adjacent elements on opposite sides of the fault. The term f_q denotes the actual slip transformed to the quadrature points. The evaluation of the derivative of the traction with respect to the displacement requires to derivate the gradient of the displacement with respect to itself. We obtain:

$$\frac{(\nabla u)_{pq}}{\partial u_k} = \frac{1}{2}\left(D^0_{lpq}\frac{\partial u^0_l}{\partial u_k} + D^1_{lpq}\frac{\partial u^1_l}{\partial u_k}\right) + c_0\left(E^0_{lq}\frac{\partial u^0_l}{\partial u_k} - E^1_{lq}\frac{\partial u^1_l}{\partial u_k}\right)n_{pq} \tag{3.17}$$

$$= \frac{1}{2}\left(D^0_{lpq}\delta^0_{lk} + D^1_{lpq}\delta^1_{lk}\right) + c_0\left(E^0_{lq}\delta^0_{lk} - E^1_{lq}\delta^1_{lk}\right)n_{pq} \tag{3.18}$$

$$= \frac{1}{2}\left(D^0_{kpq} + D^1_{kpq}\right) + c_0\left(E^0_{kq} - E^1_{kq}\right)n_{pq} \tag{3.19}$$

And further we get:

$$\frac{\partial \tau_p}{\partial u_l} = M^{-1}_{rp}e^T_{qr}w_q\frac{(\nabla u)_{kq}}{\partial u_l}n_{kq} \tag{3.20}$$

In addition, the traction needs to be derivated with respect to the current slip directly for the term $\frac{\partial \tau_i(U)}{\partial S_j}$, which is contained in the term $f_q = e_{qj}S_j$. It appears again in the gradient, so its derivative with respect to the slip is also needed.

$$\frac{(\nabla u)_{pq}}{\partial S_j} = -c_0\frac{\partial f_q}{\partial S_j}n_{pq} = -c_0 e^T_{qk}\frac{\partial S_k}{\partial S_j}n_{pq} = -c_0 e^T_{qj}n_{pq} \tag{3.21}$$

$$\tag{3.22}$$

With this expression, we obtain:

$$\frac{\partial \tau_p}{\partial S_j} = -M^{-1}_{rp}e^T_{qr}w_q c_0 e^T_{qj}n_{kq}n_{kq} \tag{3.23}$$

This derivative term does not depend on the current slip anymore but only on the geometry of the discretization, so it can be calculated once at the beginning of the simulation. All components of the remaining partial derivative in Equation 3.13 are now available and the Jacobian matrices of the slip rate with respect to the slip and to the state variable can be calculated.
To express the full Jacobian matrix of the ODE in Equation 3.8, the partial derivatives of the ageing law $g(\psi, V)$ are still missing. They can be expressed in term of the already known partial derivatives:

$$\frac{\partial g_i}{\partial S_j} = -\frac{b}{L}\frac{\partial V}{\partial S} \tag{3.24}$$

$$\frac{\partial g_i}{\partial S_j} = -\frac{V_0}{L}e^{\frac{f_0-\psi}{b}}\delta_{ij} - \frac{b}{L}\frac{\partial V}{\partial \psi} \tag{3.25}$$

3.4.2 Formulation as a DAE

3.4.2.1 Problem

The ODE form looks pretty and like an easy-to-tackle numerical problem, but may run into some trouble because of the implicit evaluation of the slip rate, which is treated independently of the numerical integration. Alternatively, the friction law could be solved for the slip rate together with the time-variant solution components. The solution vector is then extended by one quantity, and the evaluation of the slip rate with a separate iterative does not occur anymore.

$$x = \begin{pmatrix} S \\ \psi \\ V \end{pmatrix} \qquad \dot{x} = \begin{pmatrix} \dot{S} \\ \dot{\psi} \\ 0 \end{pmatrix} = \begin{pmatrix} V \\ g(\psi, V) \\ f(S, \psi, V) \end{pmatrix} \tag{3.26}$$

This system cannot be used anymore with pure explicit solvers, since no time discretization can be set up for the algebraic equation. It is however well-suited for implicit methods, which iteratively solve for the time-variant quantities S and ψ as well for the algebraic state V. A notable advantage of this formulation is that the tolerances of the numerical solver can be specified independently for S, ψ and V, whereas in the ODE formulation of the problem, the error control is limited to the parameters S and ψ.
The PETSc interface requires us to provide a DAE in the form

$$F(t, x, \dot{x}) = G(t, x) \tag{3.27}$$

where the left-hand side function $F()$ depends on the solution vector and its time derivative whereas the right-hand side function $G()$ only depends on the solution vector. For now, we choose a fully implicit formulation of the problem, in which $G()$ vanishes. The DAE can then be expressed in the following way:

$$x = \begin{pmatrix} S \\ \psi \\ V \end{pmatrix} \qquad F(t, x, \dot{x}) = \begin{pmatrix} V - \dot{S} \\ g(\psi, V) - \dot{\psi} \\ f(S, \psi, V) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = G(t, x) \tag{3.28}$$

Another class of solvers, the so called IMEX (implicit-explicit) solvers [cite IMEX], offer an interesting approach to numerically solve this problem, by mixing an implicit, iterative method for the algebraic states with an explicit method for the remaining simple time-dependent quantities. Technically, the function $F()$ is expected to be integrated implicitly, while the right-hand side function $G()$ is solved explicitly. In the given formulation, the right hand side is equal to 0, this means that the system has to be solved fully implicitly and the advantages of IMEX methods cannot be used here.

3.4.2.2 Jacobian matrix

For the DAE formulation of the system in Equation 3.28, another approach can be chosen to calculate the Jacobian matrix of the system. With now three different components in the solution vector, the Jacobian matrix contains the partial derivatives with respect to the slip rate in addition to the slip and to the

state variable. Similarly, the function F on which the Jacobian is applied is extended by the friction law. Each block i, j in the structure of the Jacobian is given by:

$$\mathbf{J}_{S_j,\psi_j,V_j} F_i(S,\psi,V) = \begin{pmatrix} \frac{\partial V_i}{\partial S_j} & \frac{\partial V_i}{\partial \psi_j} & \frac{\partial V_i}{\partial V_j} \\ \frac{\partial g_i(\psi,V)}{\partial S_j} & \frac{\partial g_i(\psi,V)}{\partial \psi_j} & \frac{g_i(\partial \psi,V)}{\partial V_j} \\ \frac{\partial f_i(S,\psi,V)}{\partial S_j} & \frac{\partial f_i(S,\psi,V)}{\partial \psi_j} & \frac{\partial f_i(S,\psi,V)}{\partial V_j} \end{pmatrix} \tag{3.29}$$

In the previous section, a complicated approach had to be followed to obtain the derivative for the slip rate because of the nonlinear friction law, which was indirectly solved for each evaluation of the slip rate. Since the slip rate is now obtained directly from the solution vector x and not from an iterative solver for the friction law, all its partial derivatives except with each component itself vanish.

$$\frac{\partial V_i}{\partial S_j} = 0 \qquad \frac{\partial V_i}{\partial \psi_j} = 0 \qquad \frac{\partial V_i}{\partial V_j} = \delta_{ij} \tag{3.30}$$

Similarly, the partial derivatives of the ageing law can be directly evaluated from its definition in Equation 3.4.

$$\frac{\partial g_i(\psi,V)}{\partial S_j} = 0 \qquad \frac{\partial g_i(\psi,V)}{\partial \psi_j} = -\frac{V_0}{L} e^{\frac{f_0-\psi_i}{b}} \delta_{ij} \qquad \frac{\partial g_i(\psi,V)}{\partial V_j} = -\frac{b}{L}\delta_{ij} \tag{3.31}$$

For the friction law, the Jacobian matrices have already been calculated in Equation 3.11 for $\frac{\partial f(S,\psi,V)}{\partial psi}$, in Equation 3.13 for $\frac{\partial f(S,\psi,V)}{\partial S}$ and in Equation 3.12 for $\frac{\partial f(S,\psi,V)}{\partial V}$.

This formulation of the Jacobian cannot be directly used in the Newton iteration to calculate the next timestep. Typically, the Jacobian has to be adapted to represent the time-stepping scheme. For instance, the implicit Euler method results requires to solve the nonlinear equation $0 = 1/h(x_n - x_{n+1}) + F(x_{n+1})$ for the solution at the next timestep x_{n+1} and the Jacobian matrix J_N for the corresponding Newton iteration is constructed from J.

$$J_N = -\frac{1}{h} I + J \tag{3.32}$$

Here, the problem arises that the evolution over time is only defined for two components in the solution vector: the slip S and the state variable ψ. The slip rate V needs to be solved algebraically at every Newton iteration, but this component should not be included in the time-stepping scheme. The nonlinear equation to calculate a step of the implicit Euler has to be adapted to include properly the slip rate.

$$0 = \begin{pmatrix} \frac{1}{h} S_n \\ \frac{1}{h} \psi_n \\ 0 \end{pmatrix} - \begin{pmatrix} \frac{1}{h} S_{n+1} \\ \frac{1}{h} \psi_{n+1} \\ 0 \end{pmatrix} + \begin{pmatrix} V_{n+1} \\ g(\psi_{n+1}, V_{n+1}) \\ f(S_{n+1}, \psi_{n+1}, V_{n+1}) \end{pmatrix} \tag{3.33}$$

The Jacobian matrix to be used in a Newton iteration to solve this problem has to be changed accordingly.

$$\mathbf{J}_N = \begin{pmatrix} -\frac{1}{h}\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{h}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{J}_S(S) & \mathbf{J}_\psi(S) & \mathbf{J}_V(S) \\ \mathbf{J}_S(\psi) & \mathbf{J}_\psi(\psi) & \mathbf{J}_V(\psi) \\ \mathbf{J}_S(V) & \mathbf{J}_\psi(V) & \mathbf{J}_V(V) \end{pmatrix} = \begin{pmatrix} -\frac{1}{h}\mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\frac{1}{h}\mathbf{I} + \frac{\partial g}{\partial \psi} & \frac{\partial g}{\partial V} \\ \frac{\partial f}{\partial S} & \frac{\partial f}{\partial \psi} & \frac{\partial f}{\partial V} \end{pmatrix} \tag{3.34}$$

If other implicit numerical schemes are used than the implicit Euler, such as higher order BDF methods, $\mathbf{J}_N$ can be easily changed by multiplying the identity matrices $\mathbf{I}$ with a coefficient that depends on the chosen method. Albeit this formulation of the Jacobian matrix is perfectly valid, it is very specific to the SEAS problem and is therefore hard to implement as such within the PETSc framework.
To solve general implicit DAEs of the form $F(t, x, \dot{x}) = G(t, x)$, PETSc offers a convenient interface to provide the Jacobian matrices. In fact, three matrices have to be provided. The first Jacobian matrix, denoted by $\mathbf{F}_{\dot{x}}$, corresponds to the variation of the left-hand side function $F()$ with respect to the derivative of the solution vector. The second matrix $\mathbf{F}_x$ describes the variation of the same function under the actual solution vector. Finally, the third Jacobian matrix $\mathbf{G}_x$ represents the variation of the right hand side function $G()$ under the influence of x. In our case, the formulation is fully explicit and

the right hand side function $G(t, x)$ is zero everywhere, and by extension, its Jacobian matrix $\mathbf{G}_x$ also vanishes. It remains to express the Jacobian matrix of the Newton iteration $\mathbf{J}_N$ in terms of $\mathbf{F}_x$ and $\mathbf{F}_{\dot{x}}$:

$$\mathbf{J}_N = \frac{dF}{dx_n} = \mathbf{F}_{\dot{x}} \frac{d\dot{x}_n}{dx_n} + \mathbf{F}_x \tag{3.35}$$

The term $\frac{d\dot{x}_n}{dx_n}$ is a constant scalar and depends only on the chosen time step scheme, which can be denoted by σ. For the implicit Euler method for instance, we have $\sigma = 1/h$. The Jacobian matrix $\mathbf{F}_x$ is exactly the one expressed in Equation 3.29 and $\mathbf{F}_x$ contains negative identity matrices at the entries for the slip S and for the state variable ψ.

$$\mathbf{F}_x = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \frac{\partial g}{\partial \psi} & \frac{\partial g}{\partial V} \\ \frac{\partial f}{\partial S} & \frac{\partial f}{\partial \psi} & \frac{\partial f}{\partial V} \end{pmatrix} \qquad \mathbf{F}_{\dot{x}} = \begin{pmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{3.36}$$

With this too matrices, the Jacobian matrix for the Newton iteration of the implicit Euler method is exactly the same as calculated in Equation 3.34.

3.4.3 Compact formulation as a DAE

3.4.3.1 Problem

It is important to remember that the slip rate is the time derivative of the slip, so the variables $\dot{S}$ and V are equal and can be interchanged in the problem formulation. As such, the friction law can be solved directly for $\dot{S}$ and the term V is not required anymore. The solution vector then remains with the components of the slip S and of the state variable ψ, as previously in the ODE formulation.

$$x = \begin{pmatrix} S \\ \psi \end{pmatrix} \qquad F(t, x, \dot{x}) = \begin{pmatrix} f(S, \psi, V) \\ g(\psi, V) - \dot{\psi} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = G(t, x) \tag{3.37}$$

As in the developed DAE formulation, the friction law is solved along with time-variant quantities S and ψ. It therefore requires an iterative solver method from an implicit time solver and it can be used only with an implicit integrator.

3.4.3.2 Jacobian matrix

The compact DAE formulation of the problem in Equation 3.37 needs again to be solved fully implicitly and involves again the three Jacobian matrices, $\mathbf{F}_{\dot{x}}$, $\mathbf{F}_x$ and $\mathbf{G}_x$, where the last one also vanishes because the right-hand side function is zero. The block matrices that appear in this formulation are calculated in exactly the same way as previously but are arranged in a different manner.

$$\mathbf{F}_{\dot{x}} = \begin{pmatrix} \frac{\partial f}{\partial V} & \mathbf{0} \\ \frac{\partial g}{\partial V} & -\mathbf{I} \end{pmatrix} \qquad \mathbf{F}_x = \begin{pmatrix} \frac{\partial f}{\partial S} & \frac{\partial f}{\partial \psi} \\ \mathbf{0} & \frac{\partial g}{\partial \psi} \end{pmatrix} \qquad \mathbf{G}_x = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{3.38}$$

3.4.4 Verification of the Jacobian matrices

All Jacobian matrices defined in this section have been checked for accuracy. For that, a second order central finite difference approximation [Olv14, p. 184] is used to approximate each term of the Jacobian.

$$\mathbf{J}_{ij} = \frac{\partial F_i}{\partial x_j} \approx \frac{F_i(x + he_j) - F_i(x - he_j)}{2h} \tag{3.39}$$

The function F has to be evaluated $2n$ times with the current solution vector x and a variation h applied in positive and negative direction to each component of x respectively. For this, the function to

evaluate the right-hand side of the ODE or respectively the left-hand side for the DAEs is called. For an appropriate choice of h, not too large to be accurate and not too small to avoid truncation errors, the relative error between the analytic expression of the Jacobian and its approximation is inferior to 10^{-5} in all components and therefore, the expressions for the Jacobian matrices can be considered to be exact.

3.4.5 Practical limitations of the Jacobian matrices

The main advantage of implicit methods over explicit methods is to allow larger time steps without loss in accuracy. The program execution is therewith expected to be accelerated. The proposed calculation of the Jacobian matrix and its application in Newton iterations comes along with some drawbacks with respect to the required calculation effort.
As described in subsubsection 3.4.1.2, the computation of the Jacobian matrix can be split up in one constant part $\frac{\partial \tau_p}{\partial S_l}$ that has to be evaluated once at the beginning of the simulation and only depends on the domain geometry and in one variable part, that needs to be updated after each evaluation of the solution vector x_n. The initialization can be quite computationally expensive, at it requires, for each column of $\frac{\partial \tau_p}{\partial S_l}$, to solve the Discontinuous Galerkin system for the entire domain with the corresponding unit vector as right-hand side vector. Each calculation of this linear system is approximately as expensive as performing one time iteration with an explicit solver, since solving the same linear system is the most expensive operation of a time iteration. For large domains with several hundreds or thousands of fault elements, which each contains a couple of fault nodes, this initialization operation takes a considerable amount of time. In comparison with an explicit scheme, a potential implicit method starts the race for the fastest solver with a penalty of about n explicit time iterations, where n is the total number of nodes on the fault. The advantages of implicit methods will therefore only pay off after a high number of iterations and make such methods of interest only if the simulated time frame is very long.
Another potential drawback of the Newton method is that a linear system of size n needs to be solved at each iteration step. This might still sound much less than solving for the displacement on the whole DG-domain of size N as it is required to evaluate the right-hand side of the ODE at each iteration too. However, the DG system has a constant matrix A whose LU-decomposition can be calculated once at the beginning of the simulation and each evaluation of the right-hand side only requires a backward substitution of complexity $\mathcal{O}\left(N^2\right)$. For the Jacobi matrix, such a pre.processing is not available since it changes along with the solution vector. Every calculation to solve the linear system in the Newton iteration is achieved with a complexity of the order $\mathcal{O}\left(n^3\right)$. In particular in large domains with many fault nodes the solver with cubic complexity may involve a substantially higher execution time than the one with quadratic complexity.

3.4.6 Iterative solvers for the Jacobian matrices

As discussed in the previous section, a linear system of equations has to be solved at each Newton iteration, and since the Jacobian matrices depend on the current solution vector, an expensive LU decomposition in $\mathcal{O}\left(n^3\right)$ has to be calculated at each step. For the two DAE formulations, the system of equations can however be reformulated in a way that iterative methods can be applied effectively, reducing the complexity of each Newton step to $\mathcal{O}\left(kn^2\right)$, where k is the number of iterations in the matrix solver. We use the fact that among all submatrices, that are used to set-up the Jacobian matrix, the only dense matrix is the constant term $\frac{\partial f}{\partial S}$, and all other terms, are diagonal matrices. Let us consider the example of the Jacobian of the compact DAE formulation in Equation 3.38, with the linear system in which it appears in the Newton iteration. The dense term is denoted by the matrix $\mathbf{B}$ and all other block matrices J_{ij} are diagonal.

$$\begin{pmatrix} \mathbf{B} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{3.40}$$

The vector x describes the change in each component at the current Newton step and is the searched unknown and b contains the left-hand side of the DAE system. Under the assumption that there are no zero entries on the diagonals, we first transform the Jacobian matrix to a lower triangular block matrix and obtain the following modified system:

$$\begin{pmatrix} \mathbf{B} - \mathbf{J}_{12}\mathbf{J}_{22}^{-1}\mathbf{J}_{21} & \mathbf{0} \\ \mathbf{J}_{12} & \mathbf{J}_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 - \mathbf{J}_{12}\mathbf{J}_{22}^{-1}b_2 \\ b_2 \end{pmatrix} \tag{3.41}$$

Now, an iterative solver can be applied on the subsystem in the first line on the dense, but diagonally dominant to solve for the components in x_1. The components x_2 can then be easily derived in $\mathcal{O}(n)$ operations with a forward substitution.
A similar transformation was obtained for the extended DAE formulation too, where the iterative solver is only required on a subsystem of size n and where the missing components can be updated linearly in time. Without the reduction, no iterative methods that converges fast could be found. For the first order ODE formulation, such a Gaussian elimination is not expedient because two out of the four submatrices are dense, however iterative methods can directly be used on the entire Jacobian matrix.

3.5 Second Order Differential Equation

So far, the original problem in section 3.1 has been formulated as a first order ODE in Equation 3.7, in which the friction law is solved implicitly within each evaluation of the right-hand side, such that explicit time integrators can be used. The solution vector has a compact size and only includes the slip S and the state variable ψ. Alternatively, the problem can be directly formulated as a DAE, where the solution vector is extended by the slip rate V, such that it can be solved through the friction law iteratively together with the time quantities S and ψ. A compact form of the DAE has also been formulated, where the derivative slip rate is directly used in the friction law instead of introducing a new variable V.
The original motivation was to define an extended ODE formulation as counterpart to the extended DAE form, such that the solution vector contains an additional component for the slip rate V. To achieve this goal, it is necessary to find a function $h(t, S, \psi, V)$ which describes the change in time of the slip rate $\frac{dV}{dt}$. Since V itself is the time derivative of the slip, this function $h()$ corresponds to the second time derivative of the slip, or the slip acceleration. The new ODE formulation is:

$$x = \begin{pmatrix} S \\ \psi \end{pmatrix} \qquad \begin{pmatrix} \ddot{S} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} h(t, S, \psi, \dot{S}) \\ g(\psi, V) \end{pmatrix} \tag{3.42}$$

$$\Leftrightarrow \qquad x = \begin{pmatrix} S \\ \psi \\ V \end{pmatrix} \qquad \dot{x} = \begin{pmatrix} \dot{S} \\ \dot{\psi} \\ \dot{V} \end{pmatrix} = \begin{pmatrix} V \\ g(\psi, V) \\ h(t, S, \psi, V) \end{pmatrix} = F \tag{3.43}$$

Here, the friction law does not appear in the right-hand side anymore and is implicitly included in the function $h()$. Unlike the previous ODE formulation, it does not require anymore to solve the friction law implicitly at every time step for the slip rate V (or any other quantity), therefore, the new formulation is in some sense a "true" ODE and not a hidden DAE anymore. Another very beneficial advantage is, that if the boundary conditions of the DG domain are linear in time, it does not require to solve the DG system at every time step anymore, which is the main performance bottleneck after initialization in the first order formulations.

3.5.1 Derivation of the second order ODE

This section will focus on the derivation of the right-hand side function $h(t, S, \psi, V)$, which is necessary to formulate the second order ODE. Starting point is the expression for the friction law in Equation 3.5. At any time in the simulation, this algebraic equation always evaluates to zero, and therefore, its time derivative is also equal to 0. With the help of the formula of total derivatives, we obtain following expansions:

$$0 = \frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S}\frac{\mathrm{d}S}{\mathrm{d}t} + \frac{\partial f}{\partial \psi}\frac{\mathrm{d}\psi}{\mathrm{d}t} + \frac{\partial f}{\partial V}\frac{\mathrm{d}V}{\mathrm{d}t} \tag{3.44}$$

$$= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S}V + \frac{\partial f}{\partial \psi}g(\psi, V) + \frac{\partial f}{\partial V}h(t, S, \psi, V) \tag{3.45}$$

The partial derivatives of f that occur are calculated in the same way as for the Jacobian matrices of the first order formulations. The terms $\frac{\partial f}{\partial \psi}$ and $\frac{\partial f}{\partial V}$ are diagonal matrices with the entries stated in Equation 3.11 and Equation 3.12 and the term $\frac{\partial f}{\partial S}$ is a constant, full matrix which depends on the geometry

of the domain and is derived in Equation 3.13. None of these terms depends on the slip S anymore, so $h()$ can be calculated independently from S. After some transformations, it can be expressed as:

$$h(t, \psi, V) = -\left(\frac{\partial f}{\partial V}\right)^{-1} \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial S} V + \frac{\partial f}{\partial \psi} g(\psi, V)\right) \tag{3.46}$$

It can be noted that this expression is very similar to the Jacobian matrix of the first order ODE formulation. If $x = (S, \psi)^T$ stands for its solution vector and $\mathbf{J}_S$ denotes the first row of block matrices of its Jacobian matrix in Equation 3.8, an alternative definition of $h()$ could be:

$$h(t, \psi, V) = \mathbf{J}_S \frac{\mathrm{d}x}{\mathrm{d}t} - \left(\frac{\partial f}{\partial V}\right)^{-1} \frac{\partial f}{\partial t} \tag{3.47}$$

The last missing term is the partial derivative of the friction law f with respect to the time variable t. In its definition, the only component that directly depends on time is the traction τ through the boundary conditions of the DG domain. Typically, at boundary nodes which are not on the seismic fault, the slip S_b there is prescribed and evolves with time to represent the natural movement of the tectonic plate. For a constant environment slip rate V_p, we calculate $S_b = V_p t$ for the general case or $S_b = V_p/2\ t$ if the domain is symmetric. The boundary slip is linear in time, so its first derivative $\dot{S}_b$ will be constant. In the DG formulation, boundary conditions are included in the term f_q of **??**, and are added to the traction τ on the fault only after linear transformations. To obtain $\frac{\partial \tau}{\partial t}$ on the fault, it is then sufficient to solve the Poisson problem with only $\dot{S}_b$. Since this term is constant, $\frac{\partial f}{\partial t}$ is also constant it is enough to solve the Poisson problem once at the beginning of the simulation and then, at each evaluation of the right-hand side of the ODE, use the same term $\frac{\partial f}{\partial t}$. To reuse existent code structures, $\frac{\partial \tau}{\partial t}$ can be pre-calculated with the DG solver for τ using a slip of 0 everywhere on the fault and by evaluating the boundary condition at the time $t = 1$.

In comparison with the first order ODE formulation, the new second order formulation needs a considerable initialization phase, but evaluates the right-hand side of the ODE much faster. In addition of the LU-decomposition of the DG matrix A, the initialization of the Jacobian $\frac{\partial f}{\partial S}$ needs to solve the Poisson problem on the DG domain of size N once per fault node, so in total n times. Moreover, the problem needs to be solved once to calculate the initial condition of V from the initial S and ψ and once to initialize the time derivative of the friction law $\frac{\partial f}{\partial t}$. At each evaluation of the right-hand side of the first order ODE, the DG system needs to be solved with its LU-decomposition in $\mathcal{O}\left(N^2\right)$ and the remaining components of the vector set up in $\mathcal{O}\left(n^2\right)$ because of the matrix-vector product in τ, so the total complexity is $\mathcal{O}\left(n^2 + N^2\right)$. In contrast, the second order formulation only needs matrix-vector multiplications on the fault node, so its total complexity per evaluation is reduced to $\mathcal{O}\left(n^2\right)$. Since $n \ll N$, it is expected that the new formulation performs much better for long simulation times.

However, if the boundary conditions are not linear in time, their derivation is not constant in time anymore and needs to be evaluated at every time step, and therefore the solution of the Poisson problem is needed again at every time step. The complexity increases back to $\mathcal{O}\left(n^2 + N^2\right)$ and the main advantage of the second order formulation is lost. For now, we do not consider this case and always assume that the boundary conditions are linear in time.

Figure 3.2 shows the maximum slip rate that occurs throughout the simulation time for both ODE formulations. Both curves overlap well, at least up to 1000 years of simulation time, so it proves that the new approach leads to the same results. This graph also indicates at which times earthquakes happen: after two initial earthquakes, a regular pattern of three close earthquakes alternates with one singular earthquake every 90 years approximately.

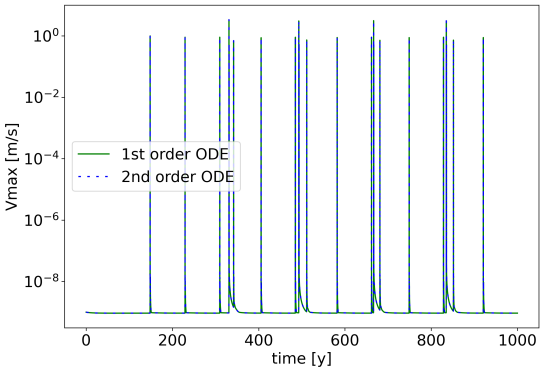


Figure 3.2: Maximum slip rate in the 1st and 2nd order formulations on a domain with 5 fault elements for 1000 years

3.5.2 Derivation of the analytic Jacobian

The Jacobian of the second order ODE formulation in Equation 3.43 has components in the slip S, the state variable ψ and in the slip rate V. The block structure is given by:

$$\mathbf{J}_{S_j,\psi_j,V_j} F_i(S,\psi,V) = \begin{pmatrix} \frac{\partial V_i}{\partial S_j} & \frac{\partial V_i}{\partial \psi_j} & \frac{\partial V_i}{\partial V_j} \\ \frac{\partial g_i(\psi,V)}{\partial S_j} & \frac{\partial g_i(\psi,V)}{\partial \psi_j} & \frac{\partial g_i(\psi,V)}{\partial V_j} \\ \frac{\partial h_i(t,\psi,V)}{\partial S_j} & \frac{\partial h_i(t,\psi,V)}{\partial \psi_j} & \frac{\partial h_i(t,\psi,V)}{\partial V_j} \end{pmatrix} \tag{3.48}$$

It can be directly seen that none of the terms V, g and h depend on the slip S, therefore all entries in the first column of block matrices evaluate to 0. A direct consequence is the singularity of the Jacobian matrix, which makes it unsuitable for the Newton iteration. Under consideration of all 0-entries, the matrix takes the form, where $\mathbf{K}$ is invertible with a partial diagonal structure:

$$\mathbf{J}_{S_j,\psi_j,V_j} F_i(S,\psi,V) = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & & \mathbf{K} \\ \mathbf{0} & & \end{pmatrix} \tag{3.49}$$

To cope with this issue, recall that g and h do not depend on S, it is thus possible to apply the Newton iteration only on ψ and V with the reduced Jacobian $\mathbf{K}$. Once the slip rate $V^{(n+1)}$ is known for the next timestep, the implicit scheme can be directly applied to S without the need to solve a nonlinear system of equations. Indeed, the value of the right-hand side of the ODE at the next timestep for $\dot{S^{(n+1)}} = V^{(n+1)}$ is directly available and can be plugged in the BDF scheme. If $S^{(n-i)}$ are the solutions at the i previous timesteps, α_{n-i} the time adaptive BDF coefficients at these steps and σ the current shift, the new slip rate $S^{(n+1)}$ can be calculated as:

$$S^{(n+1)} = \frac{1}{\sigma}\left(V^{(n+1)} - \sum_{i=0}^{k} \alpha_{n-i} S^{(n-i)}\right) \tag{3.50}$$

For the remaining unknown Jacobian $\mathbf{K}$, the entries in the first line are identical to the partial derivatives $\frac{\partial g}{\partial \psi}$ and $\frac{\partial g}{\partial V}$ Equation 3.29 and are trivial to compute. The entries in the second line are more challenging to obtain as they describe the variation of the product and sum of other Jacobian matrices.

We start from the expression for h in Equation 3.46 and introduce the new variables $C = \frac{\partial f}{\partial t}$, a constant vector from the partial time derivative, $\mathbf{A} = \frac{\partial f}{\partial S}$, a constant and dense matrix from the DG discretization defined in Equation 3.13, and $\xi(\psi,V) = \left(\frac{\partial f}{\partial V}\right)^{-1}$ and $\zeta(\psi,V) = \frac{\partial f}{\partial \psi}$, two diagonal matrices which depend on the state variable and the slip rate. With these new variables, the components of h can be expressed as:

$$h_i = -\xi_{ii}\left(C_i + \mathbf{A}_{ik} V_k + \zeta_{ii} g_i\right) \tag{3.51}$$

The first partial derivative $\frac{\partial h_i}{\partial S_j}$ can be directly evaluated to 0 since none of the terms depends on the slip S. Next, the derivative with respect to the slip rate is already more complex:

$$\frac{\partial h_i}{\partial \psi_j} = -\left[\frac{\partial \xi_{ii}}{\partial \psi_j}\left(C_i + \mathbf{A}_{ik}V_k + \zeta_{ii}g_i\right) + \xi_{ii}\left(\frac{\partial \zeta_{ii}}{\partial \psi_j}g_i + \zeta_{ii}\frac{\partial g_i}{\partial \psi_j}\right)\right]\delta_{ij} \tag{3.52}$$

The derivatives of ξ and ζ can be directly computed from Equations 3.11 and 3.12:

$$\frac{\partial \xi_{ii}}{\partial \psi_j} = \frac{2V_0\sigma_n e^{\frac{\psi_i}{a}}}{\sqrt{\frac{e^{\frac{2\psi_i}{a}}V_i^2}{4V_0^2}+1}\left(2V_0\eta\sqrt{\frac{e^{\frac{2\psi_i}{a}}V_i^2}{4V_0^2}+1}+a\sigma_n e^{\frac{\psi_i}{a}}\right)^2}\delta_{ij} \tag{3.53}$$

$$\frac{\partial \zeta_{ii}}{\partial \psi_j} = -\frac{\sigma_n V_i e^{\frac{\psi_i}{a}}}{2aV_0\left(\frac{e^{\frac{2\psi_i}{a}}V^2}{4V_0^2}+1\right)^{\frac{3}{2}}}\delta_{ij} \tag{3.54}$$

The last derivative with respect to the slip rate V is not a diagonal matrix anymore:

$$\frac{\partial h_i}{\partial V_j} = -\left[\frac{\partial \xi_{ii}}{\partial V_j}\left(C_i + \mathbf{A}_{ik}V_k + \zeta_{ii}g_i\right)\delta_{ij} + \xi_{ii}\left(\mathbf{A}_{ij} + \frac{\partial \zeta_{ii}}{\partial V_j}g_i\delta_{ij} + \zeta_{ii}\frac{\partial g_i}{\partial V_j}\delta_{ij}\right)\right] \tag{3.55}$$

Again, the unknown derivatives can be calculated:

$$\frac{\partial \xi_{ii}}{\partial V_j} = -\frac{a\sigma_n V_i e^{\frac{3\psi_i}{a}}}{2V_0\sqrt{\frac{e^{\frac{2\psi_i}{a}}V_i^2}{4V_0^2}+1}\left(2V_0\eta\sqrt{\frac{e^{\frac{2\psi_i}{a}}V_i^2}{4V_0^2}+1}+a\sigma_n e^{\frac{\psi_i}{a}}\right)^2}\delta_{ij} \tag{3.56}$$

$$\frac{\partial \zeta_{ii}}{\partial V_j} = -\frac{\sigma_n e^{\frac{\psi_i}{a}}}{2V_0\left(\frac{e^{\frac{2\psi_i}{a}}V_i^2}{4V_0^2}+1\right)^{\frac{3}{2}}}\delta_{ij} \tag{3.57}$$

By now, all components to set up the Jacobian matrix in Equation 3.48 are provided and the formulation is ready to be used with implicit methods.
In the implementation, the singularity is handled slightly differently: solving the Newton iteration with the matrix $\mathbf{K}$ in Equation 3.49 would require to copy the entries for V and ψ from the solution vector to another vector whose size matches the dimensions of $\mathbf{K}$. Instead, $\mathbf{K}$ is extended with ones on the diagonal elements related to the S, and in each Newton step, just before solving the linear system, the corresponding entries in the right-hand side vector are set to zero. As such, the change in S always vanishes and the convergence of the Newton iteration is not altered. After a successful termination, the entries for S are updated as described in Equation 3.50.

3.6 Numerical Aspects and Implementation Details

So far, 4 different methods have been proposed, which can be solved explicitly or implicitly. An overview of the available is given in Table 3.1, which in addition states the length of the solution vector. Methods with a size $2n$ contain the slip S and the state variable ψ in the solution vector, and methods with $3n$ contain in addition the slip rate V.

	explicit	implicit	Vector length
1st order ODE	yes	yes	$2n$
extended DAE	no	yes	$3n$
compact DAE	no	yes	$2n$
2nd order ODE	yes	yes	$3n$

Table 3.1: Overview of the available methods

This section provides more details about the necessary components to solve each of these methods numerically. The focus is first on adaptive time integration methods followed by a discussion about the proper definition of tolerances. Next, the Newton iteration and its convergence properties is described to solve the nonlinear systems which arise in the implicit problem formulations. Finally, the data layout is described along with a quick estimate of the memory requirements for each problem formulation. In addition, the section exposes the peculiarities and possible limitations of the respective methods.

3.6.1 Time integration

The PETSc environment is used to handle most of the time integration. Among all available integrator, the new code is only compatible with Runge-Kutta methods for explicit formulations and BDF methods for implicit formulations.
For the explicit methods, the 2nd order Bogacki-Shampine scheme with a 3rd order error estimate and the 4th order Dormand-Prince scheme with a 5th order error estimate will be compared later in the results section. Their respective Butcher tableaus are given in /:TODO////—APPENDIX—////.
For now, only the basic PETSc timestep adapter is used. The next timestep size is calculated in function of the estimation of the local truncation error ϵ and of the order of the estimate $\bar{k}$. In general, $\bar{k} = k + 1$, where k is the order of the numerical scheme.

$$h_{n+1} = C\epsilon^{-1/\bar{k}} h_n \tag{3.58}$$

The safety factor C is set to 0.9, and the scaling from one timestep to another is restricted to values between 0.1 and 10. In the native PETSc implementation for BDF, this upper bound for the factor is set to 2, but this only matters for the very first timesteps, since the relative change between timestep sizes is generally below 2 afterwards.

3.6.1.1 Implementation of the BDF schemes

For the implicit methods, any BDF scheme of order 1 to 6 can be used. For the first few timesteps, not all necessary previous solutions are available yet to perform a k-order scheme. In this case, the highest possible order is used until the order k is reached. If for example, a 4th order BDF scheme is required, the first three steps will be actually performed with the respective orders 1, 2 and 3.
The native PETSc error estimate uses the Lagrangian polynomials as described in subsubsection 1.3.2.2. The second method with an embedded higher order execution of the BDF scheme from subsubsection 1.3.2.1, was manually implemented and the user can choose between both possibilities. Again, in the first few steps, the solution vectors are not yet available to execute the BDF scheme of order $k+1$ for the error estimate. In this case, the error is evaluated with a lower order scheme $k-1$ until enough old solution vectors are available.
Further, the adaptive BDF order finding from subsubsection 1.3.2.3 has been implemented and is used if the user provided "0" for the BDF order.

3.6.1.2 Switching between different methods

The simulation supports the use of different solvers and time integration methods for the aseismic slip phase and for the earthquake phase. To detect in which phase the simulation is currently in, the maximum slip rate V_{max} at the current timestep is compared to the environment slip V_0. If $V_{max} > V_0$ is verified, it means that the simulation is currently in an earthquake, else wise it is in the aseismic slip. When the simulation detects a switch from one state to the other, and if a different problem formulation or integration method is defined by the users for both states, the program overwrites the PETSc configuration and restarts the solver at the current timestep with the new methods. It is thus possible, for instance, to use the compact DAE formulation for the aseismic slip and the 2nd order ODE formulation with an explicit Runge-Kutta scheme for the earthquake.

3.6.2 Tolerances for the time integration

Tolerances play a crucial role fort adaptive time-stepping methods. A proposed timestep is accepted only if the estimated error is inferior to a defined tolerance t, therefore a carefully chosen tolerance has a direct impact on the timestep size and consequently on the total number of time iterations to reach the final simulation time. If u_i refers to all components of the solution vector and $u_i^{(e)}$ to the components of the embedded solution used for the error estimate, then a step is acceptable if following condition is fulfilled:

$$\left\| \frac{u_i - u_i^{(e)}}{t_i} \right\|_\infty = \max_i \left| \frac{u_i - u_i^{(e)}}{t_i} \right| \leq 1 \tag{3.59}$$

The infinity norm is used because a too large deviation in one node of the fault can erroneously provoke an earthquake at a too early time and thus strongly affects the accuracy of the results for the whole system. If the 2-norm was used, as it is common for other applications, too large errors at some nodes may occur, which are then compensated by nodes where the actual and the embedded solutions match well.

3.6.2.1 Behaviour of the Error

The tolerance t_i can be defined independently for each component of the solution vector. It is calculated for each time step with an absolute tolerance t_i^a and a relative tolerance t_i^r.

$$t_i = t_i^a + \max\left(u_i, u_i^{(e)}\right) t_i^r \tag{3.60}$$

Since some components of the solution vector correspond to the values of the state variable ψ and the other components correspond to the slip S, it is appropriate to use two different tolerances for the respective quantities. Thus, t_i^a and t_i^r take the values t_ψ^a and t_ψ^r in case the component at index i refers to the state variable and the values t_S^a and t_S^r if the index i refers to the slip at the fault. The motivation behind this decision can be seen in Figure 3.3, which depicts the maximal absolute and relative errors ($\max_i \left| u_i - u_i^{(e)} \right|$ and $\max_i \left| \frac{u_i - u_i^{(e)}}{u_i} \right|$) for the respective components of ψ and S. For this simulation, all errors have been set to $t_i^r = t_i^a = 10^{-7}$, without any distinction between slip or state variable components nor between relative or absolute tolerances.

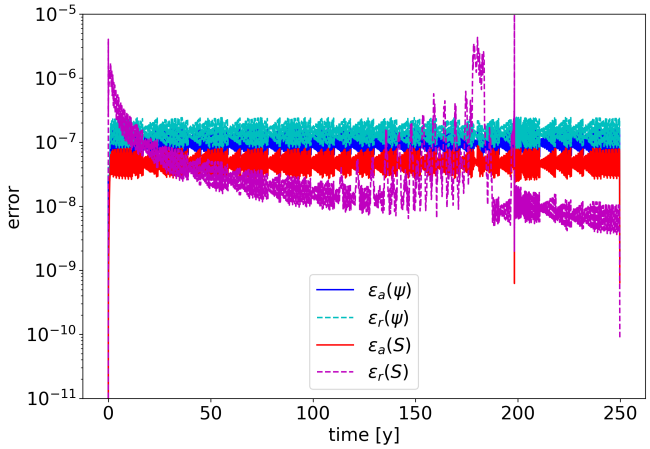


Figure 3.3: Maximum relative and absolute errors using the RKDP5 method on a fault with 200 elements with an absolute and relative error tolerance of 10^{-7}

Over the whole simulation time, the state variable ψ takes values close to 0.8 on all fault nodes, leading to a total tolerance for the corresponding components of $t_\psi \approx 1.8 \cdot 10^{-7}$. The blue line, which draws the

maximum absolute error, is clearly limited by this tolerance value and the light blue line of the relative error is located, as expected, by a factor 1/0.8 above the absolute error.
The error analysis of the slip presents a much less regular picture. The maximal absolute error lies always below the tolerance $t_S^a = 10^{-7}$, however the relative error reaches much higher values at the beginning of the simulation and before the earthquake event. The evolution of the extreme values of the slip in Figure 3.4 provides an explanation for these large errors. The maximum slip linearly increases, driven by the environment slip rate of the tectonic plates. On the open surface, the minimum slip is almost zero until the first earthquake. It then jumps to the next constant value, which is maintained until the subsequent earthquake. So, over the course of time, the slip increases overall.

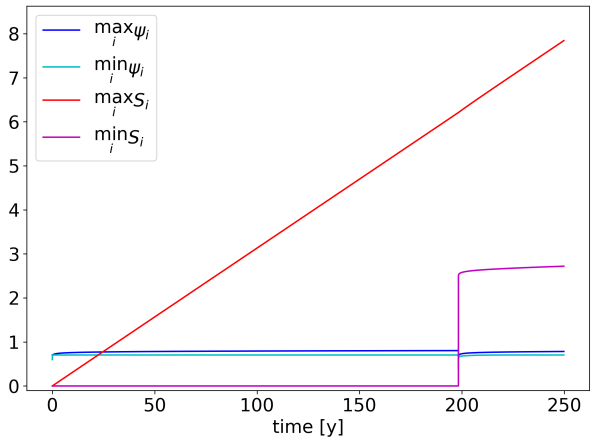


Figure 3.4: Maximum and minimum values of the state variable ψ and of the slip S over time. Simulation performed with the RKDP5 method on a fault with 200 elements with an absolute and relative error tolerance of 10^{-7}

From a physical point of view, the exact value of the slip does not matter, as only the change during the earthquake and the difference between the extreme values are relevant to describe tectonic dynamics. Therefore, only an absolute tolerance should be defined and a relative tolerance for the slip is not reasonable. For very long simulations, the total slip reaches higher orders of magnitudes after several earthquakes and if a relative tolerance was defined for the slip, it means that that early earthquakes are solved with higher accuracy than later ones.
In conclusion, it is sufficient to define absolute tolerances for the time integration of the slip and the state variable. Both quantities have values at a similar magnitude and setting $t_S^a = t_\psi^a = 10^{-7}$ is a safe bet. In a later section, we will investigate by how much this tolerance could be increased to allow for larger timesteps without a failure of the simulation.

// TODO: change this section with conclusion absolute tolerance for S, relative for ψ, relative + absolute for V //

3.6.2.2 Lower bound for the tolerances

For the 1st order ODE and both DAE formulations, the slip S and the state variable ψ are integrated over time and thus require tolerances. To get an idea about the range of values they can take, we estimate the smallest local truncation error in S and ψ that can be achieved at all. For that, we use the fact that the friction law can only be calculated iteratively up to an error $\varepsilon_{max} < 10^{-12}$ and we assume that in the worst case this error propagates entirely to the slip S or to the state variable ψ. The impact of a perturbation ε^ψ or ε^S on the friction law is set to ε_{max} and is solved with a first order Taylor polynomial.

$$\begin{cases} \varepsilon_{max} < f(S+\varepsilon^S,\psi,V) - f(S,\psi,V) \\ \varepsilon_{max} < f(S,\psi+\varepsilon^\psi,V) - f(S,\psi,V) \end{cases} \quad \Leftrightarrow \quad \begin{cases} \varepsilon_{max} < \left|\frac{\partial f_i}{\partial S_j}\right| \mathbb{1}_j \varepsilon^S + \mathcal{O}\left(\left(\varepsilon^S\right)^2\right) & \forall i \\ \varepsilon_{max} < \left|\frac{\partial f_i}{\partial \psi_j}\delta_{ij}\right| \varepsilon^\psi + \mathcal{O}\left(\left(\varepsilon^\psi\right)^2\right) & \forall i \end{cases} \tag{3.61}$$

$$\Leftrightarrow \quad \begin{cases} \varepsilon^S > \left[\min_i \left|\frac{\partial f_i}{\partial S_j}\right| \mathbb{1}_j\right]^{-1} \varepsilon_{max} \\ \varepsilon^\psi > \left|\frac{\sigma_n}{2V_0} \frac{V_i e^{\frac{\psi_i}{a}}}{\sqrt{\frac{e^{2\psi_i/a} V_i^2}{4V_0^2}+1}}\right|^{-1} \varepsilon_{max} \quad \forall i \end{cases} \tag{3.62}$$

In the first condition, $\mathbb{1}$ denotes the one vector, so we try to find the minimum absolute row sum of the constant matrix $\frac{\partial f}{\partial S}$. A look at the range of the absolute row sums in Table 3.2 tells that the searched value does not depend of the problem size, so a lower bound for the error in S can be conveniently calculated by $\varepsilon^S > \varepsilon_{max}/0.257 = 4 \cdot 10^{-12}$.

Number of fault elements	21	40	80	200	400
Maximum absolute row sum	62.3	129	283	624	1230
Minimum absolute row sum	0.257	0.257	0.257	0.257	0.257

Table 3.2: Range of the components in $\frac{\partial f_i}{\partial S_j}\mathbb{1}_j$ for different problem sizes

In the second condition, the lower bound of the exponential term in the denominator $e^{2\psi_{min}/a_{max}} = e^{2\cdot 0.5/0.025} \approx 5.5 \cdot 10^{34}$ is already extremely high, so that the term $+1$ can be neglected in the square root. Then, the expression simplifies dramatically as $\frac{\partial f}{\partial \psi} \approx \sigma_n = 50$, which can be safely used for the lower bound of the error in $\varepsilon^\psi > \varepsilon_{max}/50 = 2 \cdot 10^{-14}$.
For the second order ODE, the tolerance in V is also required, so the same procedure is applied to determine a lower bound for the error term ε^V.

$$\varepsilon_{max} < f(S,\psi,V+\varepsilon^V) - f(S,\psi,V) \quad \Leftrightarrow \quad \varepsilon_{max} < \left|\frac{\partial f_i}{\partial V_j}\delta_{ij}\right| \varepsilon^V + \mathcal{O}\left(\left(\varepsilon^V\right)^2\right) \quad \forall i \tag{3.63}$$

$$\Leftrightarrow \quad \varepsilon^V > \left|\frac{\sigma_n}{2V_0} \frac{e^{\frac{\psi_i}{a}}}{\sqrt{\frac{e^{2\psi_i/a} V_i^2}{4V_0^2}+1}} + \eta\right|^{-1} \varepsilon_{max} \quad \forall i \tag{3.64}$$

With the same approximation as earlier for the denominator, the expression simplifies to $\varepsilon^V > V_i\varepsilon_{max}/(\sigma_n + V_i\eta)$. As opposed to ε^S and ε^ψ, the error depends on the maximum slip rate in the current iteration, which is in line with the postulate from the previous section that a relative tolerance definition for the slip rate is more appropriate. Sadly, the expression for the lower bound of ε^V is not an affine function and can not be used as such as a tolerance. Instead, we can use the condition $\varepsilon^V > V_i\varepsilon_{max}/\sigma_n$, which is still a very accurate lower bound for most of the encountered slip rates. As seen in Figure 3.5, only for very high slip rates that occur at the peak of an earthquake, the linearized lower bound for ε^V exceeds significantly the exact value.

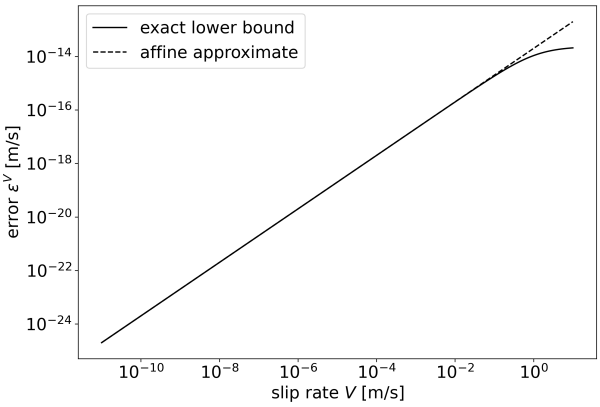


Figure 3.5: Lower bound for the error in the slip rate V compared to its linear approximate in function of the slip rate

The proposed tolerances are only of theoretical nature, as they assume that the error only stems from the iterative solver of the friction law. However, the time integration would most likely fail because numerical errors or the sum of the LTE from the different stages of a RK scheme have not been considered. Moreover the error estimate itself is not exact, and, as already discussed in subsection 2.4.2, it tend to overestimate the local truncation error, which means that even if the error is within the tolerance, the program would not understand it as such. If one wants to run the simulation with maximal accuracy, it is wise to add a safety factor to the tolerances.

3.6.2.3 Upper bound for the tolerances

The discussion so far only tackled the question of the lowest possible tolerances to reach as accurate results as possible. However, such strict tolerances may severely affect the timestep size and by consequence the length of the simulation. For many applications, satisfactory results are already obtained for larger tolerances, and the aim of this section is to investigate it for SEAS. As there is no mathematical definition for *satisfactory results* (unless convergence is the only requirement), we will run the simulation with the minimum possible tolerance described in the previous section and assess how the results deteriorate as the tolerances increase.
We consider

3.6.3 Newton Iteration

The Jacobian matrix is needed to apply implicit numerical methods to the SEAS problem. Unlike explicit methods, they evaluate the right hand side of the ODE with the current solution which is not known yet. To calculate the solution vector at a given time step, a nonlinear algebraic equation of the form $\phi(x) = 0$ needs to be solved where x is the solution vector to be determined. The Newton method is often used to solve the equation because of its ease to implement and its second-order convergence.

- Calculate an initial guess x_0
- Repeat until tolerance is reached $\|\phi(x_n)\| < TOL$:
  - $x_{n+1} = x_n - J_\phi^{-1}(\phi(x_n))\phi(x_n)$

The matrix $J_\phi^{-1}(f(x_n))$ is the Jacobi matrix of the function ϕ evaluated at the point x_n.

3.6.3.1 Convergence

Next, the convergence properties of the Newton iteration with the analytic Jacobian is investigated on the 1st order ODE formulation. For this, we solve one timestep of the implicit Euler method starting from the initial condition of the simulation. The function ϕ and its Jacobian matrix are given by:

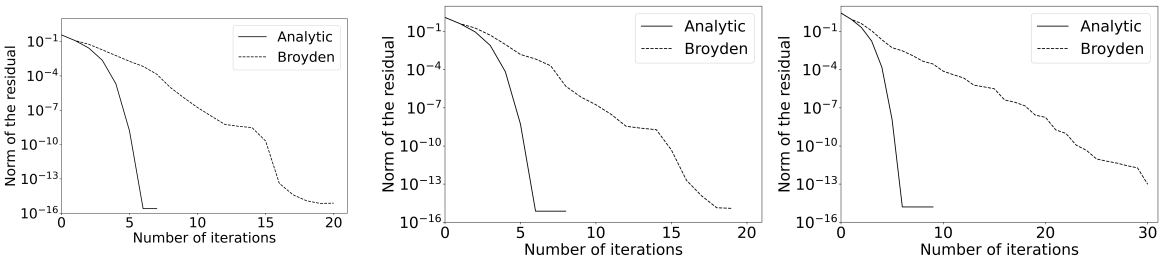
$$\phi(x) = -x + x^{(0)} + hF(x) \tag{3.65}$$

$$J_\phi(x) = -I + hJ_F(x) \tag{3.66}$$

The vector x contains both the components related to the slip S and to the state variable ψ and the right hand-side vector $F(x)$ contains their respective time derivative. The Jacobian of the proposed Newton iteration needs the Jacobian $J_F(x)$ of the right-hand side vector, of which the correctness is evaluated here. The success of the Newton iteration, thus observable second-order convergence, indicates the correctness of the Jacobian matrix.
Furthermore, the behavior of the analytic expression of the Jacobian is compared to the behavior of an iterative approximation of it. The Broyden's method [Bro65] provides an enhancement of the Newton method which updates the Jacobian matrix at each iteration without the need of its analytical expression. The main difficulty is to find an appropriate initial guess to achieve a fast convergence.

- Calculate the initial guesses x_0 and J_0
- Repeat until tolerance is reached $\|\phi(x_n)\| < TOL$:
  - $\Delta x_n = x_n - x_{n-1}$ and $\Delta\phi_n = \phi(x_n) - \phi(x_{n-1})$
  - $J_n = J_{n-1} + \frac{\Delta\phi_n - J_{n-1}\Delta x_n}{\|\Delta x_n\|^2}\Delta x_n^T$
  - $x_{n+1} = x_n - J_n\phi(x_n)$

The motivation behind this update scheme is to minimize the Frobenius norm $\|J_n - J_{n-1}\|_F$. As a matter of simplicity, the initial guess of the Jacobian is obtained with the analytical expression of it, even though its correctness has not yet been shown. Other initialization methods such as finite differences do not lead to convergence of the Broyden method.
The experiment has been performed on a symmetric, two-dimensional domain of varying size. The initial guesses for x are obtained with one step of the explicit Euler method with a timestep of $h = 10^5$s. This time step is large enough to obtain an error to the exact value at this time which needs several Newton iterations to be corrected but still small enough to ensure that the Newton iteration converges at all. The evolution of the residual $\phi(x_n)$ is shown in Figure 3.6.

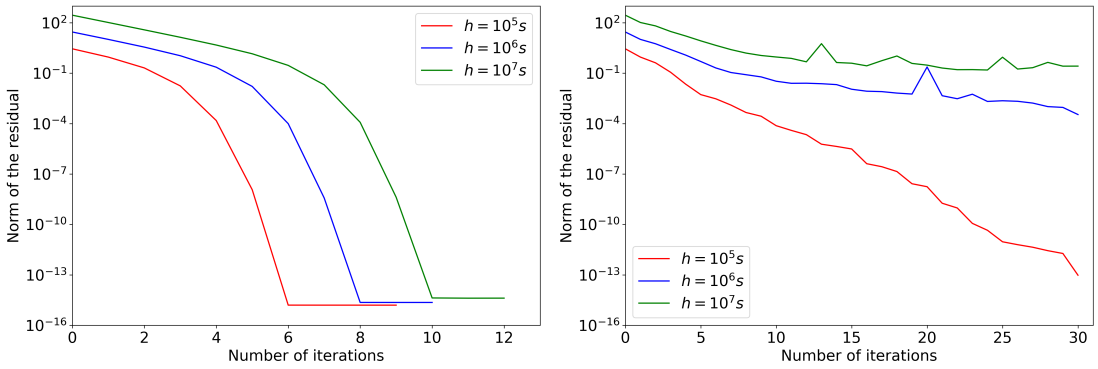


(a) 5 fault elements (b) 40 fault elements (c) 200 fault elements

Figure 3.6: Evaluation of the L2 norm of the residual $\phi(x_n)$ at each iteration of the Newton and Broyden methods

It can be immediately seen that the Newton method with an analytic expression of the Jacobian reaches much faster the maximum reachable accuracy of about 10^{-15} as if it was approximated with the Broyden method. The convergence rate even seems to be quadratic, as one would expect for the Newton iteration and the convergence is similarly fast for all tested domain sizes. The Broyden method on the other hand shows rather a linear convergence behavior, and the number of needed iterations increases with problem size. This makes the Broyden iteration particularly unsuitable for simulations on large domains. The approximated Jacobian matrix at the end of the Broyden iteration matches with high precision its

analytic counterpart, since the maximum relative difference among the entries is of the order of 10^{-5}. So far, it has been shown that the Newton iteration converges well for various domain sizes. The chosen timestep size 10^5s is still small, as in the aseismic slip, it may reach the order 10^7s. Figure 3.7 shows the norm of the residual for the timestep sizes 10^5s 10^6s and 10^7s. The direct iteration with the analytic Jacobian matrix always has a quadratic convergence, and the slightly higher number of iterations for large timesteps is essentially due to the worse initial guess. In contrast, the Broyden iteration converges much slower if the timestep size increases.



(a) Direct iteration

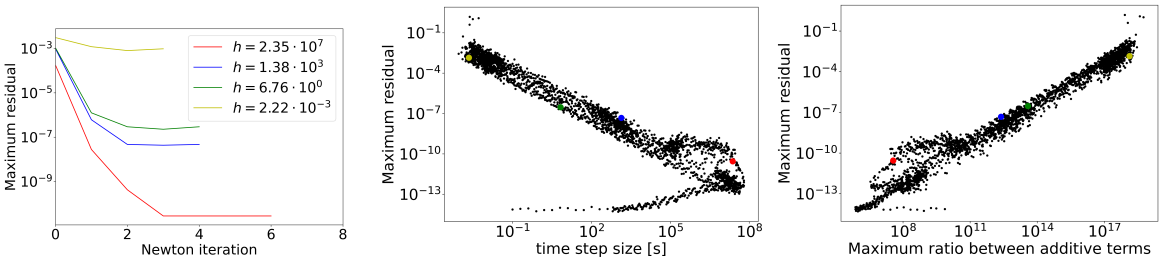
(b) Broyden iteration

Figure 3.7: Evaluation of the L2 norm of the residual $\phi(x_n)$ at each iteration of the Newton and Broyden methods for 200 fault elements

It could be shown that the Newton iteration with the analytic Jacobian has excellent convergence properties for any domain sizes and large time step sizes. On the other hand, the alternative with a Broyden iteration, which approximates the Jacobian matrix along with the solution, converges only very slowly for large domains and large timesteps which is why this method is not suited to solve the problem. In practice, the Newton iteration converges much faster because BDF schemes of higher order are used instead of the implicit Euler and the initial guess at the beginning of the iteration is obtained by extrapolating the previous solution vectors to the current simulation time. Usually, one Newton step is required to achieve the desired accuracy, which means two evaluations of the right-hand side function.

3.6.3.2 Convergence issues with the compact DAE formulation

The compact DAE formulation from Equation 3.37 shows a similar convergence behavior only during the aseismic slip. Whereas the method never diverges, the maximal achievable accuracy depends on the current time step size, in a way that for very small timesteps, the residual in the Newton iterations does not go below some large value. This bad convergence can be seen in Figure 3.8a, which shows the infinity norm of the residual at each Newton step for different time step sizes. The largest instance, of the order 10^7, is representative for the aseismic slip after the first earthquake and the other samples have been taken in the initial phase of the second earthquake, when the slip rate increases significantly and the timestep size decreases. Figure 3.8b shows the maximum residual at the end of the Newton iteration in function of the current timestep size for all timesteps in the simulation. The locations of the samples in the first graph are highlighted by colored points. It can be clearly seen that a large residual norm inversely depends on the timestep size.
To obtain these results, the time integration itself has not been performed with the compact DAE formulation, since it obviously yields a poor accuracy, but rather with the 1st order ODE formulation using a classic adaptive RK4 scheme. At each timestep, the Newton iteration has been launched until the residual norm does not decrease anymore and the final result was discarded. Therefore, the solution vectors at the previous timesteps needed in the BDF method stem from the explicit RK4 time integration and thus slightly differ to the vectors in the time integration with the compact DAE formulation. To discuss the convergence, this difference is not significant, since the convergence issues arise in either case.



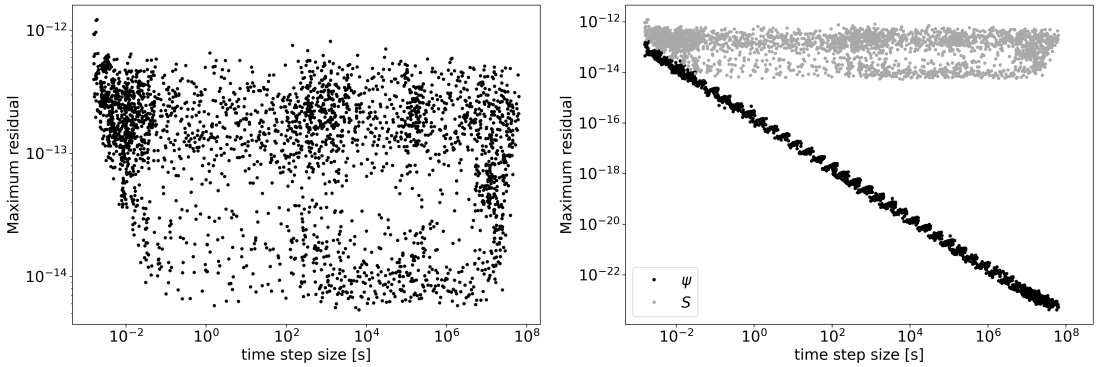
(a) ∞-norm of the residual in the Newton iterations at different time step sizes

(b) Minimum achievable residual norms in function of the timestep size

(c) Minimum achievable residual norms in function of the maximal ratio between the additive terms in the Jacobian matrix

Figure 3.8: Convergence properties of the compact DAE formulation with the 4th order BDF method on a small domain with 5 fault elements

The reason for this poor convergence can be found in the definition of the Jacobian matrix of the compact DAE formulation in Equation 3.38. As a quick remainder, the Jacobian matrix in the Newton iteration is obtained by $\mathbf{J} = \mathbf{F}_x + \sigma \mathbf{F}_{\dot{x}}$, where the shift σ depends on the order k of the BDF scheme and on the timestep sizes in the k previous timesteps. The upper left block in the Jacobian is thus the sum of $\sigma \frac{\partial f}{\partial V}$ and $\frac{\partial f}{\partial S}$. The first partial derivative is of the magnitude 10^9 in the aseismic slip and reaches values up to the order 10^{15} in an earthquake. On the other hand, the second summand takes values between 10^0 and 10^2. Since the former is a diagonal matrix, the sum only affects the diagonal elements. The timestep does not change by a lot between two time iterations, it can be assumed that they are of the same order of magnitude. The shift σ is then of the order $\sigma = \Theta\left(h_n^{-1}\right)$, where h_n is current timestep size. In the aseismic slip, σ is thus of the order 10^{-7} and can go as high as 10^3 during an earthquake. In the worst case, two terms of orders 10^{18} and 10^0 have to be summed up, which cannot be represented anymore by a double precision floating point variable. The partial derivative $\frac{\partial f}{\partial S}$ is therefore neglected in the Jacobian matrix during an earthquake and the friction law cannot be solved accurately anymore. Figure 3.8c shows the maximum residual at the end of the Newton iteration in function of the ratio $\sigma \frac{\partial f}{\partial V} / \frac{\partial f}{\partial S}$ between the two problematic summands. As the ratio approaches and surpasses the machine precision of approximately 10^{16}, the minimal achievable residual norm also increases. As a matter of fact, the bad accuracy only appears in the residual components in S, for which the problematic summation occurs, whereas the residual components in ψ remain at very good values below 10^{-12} throughout the simulation.
In the last two graphs, a band with high accuracy and low timestep size appears at the bottom. These points correspond to the initial phase of the simulation, which is always launched in the aseismic slip phase with $h_0 = 0.1s$. Since the solution vector contains the initial value in these points, the Newton iteration converges directly with a high precision.
This issue occurs only with the compact DAE formulation, because the Jacobian matrices of the ODE formulations do not require any summations and in the extended DAE formulation, the two problematic terms $\frac{\partial f}{\partial V}$ and $\frac{\partial f}{\partial S}$ are located in different submatrices of $\mathbf{F}_{\dot{x}}$ and $\mathbf{F}_x$ and are thus never added to each other. Figure 3.9a shows the same dependency as in Figure 3.8b for the extended DAE formulation. The norm of the residual at the end of the Newton iteration here does not depend on the timestep size and it varies between 10^{-12} and 10^{-15} throughout the simulation. Since both the slip and the state variable are of the order of 10^0, the achieved tolerance is excellent and close to machine precision. However, it seems that the best accuracy is only reached for large timesteps, whereas for very small timesteps, the Newton iteration only reached the "not the best but still very good" - accuracy of 10^{-12}. In the definition of the Jacobian matrix for the extended DAE formulation in Equation 3.34, there still is an addition between the shift σ ($\sigma = 1/h$ in the referred equation because it is related to the implicit Euler method) and the term $\frac{\partial g}{\partial \psi}$. Since this last term is always of the order 10^{-7}, the difference between the summands is not as extreme as for the compact formulation, but still noticeable for small time steps, where $\sigma \approx 10^3$. Looking at Figure 3.9b, the components of the residual vector relative to ψ increase if the timestep size decreases. This is the same behavior as previously, but on a much lower magnitude. Only in the earthquake phase, when timesteps below $0.1s$ are encountered, the residual in ψ exceed the maximal precision 10^{-15} of the friction law and impacts the overall accuracy.



(a) Residual of all components

(b) Residual of the individual components ψ and S

Figure 3.9: Minimum achievable residual norms in function of the timestep size for the extended DAE formulation with the 4th order BDF method on a small domain with 5 fault elements

In conclusion, the compact DAE formulation fails for the earthquake and can only be applied in the aseismic slip phase. Since the accuracy decreases along with the timestep size, the usual strategy to restart a step with a smaller timestep size if the nonlinear did not converge only leads to an even worse accuracy and is unsuitable. For this limited purpose, it still offers advantages over other methods which will be discussed in a further section.

3.6.3.3 Iterative solvers for the Newton step update

In each Newton step, an expensive linear system has to be solved in $\prime\left(n^3\right)$, which is the main bottleneck for simulations on large domain. To solve this issue, several iterative methods have been tested instead of the classical Gaussian elimination. PETSc comes along with a broad range of Krylov methods with even more preconditioners. Very good results could already obtained with the simple Jacobi method [Axe93, p. 230]. More advanced stationary iterative methods, such as Gauss-Seidel or SOR, were not used because their PETSc implementation skips the convergence test and always performs a fixed number of iterations. Further, a Krylov method has also been tested, the generalized minimal residual method (GMRES) [SS86]. In each iteration k, it minimizes the residual in the kth-Krylov of the Jacobian matrix J and returns the exact solution at latest after n steps, so the complexity is the same as for a direct method in the worst case. To better its performance, both a Jacobi and a Gauss-Seidel preconditioner have been used.
The average number of iterations to solve the linear system, as well as the impact of its accuracy on the Newton iteration and on the total number of timesteps are shown in Tables 3.3-3.6 for all four formulations. The simulation length has been choosen such that exactly one earthquake happens, except for the compact DAE formulation, which generally fails to converge for small timesteps. For the ODE formulations, the iterative solver is directly applied to the Jacobian matrix as it appears in the Newton iteration and for the DAE formulations, the system is first reduced as described in subsection 3.4.6. This is especially crucial for the extended DAE formulation, since all methods fail to converge when directly applied to the Jacobian matrix.

	Jacobi	Jacobi-GMRES	GS-GMRES	LU
Average number of iterations per Newton step	6.97	5.72	3.08	-
Average number of Newton steps	3.28	3.28	3.28	3.27
Average final Newton residual	$1.07 \cdot 10^{-12}$	$1.17 \cdot 10^{-12}$	$1.33 \cdot 10^{-12}$	$1.54 \cdot 10^{-12}$
Total number of timesteps	4896	4906	4904	4901

Table 3.3: Quality of iterative solvers for the Jacobian system on a 5th-order BDF scheme with the 1st order ODE formulation on 101 fault elements for 250 years

In the 1st order ODE formulation, the GMRES with a Guass-Seidel preconditioner only needs in average 3.08 iterations to solve the linear systems, as opposed to the Jacobi and Jacobi-GMRES methods with respectively 6.97 and 5.72 iterations. The quality of the Newton iteration and the total number of timesteps is very similar to the direct method for all three iterative methods, so the GS-GMRES method can be fully recommended for this formulation.

	Jacobi	Jacobi-GMRES	GS-GMRES	LU
Average number of iterations per Newton step	7.48	5.11	2.77	-
Average number of Newton steps	4.07	4.82	4.51	4.23
Average final Newton residual	$8.73 \cdot 10^{-12}$	$1.17 \cdot 10^{-10}$	$1.13 \cdot 10^{-10}$	$4.53 \cdot 10^{-12}$
Total number of timesteps	4920	4919	4919	4919

Table 3.4: Quality of iterative solvers for the Jacobian system on a 5th-order BDF scheme with the extended DAE formulation on 101 fault elements for 250 years

In the extended DAE formulation, the three methods converge after a similar number of steps and again, the GS-GMRES method performs the best. However, both GMRES methods require slightly more Newton steps and only allow the Newton iteration to converge up to 10^{-10} and not 10^{-12} as for the direct method or the Jacobi method. This reduced accuracy does not seem to impact the total number of timesteps, so the GS-GMRES can again be considered to be the best option, but there might be some simulation settings where the more accurate Jacobi method would be more appropriate.

	Jacobi	Jacobi-GMRES	GS-GMRES	LU
Average number of iterations per Newton step	10.85	6.58	3.45	-
Average number of Newton steps	3.98	4.01	3.97	3.97
Average final Newton residual	$2.20 \cdot 10^{-11}$	$2.22 \cdot 10^{-11}$	$2.12 \cdot 10^{-11}$	$2.04 \cdot 10^{-11}$
Total number of timesteps	686	686	686	686

Table 3.5: Quality of iterative solvers for the Jacobian system on a 5th-order BDF scheme with the compact DAE formulation on 101 fault elements for 200 years

In the compact DAE formulation, a clear victory can be declared for GS-GMRES. Overall, more iterations are needed to converge than for its extended counterpart, albeit the comparison has only a limited significance, because the latter went through an earthquake event, which is impossible for the former.

	Jacobi	Jacobi-GMRES	GS-GMRES	LU
Average number of iterations per Newton step	5.41	4.95	2.57	-
Average number of Newton steps	3.25	2.72	2.72	2.72
Average final Newton residual	$1.91 \cdot 10^{-7}$	$9.03 \cdot 10^{-16}$	$7.85 \cdot 10^{-16}$	$7.41 \cdot 10^{-16}$
Total number of timesteps	11504	8308	8321	8306

Table 3.6: Quality of iterative solvers for the Jacobian system on a 5th-order BDF scheme with the 2nd order ODE formulation on 101 fault elements for 250 years

Finally, in the 2nd order ODE formulation, there is a massive difference between the Jacobi and the GMRES methods. The Jacobi methods induces a much higher final residual in the Newton iteration, with a direct consequence on the total number of timesteps. On the other hand, with the GMRES methods, the simulation works as good as with a direct solver, and, as usual, the Gauss-Seidel preconditioner leads to fewer iterations than the Jacobi preconditioner.

In conclusion, iterative solvers are well-suited to update the Newton step. For all formulations, the GMRES method with a Gauss-Seidel preconditioner performs the best. The Jacobi preconditioner has exactly the same impact on the overall performance of the simulation, but requires more iterations to converge, so it does not bring any benefits. The Jacobi method has to be remembered as an alternative for the extended DAE formulation because the Newton iteration with GMRES is less accurate, which might

impact the overall performance and accuracy under certain conditions. However, the Jacobi method should be avoided at any cost for the 2nd order ODE formulation, because it considerably affects the accuracy and performance of the simulation.

3.6.4 Data structure

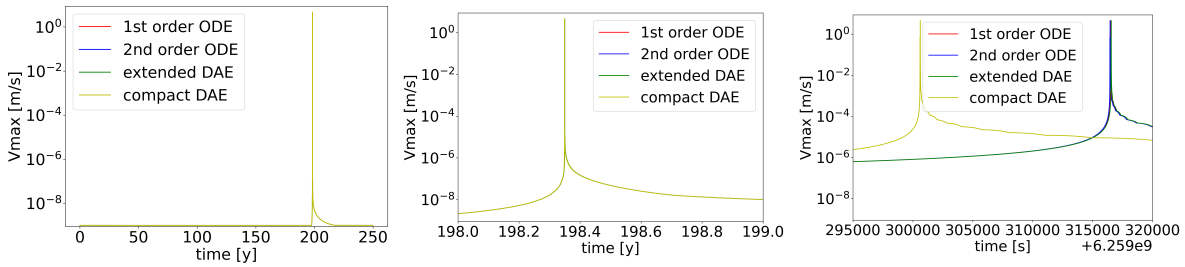
3.6.4.1 Block structure

3.6.4.2 Memory requirements

3.7 Results

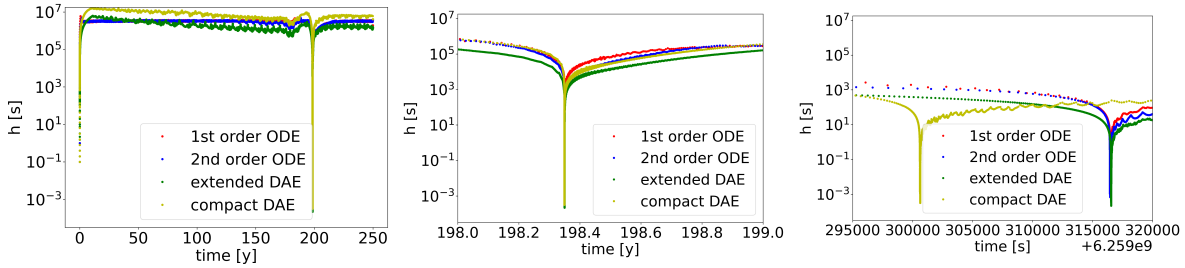
3.7.1 Comparison between the different formulations of problem

The simulation is run over a period of 250 years, in which one earthquake occurs on June 13th of the 195th year. This event can be clearly observed in Figure 3.10 which depicts the maximum slip rate over time, which reaches $4.6m \cdot s^{-1}$ as opposed to an average of $1.0 \cdot 10^{-9} m \cdot s^{-1}$ in calm times.



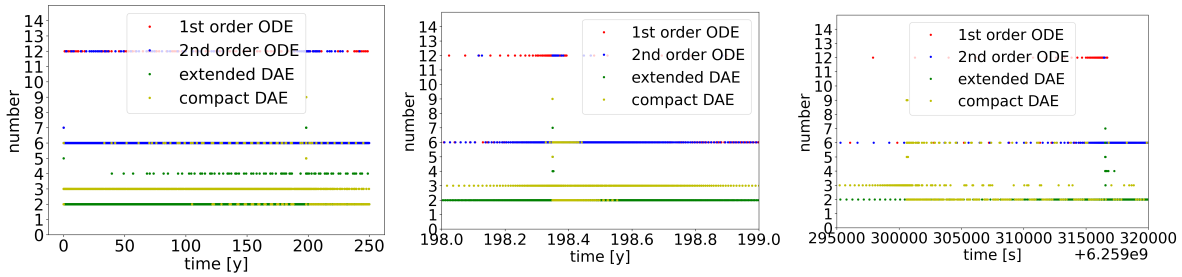
(a) Full simulation time (b) Year of the earthquake (c) Evolution of the earthquake event

Figure 3.10: Evolution of the maximal slip rate V on the fault for different solvers on the symmetric two-dimensional BP1 problem with 200 elements on the fault



(a) Full simulation time (b) Year of the earthquake (c) Evolution of the earthquake event

Figure 3.11: Evolution of the time step size h for different solvers on the symmetric two-dimensional BP1 problem with 200 elements on the fault



(a) Full simulation time

(b) Year of the earthquake

(c) Evolution of the earthquake event

Figure 3.12: Number of evaluations of the right hand side of the ODE in each time iteration for different solvers on the symmetric two-dimensional BP1 problem with 200 elements on the fault
Legend: — 1st order ODE formulation, — 2nd order ODE formulation, — extended DAE formulation, — compact DAE formulation

3.7.2 Time adaptivity on BDF methods

3.7.2.1 Quality of the error estimates

Two error estimates have been introduced for the BDF methods. First, in a given time step, the BDF scheme is applied a second time with higher order and the difference between both available solutions at the timestep yields the error estimate. The second method bases on the derivatives of Lagrangian polynomials as described in subsubsection 1.3.2.2, and has the advantage to be much cheaper to evaluate, since it does not require to solve the system again. It has already been shown in subsection 2.4.2 that for the 0D example, the first method gives a more accurate estimation of the local truncation error than the second method. Both methods tend to overestimate the error, which is for sure a safe behavior, but restrict the choice of the next timesteps. The aim of this section is to see, whether similar estimates can be observed for the current problem.
Since here no exact solution is available for the slip or the state variable (or the slip rate, if the solution vector has an extended size), the accuracy of the error estimate cannot be directly determined as previously. Instead, we assume that the actual error is inferior to the estimate, and that the embedded method is more accurate than Lagrangian polynomials. We would then expect that the timestep adapter chooses larger timesteps with the first method and finishes the program earlier. Exactly this can be observed in Figure 3.13 for the compact ODE formulation. As expected, the embedded method is faster than Lagrangian polynomials, because features, such as an increase or decrease of the timestep size happen increasingly earlier with the embedded method. Overall, fewer timesteps are required to reach a solution at a given time. However, the allowed timesteps are only slightly larger and the difference cannot be noticed on the logarithmic scale. Nonetheless, it is positive to remark that the same small features such as intermediate peaks are observed with both methods, which indicates that both error estimates recognize the same inconsistencies in the solution.

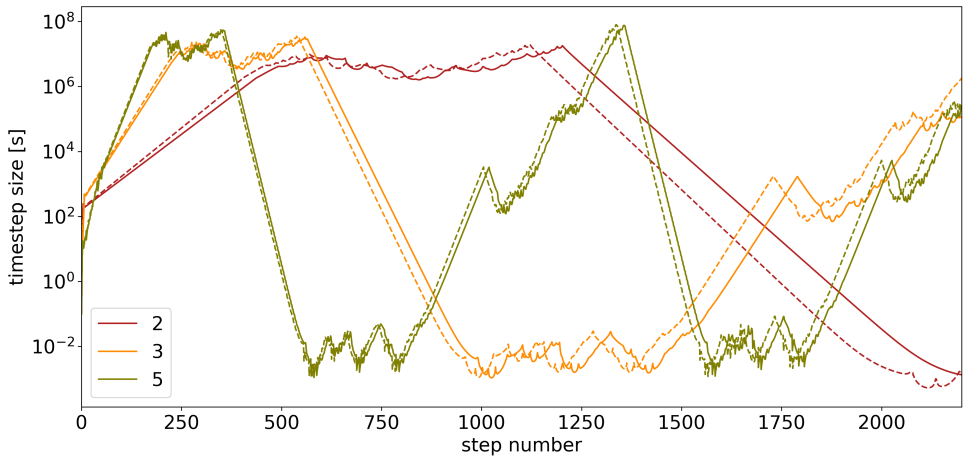


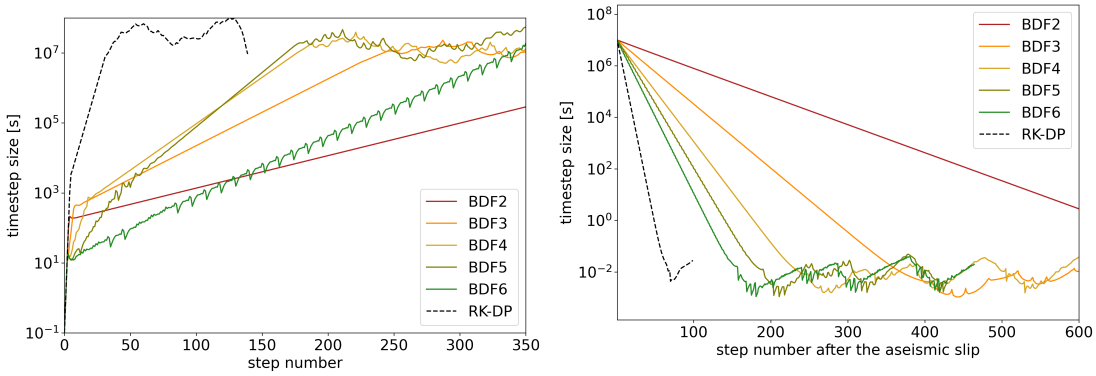
Figure 3.13: Comparison of the chosen timestep sizes with the Lagrangian polynomials (full line —) and the embedded error estimate (dashed line - -) for different BDF orders on a domain with 5 fault elements and the compact ODE formulation with $\varepsilon_a^S = \varepsilon_a^\psi = 10^{-7}$ and $\varepsilon_r^S = \varepsilon_r^\psi = 0$

Overall, the embedded method gives a more accurate error estimate than the Lagrangian polynomials, and allows for larger timestep sizes. However, the gain is only very small, as in average 2% fewer timesteps are required in sum at the end of the simulation, and does not justify the use of the embedded higher order BDF method as error estimate, as it is twice as expensive. Actually, the order of the BDF method has a much higher influence on the total number of iterations, as it can be seen in this graph and will be discussed more in detail in the next section.

3.7.2.2 Effect of the order on the timestep size

The order of the BDF method has a strong influence on the total number of required timesteps, as it can be clearly seen in Figure 3.13 from the previous section. The simulation can be roughly split into four sections: the evolution of the aseismic slip, with an approximately constant stepsize of the order of $h = 10^7$s, the evolution of the earthquake at $h = 10^{-2}$s and the transition between both phases, where the stepsize progressively increases or decreases. In addition to that, there is the initialization phase, in which the stepsize starts at $h = 0.1$s to quickly reach the stepsize of the aseismic slip.
From Figure 3.13, it seems that the stepsize of the constant evolution directly depends on the BDF order: a lower order leads to a smaller timestep and thus requires much more steps in total until the end of the aseismic slip or earthquake. The transition from the aseismic slip to the earthquake is a crucial section, because the sudden increase of the slip rate requires to rapidly scale down the timestep size. The end of the earthquake is less critical, as the slip rate drops relatively slowly which gives more scope to the stepsize to adapt to it, moreover the eventual too small stepsize there is not as problematic as a too high stepsize at the beginning of the earthquake. It is also interesting to look at the initialization phase, as there are no other constraints for a small stepsize but the limitations of the BDF scheme.
Figure 3.14a shows exactly the increase at the beginning of the simulation for the BDF methods of order two to six and Figure 3.14b shows the transition from the aseismic slip to the first earthquake. To better contextualize the results, the graph for the explicit 4th order Runge-Kutta scheme with the same tolerances is also shown. In the initialization phase, the slope is in general steeper if a higher order method is used. One notable exception is the 6th order BDF scheme, which performs almost as bad as the 2nd order scheme. Unlike the other schemes, it does not increase smoothly but has regular steps, which indicates that the 6th order scheme is not as stable as the others. Overall, the increase is perfectly exponential for the schemes 2-5, and the factor by which the timestep size increases at each step is driven by the order of the method. Interestingly enough, in the very first timesteps, the timestep size of the 2nd order scheme jumps by two orders of magnitude whereas the 5th and 6th order do not benefit at all from such a strong initial increase. At the transition from the aseismic slip to the earthquake phase, the timestep size decreases exponentially with the same pattern: a higher order scheme allows a stronger reduction of the timestep size from one step to the next. Now, the 6th order scheme is perfectly stable and has the steepest slope among all BDF methods. When compared to the explicit 4th order

Runge-Kutta method, all BDF methods in both scenarios perform extremely bad. This is however not a reason to discard BDF methods as a whole, since they are much more flexible with respect to the defined tolerances, as it will be discussed in a later stage.



(a) Maximum absolute value of the friction law

(b) Absolute change of the friciton law per timestep, averaged over 50 timesteps

Figure 3.14: Evolution of the time step size for BDF schemes with different order at key points in the simulation on a domain with 5 fault elements and the compact ODE formulation with $\varepsilon_a^S = \varepsilon_a^\psi = 10^{-7}$ and $\varepsilon_r^S = \varepsilon_r^\psi = 0$

Higher order BDF methods allow in general for higher stepsizes and are to be preferred over lower order methods. Since the costs of all BDF schemes are similar (they converge similarly fast and each Newton step always requires to solve one linear system), there is no strong reason to use a low order scheme here. Because of the stability issues with the 6th order scheme when the stepsize increases, the best performance can be expected from the 5th order scheme. There are some few cases where lower order methods perform better, such at the very beginning of the simulation. To take advantage of these peculiarities, and also to use the 6th order method if it is currently stable, the order of the BDF scheme can be adapted at each timestep as described in subsubsection 1.3.2.3. In this case, each step also calculates by how much the timestep size would change if one order higher or lower was used, and if a larger timestep can be reached, the order is changed for the next timestep. This has been implemented and the performance of the order-adaptive BDF scheme is shown in Figure 3.15. It successfully uses the scheme with the largest allowable stepsize and finishes the simulation faster than any other methods.

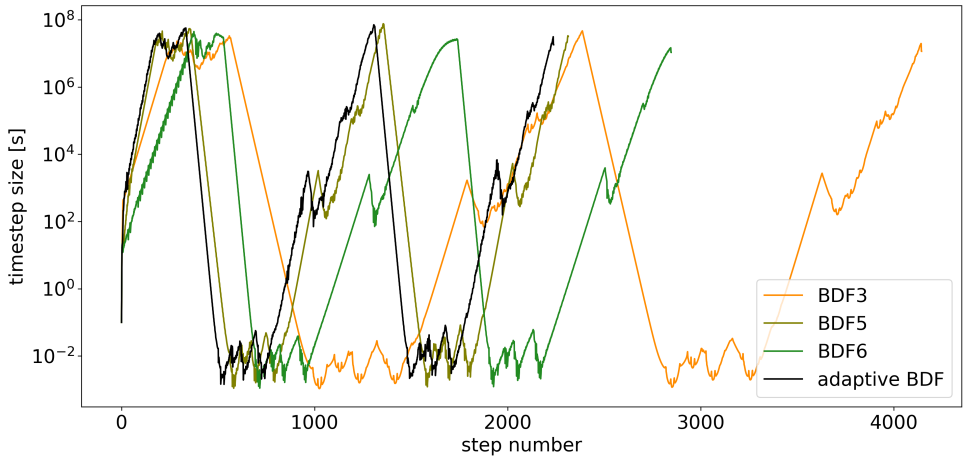


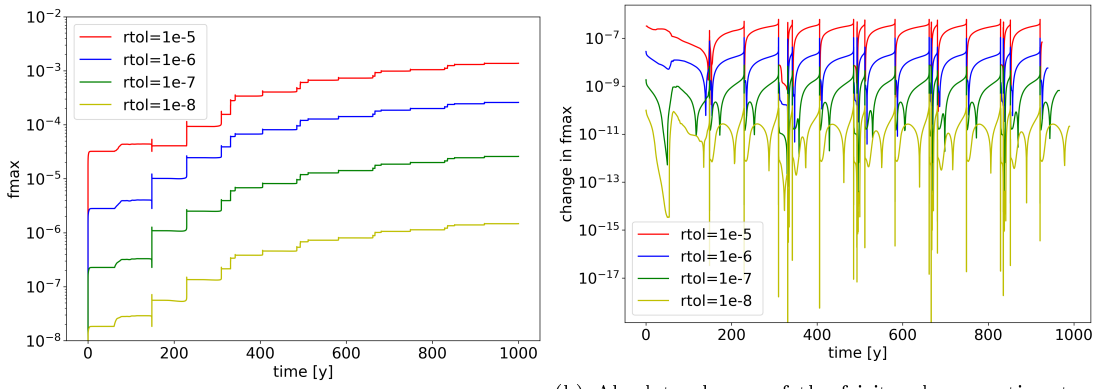
Figure 3.15: Evolution of the time step size for the adaptive-order BDF method and for a selection of different fixed orders on a domain with 5 fault elements and the compact ODE formulation with $\varepsilon_a^S = \varepsilon_a^\psi = 10^{-7}$ and $\varepsilon_r^S = \varepsilon_r^\psi = 0$

3.7.3 Error estimation of the 2nd order ODE formulation

Despite the lack of an analytic solution to the problem, this formulation of the problem offers a very powerful tool to evaluate the accuracy of the results. Since the slip rate is not calculated from the friction

law anymore, it is not ensured that it is always equal to 0 up to numerical precision, as it was the case in the first order formulations of the SEAS problem. We could then evaluate the friction law at every timestep to assess the accuracy of the numerical integration. A large deviation from 0 of the absolute value of the friction law at a given timestep means that the integrator provides poor results. For a standard execution of the simulation with the second order formulation, this metric is not available, since the evaluation of τ in the friction law requires to solve the Poisson problem, and the big advantage of the new formulation is exactly not to solve this system. For the following graphs, The value of the friction law has been exceptionally calculated.

For the first set of pictures, a very small domain with 5 fault elements has been chosen to be able to observe the evolution of the quantities over a long period of time. The tolerances for the slip and the state variable have been fixed to 10^{-7}. Figure 3.16a shows the maximum value of the friction law for varying relative relative tolerances for V, and the absolute tolerance is fixed to 0. From the initial condition, the slip rate is calculated to fulfill the friction law up to numerical precision, at about 10^{-15}. After some time steps, this high precision is lost, and at every earthquake event, the difference increases sharply. Overall, the logarithmic shape of the curves indicate a linear decrease of accuracy with time, It seems that, at every evaluation of the right-hand side function, for one given tolerance of V, a certain local truncation error is added to the residual of the friction law. At an earthquake, much more timesteps are required and the sum of the local truncation errors sum up to form an apparent sharp increase in the global error. For lower tolerances in V, the increase of the error at every evaluation is smaller the accuracy is higher. This hypothesis is confirmed by Figure 3.16b, which shows the absolute change of the friction law per timestep. It can be considered somehow as an estimate for the local truncation error (LTE), as it describes how the error increases at each step. For a given tolerance in the slip rate, an upper bound for the LTE can be observed. During the earthquake, the LTE is actually much lower than in the aseismic slip. Only the high number of steps required for this event result in the apparent sharp increase of the total error.

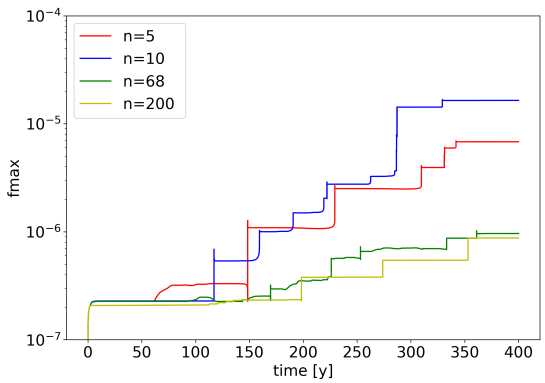


(a) Maximum absolute value of the friction law

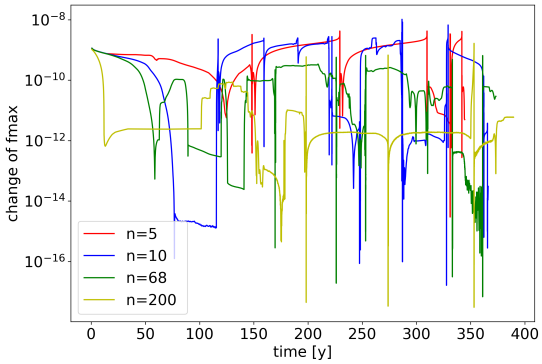
(b) Absolute change of the friciton law per timestep, averaged over 50 timesteps

Figure 3.16: Evolution of the value of the friction law of the 2nd order formulation with 5 elements on the fault over 1000 years with varying relative tolerances in the slip rate V

The local truncation error seems to depend linearly on the tolerance for the slip rate, and a similar dependency is to be expected with respect to the tolerances of the slip S and ψ, since these quantities are also required to evaluate the friction law. Next, we investigate whether other parameters, such as the spatial discretization, have an effect on the global error too. For a fixed relative tolerance for the slip of 10^{-7}, the simulation has been executed for different spatial resolutions and the evolution of the friction law in each case is shown in Figure 3.17a. Since each fault elements contains three fault nodes, one has to multiply n by three to obtain the total number of simulated fault nodes. At the beginning, all domain sizes have a similar accuracy, but after some earthquakes, it seems that the error in domains with higher resolution increases less fast. However, in Figure 3.17b, it is hard to distinguish separate upper bounds for the local truncation error with different domain sizes. The better performance of larger domains is rather due to the overall higher accuracy of simulations with higher resolutions and to a lower incidence of earthquakes, than due to a reduction of the LTE.



(a) Maximum absolute value of the friction law



(b) Absolute change of the friciton law per timestep, averaged over 50 timesteps

Figure 3.17: Evolution of the value of the friction law of the 2nd order formulation over 400 years with varying domain sizes, where n denotes the number of fault elements

Overall, the evaluation of the friction law is a suitable estimate for the global error in the system. Over the course of time, its value increases step wise at each earthquake, but overall linearly, which is due to a regular local truncation error. The main driver of the LTE is the tolerance for the slip rate V, which has been introduced to the system along with the second order ODE formulation. The upper bound for the LTE is proportional to the chosen tolerance.

3.7.4 Scalability

Chapter 4

Extension to More Complex Problems

4.1 Elasticity Equation

Solve the Elasticity equation instead of the Poisson equation in the domain with the discontinuous Galerkin method.

4.2 Three-Dimensional Simulations

Consider a full three-dimensional domain with a two-dimensional fault instead of the cross-section infinitely long block.

4.3 Seismic Waves in the Domain

Include seismic waves in the full domain after an earthquake. That waves could trigger an earthquake somewhere else, if there are several faults.

Bibliography

[Axe93] O. Axelsson. *Iterative solution methods.* Cambridge ; New York : Cambridge University Press, 1993.

[Bro65] C. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593, 1965.

[Olv14] Peter J. Olver. *Introduction to Partial Differential Equations.* Springer International Publishing, 2014.

[Rap97] Joseph Raphson. *Analysis Æequationum Universalis.* Th. Braddyll, 1697.

[Roa01] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions . *Journal of Fluids Engineering*, 124(1):4–10, 11 2001.

[SÖ2] Gustaf Söderlind. Automatic control and adaptive time-stepping. 31(1-4):281–310, 2002.

[SS86] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.