SCycle

May 23, 2020

SCycle was written primarily by Kali L. Allison (kallison@stanford.edu), based on an initial prototype by Brittany A. Erickson. Contributions to the code were made by Maxime Rivet and Weiqiang Zhu. The numerical method was additionally developed by Kenneth Duru and Brittany A. Erickson. Details are given in papers by the authors, particularly

Allison, Kali L., E. M. Dunham (2017), Earthquake cycle simulations with rate-and-state friction nd power-law viscoelasticity, Tectonophysics, doi:10.1016/j.tecto.2017.10.021.

Erickson, Brittany A., E. M. Dunham (2014), An efficient numerical method for earthquake cycles in heterogeneous media: Alternating subbasin and surface-rupturing events on faults crossing a sedimentary basin, Journal of Geophysical Research: Solid Earth, doi:110.1002/2013JB010614.

1 Installation

SCycle is written in C++ and is developped based on PETSc Balay et al. (2019). The Installation of PETSc can be found at this website: `https://www.mcs.anl.gov/petsc/documentation/installation.html`. After installing PETSc, compile SCycle source code by:

```
make -C source
```

You can also test SCycle using Docker. Below we show an example to complie SCycle using the PETSc installed in the docker image of the FEniCS Project Alnæs et al. (2015).

```
docker pull quay.io/fenicsproject/stable
docker run -it -v $(pwd):/home/fenics/shared -w /home/fenics/shared quay.io/fenicsproject/stable
make -C source
```

SCycle is compatible on both Linux and macOS. It has been tested with PETSc versions of 3.3.2 and 3.12.2 on Linux (CentOS 7.7).

2 Introduction

SCycle (pronounced like *cycle*) is a code for simulating single earthquakes and sequences of earthquakes, aka earthquake cycles. It only supports the antiplane geometry in 2D, as shown in Figure 1.

SCycle requires an input text file, provided as the first command line argument after the name of the executable. For example, if the code has been compiled into an executable named "main", and the input file is "examples/ex1.in" then use this command to run the code from the command line on a single processor:

```
./main examples/ex1.in
```

Or, to run using (for example) 4 processors, use:

```
mpirun -np 4 main examples/ex1.in
```

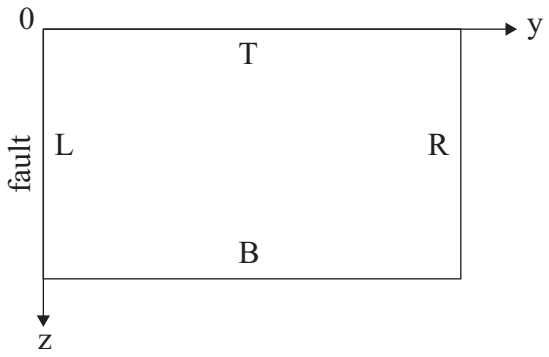

Figure 1: Diagram showing the geometry of the computational domain. Note that only the block to the right of the fault is included. The coordinate system's origin is at the upper left corner.

The running time of ex1.in takes about 1 minute using one processor of 3GHz.

Several example input files are provided in the **example** folder:

ex1.in Input file for a 1D earthquake cycle simulation with linear elastic off-fault material, using the quasi-dynamic approximation. The material properties are constant. This is 1D, meaning that the fault is a single point and the domain extends in the y-direction but not the z-direction.

ex2.in Input file for a 2D earthquake cycle simulation with linear elastic off-fault material, using the quasi-dynamic approximation. The material properties are constant.

ex3.in Input file for a 2D ice stream simulation with linear elastic ice, using the quasi-dynamic approximation. The material properties are constant, and it is assumed that the material to the left of the fault is perfectly rigid.

ex4.in Input file for a 2D thermomechanical earthquake cycle simulation with power-law viscoelastic off-fault material, using a fixed-point iteration method to estimate steady state conditions.

3 Input parameters

A summary of parameters accepted by the input file is provided in the tables below. The bold red parameters are required. Those which are not red and bold have default values indicated in bold face in the rightmost column.

Input parameters are formatted as:

```
parameterName = value # optional comment
```

where the use of white space is important. Each parameter must be placed on its own line, and may not have any white space or characters before its name. The separation between a parameter and its value must be <space><equals sign><space>, with no extra spaces or characters. In the examples, # is used as the default comment symbol, but in actuality any input which does not perfectly match the format will be interpreted as a comment (i.e. misspelled parameters will be ignored, any text that follows the value will be ignored), so any symbol may effectively be used as a comment symbol.

Some depth-variable fields can be inputted via a pair of vectors, one listing values and the other listing the depths for those values. The code will linearly interpolate between these points, and will extrapolate if the model domain extends beyond the last point. An example is shown in Figure 2.

A word of caution: the input parameters are NOT in base SI units.

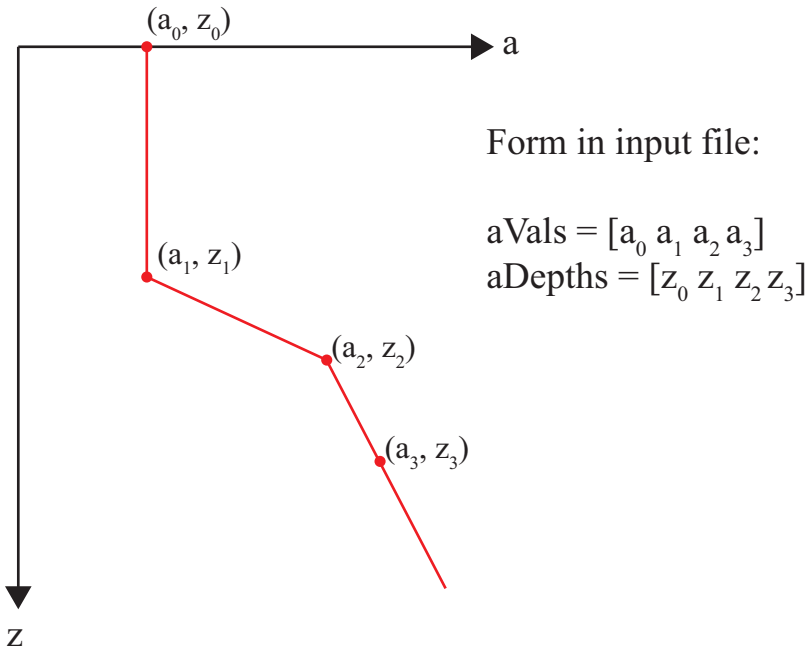

Figure 2: Example of a depth-dependent variable a, and the equivalent form for the input file.

Table 1: Input parameters to select what problem type to run:

Input Parameter	Meaning	Allowed Values
momentumBalanceType	how to treat inertial term in momentum balance equation, also whether to use fixed point iteration or not	**quasidynamic**, quasidynamic_and_dynamic, dynamic, steadyStateIts
guessSteadyStateICs	whether or not to try to converge to steady state initial conditions, based on a guess for steady state shear stress on the fault	**0 = no**, 1 = yes

Table 2: Basic input parameters for every simulation

Input Parameter	Meaning	Allowed Values
order	order of accuracy for spatial derivatives	2, **4**
Ny	# of points in y-direction	must be large enough to resolve physics
Nz	# of points in z-direction	must be large enough to resolve physics
Ly	(km) domain size in y-direction	>0
Lz	(km) domain size in z-direction	>0
sbpType	type of SBP operators used	mc = matrix-based, compatible mfc = matrix-based, fully compatible **mfc_coordTrans** = matrix-based, fully compatible, allows curvilinear coordinate transformation
bCoordTrans	if sbpType is mfc_coordTrans, then this argument tunes how extreme the grid space change in y is	>0 **default: 5**
outputDir	full path to output	string **default: data/** example: /scratch/kallison/test_

3.1 Time integration algorithms

Three explicit and two IMEX time integration algorithms are implemented. The simplest explicit method is forward Euler, called FEuler, which is provided for debugging purposes. The other two are adaptive Runge-Kutta methods, RK32 and RK43, where the first number in the name indicates the order of accuracy in time. Both methods select the size of the time step based on a user-specified subset of the explicitly integrated variables. The IMEX methods, RK32_WBE and RK43_WBE, use the same explicit integration scheme as the explicit methods, and additionally make an implicit integration call once per time step, in which the backward Euler method is used to integrate the implicit variables. Implicit variables may not be used to determine the magnitude of the time step. It is possible to vary the minimum and maximum permitted time step during run time by modifying the timeMoniter function.

Table 3: Time integration algorithms

Input Parameter	Meaning	Allowed Values
timeIntegrator	time integration algorithm	FEuler, **RK43**, RK32, RK32_WBE, RK43_WBE
timeControlType	control type for time step selection	options: P, PI, **PID**
stride1D	number of time steps between output of 1D files	any integer 0 to suppress output default: 1
stride2D	number of time steps between output of 2D files	any integer 0 to suppress output default: 1
maxStepCount	maximum number of time steps to take before terminating the simulation	any integer default: 10^8
initTime	(s) initial time	default: 0
maxTime	(s) final time	default: 10^{15}
initDeltaT	(s) size of first time step	default: 10^{-3}
minDeltaT	(s) minimum allowed time step	default: 10^{-3}
maxDeltaT	(s) maximum allowed time step (can be overridden if the user provides functions to compute this as the simulation runs)	default: 10^{10}
atol	tolerance for selection of time step	default: 10^{-9}
timeIntInds	names of explicitly integrated variables to be used to control time step size	no default values example: [psi slip]
normType	type of norm used to compute error and determine magnitude of next time step	L2_relative, **L2_absolute**

3.2 Rate-and-state friction

Rate-and-state friction is implemented as an algebraic equation relating fault strength to shear stress on the fault, which is solved for slip velocity. The regularized form is used, meaning that fault strength takes the form

$$F(\psi, V) = \sigma_N a \sinh^{-1}\left(\frac{V}{2v_0} e^{\psi/a}\right), \tag{1}$$

where ψ is the state variable and V is slip velocity. Several state evolution laws are implemented in the form

$$\dot{\psi} = G(V, \psi). \tag{2}$$

For the aging law

$$G(\psi, V) = \frac{bv_0}{d_c}\left(e^{(f_0-\psi)/b} - \frac{V}{v_0}\right). \tag{3}$$

For the slip law

$$\dot{\psi} = -V/d_c * (f - f_{\text{ss}}), \tag{4}$$
$$f_{\text{ss}} = f_0 + (a-b) * \ln(V/v_0), \tag{5}$$
$$f = a \sinh^{-1}\left(\frac{V}{2v_0} e^{\psi/a}\right). \tag{6}$$
$$\tag{7}$$

For the slip law with flash heating

$$\dot{\psi} = -V/d_c * (f - f_{\text{ss}}), \tag{8}$$
$$f_{\text{ss}} = \begin{cases} f_w + (f_{\text{LV}} - f_w)(V_w/V), & \text{if } V \geq V_w \\ f_{\text{LV}}, & \text{otherwise} \end{cases} \tag{9}$$
$$f_{\text{LV}} = f_0 + (a-b) * \ln(V/v_0) \tag{10}$$
$$f = a \sinh^{-1}\left(\frac{V}{2v_0} e^{\psi/a}\right). \tag{11}$$
$$V_w = \frac{\pi \alpha_{\text{th}}}{D}\left(\frac{\rho c (T_w - T)}{\tau_c}\right)^2. \tag{12}$$

Table 4: Basic rate-and-state friction input parameters

Input Parameter	Meaning	Allowed Values
stateLaw	evolution law for state variable	**agingLaw**, slipLaw, flashHeating
rootTol	relative tolerance for root-finding algorithm	default: 10^{-9}
f0	steady state friction coefficient at v0	default: 0.6
v0	(m/s) reference slip velocity	default: 10^{-6}
cohesionVals, cohesionDepths	(MPa) cohesion	optional, defaults to 0
DcVals, DcDepths	(m) state evolution distance, also called L in the literature	required, no default
aVals, aDepths	direct effect parameter	required, no default
bVals, bDepths	state evolution effect parameter	required, no default
sNVals, sNDepths	(MPa) effective normal stress	required, no default
sN_floor	(MPa) floor for effective normal stress (overrides sNVals, sNDepths)	optional, no default example: 5
sN_cap	(MPa) ceiling for effective normal stress (overrides sNVals, sNDepths)	optional, no default example: 50
lockedVals, lockedDepths	depth range of fault over which to force the fault to: creep at the loading rate (if value is > -0.5), hold the slip velocity at zero (if value is < 0.5), or allow friction to determine the slip velocity of the fault (if value is < -0.5 and > 0.5)	default: 0 everywhere

Table 5: Additional input parameters used only if flash heating is used.

Input Parameter	Meaning	Allowed Values
fw	fully weakened friction coefficient	no default
TwVals, TwDepths	(K) weakening temperature	no default
tau_c	(MPa) unweakened contact strength	no default
D	(μm) asperity diameter	no default

3.3 Constituitive laws for off-fault material properties

SCycle supports two types of off-fault material: linear elastic, and power-law viscoelastic.

Table 6: Input parameters that effect the behavior of the off-fault material

Input Parameter	Meaning	Allowed Values
vL	(m/s) loading velocity	default: 10^{-9}
forcingType	if iceStream, adds body forcing term to mom. bal. eq.	**no**, iceStream
momBal_bcR_qd	type of right boundary condition for the momentum balance equation	**remoteLoading**, freeSurface
momBal_bcT_qd	type of top boundary condition for the momentum balance equation	**freeSurface**
momBal_bcL_qd	type of left boundary condition for the momentum balance equation	**symmFault**, rigidFault
momBal_bcB_qd	type of bottom boundary condition for the momentum balance equation	**freeSurface**
muVals, muDepths	(GPa) shear modulus	required, no default
rhoVals, rhoDepths	(g/cm^3) density	required, no default
linSolver	algorithm used to solve the momentum balance equation	**MUMPSCHOLESKY** (direct solver using the Cholesky factorization implemented by MUMPS), MUMPSLU (direct solver using the LU factorization implemented by MUMPS), AMG (algebraic multigrid method implemented by HYPRE), CG (conjugate gradient method pre-conditioned with HYPRE's AMG method)
kspTol	tolerance for linear solver method, if an iterative method is selected	default: 10^{-9}

Table 7: Additional input parameters for linear elastic off-fault material

Input Parameter	Meaning	Allowed Values
momBal_computeSxz	determines whether or not to compute stress component σ_{xz}	**0 = no**, 1 = yes
momBal_computeSdev	determines whether or not to compute the deviatoric stress	**0 = no**, 1 = yes

Table 8: Additional input parameters for power-law viscoelastic off-fault material

Input Parameter	Meaning	Allowed Values
AVals, ADepths	power-law parameter	required, no default
BVals, BDepths	power-law parameter equal to Q/R	required, no default
nVals, nDepths	power-law stress exponent	required, no default
maxEffVisc	imposed ceiling for effective viscosity	

3.4 Fixed point iteration for power-law viscoelastic simulations

SCycle includes a fixed point iteration method to find the steady-state behavior of the system.

Input Parameter	Meaning	Allowed Values
momBal_bcR_qd	type of right boundary condition for the momentum balance equation	**remoteLoading**, freeSurface
momBal_bcT_qd	type of top boundary condition for the momentum balance equation	**freeSurface**
momBal_bcL_qd	type of left boundary condition for the momentum balance equation	**symmFault**, rigidFault
momBal_bcB_qd	type of bottom boundary condition for the momentum balance equation	**freeSurface**
fss_T	damping parameter for steady state temperature	0 - 1 default: 0.1
fss_EffVisc	damping parameter for effective viscosity	0 - 1 default: 0.1
maxEffVisc	(GPa s) ceiling value for effective viscosity	default: 10^{30}
gss_t	(millistrains) initial guess for viscous strain rate	default: 10^{-8}
maxSSIts_effVisc	maximum allowed number of iterations to converge to steady state effective viscosity	default: 50
maxSSIts_timesteps	maximum allowed number of time steps for portion of steady state iteration involving explicit time integration	default: 2×10^4
atolSS_effVisc	absolute error for determining if effective viscosity has converged	default: 10^{-3}

3.5 Fully dynamic cyle simulations

Scycle supports earthquake cycle simulations in which the coseismic period is fully dynamic (that is, inertia is included in the momentum balance equation during the earthquake) and the interseismic period is quasi-dynamic. A number of additional input parameters are provided to control the behavior of these two periods separately.

The switch from quasi-dynamic to fully dynamic occurs when the ratio

$$R = \max\left(\frac{\eta V}{\tau_{qs}}\right) \tag{13}$$

exceeds the threshold value trigger_qd2fd. Similarly, the switch from fully dynamic to quasi-dynamic occurs once R drops below the threshold value trigger_fd2qd. In marginally resolved simulations, R tends to oscillate over a wide range of values. To prevent the code from switching back and forth between regimes over just a few time steps as a result, two additional threshold values are provided, limit_qd and limit_fd. For example, the code will switch to the fully dynamic regime when R exceeds trigger_qd2fd, and then will not permit switching back to the quasi-dynamic regime until R has additionally exceeded limit_fd. Similarly, the code will switching to the quasi-dynamic regime once R drops below trigger_fd2qd (and it is permitted, as determined with limit_fd), and then will not permit switching back to fully dynamic until R additionally drops below limit_fd.

Table 9: Input parameters to control the transition from fully dynamic to quasi-dynamic and vice versa.

Input Parameter	Meaning	Allowed Values
trigger_qd2fd	threshold R to switch from quasi-dynamic to fully dynamic	any float, default: 10^{-3}
trigger_fd2qd	threshold R to switch from fully dynamic to quasi-dynamic	any float, default: 10^{-3}
limit_qd	switching from quasi-dynamic to fully dynamic is allowed if R has ever been $<$ limit_qd	any float, default: $10v_L$
limit_fd	switching from fully dynamic to quasi-dynamic is allowed if R has ever been $>$ limit_fd	any float, default: 10^{-1}

For these simulations, the user may specify different time integration parameters for the quasi-dynamic and fully dynamic periods. During the fully dynamic period, a constant time step is used. By default, this time step will be the largest permitted by the method, as derived in in Kenneth et al. (2018); alternatively, the user may specify the time step, or provide a CFL fraction. A warning will be printed if the requested time step is larger than the CFL condition.

It is also possible to specify boundary conditions for these two periods separately.

Table 10: Input parameters for time integration in fully cycle simulations.

Input Parameter	Meaning	Allowed Values
deltaT	(s) size of time step for fully dynamic period	any float, default: largest time step that meets the CFL condition
CFL	time step is computed as CFL * (max allowed time step)	any float, ignored if deltaT is also specified
stride2D_qd	number of time steps between output of 2D files during quasi-dynamic periods	any integer 0 to suppress output default: 1
stride1D_fd	number of time steps between output of 1D files during fully dynamic periods	any integer 0 to suppress output default: 1
stride2D_fd	number of time steps between output of 2D files during fully dynamic periods	any integer 0 to suppress output default: 1

Table 11: Input parameters for boundary conditions in fully cycle simulations.

Input Parameter	Meaning	Allowed Values
momBal_bcR_qd	type of right boundary condition for the momentum balance equation	**remoteLoading**, freeSurface
momBal_bcT_qd	type of top boundary condition for the momentum balance equation	**freeSurface**
momBal_bcL_qd	type of left boundary condition for the momentum balance equation	**symmFault**, rigidFault
momBal_bcB_qd	type of bottom boundary condition for the momentum balance equation	**freeSurface**
momBal_bcR_fd	type of right boundary condition for the momentum balance equation	**outgoingCharacteristics**, freeSurface
momBal_bcT_fd	type of top boundary condition for the momentum balance equation	**freeSurface**, outgoingCharacteristics
momBal_bcL_fd	type of left boundary condition for the momentum balance equation	**symmFault**, rigidFault, outgoingCharacteristics
momBal_bcB_fd	type of bottom boundary condition for the momentum balance equation	**outgoingCharacteristics**, freeSurface

3.6 Heat Equation

It is possible to additionally solve the heat equation with the any combination of the following source terms: frictional shear heating, viscous shear heating, and radioactive heat generation.

Input Parameter	Meaning	Allowed Values
thermalCoupling	whether or not to simulate the heat equation, and to allow temperature changes to feed back into the other governing equations	**coupled**, uncoupled, no
heatEquationType	what form of the heat equation to solve, steady state or including time dependence	**transient**, steadyState
withViscShearHeating	whether or not to include viscous shear heating	**yes**, no
withRadioHeatGeneration	whether or not to include radioactive heat generation	**yes**, no
linSolver_heateq	linear solver algorithm for the heat equation	MUMPSCHOLESKY (direct solver using the Cholesky factorization implemented by MUMPS), MUMPSLU (direct solver using the LU factorization implemented by MUMPS), AMG (algebraic multigrid method implemented by HYPRE), **CG** (conjugate gradient method preconditioned with HYPRE's AMG method)
kspTol_heateq	tolerance for linear solver method, if an iterative method is selected	default: 10^{-9}
rhoVals, rhoDepths	(g/cm^3) density	required, no default
kVals, kDepths	(GW/m/K) conductivity	required, no default
TVals, TDepths	(K) temperature at the top and bottom of the domain (note: only used to set boundary conditions)	required, no default
he_A0Vals, he_A0Depths	(μ W/m^3) radioactive heat generation parameter	required, no default
he_LVals, he_LDepths	(km) radioactive heat generation length scale	required, no default
wVals, wDepths	(m) frictional shear zone width	default: 0

3.7 Pore Pressure and Permeability Evolution

Scycle can solve a fully-coupled model between rate-and-state friction, pore pressure and permeability evolution as proposed by Zhu et al. (2020). Details of parameter settings and simulation results can been found in Zhu et al. (2020)'s paper. The input files can be found under the branch *Zhu_et_at_2020*.

Input Parameter	Meaning	Allowed Values
hydraulicCoupling	whether or not to simulate the fluid diffusion equation, and to allow pore pressure changes to feed back into the other governing equations	coupled, uncoupled, no
permSlipDependent,	whether or not to consider permeability enhancement by slip and healing/sealing with time	year, no
permPressureDependent,	whether or not to consider permeability decrease with effective stress increase	year, no
n_pVals	Pore volume fraction	
beta_pVals	Fluid plus pore compressibility	
eta_pVals	Fluid viscosity	
rho_fVals	Fluid density	
g	Gravity	
kL_pVals	Permeability enhancement evolution distance	
kT_pVals	Healing/sealing time scale	
kmin_pVals	Minimum permeability (for permSlipDependent case)	
kmax_pVals	Maximum permeability	
kmin2_pVals	Minimum permeability (for permPressureDependent case)	
sigma_pVals	Stress sensitivity parameter	
maxBeIteration	Max interation for fixed-point iteration	
maxBeDifference	error tolerance for fixed-point iteration	
k_pVals	initial value for k^*	
pVals	initial value for pore pressure	

References

Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., ... Wells, G. N. (2015). The fenics project version 1.5. *Archive of Numerical Software*, *3*(100). doi: 10.11588/ans.2015.100.20553

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., ... Zhang, H. (2019). *PETSc Web page.* `https://www.mcs.anl.gov/petsc`. Retrieved from `https://www.mcs.anl.gov/petsc`

Zhu, W., Allison, K. L., Dunham, E. M., & Yang, Y. (2020). Fault valving and pore pressure evolution in simulations of earthquake sequences and aseismic slip. *arXiv preprint arXiv:2001.03852*.