



QDYN

Quasi-DYNamic earthquake simulator

V 2.1

User's Manual



github.com/ydluo/qdyn

Developers

[Jean-Paul Ampuero](#) (IRD/UCA, Geoazur, France; Caltech Seismo Lab, USA)

[Martijn van den Ende](#) (Université Côte d'Azur, Geoazur, France)

[Percy Galvez](#) (KAUST, KSA; AECOM, Switzerland)

[Benjamin Idini](#) (Caltech Seismolab, USA)

[Yingdi Luo](#) (NASA JPL, USA)

Last modification on 21 August 2019
[Click here](#) to access the most recent version

Table of contents

[1 Introduction](#)

- [1.1 Summary](#)
- [1.2 Main features](#)
- [1.3 Documentation](#)
- [1.4 Support](#)
- [1.5 License](#)
- [1.6 Acknowledgements](#)
 - [1.6.1 Code contributions](#)
 - [1.6.2 Funding](#)
- [1.7 Suggested references](#)

[2 Installation](#)

- [2.1 Requirements](#)
- [2.2 Download QDYN](#)
- [2.3 Install QDYN](#)
- [2.4 Update QDYN](#)
- [2.5 Additional notes for Windows 10 users](#)

[3 Model assumptions](#)

- [3.1 Model geometry](#)
- [3.2 Boundary conditions](#)
- [3.3 Rate-and-state friction laws](#)
- [3.4 Microphysically-based friction laws](#)
- [3.5 Thermal pressurisation](#)

[4 Running a simulation](#)

- [4.1 The Matlab wrapper](#)
- [4.2 Simulation parameters structure \(pars\)](#)
- [4.3 Output structures \(ot, ox\)](#)
- [4.4 Examples](#)
 - [4.4.1 A simple 2D example](#)
 - [4.4.2 Two asperities interacting](#)
 - [4.4.3 3D simulations](#)

[5 Optimizing Performance](#)

- [5.1 Running simulations outside the Matlab environment](#)
- [5.2 Managing parallel computing](#)

[5.2.1 OpenMP](#)

[5.2.2 MPI](#)

[5.3 Managing outputs of large simulations](#)

[6 Visualizing simulation results](#)

[7 Coupling with the dynamic rupture code SPECFEM3D](#)

[7.1 The QDYN-SPECFEM Bridge \(QSB\)](#)

[7.2 Pre-requisites](#)

[7.3 The master QSB Bash Script](#)

[7.4 The job request script for clusters with job scheduling system](#)

[7.5 How to run fully coupled QDYN-SPECFEM simulations](#)

[7.6 QSB Example: fully-dynamic earthquake cycle simulation on a heterogeneous fault](#)

1 Introduction

1.1 Summary

QDYN is a boundary element software to simulate earthquake cycles (seismic and aseismic slip on tectonic faults) under the quasi-dynamic approximation (quasi-static elasticity combined with radiation damping) on faults governed by rate-and-state friction and embedded in elastic media.

QDYN includes various forms of rate-and-state friction and state evolution laws, and handles non-planar fault geometry in 3D and 2D media, as well as spring-block simulations. Loading is controlled by remote displacement, steady creep or oscillatory load. In 3D it handles free surface effects in a half-space, including normal stress coupling. The medium surrounding the fault is linear, isotropic and elastic, and may be uniform or (in 2D) contain a damaged layer.

QDYN implements adaptive time stepping, shared-memory parallelization, and can deal with multi-scale earthquake cycle simulations with fine details in both time and space. It is equipped with user-friendly Matlab and Python interfaces and graphical output utilities.

1.2 Main features

- Rate-and-state friction, with velocity cut-offs, aging and slip laws
- Microphysically based frictional model (Chen-Niemeijer-Spiers model)
- Heterogeneous frictional properties
- Slow and fast, aseismic and seismic slip transients
- Dynamic weakening (thermal pressurization)
- Non-planar faults (currently limited to variable dip, rectangular elements)
- 3D, 2D and 1D (spring-block)
- Tectonic and transient loads
- Normal stress coupling
- Faults surrounded by damaged zones
- Matlab and python wrappers, and graphic output display utilities
- Parallelized for shared memory systems (OpenMP)
- Parallelized for distributed memory systems (MPI)
- Fully coupled with SPEC3D via QSB (QDYN-SPEC3D Bridge)

1.3 Documentation

Documentation for QDYN is available through the following resources:

- This User's Manual. [Click here](#) to access the most recent version.
- The `examples` directory contains several examples, some have a `README` file
- The Matlab tools provided with the QDYN package are documented through Matlab's help. For instance `help qdyn` provides an overview of the usage of the `qdyn` Matlab interface.
- The `ToDo` file contains a list of known issues and features that we plan to implement in the future.

1.4 Support

The QDYN development team offers online support to users who report bugs, installation problems, documentation issues, feature requests, or questions about QDYN usage by submitting "issues" at <https://github.com/ydluo/qdyn/issues>.

In particular, please do not contact the QDYN developers directly by email.

Before submitting an issue please make sure that:

- you have read the QDYN documentation (see section 1.3)
- you are running the most recent version of QDYN (see sections 2.2 and 2.4)
- your problem has not been treated in previous issues. You can browse and search the list of [closed issues](#)

New issues are submitted via <https://github.com/ydluo/qdyn/issues/new>. Please include all information needed to reproduce your problem: input files, operating system, compiler, QDYN version.

1.5 License

This software is freely available for academic research purposes. If you use QDYN please include proper attributions to its authors and cite one of the references in section 1.7 in your scientific papers and reports.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or

FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

1.6 Acknowledgements

1.6.1 Code contributions

QDYN started from a 2D code written by Allan Rubin (Princeton University) in the early 2000s ([Rubin and Ampuero, 2005](#); [Ampuero and Rubin, 2008](#); [Rubin and Ampuero, 2009](#)). The main developers of QDYN are Jean Paul Ampuero, Martijn van den Ende, Percy Galvez, Benjamin Idini and Yingdi Luo. Bryan Riel contributed the double-FFT version.

The subroutines implementing Okada's formulas were provided by Shinichi Miyazaki (Kyoto University). They include subroutines written by Y. Okada.

The FFT subroutines are based on the [General Purpose FFT Package](#) written by Takuya Ooura (Kyoto University).

1.6.2 Funding

The development of QDYN has been supported by the US National Science Foundation, the Southern California Earthquake Center, Japan's Nuclear Regulation Authority (formerly Japan Nuclear Energy Safety Organization), the European Research Council, and the French National Research Agency (UCAJEDI Investments in the Future project ANR-15-IDEX-01).

1.7 Suggested references

If you use QDYN please cite the code in your publications and reports as follows:

Y. Luo, J. P. Ampuero, P. Galvez, M. van den Ende and B. Idini (2017), *QDYN: a Quasi-DYNamic earthquake simulator (v1.1) [Data set]*. Zenodo.
[doi:10.5281/zenodo.322459](https://doi.org/10.5281/zenodo.322459)

In addition, please consider citing at least one of the following papers:

M. P. A. van den Ende, J. Chen, J. P. Ampuero and A. R. Niemeijer (2018), *A comparison between rate-and-state friction and microphysical models, based on numerical simulations of fault slip*. Tectonophysics, 733, 273-295,

[doi:10.1016/j.tecto.2017.11.040](https://doi.org/10.1016/j.tecto.2017.11.040)

Y. Luo and J. P. Ampuero (2017a), *Stability of faults with heterogeneous friction properties and effective normal stress*. Tectonophysics, 733, 257-272, [doi:10.1016/j.tecto.2017.11.006](https://doi.org/10.1016/j.tecto.2017.11.006)

Y. Luo and J. P. Ampuero (2017b), *Tremor migration patterns and the collective behavior of deep asperities mediated by creep*. [EarthArXiv preprint](#), doi:10.17605/OSF.IO/MBCAV

Y. Luo, J. P. Ampuero, K. Miyakoshi and K. Irikura (2017), *Surface effects on earthquake moment-area scaling relations*. Pure Appl. Geophys., 174 (9), 3331-3342, [doi:10.1007/s00024-017-1467-4](https://doi.org/10.1007/s00024-017-1467-4)

2 Installation

2.1 Requirements

- [Make](#) and [GIT](#) facilitate code installation and updates. Both are standard Linux tools.
- A Fortran compiler.
- MPI (e.g. [MPICH](#), [Open MPI](#)) linked to your Fortran compiler (mpif90).
- Python, Matlab or Octave.
- For Windows 10 users, Linux tools can be acquired and run natively through a [Linux subsystem](#)

We mostly develop and use QDYN with Linux, the Intel Fortran compiler (ifort) and Matlab or Python. The code has been successfully compiled and used on Windows or Mac, with gfortran and Octave.

2.2 Download QDYN

QDYN is hosted on [GitHub](#). To download for the first time the latest development version of QDYN execute the following git command:

```
git clone https://github.com/ydluo/qdyn qdyn-read-only
```

This creates a directory `qdyn-read-only` which contains the whole QDYN package. You can create a directory with a different name. The code contained by the `master` branch (default) is tested and stable, but other development branches may be available. Consult the GitHub repository for the availability of development code.

2.3 Install QDYN

1. Move to the `src` directory
2. Modify the section “User Settings” of file `Makefile` following the instructions and examples therein:
 - a. In section 1, set the variable `EXEC_PATH = [the path to your executable file]`. If you leave the default value (recommended) the executable file `qdyn` is placed in the `src` directory. If you change this variable (not recommended), you must set the `EXEC_PATH` input variable accordingly when you call `qdyn.m`.
 - b. In section 2, adjust your Fortran compiler settings: set the variables

`F90 = [your compiler name], OPT = [your compiler optimisation flags] and PREPROC = [your compiler option to enable preprocessing]`. Intel Fortran (ifort) is the default compiler, but settings for several commonly used compilers are provided. Note that specific optimisation flags need to be set to enable parallelization through OpenMP (see section 4.2.1).

3. Set the parameters in the section “User Settings” of `constants.f90` following the instructions therein
4. Run `make`
5. If in step 2 you changed the path or name of the executable file, modify accordingly the line `status = system('~/.bin/qdyn')` of file `qdyn.m`

2.4 Update QDYN

If you are using the development version, after the first-time checkout you can update the package by executing the following command in your `qdyn-read-only` directory:

```
git fetch
```

Git automatically detects conflicts and attempts to resolve them. In case of unresolvable conflicts you have to fix them manually following the instructions in the [GitHub help pages](#).

2.5 Additional notes for Windows 10 users

As of 2017, Windows 10 officially supports a bash command line environment by installing a Linux subsystem (as of writing, Ubuntu and OpenSUSE are currently offered in the Windows Store). Within a subsystem, Unix-compiled executables can be run natively, and the user has access to the Canonical software repository (`apt-get install [package]`). Running QDYN in a Linux subsystem is done as follows:

1. Install your preferred Linux subsystem, see [this instruction page](#)
2. Install `make`, `gfortran`, and Open MPI as:

```
sudo apt-get install make gfortran libopenmpi-dev
```
3. Download QDYN as instructed above. Note that Windows does not have access to the Linux file system, so in order to exchange files between the subsystem and Windows, it is recommended to download QDYN to (and run simulations from) a local Windows directory (e.g. `C:\Users\bob\qdyn`). The Windows file system can be accessed in the Linux subsystem as: `cd`

- ```
/mnt/c/Users/bob/qdyn
```
4. Navigate to the QDYN `src` directory and compile QDYN as described above
  5. In the case that the required Python or command line Matlab/Octave packages are installed on the Linux subsystem, QDYN can be called directly from a wrapper. If none of these software packages are available, generate a `qdyn.in` file in Windows (e.g. through a Matlab wrapper), navigate within the subsystem to the location of `qdyn.in` (e.g. `cd /mnt/c/Users/bob/test_simulation`) and run:  

```
/mnt/c/Users/bob/qdyn/src/qdyn
```
  6. QDYN should now be running within the Linux subsystem, creating output files in `C:\Users\bob\test_simulation` that can be accessed by Windows for further processing.
  7. The Python wrapper (`pyqdyn.py`) also has built-in functionalities to call the subsystem directly from a Windows 10 environment. In order to set-up and run QDYN simulations from the Python wrapper, set `qdyn.W10_bash = True`. When doing so, the wrapper will automatically switch between the Windows and Linux environments.

## 3 Model assumptions

### 3.1 Model geometry

QDYN handles the following geometries:

- “0D fault”: a spring-block model
- “Finite 1D fault”: a straight fault (a 1D line) embedded in a 2D unbounded elastic medium. Slip is antiplane, i.e. in the direction normal to the 2D plane. The fault is infinitely long, but friction boundary conditions are applied only on a segment of finite length  $L$ ,  $x \in [-L/2, L/2]$ . Outside this segment, constant slip velocity is prescribed.
- “Periodic 1D fault”: same as above, except that here friction is applied all along the infinite fault, and the spatial distributions of friction parameters, normal stress and slip are assumed to be periodic with spatial period  $L$ . The following variants are also available:
  - Fault bisecting an elastic slab of uniform half-thickness  $H$ , with uniform, steady displacement applied along the boundaries of the slab located at a distance  $H$  from the fault.
  - Fault bisecting a low rigidity layer of uniform half-thickness  $H$  embedded in a stiffer elastic medium.
- Vertical, anti-plane, straight 1D fault in a 2D half space.
- 1.5D fault model (2.5D elastic medium) that accounts approximately for a finite fault size  $W$  in the out-of-plane direction (normal to the strike direction  $x$ ), assuming slip either tapers off to zero at the  $W$ -edge or steadily creeps beyond  $W$ . This represents for instance a map-view model of a vertical fault with finite seismogenic depth  $W$ . For more details and a derivation, see appendix A.2 of [Luo and Ampuero \(2017a\)](#). The computational cost of this reduced-dimensionality model is the same as that of 1D-fault simulations.
- 2D fault embedded in a 3D elastic space or half-space. The fault surface has fixed strike, but possibly depth-dependent dip. The fault is infinite but friction is applied only over a finite area.

### 3.2 Boundary conditions

Spring-block models are loaded by an imposed displacement applied at a load-point connected to the rigid block by an elastic spring.

On non-periodic, infinite faults embedded in a continuum elastic medium, friction boundary conditions are applied over a fault segment of finite size. This frictional segment is loaded by slip imposed along the remaining part of the fault.

On periodic faults embedded in an unbounded elastic medium, the fault is loaded by slip prescribed uniformly on the fault portions beyond the frictional width  $W$ .

On periodic faults embedded in a bounded elastic medium of half-thickness  $H$ , the fault is loaded by displacement prescribed uniformly along the boundaries of the medium located at a distance  $H$  from the fault.

In all cases, the imposed slip or displacement usually has constant velocity. An additional oscillatory component is also possible.

On the frictional portion of the fault, the shear strength equals the normal stress times the friction coefficient. Faults governed by rate-and-state friction are always slipping (although very slowly when locked during interseismic periods) and their shear stress is always equal to their frictional strength.

In the quasi-dynamic approximation adopted in QDYN, fault stresses are the sum of static elastic stresses induced by slip and a radiation damping stress. The latter approximates the effect of wave radiation: it represents exactly the stresses induced by waves radiated in the direction normal to the fault, but not the complete elastodynamic stresses.

### **3.3 Rate-and-state friction laws**

The friction coefficient is governed by one of the following rate-and-state laws:

- Conventional rate-and-state

$$\mu(V, \theta) = \mu^* + a \ln(V/V^*) + b \ln(\theta V^*/D_c)$$

- Rate-and-state with cut-offs

$$\mu(V, \theta) = \mu^* - a \ln(1 + V_1/V) + b \ln(1 + \theta V_2/D_c)$$

The state variable follows one of the following evolution equations:

- Aging law

$$d\theta/dt = 1 - \theta V/D_c$$

- Slip law

$$d\theta/dt = -\theta V/D_c \ln(\theta V/D_c)$$

All frictional properties can be spatially heterogeneous.

### **3.4 Microphysically-based friction laws**

As an alternative to rate-and-state friction, simulations can be run using a microphysical formulation for the fault rheology. The fault mechanics model is

based on the Chen-Niemeijer-Spiers (CNS) model ([Niemeijer & Spiers, 2007](#); [Chen & Spiers, 2016](#)). The implementation into QDYN is detailed in [Van den Ende et al. \(2018\)](#). Conceptually, the CNS model is based on the interplay between dilatant granular flow and non-dilatant ductile creep, from which both velocity-strengthening and velocity-weakening behaviour may emerge.

### ***3.5 Thermal pressurisation***

In addition to the (low-velocity) constitutive relations given by the rate-and-state and CNS models, QDYN includes high-velocity dynamic weakening through thermal pressurisation. The thermal pressurisation implementation is based on the spectral method of [Noda & Lapusta \(2010\)](#), which solves the coupled differential equations for the diffusion of fluids and heat in the spectral domain. Diffusion only occurs normal to the fault plane, and it is assumed that the transport properties of the medium are homogeneous in the fault-normal direction and constant in time. In the fault-parallel direction, the transport properties may be spatially heterogeneous.

## 4 Running a simulation

### 4.1 The Matlab wrapper

The core of QDYN is a Fortran code. While the format of its input and output files is well defined, we find it more convenient to set up the input parameters, perform simulations and read the output data within the Matlab environment through the wrapper function `qdyn.m`. You first need to set in Matlab the full path to the `src` directory, for instance:

```
addpath ~/qdyn-read-only/src
```

**Tip for Mac users:** if you get an error message related to gfortran libraries (e.g. `libgfortran.3.dylib`) when running `qdyn` in Matlab, do:

```
setenv('DYLD_LIBRARY_PATH', '/usr/local/bin/')
```

The second argument should be the path to your gfortran libraries (sometimes `/opt/local/lib/libgcc`).

The general usage syntax is:

```
[pars,ot,ox] = qdyn(mode,[parsin],['Property',Value,...])
```

The default input values can be listed by executing:

```
pars = qdyn('set')
```

The input parameters are:

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mode</i>   | One of the following execution modes: <ul style="list-style-type: none"><li>'set' Outputs the default parameter structure (<i>pars</i>) or overrides it with fields present in the structure <i>parsin</i> or with <i>Property/Value</i> pairs</li><li>'write' Sets parameters and writes the qdyn input file</li><li>'run' Sets parameters, writes the input file and runs a simulation</li><li>'read' Reads parameters and outputs from a previous simulation</li></ul> |
| <i>parsin</i> | Parameter structure to override the default parameters (see section 3.2 for details)                                                                                                                                                                                                                                                                                                                                                                                      |

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>'Property'</i> | Name of a field to be set in the parameter structure (see section 3.2) |
| <i>Value</i>      | Value to override the default value and the value in <i>parsin</i>     |

The output variables are:

|             |                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>pars</i> | Structure containing the parameters (see section 3.2)                                                                   |
| <i>ot</i>   | Structure containing time series outputs, global or at selected points (see section 3.3)                                |
| <i>ox</i>   | Structure containing snapshot outputs, i.e. quantities over the whole fault, output at selected times (see section 3.3) |

## 4.2 Simulation parameters structure (*pars*)

The parameters in the structure *pars*, that can be set through 'parsin' or 'Prop/Value' pairs are:

### Parameters defining the geometry of the problem and loading:

|            |                                                                                                                                                   |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| MESHDIM    | Dimension of the problem:<br>0 = Spring-block model<br>1 = 1D fault in a 2D elastic medium, antiplane slip<br>2 = 2D fault in a 3D elastic medium |
| FAULT_TYPE | Faulting type (only used if MESHDIM=2):<br>1 Strike-slip (right-lateral)<br>-1 Strike-slip (left-lateral)<br>2 Thrust<br>-2 Normal                |



|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SOLVER | <p>ODE solver mode:</p> <ol style="list-style-type: none"> <li>1 Bulirsch-Stoer</li> <li>2 Runge-Kutta-Fehlberg</li> </ol>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| MU     | Shear modulus (Pa)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| LAM    | Elastic modulus lambda for 3D simulations (Pa)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| VS     | Shear wave speed (m/s). If $VS=0$ , radiation damping is turned off                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| V_PL   | Plastic loading velocity (e.g. plate velocity; m/s)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| D      | <p>Damage level = <math>1 - (\text{damaged shear modulus}) / (\text{intact shear modulus})</math></p> <p>Currently implemented only for <math>MESHDIM=1</math> and <math>FINITE=0</math></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| H      | <p>If <math>D&gt;0</math>, half-thickness of the fault damage zone (m)<br/> If <math>D=0</math>, half-thickness of an elastic slab bisected by a fault</p> <p>Currently implemented only for <math>MESHDIM=1</math> and <math>FINITE=0</math></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| L      | <p>If <math>MESHDIM=1</math>, L is the fault length (or spatial period)<br/> If <math>MESHDIM=0</math>, <math>MU/L</math> is the spring stiffness</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| FINITE | <p>Boundary conditions when <math>MESHDIM=1</math></p> <ol style="list-style-type: none"> <li>0 = Periodic fault: the fault is infinitely long but slip is spatially periodic with period L, loaded by steady displacement at distance W from the fault</li> <li>1 = Finite fault: the fault is infinitely long but only a segment of length L has rate-and-state friction, the rest has steady slip. If running the code with this option gives the error message “kernel file src/kernel_1.tab is too short”, you should create a larger “kernel file” with the matlab function <i>TabKernelFiniteFlt.m</i></li> <li>2 = Symmetric periodic fault: like option 0 but slip is symmetric relative to the first element</li> <li>3 = Symmetric finite fault: like option 1 but slip is</li> </ol> |

|           |                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------|
|           | symmetric relative to the first element. This can be used to set a free surface next to the first element            |
| W         | Out-of-plane seismogenic width of the fault, only if MESHDIM=1 and FINITE=0                                          |
| DIP_W     | Fault dip angle (degree) if MESHDIM=2 or 4. If depth-dependent, values must be given from deeper to shallower depth. |
| Z_CORNER  | Fault bottom depth (m, negative down) if MESHDIM=2 or 4                                                              |
| SIGMA_CPL | Normal stress coupling: 0 = disable, 1 = enable                                                                      |
| APER      | Amplitude of additional time-dependent oscillatory shear stress loading (Pa)                                         |
| TPER      | Period of oscillatory loading (s)                                                                                    |

### **Simulation features (only available via Python wrapper):**

QDYN offers various non-standard simulation features. Set the following parameters to 1 to enable, or 0 to disable:

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| FEAT_STRESS_COUPL | Normal stress coupling                                    |
| FEAT_LOCALISATION | CNS only: Shear localisation                              |
| FEAT_TP           | High-velocity dynamic weakening by thermal pressurisation |

### **Rate-and-state friction (RSF) parameters:**

|       |                                             |
|-------|---------------------------------------------|
| A     | Direct effect coefficient                   |
| B     | Evolution effect coefficient                |
| DC    | Characteristic slip distance (m)            |
| MU_SS | Reference steady-state friction coefficient |
| V_SS  | Reference steady-state slip velocity (m/s)  |

|           |                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| TH_SS     | Reference steady-state state (s). The default is $TH\_SS=DC/V\_SS$ .                                                                        |
| RNS_LAW   | Type of rate-and-state friction law:<br>0 = original<br>1 = with cut-off velocities V1 and V2                                               |
| V1        | Cut-off velocity of direct effect (m/s)                                                                                                     |
| V2        | Cut-off velocity of evolution effect (m/s), controls the transition from weakening to strengthening when $a < b$ . V2 should be $\leq V1$ . |
| THETA_LAW | Type of evolution law for the state variable:<br>0 = ageing law in the "no-healing" approximation<br>1 = ageing law<br>2 = slip law         |

**Chen-Niemeijer-Spiers (CNS) model parameters (only available via Python wrapper):**

|                                 |                                                                           |
|---------------------------------|---------------------------------------------------------------------------|
| THICKNESS                       | Total thickness of gouge zone (m)                                         |
| <u>Granular flow parameters</u> |                                                                           |
| A_TILDE                         | Coefficient of logarithmic rate-dependence of grain-boundary friction (-) |
| MU_TILDE_STAR                   | Reference grain-boundary friction at Y_GR_STAR (-)                        |
| Y_GR_STAR                       | Reference strain rate corresponding with MU_TILDE_STAR (1/s)              |
| H                               | Dilatancy geometric factor (-)                                            |
| <u>Porosity parameters</u>      |                                                                           |
| PHI_C                           | Critical state porosity (-)                                               |

|                                          |                                                         |
|------------------------------------------|---------------------------------------------------------|
| PHI0                                     | Lower cut-off velocity (e.g. percolation threshold) (-) |
| PHI_INI                                  | Simulation initial porosity (-)                         |
| <u>Creep mechanisms (power-law form)</u> |                                                         |
| A                                        | List of kinetic parameters for each creep mechanism     |
| N                                        | List of stress exponents for each creep mechanism       |
| M                                        | List of porosity exponents for each creep mechanism     |

**Thermal pressurisation model parameters (only available via Python wrapper):**

|              |                                                                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| HALFW        | Half-width of the fault zone (m). If the CNS model is used, this value will be set to half the width of the gouge layer (THICKNESS)            |
| RHOC         | Mean density times specific heat capacity of the medium (J/k/m <sup>3</sup> ). This reflects a weighted average of the solid and fluid phases. |
| BETA         | Bulk compressibility (pores + fluid) (1/Pa)                                                                                                    |
| ETA          | Fluid dynamic viscosity (Pa s)                                                                                                                 |
| LAM          | Net thermal expansion coefficient (fluid - solid) (1/K)                                                                                        |
| K_T          | Thermal conductivity (J/s/K/m)                                                                                                                 |
| K_P          | Intrinsic hydraulic permeability (m <sup>2</sup> )                                                                                             |
| P_A          | Ambient fluid pressure (Pa)                                                                                                                    |
| T_A          | Ambient temperature (K)                                                                                                                        |
| DILAT_FACTOR | Coefficient to control the efficiency of dilatancy hardening (-). Set to zero to disable dilatancy hardening effects.                          |

**Initial conditions:**

|       |                                                                                    |
|-------|------------------------------------------------------------------------------------|
| SIGMA | Initial effective normal stress (Pa). Remains constant unless <i>SIGMA_CPL = 1</i> |
| TAU   | CNS only: Initial shear stress (Pa)                                                |
| V_0   | RSF only: Initial slip velocity (m/s)                                              |
| TH_0  | RSF: Initial state (s)<br>CNS: Initial porosity (-)                                |

**Discretization and accuracy parameters:**

|        |                                                                                                                                                                                                                                        |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| N      | Number of fault elements if MESHDIM=1. It must be a power of 2.                                                                                                                                                                        |
| NX     | Number of fault elements along strike if MESHDIM=2. It must be a power of 2 if FFT is used along strike (FFT_TYPE=1 or 2 in <code>constants.f90</code> ).                                                                              |
| NW     | Number of fault elements along dip if MESHDIM=2. It must be a power of 2 if FFT is used along dip (FFT_TYPE=2).                                                                                                                        |
| NPROCS | Number of processors if running in parallel with MPI (only implemented for MESHDIM=2 and FFT_TYPE=1)                                                                                                                                   |
| DW     | Along-dip length (m) of each element, from deep to shallow                                                                                                                                                                             |
| TMAX   | Threshold for stopping criterion:<br>Final simulation time (s) when NSTOP=0<br>Slip velocity threshold (m/s) when NSTOP=3                                                                                                              |
| NSTOP  | Stopping criterion<br><div style="margin-left: 40px;"> 0 = Stop at t=TMAX (s)<br/> 1 = Stop at end of slip localization phase<br/> 2 = Stop soon after first slip rate peak<br/> 3 = Stop when slip velocity exceeds TMAX (m/s) </div> |

|       |                                   |
|-------|-----------------------------------|
| DTTRY | First trial timestep (s)          |
| DTMAX | Maximum timestep (0=unrestricted) |
| ACC   | Solver accuracy                   |

#### Output control parameters:

|            |                                                                                                                                                                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OX_SEQ     | Type of snapshot outputs<br>0 = All snapshots in a single output file (fort.19)<br>1 = One output file per snapshot (fort.1001, ...)                                                                                                                                     |
| NXOUT      | Spatial interval for snapshot outputs (in number of elements)                                                                                                                                                                                                            |
| NTOUT      | Temporal interval (in number of time steps) for snapshot outputs                                                                                                                                                                                                         |
| OX_DYN     | Output specific snapshots of dynamic events defined by thresholds on peak slip velocity DYN_TH_ON and DYN_TH_OFF (see below)<br>0 = Disable<br>1 = Enable outputs for event #i:<br>Event start: fort.19998+3i<br>Event end: fort.19999+3i<br>Rupture time: fort.20000+3i |
| NXOUT_DYN  | Spatial interval (in number of elements) for dynamic snapshot outputs                                                                                                                                                                                                    |
| DYN_TH_ON  | Peak slip rate threshold (m/s) to define the beginning of a dynamic event                                                                                                                                                                                                |
| DYN_TH_OFF | Peak slip rate threshold (m/s) to define the end of a dynamic event                                                                                                                                                                                                      |
| IC         | Index of selected element for time series outputs                                                                                                                                                                                                                        |
| IOT        | Indices of elements for additional time series outputs: set $IOT(i) = 1$ to enable time series outputs at the i-th element. By                                                                                                                                           |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | default, $IOT(i)=0$ and this output is not done. Each element has a separate output file named <code>fort.xxxxx</code> , where <code>xxxxx</code> is an index (different than <code>i</code> ) that starts at 10001 and is incremented by 1 for each selected element. For instance, if $IOT=[0\ 0\ 1\ 1]$ , the output of elements $i=3$ and $i=4$ are in files <code>fort.10001</code> and <code>fort.10002</code> , respectively. |
| IASP | Auxiliary flags for elements (will not affect outputs, identification purpose only. e.g you can set elements of the VS part to -1 and VW to 0 and particular points of interests like asperities to numbers you want to use )                                                                                                                                                                                                        |

### Parameters for integration with dynamic code:

|          |                                                                                                                                      |
|----------|--------------------------------------------------------------------------------------------------------------------------------------|
| DYN_FLAG | Integration with dynamic code<br>0 =   Disable<br><br>1 =   Enable: stop QDYN at the DYN_SKIP+1-th event with seismic moment > DYN_M |
| DYN_M    | Target seismic moment of a dynamic event                                                                                             |
| DYN_SKIP | Number of dynamic events to skip (warm up cycles)                                                                                    |

### 4.3 Output structures (ot, ox)

The outputs are:

|      |                                                                                                                                                                                                                                                   |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pars | Structure containing the same fields as <i>parsin</i> (see above) plus the positions of the fault elements (X,Y,Z)                                                                                                                                |
| ot   | Structure of time series outputs, with the following fields:<br>t           Output times (s)<br><br>locl        Localization length (distance between stressing rate maxima)<br><br>cl           Crack length (distance between slip rate maxima) |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <p>p            Seismic potency</p> <p>pdot        Seismic potency rate</p> <p>Outputs at the fault location with maximum slip rate:</p> <p>xm          Location of maximum slip rate</p> <p>v            Maximum slip rate</p> <p>th          State variable theta</p> <p>om          (slip rate)*theta/DC</p> <p>tau         Shear stress</p> <p>d            Slip</p> <p>sigma       Normal stress</p> <p>Outputs at selected fault element with index IC:</p> <p>vc          slip rate</p> <p>thc         state variable</p> <p>omc         (slip rate)*theta/DC</p> <p>tauc        shear stress</p> <p>dc          slip</p> <p>If thermal pressurisation is enabled (<math>FEAT\_TP = 1</math>), the time series are appended by the fluid pressure and temperature recorded at the selected fault element and at the maximum slip rate locations.</p> |
| ox | <p>Structure of snapshot outputs, with the following fields:</p> <p>x            fault coordinates</p> <p>t            output times</p> <p>v            slip rate</p> <p>th          state variable theta</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |



|                                                                                                                                                                    |       |                                         |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|-----------------------------------------|
|                                                                                                                                                                    | vd    | slip acceleration                       |
|                                                                                                                                                                    | dtau  | shear stress relative to initial stress |
|                                                                                                                                                                    | dtaud | shear stress rate                       |
|                                                                                                                                                                    | d     | slip                                    |
|                                                                                                                                                                    | sigma | effective normal stress                 |
| If thermal pressurisation is enabled ( <code>FEAT_TP = 1</code> ), the snapshot structure is appended by the fluid pressure and temperature recorded at the fault. |       |                                         |

## 4.4 Examples

### 4.4.1 A simple 2D example

This example is in directory `examples/uniform_slip`. It's a 2D run with uniform slip and initial velocity slightly above steady state. In Matlab:

```
% get default parameters:
p = qdyn('set');
% reset some parameters:
p.N = 16; p.TMAX = 6e9; p.V_0=1.01*p.V_SS;
% run:
[p,ot,ox] = qdyn('run',p);
```

The estimated simulation time is shorter than 10 s on a single thread machine. Let's plot some outputs. Slip velocity as a function of time:

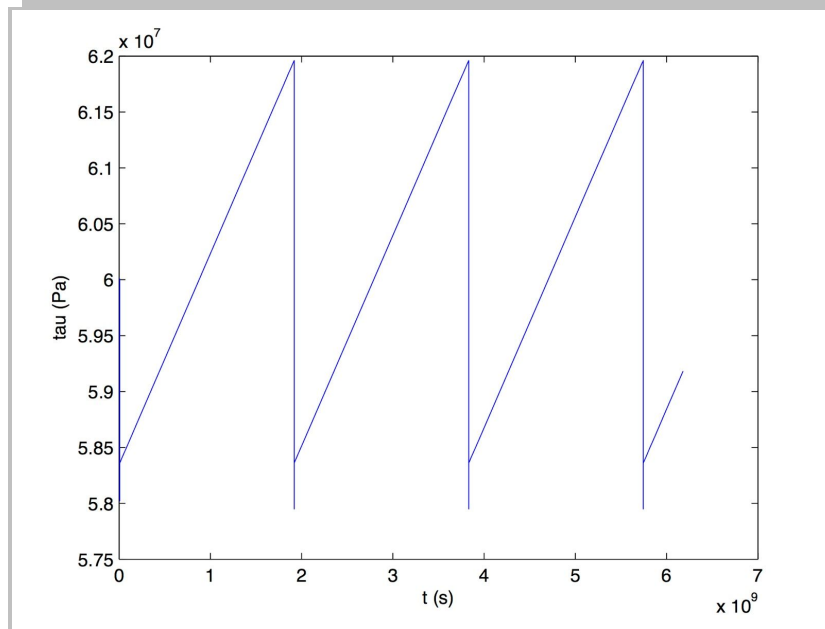
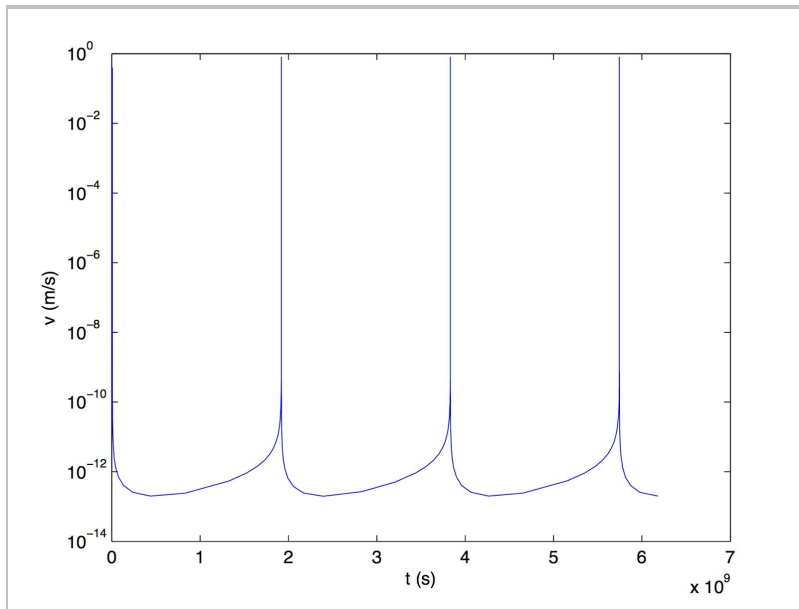
```
semilogy(ot.t,ot.v); xlabel('t (s)'); ylabel('v (m/s)')
```

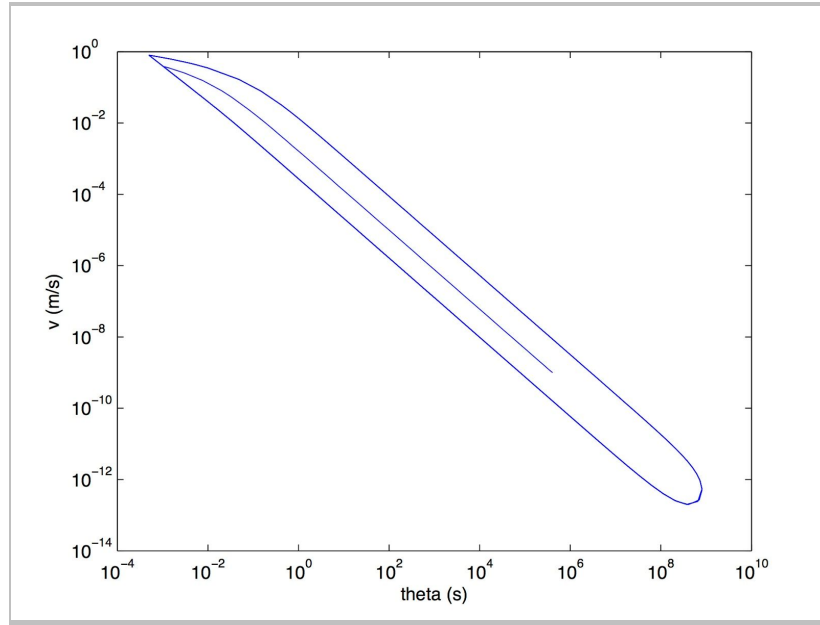
Plot shear stress as a function of time:

```
plot(ot.t,ot.tau); xlabel('t (s)'); ylabel('tau (Pa)')
```

Visualize the convergence to a limit cycle in a state-velocity plot:

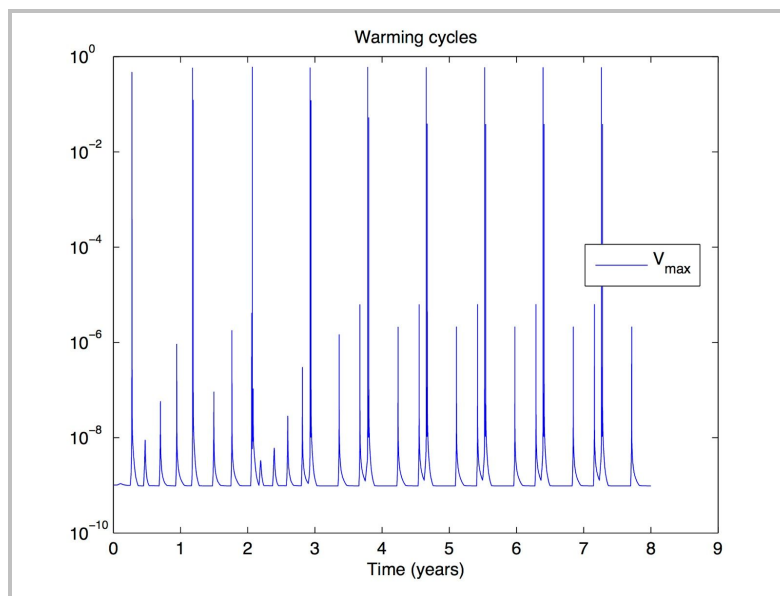
```
loglog(ot.th,ot.v); xlabel('theta (s)'); ylabel('v (m/s)')
```

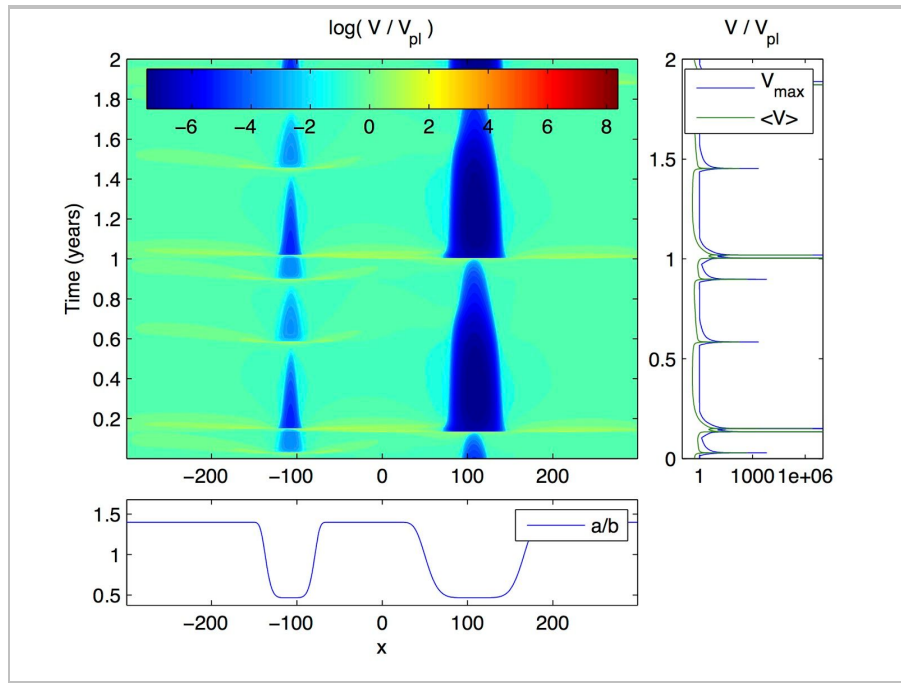




#### 4.4.2 Two asperities interacting

This example is in directory `examples/double_asperities`. A velocity-weakening asperity interacts with a smaller asperity. Both are embedded in a velocity-strengthening (creeping) fault. When the large asperity breaks, its post-seismic slip propagates bi-laterally and triggers rupture of the small asperity. During the interseismic period of the large asperity, the smaller asperity breaks twice with a decreasing recurrence interval. The estimated simulation time is about 6 mins on a single thread machine.



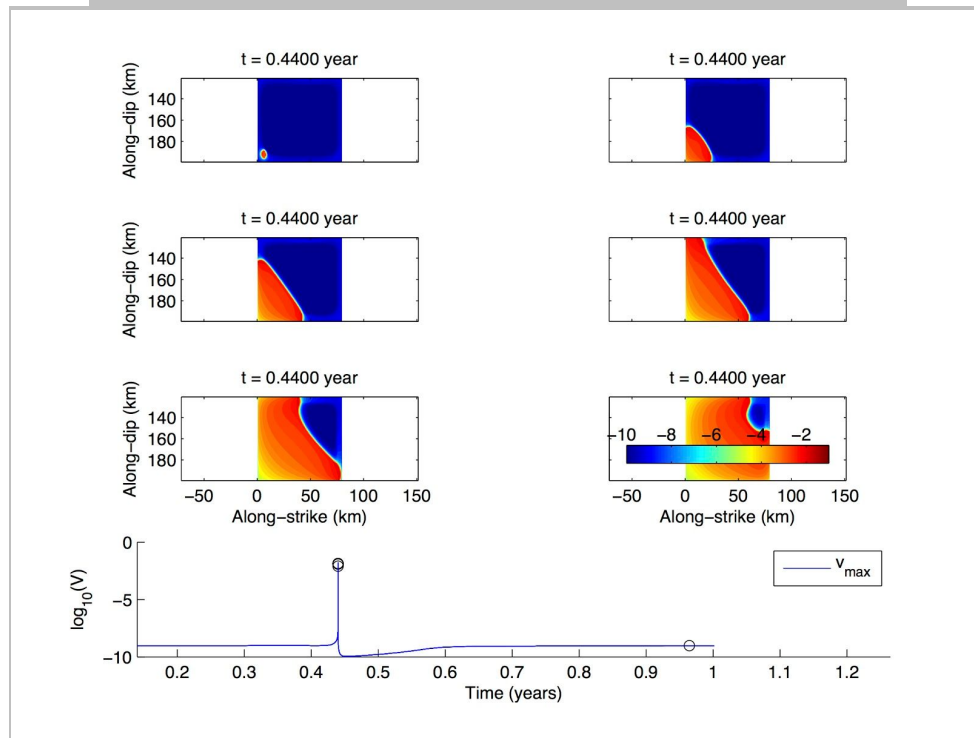
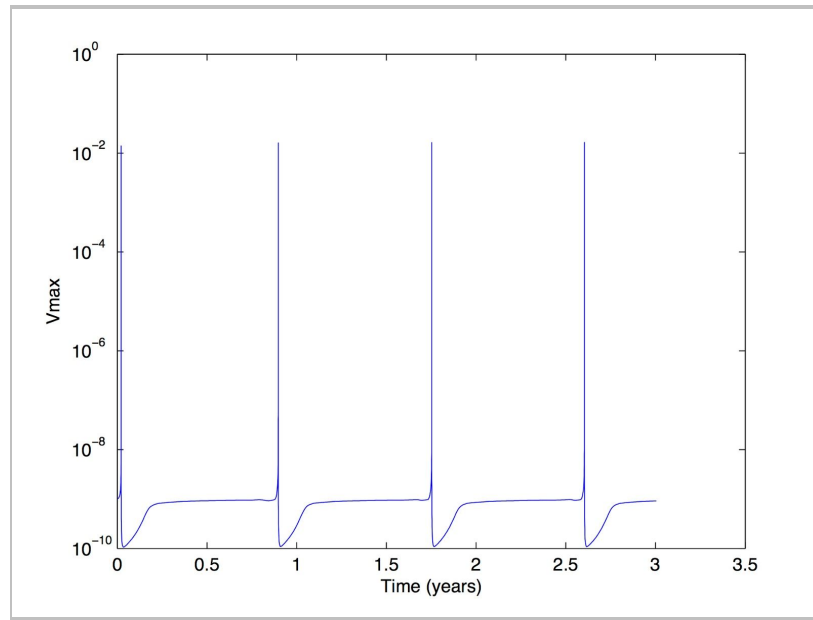


#### 4.4.3 3D simulations

An upper-layer Matlab wrapper for the base-layer Matlab wrapper `qdyn.m` is recommended for complicated simulations. We have included some examples for reference:

a) A 3D simulation: `examples/3d_fft/test_3dfft.m`

The estimated simulation time is about 10 mins on a single thread machine for the first earthquake cycle. The figures shown below are for a complete multi-cycle simulation, comprising 4 warm-up cycles and one cycle output.



b) A simplified model for the Tohoku earthquake (2D along-dip and 3D simulations): `examples/Tohoku`

For more examples and real-world applications please refer to the [wiki pages on the QDYN website](#).

## 5 Optimizing Performance

### 5.1 Running simulations outside the Matlab environment

To run simulations outside the Matlab environment, e.g. when computing on an HPC cluster: run first the Matlab wrapper only to generate the input file `qdyn.in`, then run the `qdyn` executable outside Matlab.

### 5.2 Managing parallel computing

#### 5.2.1 OpenMP

For 3D simulations on 2D faults (`MESHDIM=2` or `4`), QDYN is parallelized for shared memory multi-processor systems with OpenMP. Before compiling the code, you should set the specific compiler optimisation flags that enable OpenMP, as described in the `Makefile` (see step 2.b in section 2.3). Before performing parallel simulations, you should set the following environment variable:

```
setenv OMP_NUM_THREADS 8
```

This command allows QDYN to run on 8 threads, which will roughly speed up calculations by a factor of 8. The number of threads should be set according to demand. In general, set this value to the maximum number of threads available on your shared memory system.

#### 5.2.2 MPI

For 3D simulations with `MESHDIM=2`, QDYN can run in parallel in distributed memory clusters with MPI. The number of processors must be set in the variable `p.NPROCS`.

### 5.3 Managing outputs of large simulations

QDYN by default outputs results as a single ASCII file (`fort.19`). In most multi-cycle 3D simulations this file is very large. It is then helpful to set `OX_SEQ = 1` when calling `qdyn.m`, to generate separate “ox” files outputs for each snapshot (`fort.1001, ...`). Also, setting `OX_DYN = 1` will automatically detect seismic events (according to parameters `DYN_TH_ON` and `DYN_TH_OFF`) and generate 3 snapshots for each event.

## 6 Visualizing simulation results

The QDYN software package includes several Matlab scripts to visualize simulation results in directory `utils/post_processing`. These scripts are all self-documented:

|                                   |                                                                           |
|-----------------------------------|---------------------------------------------------------------------------|
| <code>plot_default.m</code>       | Plots slip rate of a 2D problem (along-strike)                            |
| <code>plot2d_slip.m</code>        | Plots slip of a 2D problem (along-dip)                                    |
| <code>plot3d_m.m</code>           | Plots a sequence of snapshots of slip rate of a 3D simulation             |
| <code>plot3d_faultview_3.m</code> | Plots several snapshots of slip rate for 3D simulation in a single figure |

## 7 Coupling with the dynamic rupture code SPECFEM3D

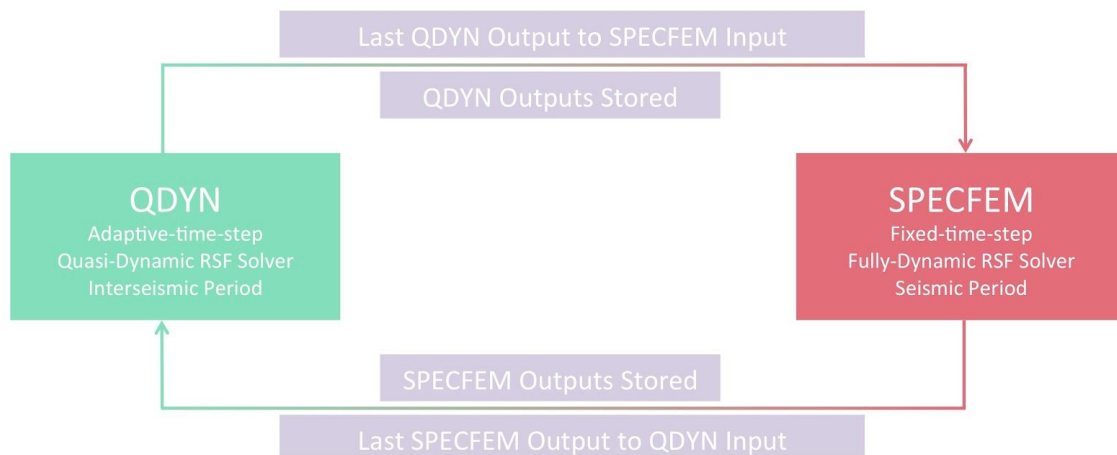
### 7.1 The QDYN-SPECFEM Bridge (QSB)

[SPECFEM3D](#) is a software for dynamic rupture simulations. It is fully dynamic, i.e. it accounts for inertial effects that are important during earthquakes. However, it is based on a solver with fixed time-step, which cannot be applied to interseismic and postseismic periods involving slow, quasi-static (aseismic) deformation.

QDYN is a software for quasi-dynamic earthquake cycle simulations. It is based on the quasi-dynamic approximation and an adaptive time-step solver. These features are accurate and efficient during periods of aseismic slip, but their accuracy degrades during seismic slip, especially in the presence of severe fault weakening mechanisms at high slip rates.

QSB, the QDYN-SPECFEM3D Bridge module, is a two-way interface between the QDYN and the SPECFEM3D softwares. QSB employs each of these codes in its optimal usage conditions to enable efficient and accurate simulations of multiple earthquake cycles containing periods of both seismic (SPECFEM3D) and aseismic (QDYN) slip. The figure below summarizes the QSB workflow and its pattern of data communication between the two codes.

#### QSB: The QDYN-SPECFEM BRIDGE





## 7.2 Pre-requisites

- Obtain a copy of the development version of SPECSEM, which includes a rate-and-state friction solver, using the following git command: `git clone https://github.com/geodynamics/specsem3d.git -b devel`.
- Obtain a copy of the QDYN package on [GITHUB](#) following the instructions described in the [QDYN online manual](#). The QSB module is made of Bash and Matlab scripts contained in directory `QSB/` of the QDYN package.

## 7.3 The master QSB Bash Script

`QDYN_SPECSEM_bridge.sh` is the master script of the QSB. It utilizes the matlab script `QDYN_to_SPECSEM_RS_F.m` to convert the output of QDYN right before an earthquake into a SPECSEM3D input and the matlab script `SEM_to_QDYN_RS_F.m` to convert the output of SPECSEM3D right after an earthquake into a QDYN input.

The control variables in `QDYN_SPECSEM_bridge.sh` are:

|                                    |                                                                                                                                                                                                    |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>N_core_allco</code>          | Number of cores in the SPECSEM simulations                                                                                                                                                         |
| <code>N_loop</code>                | Number of earthquakes to be simulated                                                                                                                                                              |
| <code>QDYN_dir_work</code>         | QDYN working directory                                                                                                                                                                             |
| <code>QDYN_dir_out_store</code>    | Directory to store QDYN outputs. Data for each earthquake is stored in a separate sub-directory named <code>QDYN_dir_out_store/run{i}</code> where $i = 1, 2, \dots$ is the earthquake index       |
| <code>SPECSEM_dir_work</code>      | SPECSEM working directory                                                                                                                                                                          |
| <code>SPECSEM_dir_in</code>        | Directory where QDYN will place output data for SPECSEM input                                                                                                                                      |
| <code>SPECSEM_dir_out</code>       | Directory where SPECSEM will place output data for QDYN input                                                                                                                                      |
| <code>SPECSEM_dir_out_store</code> | Directory to store SPECSEM outputs. Data for each earthquake is stored in a separate sub-directory named <code>SPECSEM_dir_out_store/run{i}</code> where $i = 1, 2, \dots$ is the earthquake index |

## 7.4 The job request script for clusters with job scheduling system

In most High Performance Computing clusters, job requests are submitted through a job scheduling system to better utilize computational resources. The sample job request script `QSB/run_bridge_sh` should serve on most clusters. The script may need slight modifications to adapt it to your job scheduler; please contact your system administrator to get the most accurate information.

To submit your QSB job, edit the control variables in `run_bridge_sh`, then run the following command:

```
qsub run_bridge_sh
```

The control variables in the request script `run_bridge_sh` are:

|                                                     |                                                                                                                                                                                                  |
|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>#PBS -l nodes=[nodes]</code>                  | Number of cores requested, set to the same value as <code>N_core_allco</code>                                                                                                                    |
| <code>#PBS -l walltime=[time]</code>                | Total walltime requested, process will be terminated briefly after exceeding the requested walltime<br>total queue time is in general assessed over cluster load and <code>[nodes]*[time]</code> |
| <code>#PBS -m bae</code>                            | Get notifications at the beginning, end and if an error occurs                                                                                                                                   |
| <code>#PBS -M [email]</code>                        | Notification will be sent to <code>[email]</code>                                                                                                                                                |
| <code>./QDYN-SPECFEM_bridge.sh &gt; [output]</code> | Submit job QDYN-SPECFEM_bridge.sh, simulation progress stores (and overwrites previous existing) outputs file <code>[output]</code>                                                              |

## 7.5 How to run fully coupled QDYN-SPECFEM simulations

**Step 1:** Setup the control variables in the script `QDYN_SPECFEM_bridge.sh`. For most variables you can keep the default settings found in the script.

**Step 2:** Setup a QDYN simulation and generate first-run input file `qdyn.in`

**Step 3: Setup the SPECfEM simulation:**

- Set switch `RATE_AND_STATE = .true.` and `RSF_HETE = .true.` in `src/specfem3D/DATA/Par_file_faults`
- Generate a spectral element mesh with same fault geometry (length, width and dip angles) as the QDYN mesh. The average spacing of the GLL nodes on the fault in the SPECfEM mesh should be similar to the grid spacing in QDYN. If the coordinates of the SPECfEM mesh have different starting values, set the values of `x_off`, `y_off` and `z_off` in `QDYN_to_SEM_RSf_f.m` and `SEM_to_QDYN_RSf.m` accordingly.
- Partition the mesh into `N_core_allco` processors. Please refer to the SPECfEM3D manual for further details.
- Set values in `DATA/Par_file`, `DATA/Par_file_faults` and `DATA/FAULT_STATIONS` accordingly. You can find sample files in `EXAMPLES/fault_examples/tpv103/DATA`. Set the 6 components of CMT source in file `CMTsolution` to 0. Please refer to the SPECfEM3D manual for further details.
- Set the `t_dyn` = [SPECfEM target simulation time] in `QDYN_to_SEM_RSf_f.m`
- Store the coordinates of the fault nodes in file `nodesonfault`, a text file with five columns, IX IZ X Y Z, where IX and IZ are node indices (actually not used) and X, Y and Z are fault node coordinates with units of meters. An example is provided in `QSB/nodesonfault`. You can create it by making a test run of SPECfEM and then running the provided matlab script `SEM_write_nodesonfault.m`.

**Step 4:** Run the Bash script `QDYN_SPECfEM_bridge.sh`, or, on a cluster with scheduler, submit the request script `run_bridge.sh`. You can modify `run_bridge.sh` to change the name of the progress monitoring file (default is `QSB/output.txt`) and your email notification address (see previous section)

**Step 5:** Monitor the simulation progress in file `QSB/output.txt`. Once the simulation of `N_loop` earthquakes is over, an email notification will be sent

**Step 6:** Process the outputs, matlab scripts `plot_QDYN_seq.m` and `plot_SEM_seq.m` are provided for your convenience to visualize QDYN and SPECfEM outputs, respectively.

## ***7.6 QSB Example: fully-dynamic earthquake cycle simulation on a heterogeneous fault***

**Step 1:** We want to run SPECfEM with 108 cores and simulate 2 earthquakes, so we set `N_core_allco=108`, and `N_loop=2` in the script `QDYN_SPECfEM_bridge.sh`. We keep the default values for other variables.

**Step 2:** We prepare the example `hete_3d_ss.m` to run QDYN simulations on a 3D strike-slip fault with 50 km depth and 512 km length and heterogeneous frictional properties. In the script we set `p.OX_SEQ=1`, `p.OX_DYN=1` and `p.DYN_TH_ON=0.1` to generate a snapshot output `fort.20001` when the maximum slip rate reaches 0.1 m/s. We also set `p.NSTOP=3` and `p.TMAX=0.1001` to stop QDYN soon after, when maximum slip rate reaches 0.1001 m/s.

**Step 3:** We generate a SPECFEM mesh matching the QDYN mesh and store the nodes coordinates in `nodesonfault`. In SPECFEM we set the time step to 0.005 s (as dictated by the mesh) and run 60000 steps with a total target simulation time of 300 s, which is sufficient to allow an event to nucleate and rupture the whole 512 km fault from one side to another (extreme scenario). Accordingly, we set `t_dyn = 300 s` in `QDYN_to_SEM_RSf_f.m`

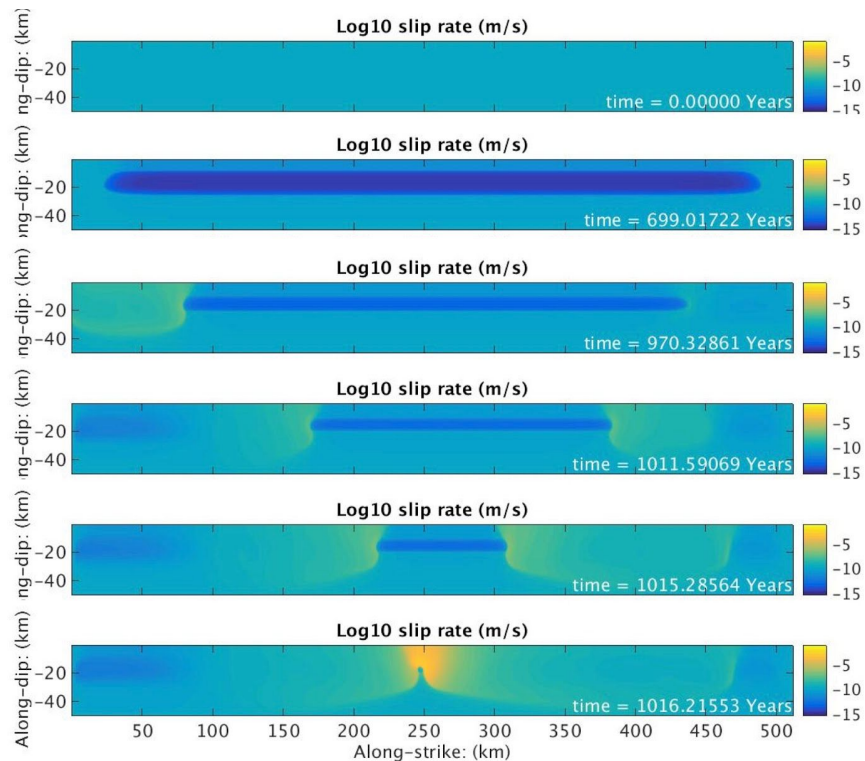
**Step 4:** We run the script `QDYN_SPECFEM_bridge.sh` on our cluster with command `qsub run_bridge_sh`

**Step 5:** The monitoring file shows typically [this](#):

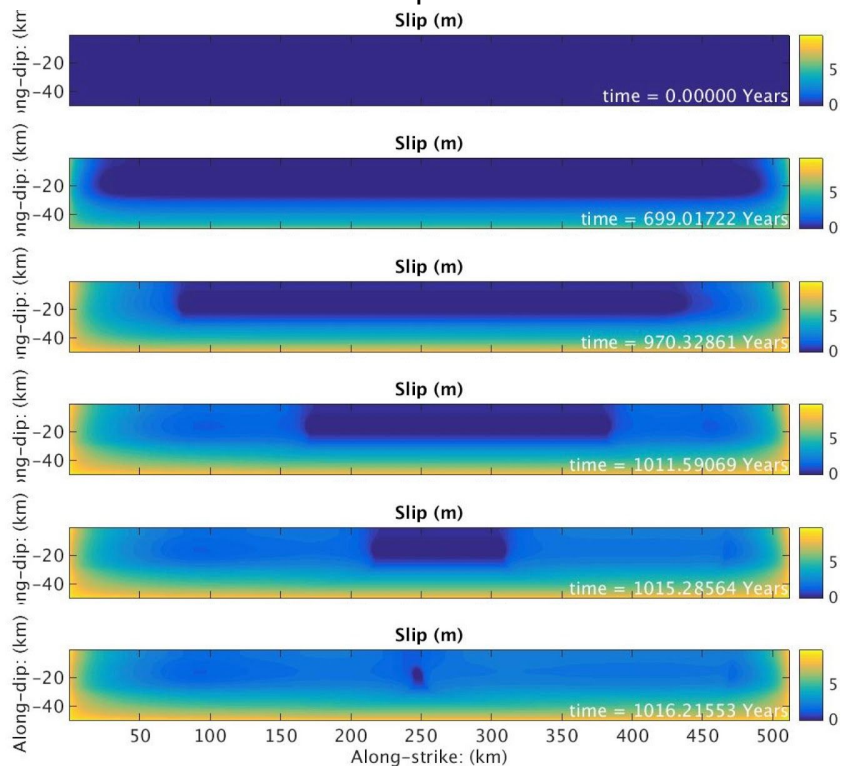
```
N_core_allco = 108
N_loop = 2
Tue Aug 4 03:01:49 PDT 2015
QSB: run no. 1
QSB: run no. 1 QDYN simulation ...
...
...
...
```

**Step 6:** Once the simulation of the 2 earthquakes is over, we process the outputs using matlab scripts `plot_QDYN_seq.m` and `plot_SEM_seq.m`. The figures below show results of a simulation comprising two earthquake cycles. They include the QDYN simulation of the two interseismic periods and the SPECFEM3D simulation of two earthquakes. We show also the results of the first earthquake computed by QDYN only, to highlight the differences introduced by the fully dynamic effects.

## QDYN run Event # 1 Interseismic

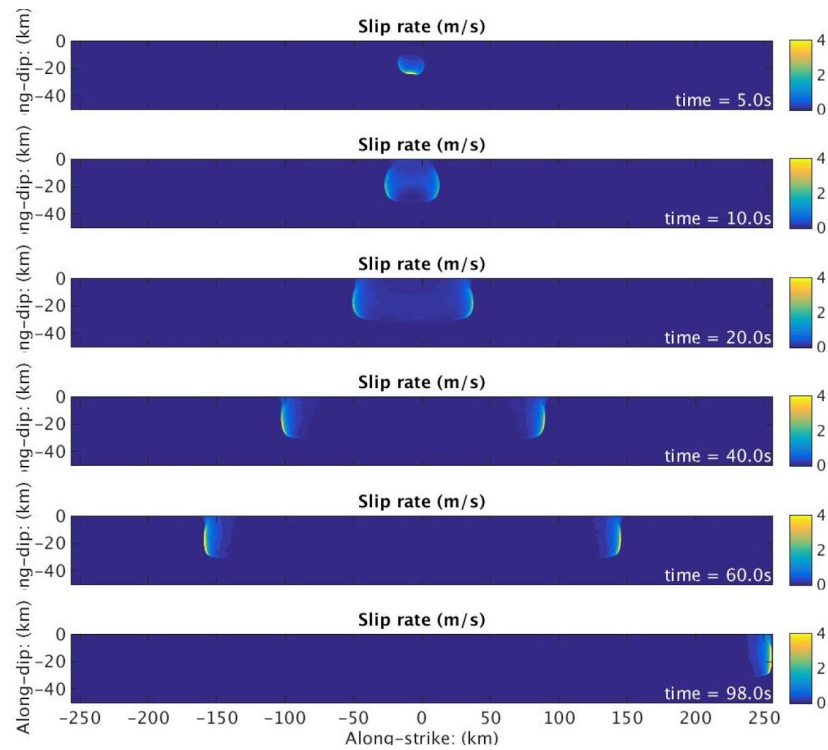


## Slip rate

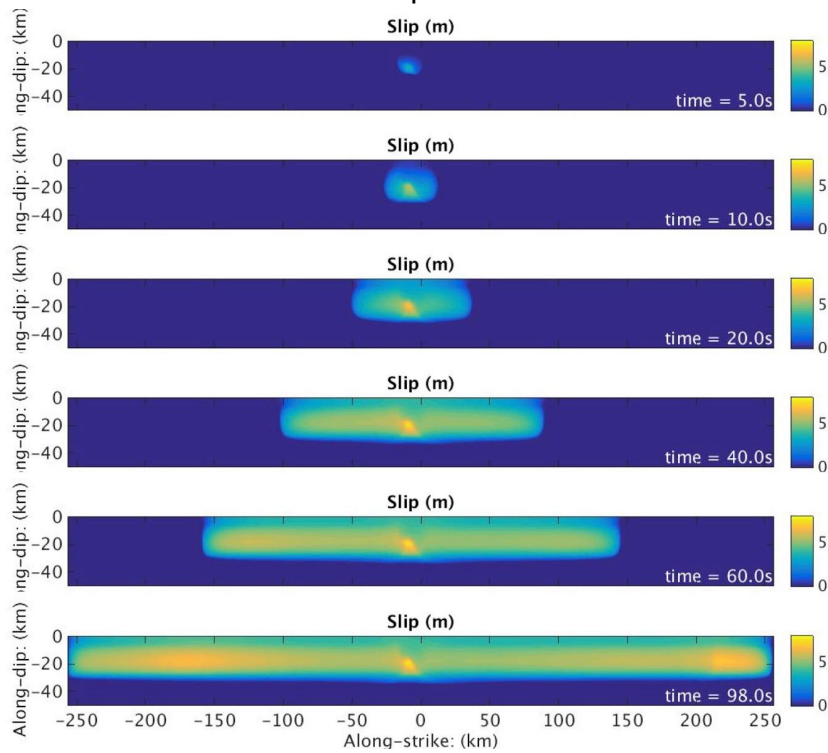


## Slip

# SPECFEM run Event # 1 Seismic

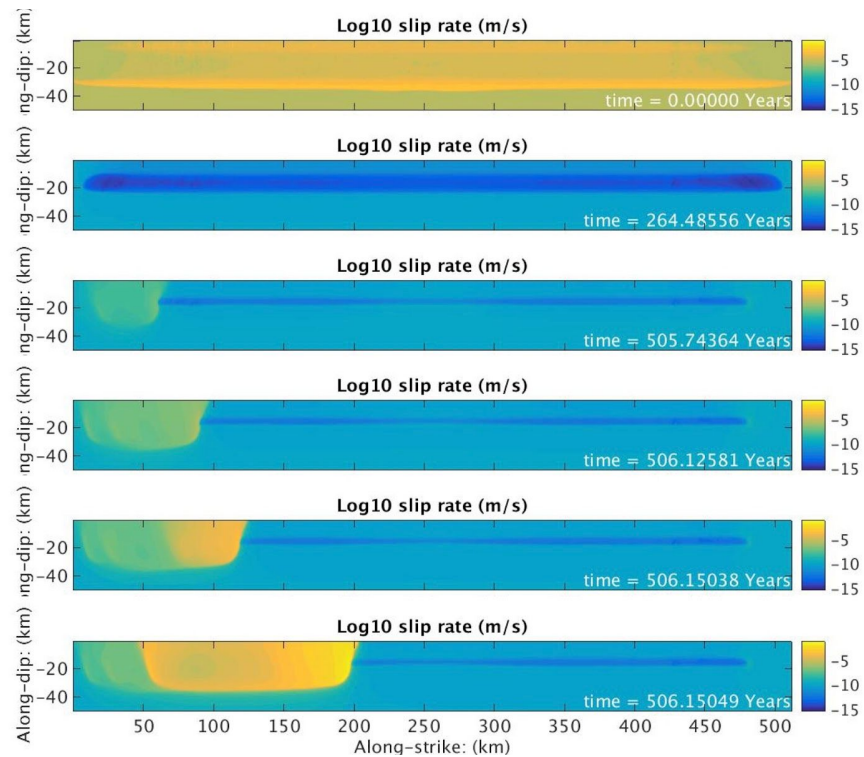


## Slip rate

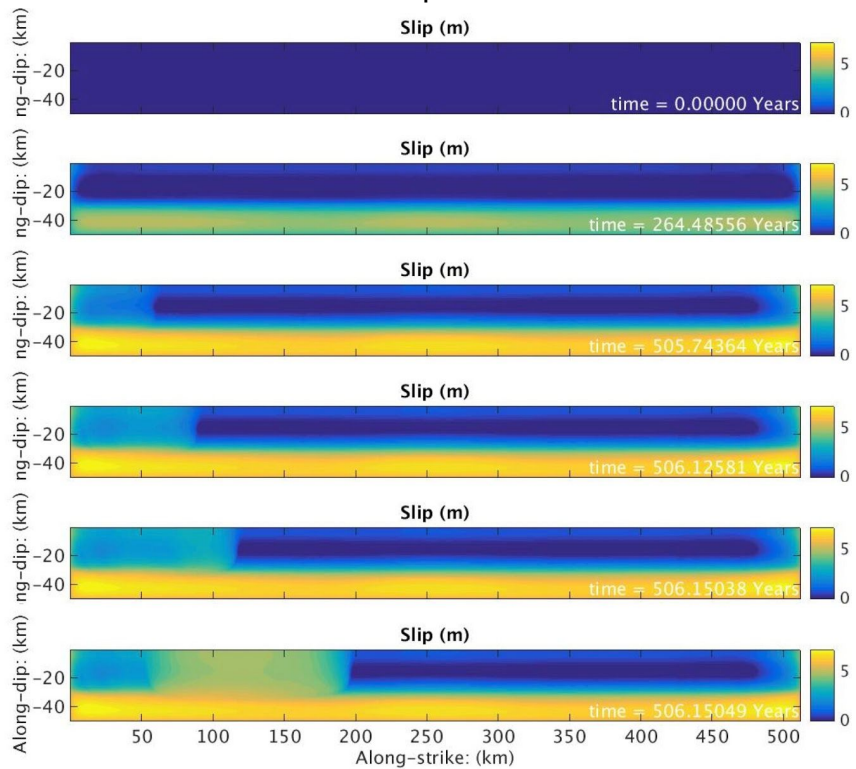


## Slip

## QDYN run Event # 2 Interseismic



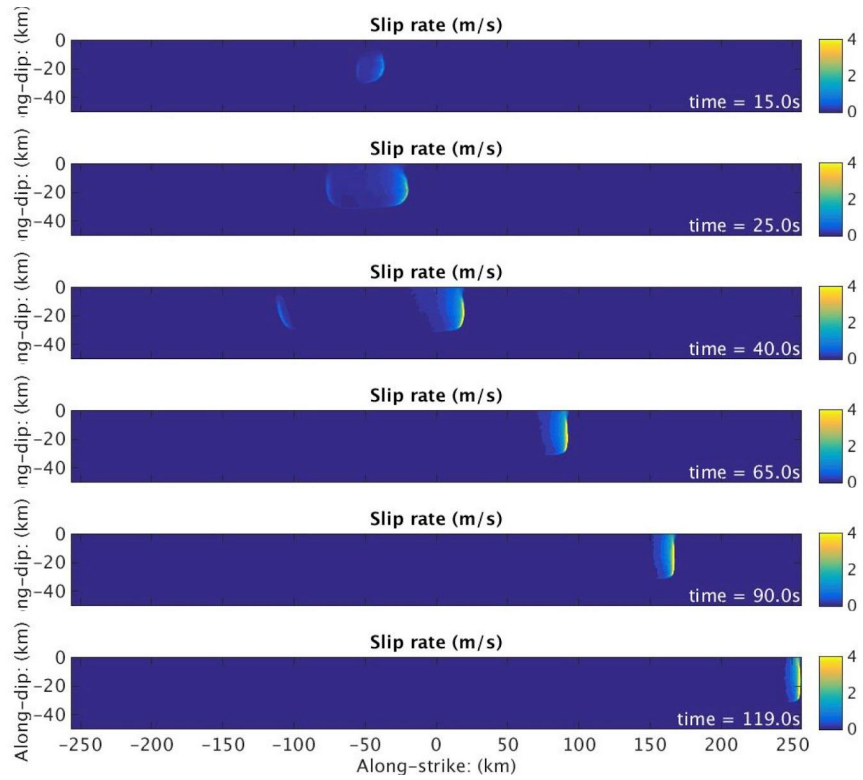
## Slip rate



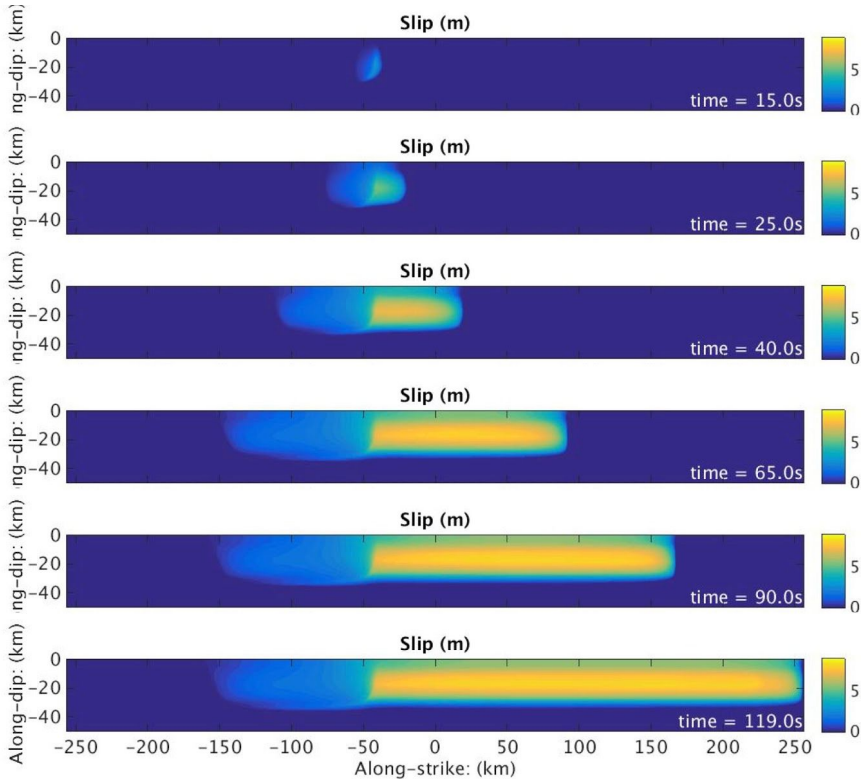
## Slip



## SPECFEM run Event # 2 Seismic



## Slip rate



## Slip



# QDYN run Event # 1 Seismic

