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.

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

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

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.

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

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.

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

$$0 = g(\psi, V) \tag{1.5}$$

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 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. 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.

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

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 ψ_n and ψ_{n+1} are evaluated with respect to the unknown ψ_{n+2}.

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

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

The idea is to add equations (1.7) and (1.8), 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:

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

By the choice of α in Equation 1.10, 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 1.11.

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

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

Because of the DAE form, the values of $\psi n + 1$ 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:

$$F(\Psi) = 0 \tag{1.12}$$

where Ψ stands respectively for ψ_{n+1} and ψ_{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.

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

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

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-adapative BDF3 scheme is given by:

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

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

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

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 ϵ. The controller C is a function

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

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 Φ.

$$r_{n+1} = \Phi h_n^k \tag{1.19}$$

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:

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

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:

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

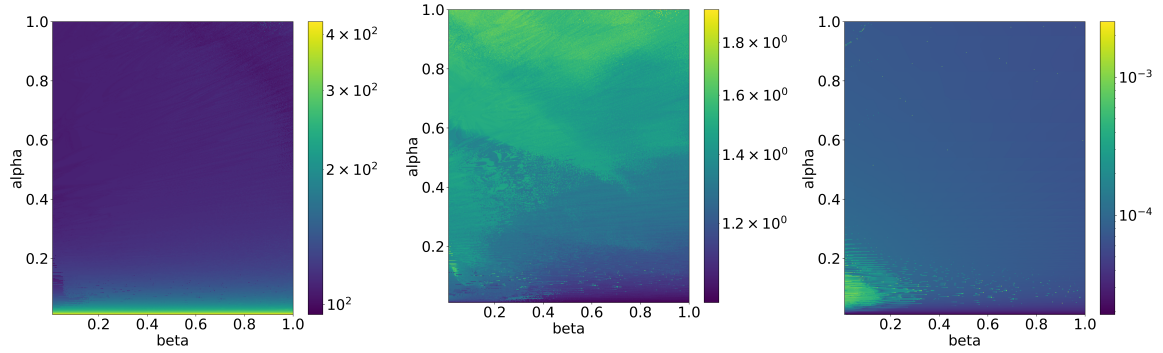
The initial assumption in Equation 1.19 that the error exposes an asymptotic behaviour is not always given. Especially for stiff problems, such a relationship between the timestep size and the local error is only true for very small timesteps.

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:

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

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 α and β. 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.

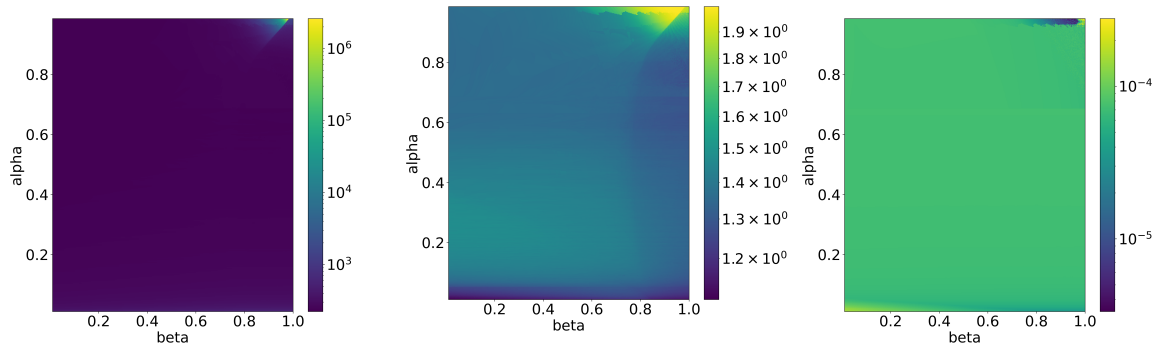


(a) Number of required timesteps

(b) Average number of iterations per timestep

(c) Absolute error of the variable ψ

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

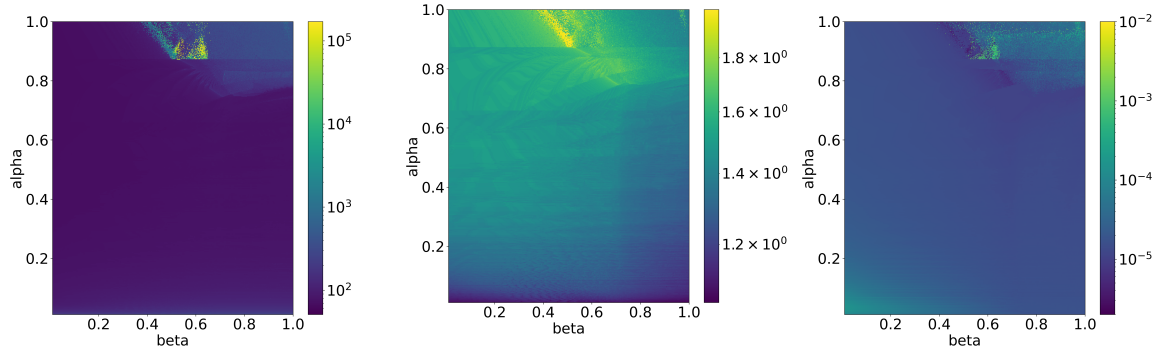


(a) Number of required timesteps

(b) Average number of iterations per timestep

(c) Absolute error of the variable ψ

Figure 1.2: Impact of the parameters α and β of the PI controller on the number of required iterations to solve the model problem from Equation 1.1 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 1.3: Impact of the parameters α and β of the PI controller on the number of required iterations to solve the model problem from Equation 1.1 using the implicit BDF method of second order (**BDF23**) for a simulation time of $t = 5.0s$

We chose as parameters:

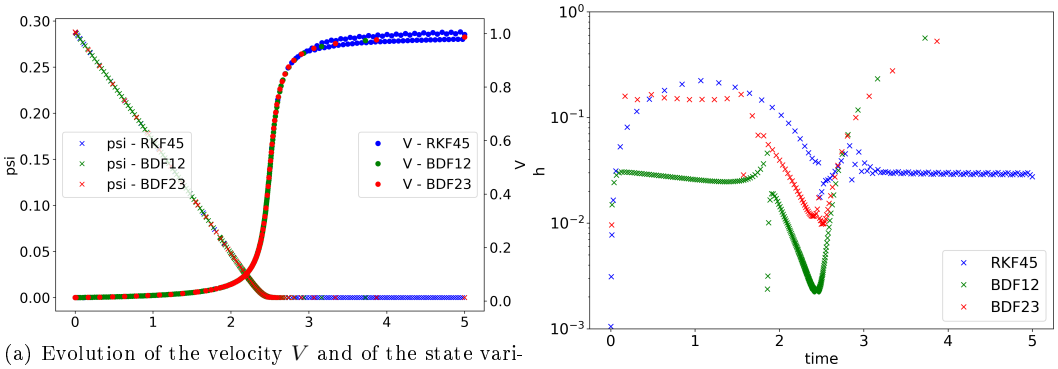
- for **RKF45**: $\alpha = 0.4$ and $\beta = 0.6$
- for **BDF12**: $\alpha = 0.9$ and $\beta = 0.4$
- for **BDF23**: $\alpha = 0.9$ and $\beta = 0.4$

—Put some more explanations here —
—Find the balance between number of iterations and accuracy —
—Numerical instabilities for the implicit methods for high values of α and β —
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 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 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 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.

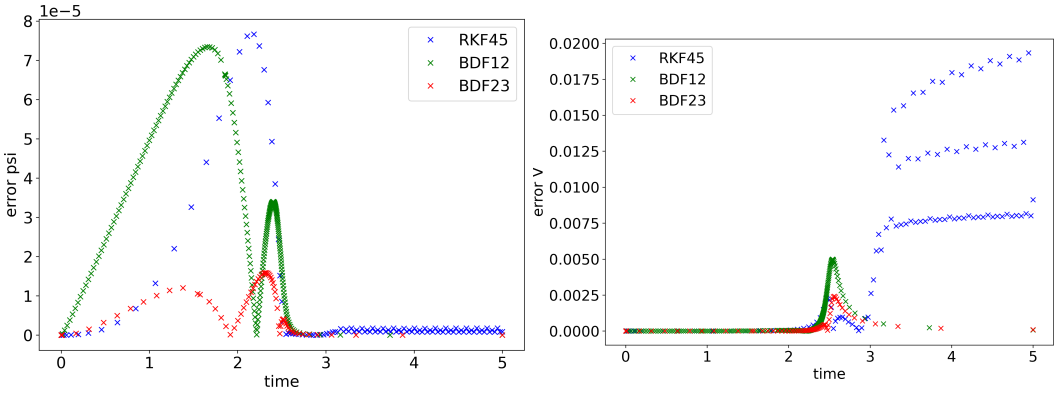


(a) Evolution of the velocity V and of the state variable ψ

(b) Evolution of the timestep size h_n

Figure 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 ψ. It is of interest to compare this error with the absolute error in velocity, shown in Figure 1.5b. 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 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 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.

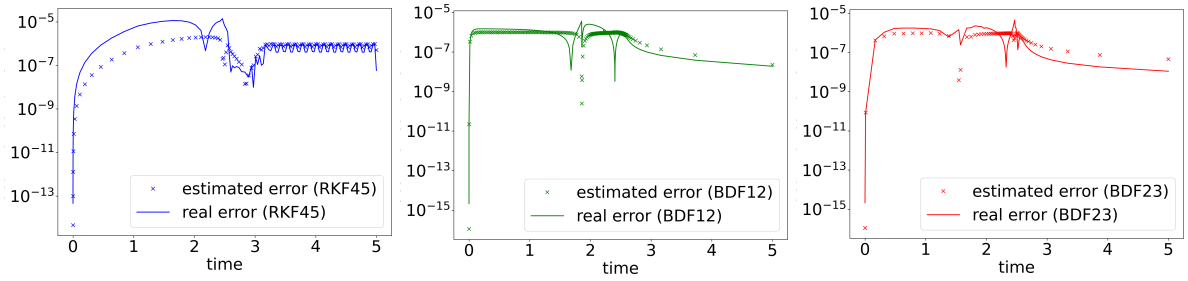


(a) Evolution of the absolute error of the state variable ψ

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

Figure 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 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 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:

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

Friction law:

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

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.

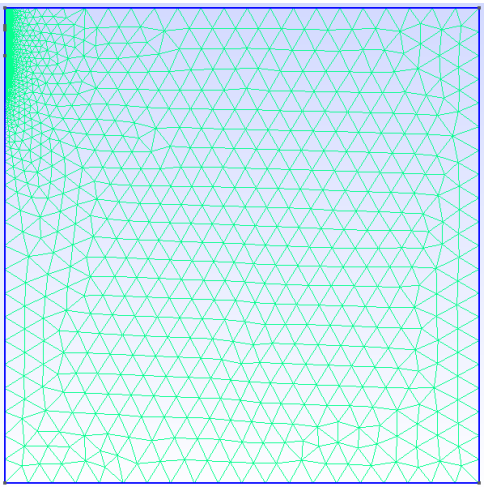


Figure 2.1: Space discretization of the BP1 problem with 200 elements on the fault

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 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.

2.4 Formulation of the DAE

The problem stated in section 2.1 can be written in the form of a DAE:

$$0 = AU - b(S) \tag{2.3}$$

$$0 = f(U_i, \psi_i, V_i) = \tau(U_i) - \sigma_n(U_i) a \operatorname{arsinh}\left(\frac{V_i}{2V_0}\right) e^{\frac{\psi}{a}} - \eta V_i \tag{2.4}$$

$$\dot{\psi}_i = g(\psi_i, V_i) = \frac{bV_0}{L}\left(e^{\frac{f_0 - \psi_i}{b}} - \frac{V_i}{V_0}\right) \tag{2.5}$$

$$\dot{S}_i = V_i \tag{2.6}$$

The index i refers to all interpolation points of the elements located on the fault, meaning that S_i, V_i and ψ_i are respectively the displacement, the velocity and the value of the state variable on the fault. In the two-dimensional case, the displacement S_i at the fault is related to U_i by $[\![U_i]\!] = U_i^+ - U_i^- = -S_i$. The displacements U_i^+ and U_i^- correspond to the displacements on the two sides of the fault, for the current symmetric problem, they have the same magnitude with opposite signs.

2.4.1 Analytical Derivation of the Jacobian matrix

To use implicit time solvers, the Jacobian matrix of the right-hand side of the ODE in (2.5) and (2.6) 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]. In our case, it is possible to evaluate the Jacobian analytically. If we define the solution vector x and the corresponding ODE as

$$x = \begin{pmatrix} \vdots \\ \S_i \\ \psi_i \\ \vdots \end{pmatrix} \qquad \text{and} \qquad \dot{x} = \begin{pmatrix} \vdots \\ \dot{S}_i \\ \dot{\psi}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ V_i \\ g(\psi_i, V_i) \\ \vdots \end{pmatrix} = F_i(V_i, \psi_i) \tag{2.7}$$

The Jacobian $J_{\psi,S}(F)$ consists of four block matrices, which each contains the Jacobians:

$$J_{\psi_h,S_j}\left(F\left(\psi_i, V_i(\psi_i, S_i)\right)\right) = \begin{pmatrix} \frac{\partial V_i}{\partial S_j} & \frac{\partial V_i}{\partial \psi_j} \\ \frac{\partial g(\psi_i,V_i)}{\partial S_j} & \frac{\partial g(\psi_i,V_i)}{\partial \psi_j} \end{pmatrix} \tag{2.8}$$

To calculate the partial derivatives of $V_i(\psi_i, S_i)$, we need to consider the friction law $f(U, \psi, V)$. For any chosen values for ψ and S, V is evaluated in a way that $f(U(S), \psi, V(\psi, S))$ vanishes. Therefore, we can state that $df/d\psi = 0$ and $df/dS = 0$. With the expression of the total derivative, we can calculate the partial derivatives of the velocity.

$$0 = \frac{df_k}{dS_j} = \frac{\partial f_k}{\partial S_j} + \frac{dV_i}{dS_j}\frac{\partial f_k}{\partial V_i} + \frac{d\psi_i}{dS_j}\frac{\partial f_k}{\partial \psi_i} \qquad \Leftrightarrow \qquad \frac{dV_i}{dS_j} = -\left(\frac{\partial f_k}{\partial S_j} + \frac{d\psi_l}{dS_j}\frac{\partial f_k}{\partial \psi_l}\right)\left(\frac{\partial f}{\partial V}\right)^{-1}_{ki} \tag{2.9}$$

$$0 = \frac{df_k}{d\psi_j} = \frac{\partial f_k}{\partial \psi_j} + \frac{dV_i}{d\psi_j}\frac{\partial f_k}{\partial V_i} + \frac{dS_i}{d\psi_j}\frac{\partial f_k}{\partial S_i} \qquad \Leftrightarrow \qquad \frac{dV_i}{d\psi_j} = -\left(\frac{\partial f_k}{\partial \psi_j} + \frac{dS_l}{d\psi_j}\frac{\partial f_k}{\partial S_l}\right)\left(\frac{\partial f}{\partial V}\right)^{-1}_{ki} \tag{2.10}$$

The derivatives of the slip rate with respect to the state variable and the slip can be reformulated as:

$$\frac{dV_i}{dS_j} = \frac{\partial V_i}{\partial S_j} + \frac{d\psi_l}{dS_j}\frac{\partial V_i}{\partial \psi_l} \qquad \Leftrightarrow \qquad \frac{\partial V_i}{\partial S_j} = \frac{dV_i}{dS_j} - \frac{d\psi_l}{dS_j}\frac{\partial V_i}{\partial \psi_l} \tag{2.11}$$

$$\frac{dV_i}{d\psi_j} = \frac{\partial V_i}{\partial \psi_j} + \frac{dS_l}{d\psi_j}\frac{\partial V_i}{\partial S_l} \qquad \Leftrightarrow \qquad \frac{\partial V_i}{\partial \psi_j} = \frac{dV_i}{dS_j} - \frac{dS_l}{d\psi_j}\frac{\partial V_i}{\partial S_l} \tag{2.12}$$

As such, the two upper block matrices in the full Jacobian from Equation 2.8 are obtained by solving the following system.

$$\begin{cases} \frac{\partial V_i}{\partial S_j} = -\left(\frac{\partial f_k}{\partial S_j} + \frac{d\psi_l}{dS_j}\frac{\partial f_k}{\partial \psi_l}\right)\left(\frac{\partial f}{\partial V}\right)^{-1}_{ki} - \frac{d\psi_l}{dS_j}\frac{\partial V_i}{\partial \psi_l} \\ \frac{\partial V_i}{\partial \psi_j} = -\left(\frac{\partial f_k}{\partial \psi_j} + \frac{dS_l}{d\psi_j}\frac{\partial f_k}{\partial S_l}\right)\left(\frac{\partial f}{\partial V}\right)^{-1}_{ki} - \frac{dS_l}{d\psi_j}\frac{\partial V_i}{\partial S_l} \end{cases} \tag{2.13}$$

This problem can be put in a matrix-vector form using block matrices. Let us define the following matrices to describe the problem:

$$\mathbf{A}_{ij} = \left(\frac{\partial f_k}{\partial S_j} + \frac{d\psi_l}{dS_j}\frac{\partial f_k}{\partial \psi_l}\right)\left(\frac{\partial f}{\partial V}\right)^{-1}_{ki} \qquad \mathbf{B}_{ij} = \left(\frac{\partial f_k}{\partial \psi_j} + \frac{dS_l}{d\psi_j}\frac{\partial f_k}{\partial S_l}\right)\left(\frac{\partial f}{\partial V}\right)^{-1}_{ki} \tag{2.14}$$

$$\mathbf{C}_{ij} = \frac{d\psi_i}{dS_j} \qquad \mathbf{D}_{ij} = \frac{dS_i}{d\psi_j} \qquad \mathbf{X}_{ij} = \frac{\partial V_i}{\partial \psi_j} \qquad \mathbf{Y}_{ij} = \frac{\partial V_i}{\partial S_j} \tag{2.15}$$

The system in Equation 2.13 is equivalent to following block matrix system, with $\mathbf{I}$ being the identity matrix, which has to be solved for all rows $X^{(i)}$ and $Y^{(i)}$ of the respective matrices $\mathbf{X}$ and $\mathbf{Y}$. Note that the matrices are multiplied from the right because of the formulation of the problem, so the system is solved for row vector and not column vectors as it is usual.

$$\begin{pmatrix} X^{(i)} & Y^{(i)} \end{pmatrix}\begin{pmatrix} \mathbf{I} & \mathbf{D} \\ \mathbf{C} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} -A^{(i)} & -B^{(i)} \end{pmatrix} \tag{2.16}$$

With block matrix arithmetic, the solution of this system is calculated as:

$$\begin{pmatrix} X^{(i)} & Y^{(i)} \end{pmatrix} = \begin{pmatrix} -A^{(i)} & -B^{(i)} \end{pmatrix}\begin{pmatrix} (\mathbf{I}-\mathbf{DC})^{-1} & -\mathbf{D}(\mathbf{I}-\mathbf{CD})^{-1} \\ -\mathbf{C}(\mathbf{I}-\mathbf{DC})^{-1} & (\mathbf{I}-\mathbf{CD})^{-1} \end{pmatrix} \tag{2.17}$$

The two arising inverse matrices are relatively easy to calculate, as it will be shown later. In index notation, the entries of the Jacobian matrices are:

$$\frac{\partial V_i}{\partial \psi_j} = \left(\mathbf{B}_{im}\mathbf{C}_{mn} - \mathbf{A}_{in}\right)\left(\delta_{kl} - \mathbf{C}_{km}\mathbf{D}_{ml}\right)^{-1}_{nj} \tag{2.18}$$

$$\frac{\partial V_i}{\partial S_j} = \left(\mathbf{A}_{im}\mathbf{D}_{mn} - \mathbf{B}_{in}\right)\left(\delta_{kl} - \mathbf{D}_{km}\mathbf{C}_{ml}\right)^{-1}_{nj} \tag{2.19}$$

The mutual dependency of the two solution components, described by the variations $\frac{\partial S}{\partial \psi}$ and $\frac{\partial \psi}{\partial S}$ can be obtained using the respective time derivatives.

$$\frac{d\psi_i}{dt} = \frac{dS_j}{dt}\frac{d\psi_i}{dS_j} \quad \Leftrightarrow \quad g_i(\psi, V) = V_j \frac{d\psi_i}{dS_j} \quad \Leftrightarrow \quad \frac{d\psi_i}{dS_j} = \frac{g_i(\psi, V)}{V_j} \tag{2.20}$$

$$\frac{dS_i}{dt} = \frac{d\psi_j}{dt}\frac{dS_i}{d\psi_j} \quad \Leftrightarrow \quad V_i = g_j(\psi, V)\frac{dS_i}{d\psi_j} \quad \Leftrightarrow \quad \frac{dS_i}{d\psi_j} = \frac{V_i}{g_j(\psi, V)} \tag{2.21}$$

This simplifies the evaluation of the inverse matrix terms, with n the total number of nodes on the fault:

$$\delta_{kl} - \mathbf{C}_{km}\mathbf{D}_{ml} = \delta_{kl} - \frac{d\psi_k}{dS_m}\frac{dS_m}{d\psi_l} = \delta_{kl} - n\frac{g_i(\psi, V)}{g_j(\psi, V)} \tag{2.22}$$

$$\delta_{kl} - \mathbf{D}_{km}\mathbf{C}_{ml} = \delta_{kl} - \frac{dS_k}{d\psi_m}\frac{d\psi_m}{dS_l} = \delta_{kl} - n\frac{V_i}{V_j} \tag{2.23}$$

To calculate the Jacobian terms in Equation 2.9 and Equation 2.10, the partial derivatives have to be determined. Two of them are straightforward to calculate and only generate values on their diagonal. This is especially beneficial for $\frac{\partial f_i}{\partial V_j}$, since its inverse is much easier to calculate if the matrix is diagonal.

$$\frac{\partial f_i}{\partial V_j} = \left(-\frac{\sigma_n a}{2V_0}\frac{e^{\frac{\psi_i}{a}}}{\sqrt{\frac{V_i^2}{4V_0^2}+1}} - \eta\right)\delta_{ij} \tag{2.24}$$

$$\frac{\partial f_i}{\partial \psi_j} = \left(-\sigma_n \operatorname{arsinh}\left(\frac{V_i}{2V_0}\right)e^{\frac{\psi_i}{a}}\right)\delta_{ij} \tag{2.25}$$

The evaluation of the last missing partial derivative is more complex to obtain and will be one reason for a complete filling of the Jacobi matrix.

$$\frac{\partial f_i}{\partial S_j} = \frac{\partial \tau_i(U)}{\partial S_j} = \frac{\partial \tau_i(U)}{\partial U_k}\frac{\partial U_k}{\partial S_j} \tag{2.26}$$

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:

$$\frac{\partial b_i(S)}{\partial S_j} = \frac{\partial}{\partial S_j}\int_e \eta\{\{c_{mnkl}\epsilon_{kl}(w)\eta_n^e\}\}[\![U_m]\!] + \frac{\delta_e}{|e|^\beta}[\![U_m]\!][\![w_m]\!]dx \tag{2.27}$$

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:

$$\frac{\partial b_i(S)}{\partial S_j} = -\int_e \eta\{\{c_{jnkl}\epsilon_{kl}(w)\eta_n^e\}\} + \frac{\delta_e}{|e|^\beta}[\![w_j]\!]dx \tag{2.28}$$

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:

$$\frac{\partial U_i}{\partial S_j} = A^{-1}_{ik} b_k(e^i) \tag{2.29}$$

The traction term $\tau(U)$ 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 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:

$$\frac{(\nabla u)_{pq}}{\partial u_k} = \frac{1}{2}\left(D^0_{lpq}\frac{\partial u^0_l}{\partial u_k} + D^1_{lpq}\frac{\partial u^1_l}{\partial u_k}\right) + c_0\left(E^0_{lq}\frac{\partial u^0_l}{\partial u_k} - E^1_{lq}\frac{\partial u^1_l}{\partial u_k}\right)n_{pq} \tag{2.30}$$

$$= \frac{1}{2}\left(D^0_{lpq}\delta^0_{lk} + D^1_{lpq}\delta^1_{lk}\right) + c_0\left(E^0_{lq}\delta^0_{lk} - E^1_{lq}\delta^1_{lk}\right)n_{pq} \tag{2.31}$$

$$= \frac{1}{2}\left(D^0_{kpq} + D^1_{kpq}\right) + c_0\left(E^0_{kq} - E^1_{kq}\right)n_{pq} \tag{2.32}$$

And further we get:

$$\frac{\partial \tau_p}{\partial u_l} = M^{-1}_{rp}e^T_{qr}w_q\frac{1}{2}\left(D^0_{lkq} + D^1_{lkq}\right) + c_0\left(E^0_{lq} - E^1_{lq}\right)n_{kq}n_{kq} \tag{2.33}$$

This derivative term does not depend on the current displacement anymore but only on the geometry of the discretization, so it can be calculated once at the beginning of the simulation. Now, all components are available to calculate the partial derivative of the friction law with respect to the slip in Equation 2.26 and further to evaluate the two Jacobian matrices of the slip rate $\frac{\partial V}{\partial S}$ and $\frac{\partial V}{\partial \psi}$. To obtain the full expression of the Jacobian matrix in Equation 2.8, the partial derivatives of the ageing law $g(\psi, V(\psi, S))$ with respect to ψ and S are still required.

$$\frac{\partial g_i(\psi, V)}{\partial \psi_j} = -\frac{V_0}{L}e^{\frac{f_0-\psi_i}{b}}\delta_{ij} - \frac{b}{L}\frac{\partial V_i}{\partial \psi_j} \tag{2.34}$$

$$\frac{\partial g_i(\psi, V)}{\partial S_j} = -\frac{V_0}{L}\frac{\partial \psi_i}{\partial S_j}e^{\frac{f_0-\psi_i}{b}} - \frac{b}{L}\frac{\partial V_i}{\partial S_j} \tag{2.35}$$

2.4.2 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 $\|\phi(x_n)\| < TOL$:
  - $x_{n+1} = x_n - J^{-1}_{\phi}(\phi(x_n))\phi(x_n)$

The matrix $J^{-1}_{\phi}(f(x_n))$ is the Jacobi matrix of the function ϕ 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 ϕ and its Jacobian matrix are given by:

$$\phi(x) = -x + x^{(0)} + \Delta t F(x) \tag{2.36}$$

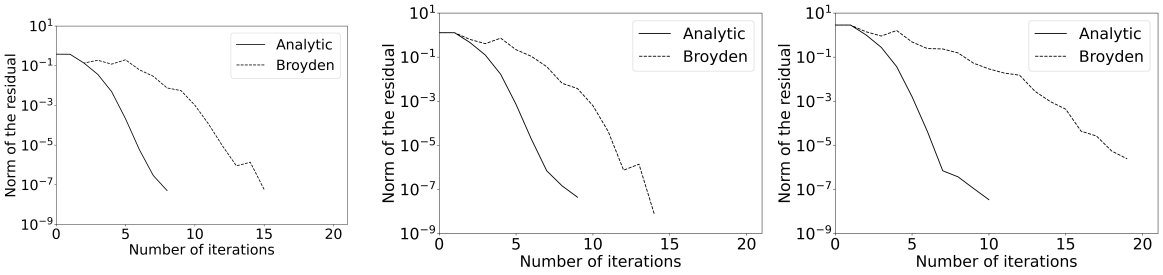
$$J_{\phi}(x) = -I + \Delta t J_F(x) \tag{2.37}$$

The vector x contains both the components related to the slip S and to the state variable ψ and the right hand-side vector $F(x)$ is its time derivative as described in Equation 2.7. 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.



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

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 ????

2.4.3 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 2.4.1, 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 $\mathcal{O}\left(N^2\right)$. 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. 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 ???).

2.5 Numerical treatment in SEAS

To solve this problem, the time solvers of the PETSc library [cite PETSc XXXX] are used. In general, they require a DAE of the form:

$$F(\dot{u}, u, t) = G(u, t) \tag{2.38}$$

For SEAS, the solution vector u combines the velocity V and the state variable ψ 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 2.3

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 2.4 with the bisection method,
   - and the actual right hand side $G(u, t)$ of the ODE is evaluated using equations 2.5 and 2.6.

2.5.1 Explicit methods to solve the DAE

Runge-Kutta schemes with error correction term

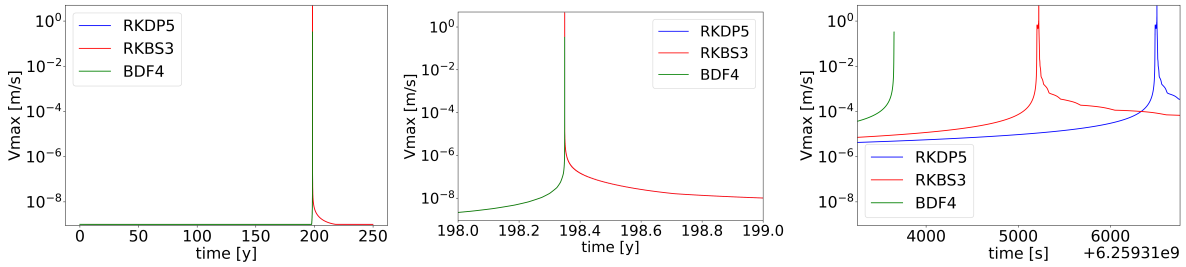
- Bogacki-Shampine (RKBS3): 2nd order with 3rd order error correction

- Dormand-Prince (RKDP5): 4th order with 5th order error correction

2.5.2 Implicit methods to solve the DAE

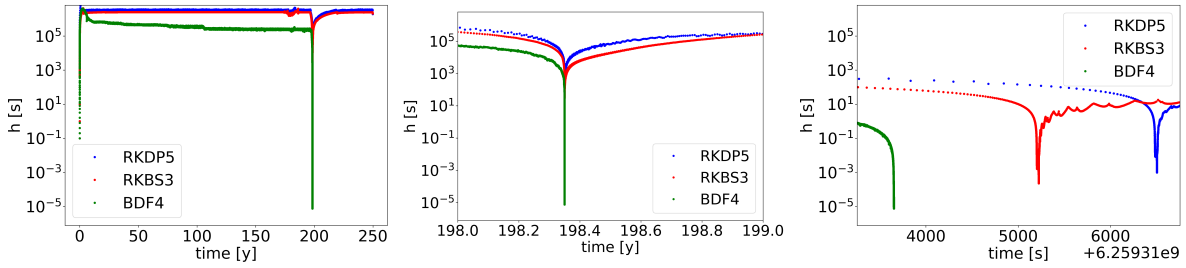
2.6 Results

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 2.3 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.



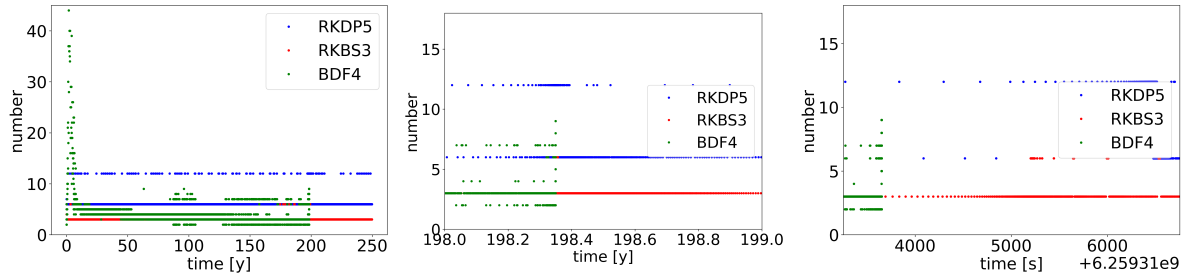
(a) Full simulation time (b) Year of the earthquake (c) Evolution of the earthquake event

Figure 2.3: 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 2.4: 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 2.5: 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

2.7 Relative and Absolute Tolerances

2.7.1 Behaviour of the Error

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 is following condition is fulfilled:

$$\left\| \frac{u_i - u_i^{(e)}}{t_i} \right\|_\infty = \max_i \left| \frac{u_i - u_i^{(e)}}{t_i} \right| \quad \leq \quad 1 \tag{2.39}$$

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.

$$t_i = t_i^a + \max\left(u_i, u_i^{(e)}\right) t_i^r \tag{2.40}$$

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 2.6, 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.

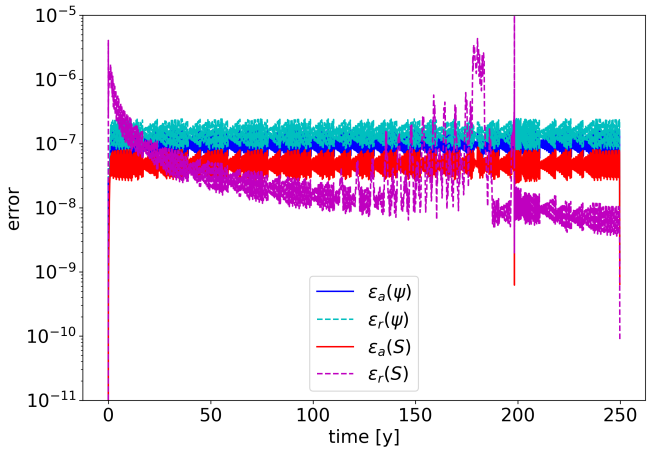


Figure 2.6: Maximum relative and absolute errors using the RKDP5 method on a fault with 200 elements with an absolute and relative error tolerance of 10^{-7}

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 2.7 provides an explanation for these large errors. In the beginning, the slip at each node is below 1, thus the overall tolerance

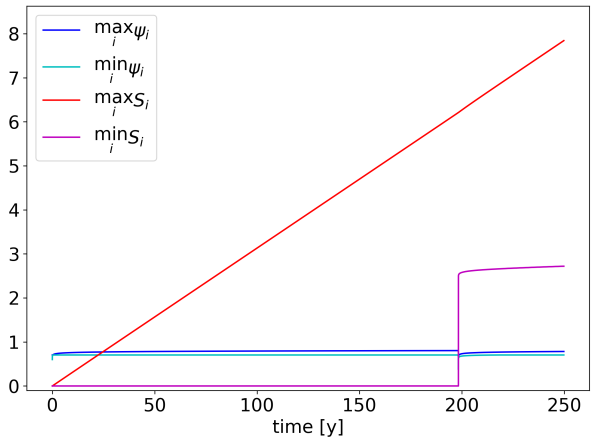


Figure 2.7: Maximum and minimum values of the state variable ψ and of the slip S over time. Simulation performed with the RKDP5 method on a fault with 200 elements with an absolute and relative error tolerance of 10^{-7}

2.7.2 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 ψ 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 ψ is described by the friction law in Equation 2.4, which is evaluated at a constant velocity V^*. For a maximal absolute slip error ϵ_S^a, the largest acceptable absolute error of the state variable ϵ_ψ^a can be calculated by equalizing the induced error in the friction law under constant velocity.

$$f_{V^*}(S + \epsilon_S^a, \psi) - f_{V^*}(S, \psi) = f_{V^*}(S, \psi + \epsilon_\psi^a) - f_{V^*}(S, \psi) \tag{2.41}$$

$$\tau(U(S + \epsilon_S^a)) - \tau(U(S)) = a\sigma_n \text{arsinh}\left(\frac{V^*}{2V_0}\right) \frac{1}{a} \left(e^{\frac{\psi + \epsilon_\psi^a}{a}} - e^{\frac{\psi}{a}}\right) \tag{2.42}$$

$$\tau(B(S + \epsilon_S^a)) - \tau(BS) = a\sigma_n \text{arsinh}\left(\frac{V^*}{2V_0}\right) e^{\frac{\psi}{a}} \left(e^{\frac{\epsilon_\psi^a}{a}} - 1\right) \tag{2.43}$$

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.

$$(\nabla u(S))_{pq} = \frac{1}{2}\left(D^0_{lpq}u^0_l + D^1_{lpq}u^1_l\right) + c_0\left(E^0_{lq}u^0_l - E^1_{lq}u^1_l - f_q\right)n_{pq} \tag{2.44}$$

$$\begin{aligned}(\nabla u(S+\epsilon^a_S))_{pq} =& \frac{1}{2}\left(D^0_{lpq}\left(u^0_l + B_{lj}\mathbb{1}_j\epsilon^a_S\right) + D^1_{lpq}\left(u^1_l + B_{lj}\mathbb{1}_j\epsilon^a_S\right)\right)\\ &+ c_0\left(E^0_{lq}\left(u^0_l + B_{lj}\mathbb{1}_j\epsilon^a_S\right) - E^1_{lq}\left(u^1_l + B_{lj}\mathbb{1}_j\epsilon^a_S\right) - f_q\right)n_{pq}\end{aligned} \tag{2.45}$$

$$(\nabla u(S+\epsilon^a_S))_{pq} - (\nabla u(S))_{pq} = \left(\frac{1}{2}\left(D^0_{lpq} + D^1_{lpq}\right) + c_0\left(E^0_{lq} - E^1_{lq}\right)n_{pq}\right)B_{lj}\mathbb{1}_j\epsilon^a_S \tag{2.46}$$

$$\tau_p(B(S+\epsilon^a_S)) - \tau_p(BS) = M^{-1}_{rp}e^T_{qr}w_q\left((\nabla u(S+\epsilon^a_S))_{kq} - (\nabla u(S))_{kq}\right)n_{kq} \tag{2.47}$$

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

$$\epsilon^a_\psi = a\ln\left(1 + \frac{C_{pj}\mathbb{1}_j}{a\sigma_n\operatorname{arsinh}\left(\frac{V^*}{2V_0}\right)e^{\frac{\psi}{a}}}\epsilon^a_S\right) \tag{2.48}$$

To provide an upper bound for ϵ^a_ψ, the smallest value of the proportionality factor of ϵ^a_S 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, ψ should be replaced by an upper limit ψ_{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 ψ can be evaluated as following:

$$\epsilon^a_\psi = a_{min}\ln\left(1 + \frac{\min_p |C_{pj}\mathbb{1}_j|}{a_{max}\sigma_n\operatorname{arsinh}\left(\frac{V_{max}}{2V_0}\right)e^{\frac{\psi_{max}}{a_{min}}}}\epsilon^a_S\right) \tag{2.49}$$

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 $V_{max} = 5.0ms^{-1}$ 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 ψ 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.

2.7.3 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^r_V in addition to the already discussed absolute error tolerance t^a_V. 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 ψ, 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^*.

$$f_{S^*}(V+\epsilon_V^a,\psi) - f_{S^*}(V,\psi) = f_{S^*}(V,\psi+\epsilon_\psi^a) - f_{S^*}(V,\psi) \tag{2.50}$$

$$f_{S^*}(V,\psi) + \frac{d}{dV} f_{S^*}(V,\psi)\epsilon_V^a - f_{S^*}(V,\psi) + \mathcal{O}\left(\left(\epsilon_V^a\right)^2\right) = a\sigma_n \operatorname{arsinh}\left(\frac{V}{2V_0}\right)\frac{1}{a}\left(e^{\frac{\psi+\epsilon_\psi^a}{a}} - e^{\frac{\psi}{a}}\right) \tag{2.51}$$

$$\left(\frac{a\sigma_n e^{\frac{\psi}{a}}}{2V_0\sqrt{\left(\frac{V}{2V_0}\right)^2+1}} - \eta\right)\epsilon_V^a + \mathcal{O}\left(\left(\epsilon_V^a\right)^2\right) = a\sigma_n \operatorname{arsinh}\left(\frac{V}{2V_0}\right) e^{\frac{\psi}{a}}\left(e^{\frac{\epsilon_\psi^a}{a}} - 1\right) \tag{2.52}$$

$$\epsilon_\psi^a = a\ln\left(1 + \left(\frac{1}{\operatorname{arsinh}\left(\frac{V}{2V_0}\right)\sqrt{V^2+(2V_0)^2}} - \frac{\eta}{a\sigma_n \operatorname{arsinh}\left(\frac{V}{2V_0}\right)e^{\frac{\psi}{a}}}\right)\epsilon_V^a + \mathcal{O}\left(\left(\epsilon_V^a\right)^2\right)\right) \tag{2.53}$$

The exponential $e^{\frac{\psi}{a}}$ with the extremely high value appears again in one of the two ratios. The term with η 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:

$$\epsilon_\psi^a = a_{min}\ln\left(1 + \frac{1}{\operatorname{arsinh}\left(\frac{1}{2}\right)\sqrt{5}V_0}\epsilon_V^a\right) \tag{2.54}$$

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.

$$f_{S^*}((1+\epsilon_V^r)V,\psi) - f_{S^*}(V,\psi) = f_{S^*}(V,(1+\epsilon_\psi^r)\psi) - f_{S^*}(V,\psi) \tag{2.55}$$

$$f_{S^*}(V,\psi) + \frac{d}{dV} f_{S^*}(V,\psi)\epsilon_V^r V - f_{S^*}(V,\psi) + \mathcal{O}\left(\left(\epsilon_V^r V\right)^2\right) = a\sigma_n \operatorname{arsinh}\left(\frac{V}{2V_0}\right)\frac{1}{a}\left(e^{\frac{(1+\epsilon_\psi^r)\psi}{a}} - e^{\frac{\psi}{a}}\right) \tag{2.56}$$

$$\left(\frac{a\sigma_n e^{\frac{\psi}{a}}}{2V_0\sqrt{\left(\frac{V}{2V_0}\right)^2+1}} - \eta\right)\epsilon_V^r V + \mathcal{O}\left(\left(\epsilon_V^r V\right)^2\right) = a\sigma_n \operatorname{arsinh}\left(\frac{V}{2V_0}\right) e^{\frac{\psi}{a}}\left(e^{\frac{\epsilon_\psi^r}{\psi}a} - 1\right) \tag{2.57}$$

$$\epsilon_\psi^r = \frac{a}{\psi}\ln\left(1 + \left(\frac{1}{\operatorname{arsinh}\left(\frac{V}{2V_0}\right)\sqrt{V^2+(2V_0)^2}} - \frac{\eta}{a\sigma_n \operatorname{arsinh}\left(\frac{V}{2V_0}\right)e^{\frac{\psi}{a}}}\right)\epsilon_V^r V + \mathcal{O}\left(\left(\epsilon_V^r V\right)^2\right)\right) \tag{2.58}$$

Again, the term with η 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 ψ can be estimated as:

$$\epsilon_\psi^r = \frac{a_{min}}{\psi_{max}}\ln\left(1 + \frac{1}{\operatorname{arsinh}\left(\frac{V_{max}}{2V_0}\right)}\epsilon_V^r\right) \tag{2.59}$$

The absolute and relative errors in ψ associated to the respective errors in the slip rate are represented in Figure 2.8. The curves show the evaluation of Equation 2.54 and Equation 2.59 for small error values in V. The various parameters are set to $a_{min} = 0.01$, $\psi_{max} = 0.85$, $V_0 = 10^{-6} ms^{-1}$ and $V_{max} = 5.0 ms^{-1}$. 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.

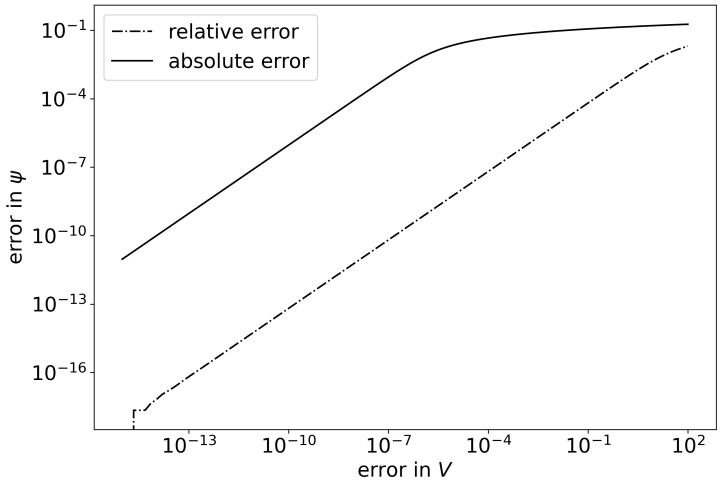


Figure 2.8: Absolute and relative error in ψ in function of the error in the slip rate

Bibliography

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

[Rap97] Joseph Raphson. *Numerical Initial Value Problems in Ordinary Differential Equations.* 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.