

thesis of me

some terrible student

November 2020

# Chapter 1

## 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.

$$\frac{d\psi}{dt} = f(\psi, V) = 1 - \frac{V\psi}{L} \quad (1.1)$$

$$0 = g(\psi, V) = \tau - \sigma_n a \sinh^{-1} \left( \frac{V}{2V_0} e^{\frac{f_0 + b \log(V_0 \psi)}{a}} \right) - \eta V \quad (1.2)$$

### 1.1 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 1.3, the slip rate increases from 0 to 1 over a time span  $t_w$  at the time  $t_e$ .

$$V^*(t) = \frac{1}{\pi} \tan^{-1} \left( \frac{t - t_e}{t_w} + \frac{\pi}{2} \right) \quad (1.3)$$

The manufactured evolution of the state variable  $\psi^*(t)$  can be calculated by solving the algebraic equations (1.2) 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 1.4.

$$\frac{d\psi}{dt} = f(\psi, V) - f(\psi^*, V^*) + \frac{d\psi^*}{dt} \quad (1.4)$$

$$0 = g(\psi, V) \quad (1.5)$$

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.

### 1.2 Error Evaluation

general concept: compute in addition to the solution another, higher order approximation. The difference between the two solutions is the error approximation

### 1.2.1 Explicit methods

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

### 1.2.2 Implicit methods

Implicit methods are well-suited to solve stiff problems and allow for higher timesteps than explicit methods. Instead of evaluating the time-derivative of Equation 1.1 for a known value  $\psi_n$ , it is evaluated at the next timestep with  $\psi_{n+1}$ , 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. To evaluate the local truncation error at a given timestep, the BDF method with the next highest order is used and the difference between the solutions is taken as the error estimate.

The first order BDF-scheme corresponds to the backward Euler method in Equation 1.6.

$$\psi_{n+1} = \psi_n + h_n f(\psi_{n+1}, V_{n+1}) \quad (1.6)$$

The second order BDF-scheme is then needed to estimate the local truncation error. Because of the adaptive time-stepping, the traditional coefficients of BDF2 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  $\psi_n$  and  $\psi_{n+1}$  are evaluated with respect to the unknown  $\psi_{n+2}$ .

$$\psi_n = \psi_{n+2} - (h_n + h_{n+1}) \frac{d\psi_{n+2}}{dt} + \frac{(h_n + h_{n+1})^2}{2} \frac{d^2\psi_{n+2}}{dt^2} + \mathcal{O}((h_n + h_{n+1})^3) \quad (1.7)$$

$$\psi_{n+1} = \psi_{n+2} - h_{n+1} \frac{d\psi_{n+2}}{dt} + \frac{h_{n+1}^2}{2} \frac{d^2\psi_{n+2}}{dt^2} + \mathcal{O}(h_{n+1}^3) \quad (1.8)$$

The idea is to add equations (1.7) and (1.8), where the latter is multiplied by a factor  $\alpha$  in a way that the second-derivative term drops. The addition of the two Taylor-expansions yields:

$$\psi_n + \alpha\psi_{n+1} = (1+\alpha)\psi_{n+2} - (h_n + (1+\alpha)h_{n+1}) \frac{d\psi_{n+2}}{dt} + \frac{(h_n + h_{n+1})^2 + \alpha h_{n+1}^2}{2} \frac{d^2\psi_{n+2}}{dt^2} + \mathcal{O}(h^3) \quad (1.9)$$

By the choice of  $\alpha$  in Equation 1.10, the coefficient in front of the second time derivative of  $\psi_{n+2}$  vanishes and the second order time adaptive BDF method is given by Equation 1.11.

$$\alpha = - \left( \frac{h_n}{h_{n+1}} \right)^2 - 2 \frac{h_n}{h_{n+1}} - 1 \quad (1.10)$$

$$-(1+\alpha)\psi_{n+2} + \alpha\psi_{n+1} + \psi_n = -(h_n + (1+\alpha)h_{n+1}) f(\psi_{n+2}, V_{n+2}) \quad (1.11)$$

Because of the DAE form, the values of  $\psi_{n+1}$  and  $\psi_{n+2}$  in Equation 1.6 and  $\psi_{n+2}$  in Equation 1.11 cannot be calculated analytically. To solve the equation, it is transformed into an algebraic equation, in which the right hand side is 0, to obtain the form:

$$F(\Psi) = 0 \quad (1.12)$$

where  $\Psi$  stands respectively for  $\psi_{n+1}$  and  $\psi_{n+2}$ . It is solved iteratively with the Newton-Raphson method [Rap97] in Equation 1.13 and the secant method to approximate the derivative  $F'(\Psi) = \frac{dF(\Psi)}{d\Psi}$  in Equation 1.14.

$$\Psi_{k+1} = \Psi_k + \frac{F(\Psi_k)}{F'(\Psi_k)} \quad (1.13)$$

$$F'(\Psi_k) = \frac{F(\Psi_k) - F(\Psi_{k-1})}{\Psi_k - \Psi_{k-1}} \quad (1.14)$$

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.

Analogously, a second order method can be developed if we use the BDF2 scheme to calculate the solution and the BDF3 scheme to estimate the local truncation error. The time-adaptive BDF3 scheme is given by:

$$\alpha = -\frac{(h_n + h_{n+1})(h_n + h_{n+1} + h_{n+2})^2}{h_{n+1}(h_{n+1} + h_{n+2})^2} \quad (1.15)$$

$$\beta = \frac{h_n(h_n + h_{n+1} + h_{n+2})^2}{h_n^2 h_{n+1}} \quad (1.16)$$

$$\psi_n + \alpha\psi_{n+1} + \beta\psi_{n+2} = (1 + \alpha + \beta)\psi_{n+3} - (h_n + (1 + \alpha)h_{n+1} + (1 + \alpha + \beta)h_{n+2})f(\psi_{n+3}, V_{n+3}) \quad (1.17)$$

## 1.3 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  $\epsilon$ . The controller  $C$  is a function

$$h_{n+1} = C(\epsilon, r_{n+1}, h_n) \quad (1.18)$$

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.

### 1.3.1 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  $\Phi$ .

$$r_{n+1} = \Phi h_n^k \quad (1.19)$$

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:

$$h_{n+1} = \left(\frac{\epsilon}{\Phi}\right)^{1/k} \Leftrightarrow h_{n+1} = \left(\frac{\epsilon}{\Phi h_n^k}\right)^{1/k} h_n \Leftrightarrow h_{n+1} = \left(\frac{\epsilon}{r_{n+1}}\right)^{1/k} h_n \quad (1.20)$$

In practice, there is never such a constant error factor  $\Phi$ , but it will take a different value  $\phi_n$  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  $\epsilon$  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:

$$h_{n+1} = \left(\frac{\theta\epsilon}{r_{n+1}}\right)^{1/k} h_n \quad (1.21)$$

The initial assumption in Equation 1.19 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.

## 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:

$$h_{n+1} = \left( \frac{\theta\epsilon}{r_{n+1}} \right)^{k_I} \left( \frac{r_n}{r_{n+1}} \right)^{k_P} h_n \quad (1.22)$$

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  $\alpha$  and  $\beta$ . The results are shown in Figures 1.1, 1.2 and 1.3. 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.

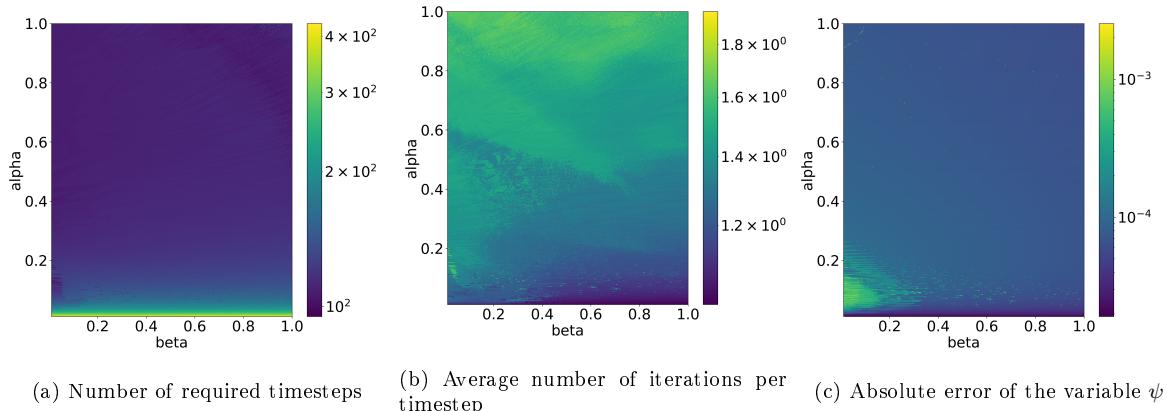


Figure 1.1: Impact of the parameters  $\alpha$  and  $\beta$  of the PI controller on the number of required iterations to solve the model problem from Equation 1.1 using the explicit Runge-Kutta-Fehlberg method (**RKF45**) for a simulation time of  $t = 5.0s$

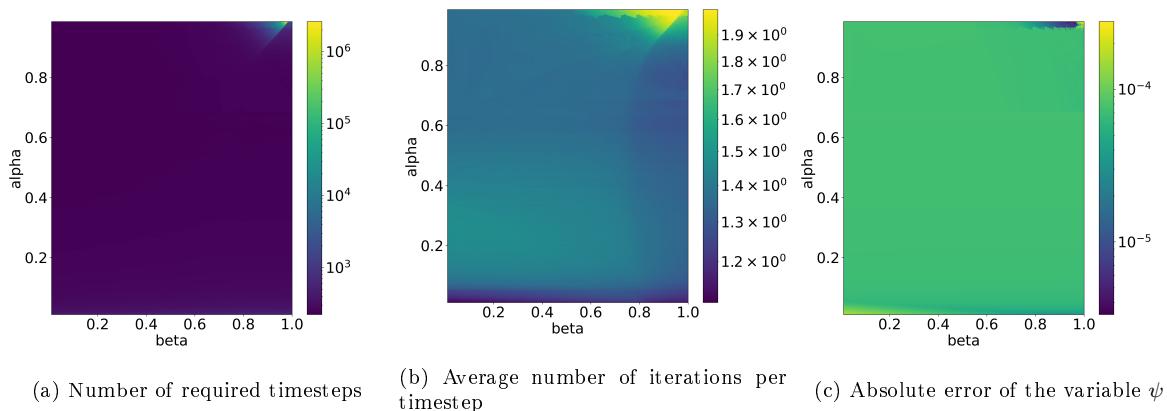


Figure 1.2: Impact of the parameters  $\alpha$  and  $\beta$  of the PI controller on the number of required iterations to solve the model problem from Equation 1.1 using the implicit BDF method of first order (**BDF12**) for a simulation time of  $t = 5.0s$

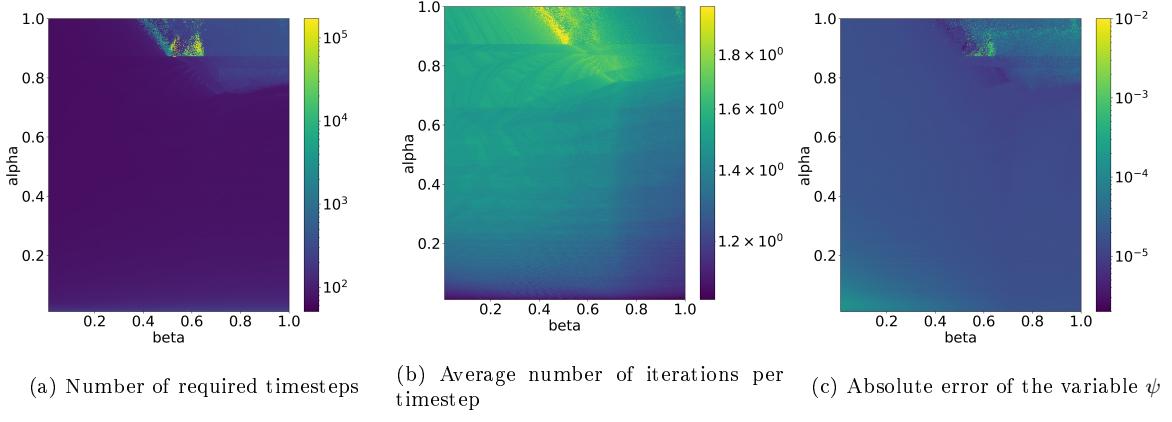


Figure 1.3: Impact of the parameters  $\alpha$  and  $\beta$  of the PI controller on the number of required iterations to solve the model problem from Equation 1.1 using the implicit BDF method of second order (**BDF23**) for a simulation time of  $t = 5.0s$

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  $\alpha$  and  $\beta$  —  
failure of the error estimate? too small timesteps?

## 1.4 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 1.2 and their ideal parameter choice for their use with a PI controller has been discussed. Now, the quality of the solutions is compared.

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 1.4a depicts the solution variables  $\psi$  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  $\psi$  match too, as  $\psi$  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 1.4b, 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  $\psi$  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 five times smaller. The second phase is marked by the fast increase in velocity and the stabilization of  $\psi$ . 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  $\psi$  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.

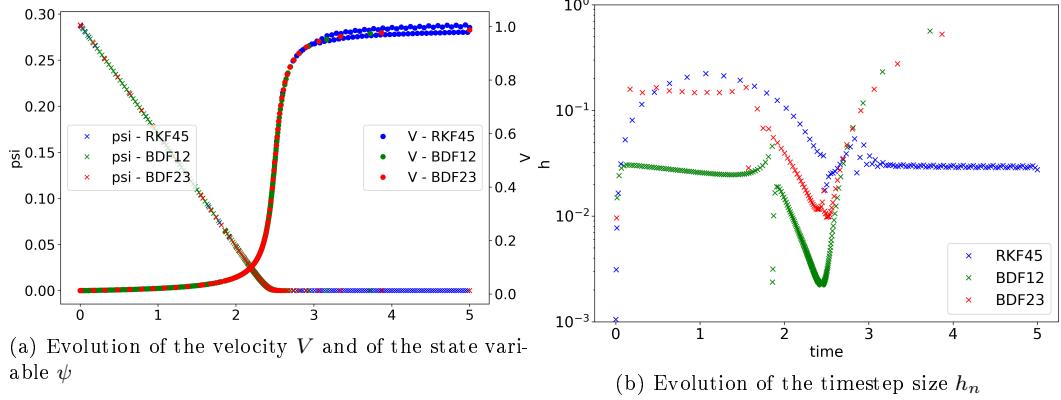
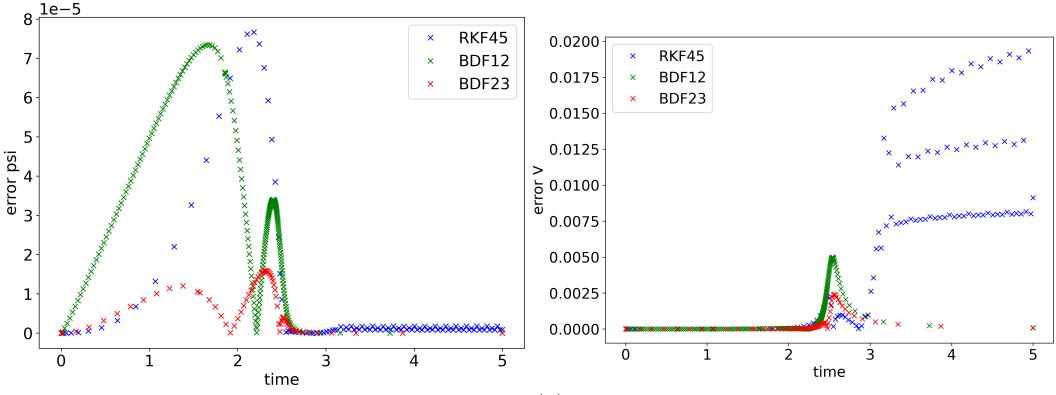


Figure 1.4: Evolution of the solution and the timestep sizes of the single state problem defined in Equation 1.2 with the implemented numerical schemes

Now, the accuracy of the three numerical methods is compared. Therefore, Figure 1.5a depicts the evolution of the absolute difference between the numerical solution and the analytical solution of the state variable  $\psi$ . It is of interest to compare this error with the absolute error in velocity, shown in Figure 1.5b. As  $\psi$  decreases, the error in  $\psi$  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  $\psi$  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 1.4a, 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  $\psi$  and in the velocity are not directly correlated, thus a small error in  $\psi$  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  $\psi$  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  $\psi$  (b) Evolution of the absolute error of the velocity  $V$

Figure 1.5: Evolution of the error for the single state problem defined in Equation 1.2 with the implemented numerical schemes

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 1.6, 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 largest 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.

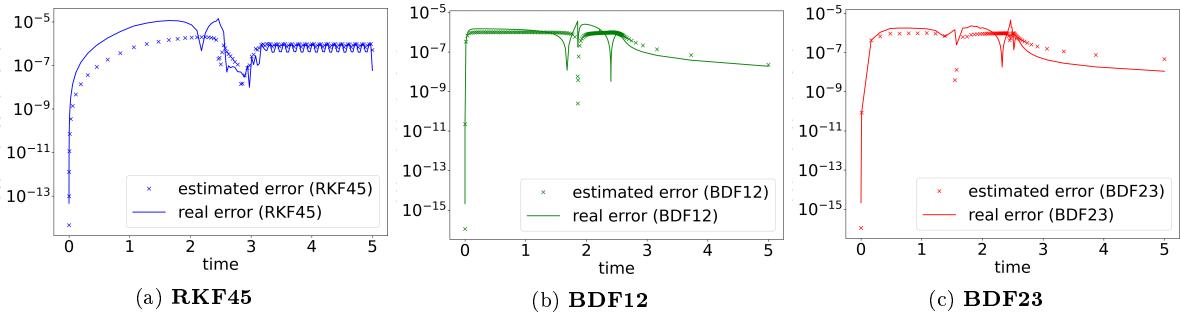


Figure 1.6: Evolution of the local truncation error and of the error estimate for the single state problem defined in Equation 1.2 with the implemented numerical schemes

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 2

## Two-Dimensional SEAS model

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

### 2.1 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:

$$\dot{\psi} = g(\psi, V) = \frac{bV_0}{L} e^{\frac{f_0 - \psi}{b}} - \frac{V}{V_0} \quad (2.1)$$

Friction law:

$$0 = \tau(U) - a\sigma_n(U)\text{arsinh}\left(\frac{V}{2V_0}e^{\frac{\psi}{a}}\right) - \eta V \quad (2.2)$$

### 2.2 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 2.1 depicts the half-domain on which the solution is calculated. The fault here is located on the left side of the domain.

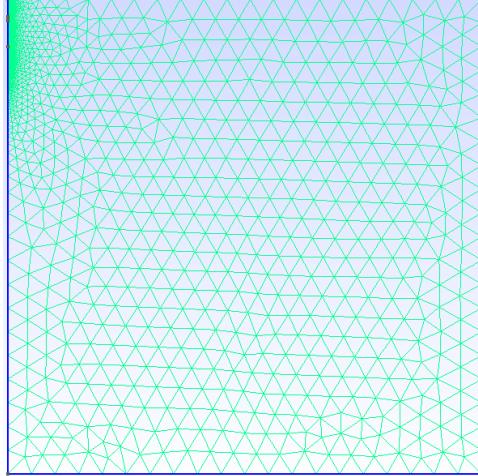


Figure 2.1: Space discretization of the BP1 problem with 200 elements on the fault

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  $V_0 = 10^{-6} m \cdot s^{-1}$ .

### 2.3 Formulation of the Discontinuous 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.

### 2.4 First Order Differential Equation

The problem stated in section 2.1 is a first order DAE, as the involving terms only depend on the unknown quantities and at most there first time derivatives:

$$\dot{S}_i = V_i \quad (2.3)$$

$$\dot{\psi}_i = g(\psi_i, V_i) = \frac{bV_0}{L} \left( e^{\frac{f_0 - \psi_i}{b}} - \frac{V_i}{V_0} \right) \quad (2.4)$$

$$0 = f_i(U, S, \psi, V) = \tau_i(U, S) - a\sigma_n \text{arsinh} \left( \frac{V_i}{2V_0} e^{\frac{\psi_i}{a}} \right) - \eta V_i \quad (2.5)$$

$$0 = AU - b(S) \quad (2.6)$$

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  $\psi_i$  are respectively the slip, the velocity and the value of the state variable on the fault. The matrix-vector system in Equation 2.6 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  $\llbracket U_j \rrbracket = U_j^+ - U_j^- = -S_i$ , 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.

### 2.4.1 Formulation as a first order ODE

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 2.5 from the state with an iterative solver.

$$x = \begin{pmatrix} S \\ \psi \end{pmatrix} \quad \dot{x} = \begin{pmatrix} \dot{S} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} V(S, \psi) \\ g(\psi, V) \end{pmatrix} = F \quad (2.7)$$

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

### 2.4.2 Formulation as a DAE

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.

$$x = \begin{pmatrix} S \\ \psi \\ V \end{pmatrix} \quad \dot{x} = \begin{pmatrix} \dot{S} \\ \dot{\psi} \\ \dot{V} \end{pmatrix} = \begin{pmatrix} V \\ g(\psi, V) \\ f(S, \psi, V) \end{pmatrix} \quad (2.8)$$

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  $\psi$  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  $S$ ,  $\psi$  and  $V$ , whereas in the ODE formulation of the problem, the error control is limited to the parameters  $S$  and  $\psi$ .

The PETSc interface requires us to provide a DAE in the form

$$F(t, x, \dot{x}) = G(t, x) \quad (2.9)$$

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:

$$x = \begin{pmatrix} S \\ \psi \\ V \end{pmatrix} \quad F(t, x, \dot{x}) = \begin{pmatrix} V - \dot{S} \\ g(\psi, V) - \dot{\psi} \\ f(S, \psi, V) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = G(t, x) \quad (2.10)$$

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.

### 2.4.3 Compact formulation as a DAE

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  $\psi$ , as previously in the ODE formulation.

$$x = \begin{pmatrix} S \\ \psi \end{pmatrix} \quad F(t, x, \dot{x}) = \begin{pmatrix} f(S, \psi, V) \\ \dot{\psi} - g(\psi, V) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = G(t, x) \quad (2.11)$$

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

## 2.5 Formulation of the Jacobian

To use implicit time solvers, the Jacobian matrix of the right-hand side of the ODE in (2.4) and (2.3) has to be computed in the Newton iteration. 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.

### 2.5.1 Analytical Derivation of the Jacobian matrix

Next, we try to derive an analytical expression of the Jacobian for both the ODE and the DAE formulation of the problem, so that the Newton iterations when using implicit methods can be solved efficiently.

#### Jacobian of the ODE system

For the ODE formulation in Equation 2.7, 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:

$$\mathbf{J}_{\psi_j, S_j}(F_i(\psi, V(\psi, S))) = \begin{pmatrix} \frac{\partial V_i}{\partial S_j} & \frac{\partial V_i}{\partial \psi_j} \\ \frac{\partial g(\psi_i, V_i)}{\partial S_j} & \frac{\partial g(\psi_i, V_i)}{\partial \psi_j} \end{pmatrix} \quad (2.12)$$

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 2.5, 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 2.16, 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:

$$\frac{\partial V}{\partial S} = - \left( \frac{\partial f}{\partial V} \right)^{-1} \frac{\partial f}{\partial S} \quad (2.13)$$

$$\frac{\partial V}{\partial \psi} = - \left( \frac{\partial f}{\partial V} \right)^{-1} \frac{\partial f}{\partial \psi} \quad (2.14)$$

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

$$\frac{\partial f_i(S, \psi, V)}{\partial \psi_j} = \left( -\frac{\sigma_n}{2V_0} \frac{Ve^{\frac{\psi_i}{a}}}{\sqrt{\frac{e^{\frac{2\psi_i}{a}}V_i^2}{4V_0^2} + 1}} \right) \delta_{ij} \quad (2.15)$$

$$\frac{\partial f_i(S, \psi, V)}{\partial V_j} = \left( -\frac{\sigma_n a}{2V_0} \frac{e^{\frac{\psi_i}{a}}}{\sqrt{\frac{e^{\frac{2\psi_i}{a}} V_i^2}{4V_0^2} + 1}} - \eta \right) \delta_{ij} \quad (2.16)$$

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.

$$\frac{\partial f_i}{\partial S_j} = \frac{d\tau_i(U)}{dS_j} = \frac{\partial \tau_i(U)}{\partial U_k} \frac{\partial U_k}{\partial S_j} + \frac{\partial \tau_i(U)}{\partial S_j} \quad (2.17)$$

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:

$$\frac{\partial b_i(S)}{\partial S_j} = \frac{\partial}{\partial S_j} \int_e \eta \{ \{ c_{mnkl} \epsilon_{kl}(w) \eta_n^e \} \} [U_m] + \frac{\delta_e}{|e|^\beta} [U_m] [w_m] dx \quad (2.18)$$

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:

$$\frac{\partial b_i(S)}{\partial S_j} = - \int_e \eta \{ \{ c_{jnkl} \epsilon_{kl}(w) \eta_n^e \} \} + \frac{\delta_e}{|e|^\beta} [w_j] dx \quad (2.19)$$

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:

$$\frac{\partial U_i}{\partial S_j} = A_{ik}^{-1} b_k(e^i) \quad (2.20)$$

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  $\tau_p = M_{rp}^{-1} e_{qr}^T w_q (\nabla u)_{kq} n_{kq}$ , 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  $(\nabla u)_{pq} = \frac{1}{2} (D_{lpq}^0 u_l^0 + D_{lpq}^1 u_l^1) + c_0 (E_{lq}^0 u_l^0 - E_{lq}^1 u_l^1 - f_q) n_{pq}$ . 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:

$$\frac{(\nabla u)_{pq}}{\partial u_k} = \frac{1}{2} \left( D_{lpq}^0 \frac{\partial u_l^0}{\partial u_k} + D_{lpq}^1 \frac{\partial u_l^1}{\partial u_k} \right) + c_0 \left( E_{lq}^0 \frac{\partial u_l^0}{\partial u_k} - E_{lq}^1 \frac{\partial u_l^1}{\partial u_k} \right) n_{pq} \quad (2.21)$$

$$= \frac{1}{2} (D_{lpq}^0 \delta_{lk}^0 + D_{lpq}^1 \delta_{lk}^1) + c_0 (E_{lq}^0 \delta_{lk}^0 - E_{lq}^1 \delta_{lk}^1) n_{pq} \quad (2.22)$$

$$= \frac{1}{2} (D_{kpq}^0 + D_{kpq}^1) + c_0 (E_{kq}^0 - E_{kq}^1) n_{pq} \quad (2.23)$$

And further we get:

$$\frac{\partial \tau_p}{\partial u_l} = M_{rp}^{-1} e_{qr}^T w_q \frac{(\nabla u)_{kq}}{\partial u_l} n_{kq} \quad (2.24)$$

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.

$$\frac{(\nabla u)_{pq}}{\partial S_j} = -c_0 \frac{\partial f_q}{\partial S_j} n_{pq} = -c_0 e_{qk}^T \frac{\partial S_k}{\partial S_j} n_{pq} = -c_0 e_{qj}^T n_{pq} \quad (2.25)$$

$$(2.26)$$

With this expression, we obtain:

$$\frac{\partial \tau_p}{\partial S_j} = -M_{rp}^{-1} e_{qr}^T w_q c_0 e_{qj}^T n_{kq} n_{kq} \quad (2.27)$$

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 2.17 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 2.12, 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:

$$\frac{\partial g_i}{\partial S_j} = -\frac{b}{L} \frac{\partial V}{\partial S} \quad (2.28)$$

$$\frac{\partial g_i}{\partial S_j} = -\frac{V_0}{L} e^{\frac{f_0 - \psi}{b}} \delta_{ij} - \frac{b}{L} \frac{\partial V}{\partial \psi} \quad (2.29)$$

### Jacobian of the DAE system

For the DAE formulation of the system in Equation 2.10, 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:

$$\mathbf{J}_{S_j, \psi_j, V_j} F_i(S, \psi, V) = \begin{pmatrix} \frac{\partial V_i}{\partial S_j} & \frac{\partial V_i}{\partial \psi_j} & \frac{\partial V_i}{\partial V_j} \\ \frac{\partial g_i(\psi, V)}{\partial S_j} & \frac{\partial g_i(\psi, V)}{\partial \psi_j} & \frac{\partial g_i(\psi, V)}{\partial V_j} \\ \frac{\partial f_i(S, \psi, V)}{\partial S_j} & \frac{\partial f_i(S, \psi, V)}{\partial \psi_j} & \frac{\partial f_i(S, \psi, V)}{\partial V_j} \end{pmatrix} \quad (2.30)$$

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.

$$\frac{\partial V_i}{\partial S_j} = 0 \quad \frac{\partial V_i}{\partial \psi_j} = 0 \quad \frac{\partial V_i}{\partial V_j} = \delta_{ij} \quad (2.31)$$

Similarly, the partial derivatives of the ageing law can be directly evaluated from its definition in Equation 2.4.

$$\frac{\partial g_i(\psi, V)}{\partial S_j} = 0 \quad \frac{\partial g_i(\psi, V)}{\partial \psi_j} = -\frac{V_0}{L} e^{\frac{f_0 - \psi_i}{b}} \delta_{ij} \quad \frac{\partial g_i(\psi, V)}{\partial V_j} = -\frac{b}{L} \delta_{ij} \quad (2.32)$$

For the friction law, the Jacobian matrices have already been calculated in Equation 2.15 for  $\frac{\partial f(S, \psi, V)}{\partial \psi}$ , in Equation 2.17 for  $\frac{\partial f(S, \psi, V)}{\partial S}$  and in Equation 2.16 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  $J$ .

$$J_N = -\frac{1}{h} I + J \quad (2.33)$$

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  $\psi$ . 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.

$$0 = \begin{pmatrix} \frac{1}{h} S_n \\ \frac{1}{h} \psi_n \\ 0 \end{pmatrix} - \begin{pmatrix} \frac{1}{h} S_{n+1} \\ \frac{1}{h} \psi_{n+1} \\ 0 \end{pmatrix} + \begin{pmatrix} V_{n+1} \\ g(\psi_{n+1}, V_{n+1}) \\ f(S_{n+1}, \psi_{n+1}, V_{n+1}) \end{pmatrix} \quad (2.34)$$

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

$$\mathbf{J}_N = \begin{pmatrix} -\frac{1}{h}\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{h}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{J}_S(S) & \mathbf{J}_\psi(S) & \mathbf{J}_V(S) \\ \mathbf{J}_S(\psi) & \mathbf{J}_\psi(\psi) & \mathbf{J}_V(\psi) \\ \mathbf{J}_S(V) & \mathbf{J}_\psi(V) & \mathbf{J}_V(V) \end{pmatrix} = \begin{pmatrix} -\frac{1}{h}\mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\frac{1}{h}\mathbf{I} + \frac{\partial g}{\partial \psi} & \frac{\partial g}{\partial V} \\ \frac{\partial f}{\partial S} & \frac{\partial f}{\partial \psi} & \frac{\partial f}{\partial V} \end{pmatrix} \quad (2.35)$$

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  $\mathbf{F}_{\dot{x}}$ :

$$\mathbf{J}_N = \frac{dF}{dx_n} = \mathbf{F}_{\dot{x}} \frac{d\dot{x}_n}{dx_n} + \mathbf{F}_x \quad (2.36)$$

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  $\sigma$ . 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 2.30 and  $\mathbf{F}_x$  contains negative identity matrices at the entries for the slip  $S$  and for the state variable  $\psi$ .

$$\mathbf{F}_x = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \frac{\partial g}{\partial \psi} & \frac{\partial g}{\partial V} \\ \frac{\partial f}{\partial S} & \frac{\partial f}{\partial \psi} & \frac{\partial f}{\partial V} \end{pmatrix} \quad \mathbf{F}_{\dot{x}} = \begin{pmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \quad (2.37)$$

With this two matrices, the Jacobian matrix for the Newton iteration of the implicit Euler method is exactly the same as calculated in Equation 2.35.

### Jacobian of the compact DAE system

The compact DAE formulation of the problem in Equation 2.11 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.

$$\mathbf{F}_{\dot{x}} = \begin{pmatrix} \frac{\partial f}{\partial V} & \mathbf{0} \\ -\frac{\partial g}{\partial V} & \mathbf{I} \end{pmatrix} \quad \mathbf{F}_x = \begin{pmatrix} \frac{\partial f}{\partial S} & \frac{\partial f}{\partial \psi} \\ \mathbf{0} & \frac{\partial g}{\partial \psi} \end{pmatrix} \quad \mathbf{G}_x = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \quad (2.38)$$

### 2.5.2 Verification of the Jacobian

The Jacobian matrix is needed to apply implicit numerical methods to solve 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  $\phi$  evaluated at the point  $x_n$ .

To verify the correctness of the analytic expression of the Jacobi matrix which has been set up in the previous section, it is applied in a Newton iteration to solve one timestep of the implicit Euler method. The function  $\phi$  and its Jacobian matrix are given by:

$$\phi(x) = -x + x^{(0)} + \Delta t F(x) \quad (2.39)$$

$$J_\phi(x) = -I + \Delta t J_F(x) \quad (2.40)$$

The vector  $x$  contains both the components related to the slip  $S$  and to the state variable  $\psi$  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  $\Delta t = 10^5$ 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 2.2.

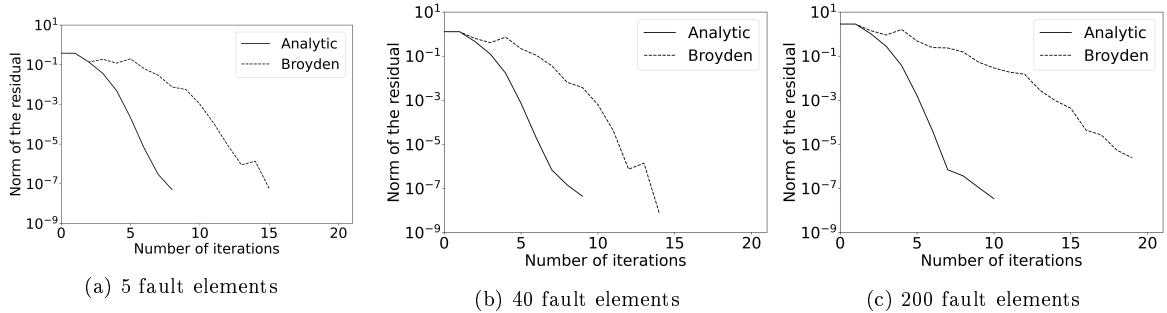


Figure 2.2: Evaluation of the L2 norm of the residual  $\phi(x_n)$  at each iteration of the Newton and Broyden methods

It can be immediately seen that the Newton method with an analytical expression of the Jacobian reaches much faster the required tolerance of  $10^{-8}$  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. ???? Add some convergence test ????

### 2.5.3 Limitations of the use of the Jacobian

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 subsection 2.5.1, 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  $\mathcal{O}(N^2)$ . 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  $\mathcal{O}(n^3)$ . 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. Notwithstanding, for such large domains, the question has to be asked whether a direct solver is still appropriate or whether iterative solvers yield acceptable results. Such considerations are however beyond the scope of this thesis (so far ???).

## 2.6 Second Order Differential Equation

So far, the original problem in section 2.1 has been formulated as a first order ODE in Equation 2.7, 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  $\psi$ . 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  $\psi$ . 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:

$$x = \begin{pmatrix} S \\ \psi \end{pmatrix} \quad \begin{pmatrix} \ddot{S} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} h(t, S, \psi, \dot{S}) \\ g(\psi, V) \end{pmatrix} \quad (2.41)$$

$$\Leftrightarrow \quad x = \begin{pmatrix} S \\ \psi \\ V \end{pmatrix} \quad \dot{x} = \begin{pmatrix} \dot{S} \\ \dot{\psi} \\ \dot{V} \end{pmatrix} = \begin{pmatrix} V \\ g(\psi, V) \\ h(t, S, \psi, V) \end{pmatrix} = F \quad (2.42)$$

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.

### 2.6.1 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 2.5. 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:

$$0 = \frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S} \frac{dS}{dt} + \frac{\partial f}{\partial \psi} \frac{d\psi}{dt} + \frac{\partial f}{\partial V} \frac{dV}{dt} \quad (2.43)$$

$$= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S} V + \frac{\partial f}{\partial \psi} g(\psi, V) + \frac{\partial f}{\partial V} h(t, S, \psi, V) \quad (2.44)$$

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 2.15 and Equation 2.16 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 2.17. After some transformations, the function  $h()$  can be expressed as:

$$h(t, S, \psi, V) = - \left( \frac{\partial f}{\partial V} \right)^{-1} \left( \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S} V + \frac{\partial f}{\partial \psi} g(\psi, V) \right) \quad (2.45)$$

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 2.12, an alternative definition of  $h()$  could be:

$$h(t, S, \psi, V) = \mathbf{J}_S \frac{dx}{dt} - \left( \frac{\partial f}{\partial V} \right)^{-1} \frac{\partial f}{\partial t} \quad (2.46)$$

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  $\tau$  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  $f_g$  of ??, and are added to the traction  $\tau$  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  $\tau$  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  $\psi$  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  $\mathcal{O}(N^2)$  and the remaining components of the vector set up in  $\mathcal{O}(n^2)$  because of the matrix-vector product in  $\tau$ , so the total complexity is  $\mathcal{O}(n^2 + N^2)$ . In contrast, the second order formulation only needs matrix-vector multiplications on the fault node, so its total complexity per evaluation is reduced to  $\mathcal{O}(n^2)$ . 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  $\mathcal{O}(n^2 + N^2)$  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 2.3 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.

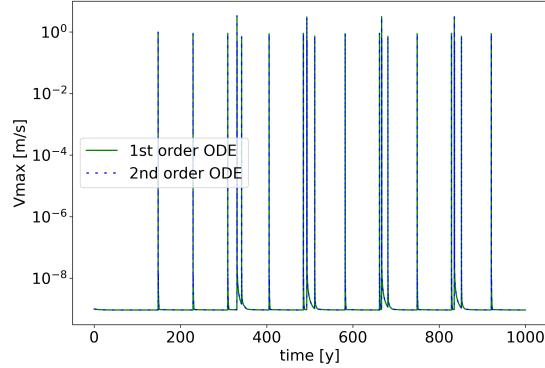


Figure 2.3: Maximum slip rate in the 1st and 2nd order formulations on a domain with 5 fault elements for 1000 years

### 2.6.2 Derivation of the analytic Jacobian

This will be tricky, involves Hessian matrices and stuff of the friction law.  $\rightarrow$  next thing to implement

### 2.6.3 Error estimation

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  $\tau$  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 2.4a shows the maximum value of the friction law for varying 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 2.4b, 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.

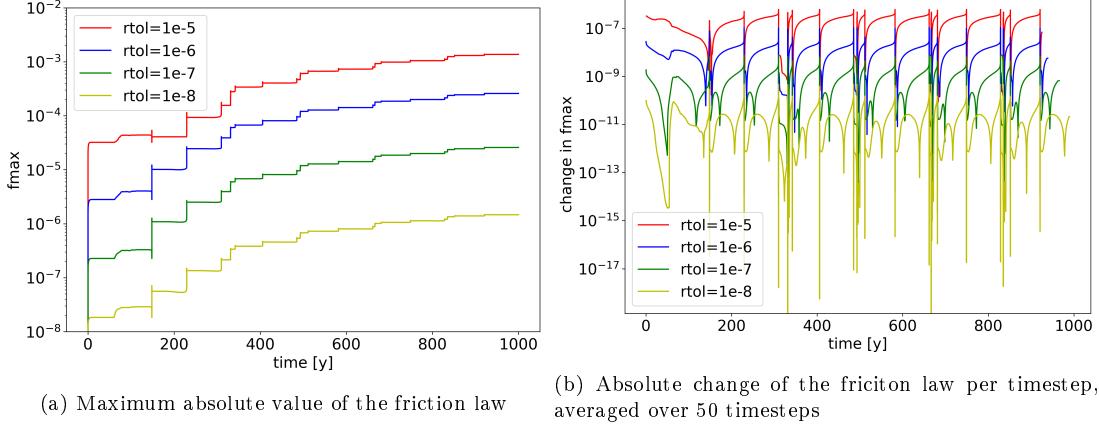


Figure 2.4: Evolution of the value of the friction law of the 2nd order formulation with 5 elements on the fault over 1000 years with varying relative tolerances in the slip rate  $V$

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  $\psi$ , 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 2.5a. Since each fault element 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 2.5b, 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.

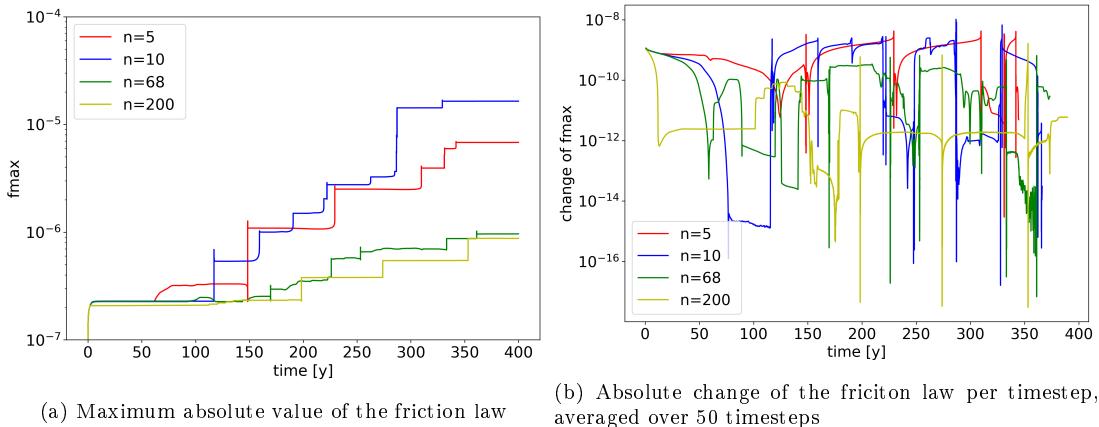


Figure 2.5: Evolution of the value of the friction law of the 2nd order formulation over 400 years with varying domain sizes, where  $n$  denotes the number of fault elements

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.

#### 2.6.4 Analytic derivation of an upper bound for the LTE

In this section, we try to mathematically derive an upper bound for the local truncation error from the defined tolerances in  $S$ ,  $\psi$  and  $V$ . With it, we can roughly estimate the error in the friction law without having to solve the Poisson problem at every timestep.

#### 2.6.5 Normalization of the slip rate

Does it make sense to evaluate the slip rate from the friction law sometimes to normalize the friction law back to 1 if the error is too high?

### 2.7 Numerical treatment in SEAS

For SEAS, the solution vector  $u$  combines the velocity  $V$  and the state variable  $\psi$  for all interpolation points at the edge of the fault elements. The functional  $F$  is not used and takes as value only the time derivative of the solution vector  $\dot{u}$ . The call of the right-hand-side function  $G(u, t)$  combines the solving of the algebraic and the differential equations. At each step, following steps are performed:

1. The current displacement in the whole system is calculated to fulfill the Poisson or elasticity equations by solving Equation 2.6
2. The values of  $\tau(U)$  and  $\sigma_n(U)$  are calculated on the fault according to the updated displacements
3. The iteration over all interpolation points on the fault is performed. At each interpolation step:
  - the velocity  $V_i$  is obtained from the friction law in Equation 2.5 with the bisection method,
  - and the actual right hand side  $G(u, t)$  of the ODE is evaluated using equations 2.4 and 2.3.

#### 2.7.1 Explicit methods

Runge-Kutta schemes with error correction term

- Bogacki-Shampine (RKBS3): 2nd order with 3rd order error correction
- Dormand-Prince (RKDP5): 4th order with 5th order error correction

#### 2.7.2 Implicit methods

BDF methods of order 1-6 describe coefficients and error estimation with Lagrange polynomials t method

#### 2.7.3 Implicit - explicit methods

Not really a thing yet, but consider for later. Needs to reformulate the DAE with a nonzero  $G()$ , can only be used for the ageing law  $g()$ .

## 2.7.4 Switching between different formulations

-> next thing to implement

## 2.7.5 Definition of the tolerances

## 2.7.6 Time adaptivity

## 2.8 Results

### 2.8.1 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 2.6 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.

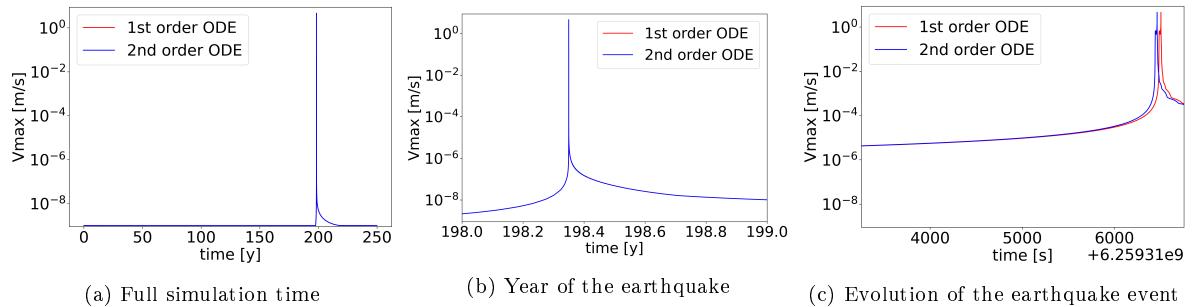


Figure 2.6: Evolution of the maximal slip rate  $V$  on the fault for different solvers on the symmetric two-dimensional BP1 problem with 200 elements on the fault

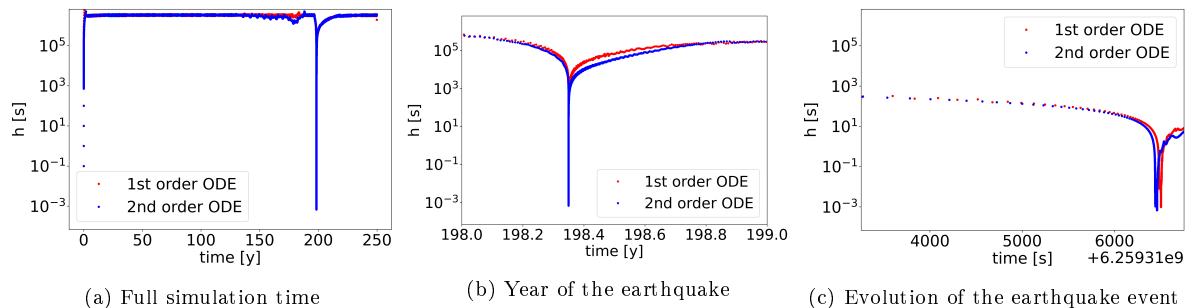


Figure 2.7: Evolution of the time step size  $h$  for different solvers on the symmetric two-dimensional BP1 problem with 200 elements on the fault

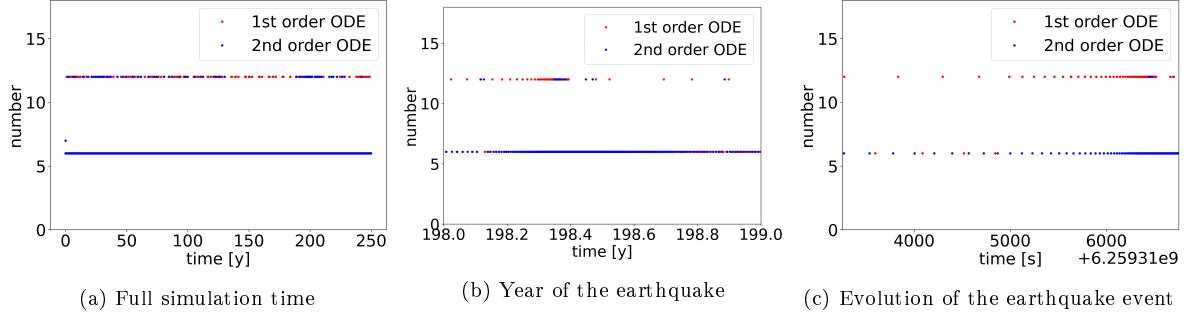


Figure 2.8: Number of evaluations of the right hand side of the ODE in each time iteration for different solvers on the symmetric two-dimensional BP1 problem with 200 elements on the fault

**Legend:** — 1st order ODE formulation, — 2nd order ODE formulation, — extended DAE formulation, — compact DAE formulation

## 2.9 Relative and Absolute Tolerances

→ put that in the numerics section

### 2.9.1 Behaviour of the Error

Tolerances play a crucial role for 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:

$$\left\| \frac{u_i - u_i^{(e)}}{t_i} \right\|_{\infty} = \max_i \left| \frac{u_i - u_i^{(e)}}{t_i} \right| \leq 1 \quad (2.47)$$

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.

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  $t_i^r$ .

$$t_i = t_i^a + \max(u_i, u_i^{(e)}) t_i^r \quad (2.48)$$

Since some components of the solution vector correspond to the values of the state variable  $\psi$  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  $t_{\psi}^a$  and  $t_{\psi}^r$  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 2.9, which depicts the maximal absolute and relative errors ( $\max_i |u_i - u_i^{(e)}|$  and  $\max_i \left| \frac{u_i - u_i^{(e)}}{u_i} \right|$ ) for the respective components of  $\psi$  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.

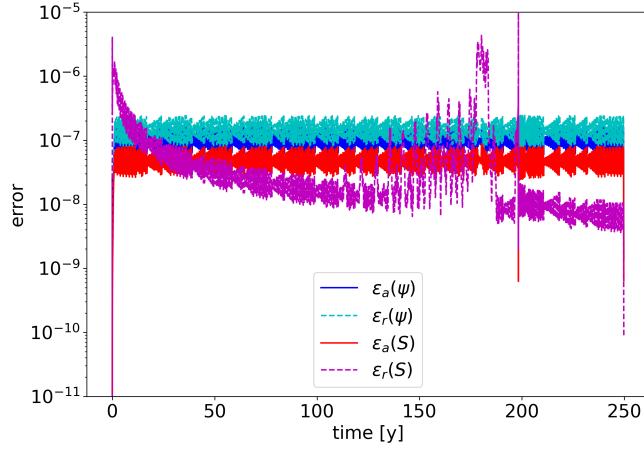


Figure 2.9: Maximum relative and absolute errors using the RKDP5 method on a fault with 200 elements with an absolute and relative error tolerance of  $10^{-7}$

Over the whole simulation time, the state variable  $\psi$  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 2.10 provides an explanation for these large errors. In the beginning, the slip at each node is below 1, thus the overall tolerance

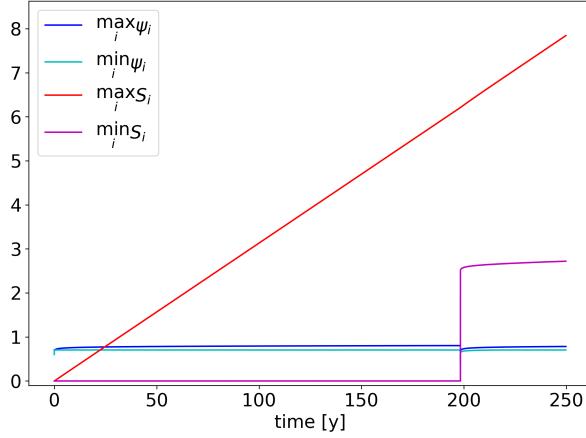


Figure 2.10: Maximum and minimum values of the state variable  $\psi$  and of the slip  $S$  over time. Simulation performed with the RKDP5 method on a fault with 200 elements with an absolute and relative error tolerance of  $10^{-7}$

### 2.9.2 Tolerance of the State Variable Dependent on the Slip

Overall, the absolute error of the slip is inferior to the absolute error of the state variable. From a physical point of view, the slip is a relevant quantity to describe earthquakes whereas the state variable  $\psi$  is only required to solve the DAE. Therefore, the absolute and relative tolerances are defined as external requirements for the slip only, and the tolerances for the state variable have to be chosen in a way to

achieve the best numerical performance without loosing physical accuracy. In the following equations, we try to evaluate the highest acceptable tolerances for the state variable in function of the provided tolerances for the slip. The relation between  $S$  and  $\psi$  is described by the friction law in Equation 2.5, which is evaluated at a constant velocity  $V^*$ . For a maximal absolute slip error  $\epsilon_S^a$ , the largest acceptable absolute error of the state variable  $\epsilon_\psi^a$  can be calculated by equalizing the induced error in the friction law under constant velocity.

$$f_{V^*}(S + \epsilon_S^a, \psi) - f_{V^*}(S, \psi) = f_{V^*}(S, \psi + \epsilon_\psi^a) - f_{V^*}(S, \psi) \quad (2.49)$$

$$\tau(U(S + \epsilon_S^a)) - \tau(U(S)) = a\sigma_n \operatorname{arsinh} \left( \frac{V^*}{2V_0} \right) \frac{1}{a} \left( e^{\frac{\psi + \epsilon_\psi^a}{a}} - e^{\frac{\psi}{a}} \right) \quad (2.50)$$

$$\tau(B(S + \epsilon_S^a)) - \tau(BS) = a\sigma_n \operatorname{arsinh} \left( \frac{V^*}{2V_0} \right) e^{\frac{\psi}{a}} \left( e^{\frac{\epsilon_\psi^a}{a}} - 1 \right) \quad (2.51)$$

As already discussed in the formulation of the Jacobian matrix, the displacement  $U(S)$  is linear in  $S$  and can therefore be expressed with the linear transformation matrix  $B$  as  $U(S) = BS$ . Further transformations of the left side of the equation are performed in index notation, where  $\mathbb{1}$  denotes a vector in which all components are equal to one.

$$(\nabla u(S))_{pq} = \frac{1}{2} (D_{lpq}^0 u_l^0 + D_{lpq}^1 u_l^1) + c_0 (E_{lq}^0 u_l^0 - E_{lq}^1 u_l^1 - f_q) n_{pq} \quad (2.52)$$

$$\begin{aligned} (\nabla u(S + \epsilon_S^a))_{pq} &= \frac{1}{2} (D_{lpq}^0 (u_l^0 + B_{lj} \mathbb{1}_j \epsilon_S^a) + D_{lpq}^1 (u_l^1 + B_{lj} \mathbb{1}_j \epsilon_S^a)) \\ &\quad + c_0 (E_{lq}^0 (u_l^0 + B_{lj} \mathbb{1}_j \epsilon_S^a) - E_{lq}^1 (u_l^1 + B_{lj} \mathbb{1}_j \epsilon_S^a) - f_q) n_{pq} \end{aligned} \quad (2.53)$$

$$(\nabla u(S + \epsilon_S^a))_{pq} - (\nabla u(S))_{pq} = \left( \frac{1}{2} (D_{lpq}^0 + D_{lpq}^1) + c_0 (E_{lq}^0 - E_{lq}^1) n_{pq} \right) B_{lj} \mathbb{1}_j \epsilon_S^a \quad (2.54)$$

$$\tau_p(B(S + \epsilon_S^a)) - \tau_p(BS) = M_{rp}^{-1} e_{qr}^T w_q ((\nabla u(S + \epsilon_S^a))_{kq} - (\nabla u(S))_{kq}) n_{kq} \quad (2.55)$$

We can see that the error in  $\tau$  does not depend on the slip  $S$  anymore and is proportional to the absolute slip error  $\epsilon_S^a$ . We can thus define a matrix  $C$  such that  $\tau_p(B(S + \epsilon_S^a)) - \tau_p(BS) = C_{pj} \mathbb{1}_j \epsilon_S^a$ . The absolute error of the state variable is then directly proportional to  $\epsilon_S^a$ .

$$\epsilon_\psi^a = a \ln \left( 1 + \frac{C_{pj} \mathbb{1}_j}{a \sigma_n \operatorname{arsinh} \left( \frac{V^*}{2V_0} \right) e^{\frac{\psi}{a}}} \epsilon_S^a \right) \quad (2.56)$$

To provide an upper bound for  $\epsilon_\psi^a$ , the smallest value of the proportionality factor of  $\epsilon_S^a$  has to be determined because the logarithm function is increasing and the smallest allowed value of the error is searched. For the numerator of the ratio, to calculate  $\min_p |C_{pj} \mathbb{1}_j|$  minimizes the ratio. Note that the matrix  $C$  depends on the geometry so this term cannot be provided in general in beforehand but has to be chosen with respect to the used space discretization. To minimize the ratio, the terms on the denominator have to be maximized. Because the arsinh function increases monotonously, the choice of  $V_{max}$  for the velocity component provides an appropriate upper bound. Similarly,  $\psi$  should be replaced by an upper limit  $\psi_{max}$ . The value of the parameter  $a$  depends on the fault location and varies between  $a_{min}$  and  $a_{max}$ . It appears once at the beginning of the denominator, where its maximal value shall be chosen, and once in the exponential, where its minimal value is required. If all those values are plugged into the expression, the error in  $\psi$  can be evaluated as following:

$$\epsilon_\psi^a = a_{min} \ln \left( 1 + \frac{\min_p |C_{pj} \mathbb{1}_j|}{a_{max} \sigma_n \operatorname{arsinh} \left( \frac{V_{max}}{2V_0} \right) e^{\frac{\psi_{max}}{a_{min}}}} \epsilon_S^a \right) \quad (2.57)$$

The upper bounds for the velocity and for the state variable can be deduced by already obtained simulation results and an appropriate choice could be the values  $V_{max} = 5.0 \text{ ms}^{-1}$  and  $\psi_{max} = 0.85$ . In the simulation settings,  $a$  increases from  $a_{min} = 0.010$  at the surface to  $a_{max} = 0.025$  at high depths. With these values, all terms take reasonable values, except for the exponential  $e^{\frac{\psi_{max}}{a_{min}}} = e^{85} \approx 8 \cdot 10^{36}$ . In the vicinity of 1, the logarithm can be approximated by a linear function with gradient 1. Under consideration of the initial factor  $a_{min}$ , the absolute error tolerance in  $\psi$  has to be 38 magnitudes below the tolerance in the slip to prevent any additional error. Such a low tolerance is not achievable in any realistic scenario, so

the state variable participates in making the simulation less accurate. When the tolerances for the state variable have to be set, and only an absolute tolerance for the slip is provided, it is the best to chose them as low as possible, and still the state variable will contribute to the error of the simulation. On the other hand, very low tolerances prevent the choice of larger timesteps and an appropriate choice of the tolerance for the state variable which is not too low to ensure reasonable execution times still needs to be determined.

### 2.9.3 Tolerance of the State Variable Dependent on the Slip Rate

So far, the error in the slip rate  $V$  is not directly regulated by any tolerance value. Since it is calculated from the slip and from the state variable, the error in  $V$  depends on the chosen tolerances for those two parameters. As a physical quantity, it might be interesting to provide a tolerance for the slip rate too. Unlike for the slip, the tolerance for the slip rate should also include a relative error tolerance  $t_V^r$  in addition to the already discussed absolute error tolerance  $t_V^a$ . Indeed, the slip rate takes values around  $10^{-6}ms^{-1}$  in the aseismic phase and peaks at values above  $10^9ms^{-1}$  during the earthquake event. The exclusive use of an absolute error tolerance, with a value below the slip rate of the aseismic evolution, is too restrictive during the earthquake. An appropriate relative error tolerance, which is scaled by the current velocity value and added to the absolute one can reasonably restrict the error. In the aseismic phase, the relative error tolerance does not have much effect since it is scaled down by the very low slip rate and outmatched by the presumably higher absolute tolerance. For example, the choices  $t_V^a = 10^{-10}$  and  $t_V^r = 10^{-7}$  allow for a relative error of the order  $10^{-4}$  during the aseismic phase and of the order  $10^{-7}$  during the earthquake. Such a choice reflects the concept that a higher accuracy is needed to simulate an earthquake than to simulate aseismic slip.

The typical approach would be to estimate it with the difference between the values obtained from the numerical solution and its embedded solution. The newly chosen timestep shall then only allow solutions which fulfill the tolerances in the entire state vector and the additional tolerance in error of the slip rate. Alternatively, the error tolerance of the state variable, which is not properly defined from physical constrains yet can be used to restrict the slip rate. To obtain the relation between error in slip rate  $V$  and state variable  $\psi$ , the same analysis as in the previous subsection has to be performed. This time, the error in the friction law is evaluated under the assumption of a constant slip  $S^*$ .

$$f_{S^*}(V + \epsilon_V^a, \psi) - f_{S^*}(V, \psi) = f_{S^*}(V, \psi + \epsilon_\psi^a) - f_{S^*}(V, \psi) \quad (2.58)$$

$$f_{S^*}(V, \psi) + \frac{d}{dV} f_{S^*}(V, \psi) \epsilon_V^a - f_{S^*}(V, \psi) + \mathcal{O}((\epsilon_V^a)^2) = a \sigma_n \text{arsinh} \left( \frac{V}{2V_0} \right) \frac{1}{a} \left( e^{\frac{\psi + \epsilon_\psi^a}{a}} - e^{\frac{\psi}{a}} \right) \quad (2.59)$$

$$\left( \frac{a \sigma_n e^{\frac{\psi}{a}}}{2V_0 \sqrt{\left( \frac{V}{2V_0} \right)^2 + 1}} - \eta \right) \epsilon_V^a + \mathcal{O}((\epsilon_V^a)^2) = a \sigma_n \text{arsinh} \left( \frac{V}{2V_0} \right) e^{\frac{\psi}{a}} \left( e^{\frac{\epsilon_\psi^a}{a}} - 1 \right) \quad (2.60)$$

$$\epsilon_\psi^a = a \ln \left( 1 + \left( \frac{1}{\text{arsinh} \left( \frac{V}{2V_0} \right) \sqrt{V^2 + (2V_0)^2}} - \frac{\eta}{a \sigma_n \text{arsinh} \left( \frac{V}{2V_0} \right) e^{\frac{\psi}{a}}} \right) \epsilon_V^a + \mathcal{O}((\epsilon_V^a)^2) \right) \quad (2.61)$$

The exponential  $e^{\frac{\psi}{a}}$  with the extremely high value appears again in one of the two ratios. The term with  $\eta$  can thus be neglected to estimate the error. In the remaining summand, an appropriate estimate value for the slip rate  $V$  needs to be determined. As stated previously, the absolute error tolerance is only relevant for the aseismic phase, in which the slip rate never exceeds the tectonic slip  $V_0$ , which can be taken as an upper bound for the slip rate here. The absolute error is then calculated by:

$$\epsilon_\psi^a = a_{min} \ln \left( 1 + \frac{1}{\text{arsinh} \left( \frac{1}{2} \right) \sqrt{5} V_0} \epsilon_V^a \right) \quad (2.62)$$

During an earthquake, the relative error tolerance of the slip rate is relevant. To evaluate it, we now investigate the relation between the relative errors of the slip rate and the state variable.

$$f_{S^*}((1 + \epsilon_V^r)V, \psi) - f_{S^*}(V, \psi) = f_{S^*}(V, (1 + \epsilon_\psi^r)\psi) - f_{S^*}(V, \psi) \quad (2.63)$$

$$f_{S^*}(V, \psi) + \frac{d}{dV} f_{S^*}(V, \psi) \epsilon_V^r V - f_{S^*}(V, \psi) + \mathcal{O}((\epsilon_V^r V)^2) = a \sigma_n \operatorname{arsinh} \left( \frac{V}{2V_0} \right) \frac{1}{a} \left( e^{\frac{(1+\epsilon_\psi^r)\psi}{a}} - e^{\frac{\psi}{a}} \right) \quad (2.64)$$

$$\left( \frac{a \sigma_n e^{\frac{\psi}{a}}}{2V_0 \sqrt{\left( \frac{V}{2V_0} \right)^2 + 1}} - \eta \right) \epsilon_V^r V + \mathcal{O}((\epsilon_V^r V)^2) = a \sigma_n \operatorname{arsinh} \left( \frac{V}{2V_0} \right) e^{\frac{\psi}{a}} \left( e^{\frac{\epsilon_\psi^r}{\psi} a} - 1 \right) \quad (2.65)$$

$$\epsilon_\psi^r = \frac{a}{\psi} \ln \left( 1 + \left( \frac{1}{\operatorname{arsinh} \left( \frac{V}{2V_0} \right) \sqrt{V^2 + (2V_0)^2}} - \frac{\eta}{a \sigma_n \operatorname{arsinh} \left( \frac{V}{2V_0} \right) e^{\frac{\psi}{a}}} \right) \epsilon_V^r V + \mathcal{O}((\epsilon_V^r V)^2) \right) \quad (2.66)$$

Again, the term with  $\eta$  can be neglected because of the exponential. For elements, where the relative tolerance is relevant, the slip rate is much larger than the tectonic slip, so  $V_0^2$  can also be neglected below the square root when added to  $V^2$ . The relative error in  $\psi$  can be estimated as:

$$\epsilon_\psi^r = \frac{a_{min}}{\psi_{max}} \ln \left( 1 + \frac{1}{\operatorname{arsinh} \left( \frac{V_{max}}{2V_0} \right)} \epsilon_V^r \right) \quad (2.67)$$

The absolute and relative errors in  $\psi$  associated to the respective errors in the slip rate are represented in Figure 2.11. The curves show the evaluation of Equation 2.62 and Equation 2.67 for small error values in  $V$ . The various parameters are set to  $a_{min} = 0.01$ ,  $\psi_{max} = 0.85$ ,  $V_0 = 10^{-6} \text{ms}^{-1}$  and  $V_{max} = 5.0 \text{ms}^{-1}$ . To use the previous example again, if an absolute error tolerance in the slip rate of  $t_V^a = 10^{-10}$  and a relative tolerance of  $t_V^r = 10^{-7}$  are required, it would translate to the tolerances in the state variable to  $t_\psi^a = 9.29 \cdot 10^{-7}$  and  $t_\psi^r = 2.77 \cdot 10^{-11}$ . However, these values cannot be used as such in the simulation, since the assumption that the absolute tolerance can be neglected during the earthquake and inversely that the relative tolerance is insignificant in the aseismic phase only holds for the slip rate but not for the state variable, whose value remains between 0.75 and 0.85. It makes more sense to define an absolute tolerance for the aseismic slip  $t_\psi^{as} = 9.29 \cdot 10^{-7}$  and for the earthquake  $t_\psi^{eq} = \psi_{min} 2.77 \cdot 10^{-11} = 6.48 \cdot 10^{-11}$  in the state variable.

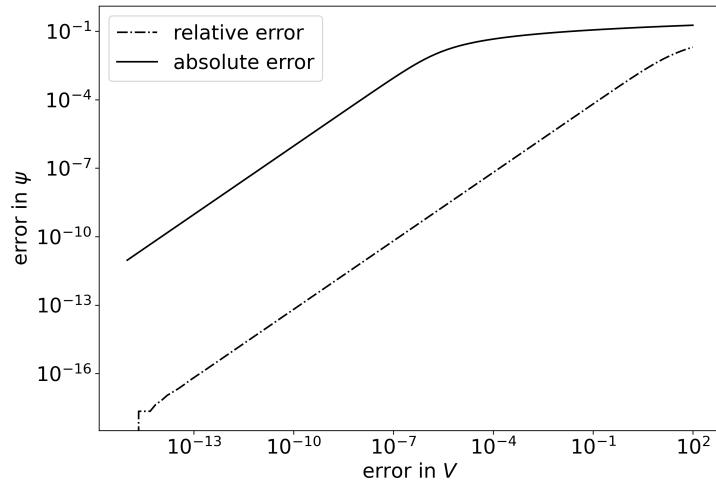


Figure 2.11: Absolute and relative error in  $\psi$  in function of the error in the slip rate

$$F(x, y) = 0, y(x) \quad x \in \mathbb{R}^n, y \in \mathbb{R}^m, F \in \mathbb{R}^m$$

$$0 = \frac{dF_k}{dx_j} = \frac{\partial F_k}{\partial x_i} \frac{dx_i}{dx_j} + \frac{\partial F_k}{\partial y_i} \frac{dy_i}{dx_j} \quad (2.68)$$

$$\frac{dy_i}{dx_j} = \frac{\partial y_i}{\partial x_l} \frac{dx_l}{dx_j} \quad (2.69)$$

$$\Rightarrow 0 = \frac{\partial F_k}{\partial x_i} \frac{dx_i}{dx_j} + \frac{\partial F_k}{\partial y_i} \frac{\partial y_i}{\partial x_l} \frac{dx_l}{dx_j} \quad (2.70)$$

If the components of  $x$  do not depend on each other (no ODE):  $\frac{dx_i}{dx_j} = \delta_{ij}$

We obtain exactly the implicit function theorem:

$$0 = \frac{\partial F_k}{\partial x_j} + \frac{\partial F_k}{\partial y_i} \frac{\partial y_i}{\partial x_j} \quad (2.71)$$

$$\Rightarrow \frac{\partial y}{\partial x} = - \left( \frac{\partial F}{\partial y} \right)^{-1} \frac{\partial F}{\partial x} = -F_y^{-1} F_x \quad (2.72)$$

If the components of  $x$  depend on each other (autonomous ODE):  $\frac{dx_i}{dx_j} \neq \delta_{ij}$  is singular. The problem cannot be solved.

# Chapter 3

## Extension to More Complex Problems

### 3.1 Elasticity Equation

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

### 3.2 Three-Dimensional Simulations

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

### 3.3 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

- [Bro65] C. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593, 1965.
- [Jam78] M. James. The generalised inverse. *The Mathematical Gazette*, 62(420):109–114, 1978.
- [Pen55] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.
- [Rap97] Joseph Raphson. *Numerical Initial Value Problems in Ordinary Differential Equations*. Th. Braddell, 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.