ShellSet v1.1.0 - Parallel Dynamic Neotectonic Modelling: A case study using Earth5-049

Jon B. May[1], Peter Bird[2,1], and Michele M. C. Carafa[1]

[1]Istituto Nazionale di Geofisica e Vulcanologia (INGV), Sezione di Sismologia e Tettonofisica, L'Aquila, Italy
[2]Department of Earth, Planetary, and Space Sciences, UCLA, Los Angeles, California, U.S.A

Correspondence: Jon B. May (jonbryan.may@ingv.it)

Abstract.

We present a parallel combination of existing, well known, and robust software used in modelling the neotectonics of planetary lithosphere, which we call ShellSet. The added parallel framework allows multiple models to be run at the same time and with varied input parameters. Additionally, we have added a grid search option to automatically generate models within a given parameter space. ShellSet offers significant advantages over the original programs through its simplicity, efficiency, and speed. We demonstrate the speedup obtained by ShellSet's parallel framework by presenting timing information for a parallel grid search, varying the number of threads and models, on a typical computer. A possible use case for ShellSet is shown using two examples in which we improve upon an existing global model. Initially we improve the model using the same data before further improving the model through the addition of a new scoring data set.

1 Introduction

Recent decades have seen significant progress made in the numerical modelling of lithospheric and crustal-scale processes. Simulations have become increasingly complex, and it often remains difficult to determine the best set of model parameters for a specific simulation, especially if non-linear rheologies or 3D effects are involved. Currently, the determination of the best set of model parameters is usually done by iteratively changing the model input parameters, performing numerous simulations, and scoring them against a particular class of observations, for example GPS measurements; stress tensor orientations; or earthquake production rates.

There has been a significant improvement in computational power, efficiency, and machine availability since the first computers were utilised for scientific purposes. Research institutions, both academic and industrial (public and private), provide employees with personal machines for their work. Even the least powerful of these machines is typically capable of simulations, in serial and parallel, that would have required much more expensive hardware only a decade previously. This allows many more simulations to be performed within any given period, and larger models to be tested, which means scientific knowledge is being improved almost by the second. In short, current day scientists have access to more computing power than ever.

Unfortunately, this rapid improvement in computing power has, in some cases, outstripped the ability of widely used, scientifically robust, programs to efficiently use the hardware available. For example, many existing geodynamic simulation programs were developed before this accelerated rise in computing performance and remain unable to take advantage of modern computing capabilities.

One such program is Shells, by Kong and Bird (1995). Shells is a dynamic neotectonic modelling program. The program was first developed and released in 1995, since then it has had several updates, the most recent of which was released in 2019. Shells is serial in nature, save for some MKL libraries used to solve systems of linear equations, and so is unable to fully utilise the currently available computing hardware.

In this work we present a single program which is a combination of the program Shells with an optional program used to alter the input Finite Element Grid (created by OrbWin), OrbData5 (which we will refer to as OrbData) and scoring program OrbScore2 (which we will refer to as OrbScore), both of which function in conjunction with Shells.

Our program is based entirely on open source software with OrbData, Shells and OrbScore all being available at: http://peterbird.name/index.htm and all dependencies (see Sect. 3.1) are provided for free by Intel. The final program, which we call ShellSet, runs in a Linux OS environment either using a guest Linux OS on Windows machines or directly on Linux OS machines, including high performance computing (HPC) structures.

This work serves as an introduction to ShellSet, an MPI parallel combination of existing, well known, robust and widely used software, to the neotectonic modelling community.

2 Software

The base of ShellSet are three existing programs: OrbData, Shells, and OrbScore. Despite recent updates, these three programs remain both serial and separate by design. This requires the user to manually control all files required by each program, which may be complicated further depending on the settings for Shells forward simulations. For example, previously generated output is often required to be read before the next Shells solution iteration begins. We outline the function of each of these programs in the following sections.

2.1 OrbData

Before deformation of the lithosphere can be modelled its present structure must be defined, including its surface elevations; layer (crust and mantle-lithosphere) thicknesses; densities; and parameters which define its internal temperatures. These quantities must be specified for each node in the finite element grid which defines the domain of the model. OrbData calculates this lithosphere structure, which is used as an input to both Shells and OrbScore, by simple local operations on published data sets, often in spatially gridded formats. OrbData does not affect the topology of the finite element grid or its included fault network.

OrbData computes the crustal and mantle lithosphere thicknesses based on assumptions of local isostasy and either: (a) a steady state geotherm assumption; or (b) seismically determined layer thicknesses. It incorporates seismic constraints by adding two adjustable parameters, the density anomaly of the lithosphere of compositional origin and an extra quadratic curvature of geotherm due to transient cooling/heating. These extra values are then incorporated into the finite element grid at each node. For a description of the various algorithms OrbData uses, see Bird et al. (2008).

Whether the finite element grid needs updating by OrbData depends on which input variables have been altered. We therefore consider OrbData to be *optional* - it is strictly required in some cases but unnecessary in others. ShellSet removes this consideration from the user by automatically deciding whether or not a certain parameter change requires an update to the finite element grid; this decision is hard coded into the program and is based on which input parameters are altered within a test.

2.2 Shells

Shells is the leading program among those who want to conduct physics-based simulations of planetary tectonics with the efficiency of 2D spherical FE grids. All competing programs use 3D grids, requiring vastly greater computing resources, and limiting the number of experiments that are practical.

Shells uses the thermal and compositional structure of thin spherical shells (usually called "plates") of planetary lithosphere, together with the physics of quasi-static creeping flow, to predict patterns of velocity, straining, and fault-slip on the surface of a planet. Since the first publications outlining the first release version, Kong and Bird (1995) and Bird (1998), Shells has been updated and used in numerous publications including Liu and Bird (2002a, b); Bird et al. (2006, 2008); Kalbas et al. (2008); Stamps et al. (2010); Jian-jian et al. (2010); Austermann et al. (2011); Carafa et al. (2015), and more recently Tunini et al. (2017). A primary goal has been to understand the balance of forces that move the plates while a secondary goal is to predict fault slip rates and distributed strain rates for seismic hazard estimation.

Shells is serial in its application except in two parts of its calculations where it makes use of the thread safe Intel MKL routines dgbsv and dgesv. These routines apply OpenMP type threading within their execution if the problem size is large enough. The number of parallel threads used within each MKL routine is decided at run time by the routines, although it will default to the number of physical cores, environment variables can be altered to specify a strict number or to allow a dynamic selection of threads for MKL routines.

Shells produces six testable predicted fields in each model: relative velocities of geodetic benchmarks; most-compressive horizontal principal stress azimuths; long-term fault heave and throw rates; rates of seafloor spreading; the distribution of seismicity on the map; and fast-polarization directions of split SKS arrivals. Each of which can be scored against observed data.

2.3 OrbScore

OrbScore is used to score any of the six testable Shells predictions, for relative realism, against supplied real data. By using these calculated misfit scores it is possible to compare different models and perform a tuning of the input parameters to obtain the best misfit score for a particular scoring option. This best score then provides the optimal input values for the tested variables. More information about OrbScore, and example input files, can also be found within the "guide to dynamic neotectonic modelling with Shells" at http://peterbird.name/guide/Step_22.htm.

3 ShellSet

Since the three individual programs are quick in their current form, and are intrinsically linked in their work, we focused this work on combining them into a single entity. This combination has multiple benefits for the user and their work when compared with the three underlying programs. We will now outline some of the changes made to the program and improvements that ShellSet offers over the separate programs.

We have placed this combination into a simple MPI framework which allows the running of multiple models in parallel. The migration of file control to the program was necessary to run multiple models in parallel without constant user input. Previously the user would have to start multiple models and control each individually in separate instances, now the program controls the run for each, running multiple in parallel if the compute resources exist. This simplifies the user interface while simultaneously leveraging parallel computing to obtain multiple results at the same time. ShellSet also maintains the parallelism inherent in the Intel MKL libraries, each model run by an MPI thread is able leverage a team of MKL threads within its call to the dgbsv and dgesv subroutines if there are available computing resources.

Since the number of threads used within the Intel MKL library calls is automatically calculated at run time based on the problem size, we have added a control on this value. This prevents an over or under-allocation of compute resources using the total number of physical cores in the system and the number of MPI threads, from here it is trivial to calculate a maximum number of free cores per MPI thread. Performance tests, not reported, show that this automated selection of MKL threads leads to the optimal selection, over each number of threads, for the two examples shown in Sect. 4.

Certain tests require the iteration of the program Shells, each subsequent run requiring a file generated by the previous one. ShellSet has an added option to potentially exit early from these Shells iterations which is performed if any two consecutive Shells iterations have misfit scores that are equal within a preselected tolerance. This option is activated by the user at run time using a command line argument, along with the difference value which defines a converged pair of results.

We have added two options for the testing of multiple variables in parallel: a list input and a grid search. Each of these options requires one small input file to be completed which defines the variables being tested and their values, in the case of the list input or a range in the case of grid search. The grid search allows optimal variable values to be found iteratively by comparing results obtained from different simulations, using their misfit scores, before selecting the best models and refining the search area. This allows an automated search of a selected parameter space to be performed by the program in parallel. This parallel search of a parameter space finds optimal model values for desired variables quickly, efficiently, and allows the program to search each individual parameter to degrees of accuracy that are useful to the user but ordinarily tedious to perform manually.

The inclusion of a search algorithm necessitated an additional set of checks to be performed on combinations of input parameters. Originally the programs performed only basic checks (unrealistic value entries, etc.) as a human user was assumed to be in control of all input values. The grid search option essentially replaces the human user with a machine user, which theoretically allows for possible unrealistic selections of values for input variables. The new checks are added into a Fortran module in a separate file to simplify future updates. We currently place only the most general conditions on the variables - any user is expected to add to this subroutine the conditions needed depending on their local, or global, model requirements.

The main output of ShellSet is a single file which defines the command line arguments used to start the program along with a list of results, in which a row represents a model. Every model has its variable values and all misfit scores recorded.

This combination into a single program and the aforementioned updates have improved user satisfaction, while the parallel running of models saves valuable research time. This allows the exploration of multiple research hypotheses in a shorter time. A local performance analysis is shown in Sect. 5.

Combining the programs into a single entity creates a streamlined process and allows the removal of most of the user interface, leaving only a few simple interactions between user and program which saves time and reduces errors; it also reduces file reading since variables can persist in memory.

The simplified user interface has three new parts: an input file for the model generation option (list or grid search); an input file containing a list of input filenames for each of the original programs; and a selection of command line arguments which provide run-time information to ShellSet. These steps, while providing the same input required by the individual programs, remove the user control over any files that are at the interface between OrbData, Shells and OrbScore, as well as between multiple Shells runs, removing the possibility of errors or delay after program initiation. The nine command line arguments allow the user to personalise the test at run time. These nine arguments allow the user to: define the maximum iterations of the Shells within the iterative loop; choose between the List and Grid input options; create a new directory where all IO files will be stored; optionally make all errors fatal to the program; optionally produce extra program information in a new output file; decide whether to exit the Shells iteration loop earlier than the defined iteration number, and the criteria to judge when to perform this exit; select the misfit type used to select best models; create a special output file to store models with misfit scores better than a specified value; optionally specify a misfit score at which the grid search algorithm will stop.

In a further addition to ShellSet we offer a new option of a geometric mean misfit which can be used to rank models. This geometric mean can be comprised of any set of the five misfit scores generated: seafloor spreading rate; geodetic velocity; stress direction; seismic anisotropy; and fault slip rate. The geometric mean is calculated using the following:

$$GM = \sqrt[n]{S_1 * S_2 * ... * S_n} \tag{1}$$

where S is the calculated model misfit. By default, the geometric mean is always calculated by ShellSet using all of any of the five noted misfits that were run and reported in the output file; however it may be used to select the best models within the grid search at run time using the command line argument and defining which misfits to use in its calculation.

Included within the ShellSet package, but separate from its main work, are two Python programs which add to usability.

First, we include a simple GUI which helps the user to create/update key input files and select run-time options before launching ShellSet. The GUI helps the user to update the files related to parameter input values, the input file list, and the list and grid input files. It contains checks on some typical errors for both file editing and program initialisation. The GUI dramatically simplifies the test setup and launching for new users while more experienced users can choose to manually edit the input files using any text editor before launching from the terminal as with Shells.

Second, we have added a plotting routine that can generate 1D, 2D or 3D plots from the main ShellSet output file. Plots generated by this routine can be seen in Fig. 1 and Fig. 2.

3.1 Dependencies

The following is a list of ShellSet dependencies:

1. Linux OS
   Either as the main OS or as a guest OS (e.g., WSL2 under Windows)
2. Intel Fortran compiler
3. Intel Math Kernel Library
4. Message Passing Interface environment

ShellSet was developed inside Windows Subsystem for Linux 2 (WSL2) on a Windows host. WSL2 is a free program allowing Windows OS users to run a Linux OS machine within a terminal.

The second, third and fourth items of the list are all freely available from Intel in the oneAPI toolkits, specifically the Base and HPC toolkits.

ShellSet was designed for use on a standard laptop or workstation, however it will function on larger HPC machines if available.

4 Real world examples

We now present two real world example uses of ShellSet and its grid search option. Both examples follow the same path as in Bird et al. (2008) and search a similar 2D parameter space. All files for both examples are included with the ShellSet package download, see section on *code and data availability*.

The first example is a direct comparison between the original results, see Table 2 of Bird et al. (2008), and our automated search. In the second example we have added an extra data set used to score the models that was not available to the authors of Bird et al. (2008). While the authors manually updated input files to run their simulations, we will use ShellSet's grid search to search a comparable parameter space that we define as [0.025, 0.8] for fFric and [1E12, 1E13] for tauMax.

We first use the same method as Bird et al. (2008) to grade and select the best models: the geometric mean of the misfits obtained from seafloor spreading rates (SSR); geodetic velocities (GV); most compressive horizontal principal stress directions (SD); and fast-polarization azimuths of split SKS waves (SA), see Eq. 2 and Eq. 3. Information on each of the listed data sets can be found in Sect. 6.1, 6.2, 6.3 and 6.4 of Bird et al. (2008) respectively. Furthermore, we use the same data sets to generate our model scores and the same input files at the start of each model.

Due to changes in software since 2008 we expect different results to be calculated for the same models. Firstly, there have been minor updates to the original software and third-party libraries since the publication which will affect its results. Secondly, there have been numerous updates to the compilers used in the intervening years. Thirdly, ShellSet is compiled to run on Linux OS whereas the original results were obtained on a version compiled to run in Windows OS. These differences will lead to, for example, different handling of variable precision at minor decimal places which, over several iterations, will compound to alter model results in minor but noticeable ways. In order to account for these differences we have rerun the previous best model, see model Earth5-049 in Table 2 Bird et al. (2008), and report the updated results in each example named "New Earth5-049".

Preliminary testing showed that the requirement of relative velocity convergence to within 0.0001 was too strict for ShellSet; this is likely due to the aforementioned software differences. This gave us choices regarding the linear system solution iterations. We could alter the tolerance limit, change the number of allowed iterations from 50, or both. We decided to loosen the tolerance to 0.0005 while keeping the iterations set to 50 in order to complete all models in a reasonable time.

As in Bird et al. (2008) each model begins with a trHMax limit of 0.0 for the first call of Shells before updating the limit to 2×10^7. Parameter trHMax defines the upper limit of basal shear tractions, so an initial value of 0.0 means that the first iteration of Shells imposes no basal shear tractions.

Detailed analysis of the results is not within the scope of this work, however for each example we will offer some observations and a basic analysis.

4.1 Using ShellSet to recreate Bird et al. 2008

In this first example we show a recreation of the test done by the authors of Bird et al. (2008) using ShellSet.

Our grid search was set up to perform 4 iterations of Shells before a final call with updated boundary conditions for a total of 5 Shells iterations per model. After the final run OrbScore was used to generate misfits and geometric mean misfit for each model, which were used to rank the models. The geometric mean, from Eq. 1, for this example is given as

$$GM = \sqrt[4]{S_{SSR} * S_{GV} * S_{SD} * S_{SA}} \tag{2}$$

The grid search partitioned the domain into 9 cells, in a 3×3 grid, each represented by its central model. The best 2 models of these 9 were then chosen to continue to the second level where each represented cell is further divided into a 3×3 grid. This was repeated with the 18 models at this second level to generate another 18 models. This 3-level grid search gives a total of 41 distinct models, once the 4 repeated models are excluded, and a final resolution of 1/27 of the original ranges of 9×10^{12}, and 0.775 for tauMax and fFric respectively, or 3.33×10^{11} and 0.0287.

Table 1 shows some of the results we have obtained using ShellSet. The first two rows of the table show the previous best model, *Earth5-049*, and the scores when this model was rerun, *New Earth5-049*. For brevity we only report here the models at the first level, to demonstrate the initial grid (models 1-9), then every model for which the geometric mean misfit is better than *New Earth5-049*. The entire search history, excluding the first level, can be seen in Table C1 of appendix C.

Table 1. Previous best model of Bird et al. (2008) compared with ShellSet grid search models.

Model	fFric	tauMax	SSR	GV	SD	SA	Geometric mean
Earth5-049	0.1	2×10^{12}	8.02	16.19	31.28	26.43	18.10
New Earth5-049	0.1	2×10^{12}	7.96	12.18	30.68	25.43	16.58
1	0.15417	2.5×10^{12}	9.97	13.02	31.10	24.93	17.81
2	0.4125	2.5×10^{12}	18.84	17.43	33.82	24.90	22.93
3	0.67083	2.5×10^{12}	26.57	21.09	35.00	25.36	26.56
4	0.15417	5.5×10^{12}	13.65	21.85	31.04	23.04	21.49
5	0.4125	5.5×10^{12}	21.90	24.11	33.43	22.28	25.04
6	0.67083	5.5×10^{12}	28.55	24.74	34.25	22.95	27.30
7	0.15417	8.5×10^{12}	14.90	23.66	32.51	21.94	22.40
8	0.4125	8.5×10^{12}	24.02	26.29	34.15	21.31	26.03
9	0.67083	8.5×10^{12}	30.18	27.00	35.27	22.02	28.20
10	0.06806	1.5×10^{12}	7.14	12.04	31.28	27.52	16.50
28	0.03935	1.17×10^{12}	6.57	12.27	32.00	27.65	16.34
29	0.06806	1.17×10^{12}	7.03	11.70	31.57	27.10	16.29
30	0.09676	1.17×10^{12}	7.59	11.64	31.49	26.94	16.55
31	0.03935	1.5×10^{12}	6.69	12.68	31.73	27.50	16.49
32	0.06806	1.5×10^{12}	7.14	12.04	31.28	27.52	16.50
33	0.09676	1.5×10^{12}	7.74	11.96	31.03	26.35	16.58
35	0.06806	1.83×10^{12}	7.31	12.27	30.99	27.01	16.55
36	0.09676	1.83×10^{12}	7.86	12.09	30.87	25.66	16.56

Scores are reported to 2 decimal places but calculated to greater accuracy.

It is immediately clear that within the first 9 models the best geometric mean results are obtained with a minimum fFric and tauMax within their respective ranges, with fFric being the more important of the two. This is made clearer in Fig. 1, which maps the entire grid search.

There are 8 distinct models (32 being a repetition of model 10) that obtained an equal or better geometric mean score than *New Earth5-049*. Each one of these 8 models achieved a better SSR score than *New Earth5-049*, 5/8 have a better GV score (models 10, 29, 30, 33, 36), none of these 8 models have a better score in either SD or SA. The absolute best model is model 29 with a geometric mean of 16.29, an improvement of 0.29. Figure C1 plots velocity differences between the best model of Bird et al. (2008) and the best model of ShellSet. The map, focused on Asia, demonstrates the alterations made to the calculated velocities at each node. It shows a calculated increase in velocity around Japan, Pakistan and Nepal.

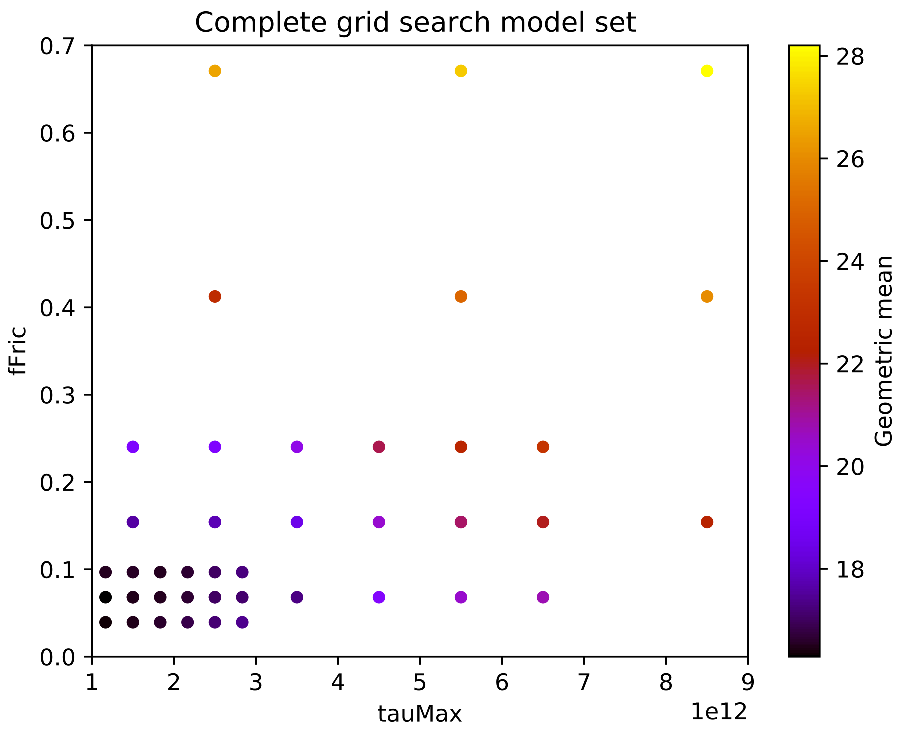

Figure 1. Visualisation of the grid search model set generated by ShellSet. The best results are found in the lower left corner of the search area, corresponding to lower values for fFric and tauMax. This plot was generated with ShellSet's included Python scatter plotter.

Examining the full set of models within this test, using Table 1 and Table C1, we can see that over the entire model set no single model has better or equal results in more than 2/4 of the individual scores that comprise the geometric mean when compared to *New Earth5-049*.

4.2 ShellSet with additional Fault Slip Rate data set

The second example performs the same test as the first example with an added data set that defines the fault slip rate (FSR), see appendix B for further information on this data set.

This additional data set was used to generate an extra misfit for each model, which was then used within the calculation of the geometric mean misfit, now defined as

$$GM = \sqrt[5]{S_{SSR} * S_{GV} * S_{SD} * S_{SA} * S_{FSR}} \tag{3}$$

In the first 2 rows of Table 2 we report the best model of Bird et al. (2008), *Earth5-049*, and the best model of Example 1, *Ex1 29*. As in the first example we rerun these two best models adding the new FSR data set to generate fair comparison models, *New Earth5-049* and *New Ex1 29* respectively.

We note that the fault slip rate data set was not available to the authors of Bird et al. (2008).

Table 2. Previous best models of Bird et al. (2008) and example 1 compared with ShellSet grid search results.

Model	fFric	tauMax	SSR	GV	SD	SA	FSR	Geometric mean
Earth5-049	0.1	2×10^{12}	8.02	16.19	31.28	26.43	×	18.10
Ex1 29	0.06806	1.17×10^{12}	7.03	11.70	31.57	27.10	×	16.29
New Earth5-049	0.1	2×10^{12}	7.96	12.18	30.68	25.43	4.42	12.73
New Ex1 29	0.06806	1.17×10^{12}	7.03	11.70	31.57	27.10	4.78	12.74
1	0.15417	2.5×10^{12}	9.97	13.02	31.10	24.93	3.84	13.10
2	0.4125	2.5×10^{12}	18.84	17.43	33.82	24.90	3.15	15.41
3	0.67083	2.5×10^{12}	26.57	21.09	35.00	25.36	3.22	17.41
4	0.15417	5.5×10^{12}	13.65	21.85	31.04	23.04	4.83	15.95
5	0.4125	5.5×10^{12}	21.90	24.11	33.43	22.28	3.23	16.63
6	0.67083	5.5×10^{12}	28.55	24.74	34.25	22.95	3.20	17.78
7	0.15417	8.5×10^{12}	14.90	23.66	32.51	21.94	4.62	16.34
8	0.4125	8.5×10^{12}	24.02	26.29	34.15	21.31	3.53	17.46
9	0.67083	8.5×10^{12}	30.18	27.00	35.27	22.02	3.43	18.51
28	0.12546	1.17×10^{12}	8.46	11.85	32.06	25.78	3.93	12.66
39	0.09676	1.17×10^{12}	7.59	11.64	31.49	26.94	4.35	12.67
42	0.09676	1.5×10^{12}	7.74	11.96	31.03	26.35	4.39	12.72
45	0.09676	1.83×10^{12}	7.86	12.09	30.87	25.66	4.44	12.73

Scores are reported to 2 decimal places but calculated to greater accuracy.

As in example 1 the search favours lower values for fFric and tauMax however, in this case the more important variable was tauMax.

This search has generated 4 models that are an improvement, or equal, to *New Earth5-049*. Three of these models, 39, 42 and 45, also have better geometric mean scores in example 1, Table 1, where they are model 30, 33 and 36 respectively. The absolute best model is 28 with a geometric mean that is 0.06 lower than the previous best model at 12.66.

The alteration to the geometric mean caused alternate models to be selected between levels which takes the grid search through a different path to find its optimal model. This is made clear by comparing Fig. 1 and Fig. 2.

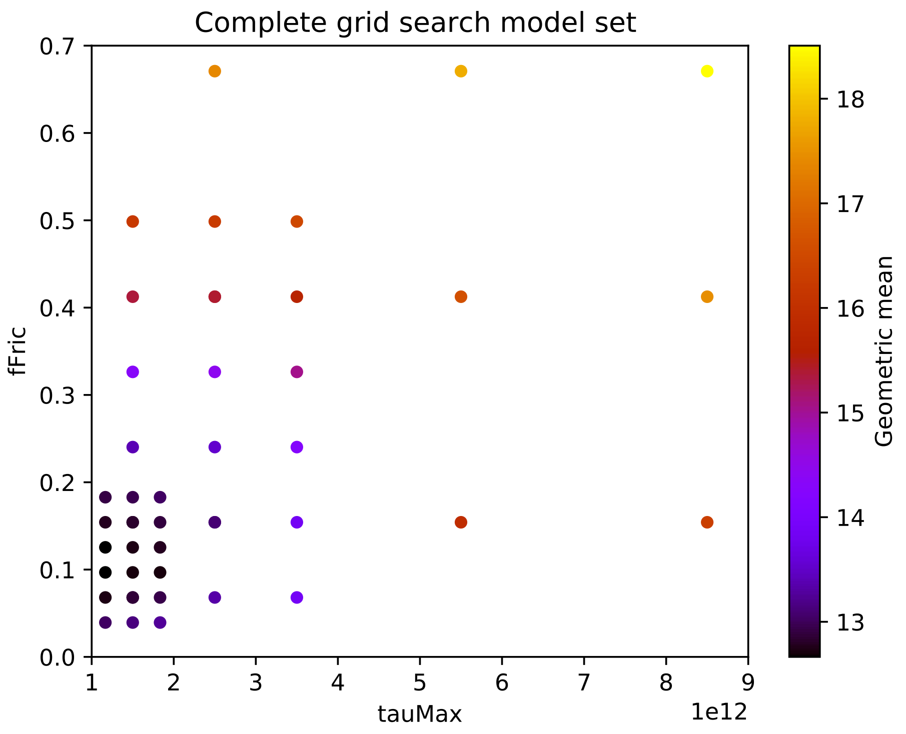

Figure 2. Visualisation of the grid search model set generated by ShellSet. The best results are found in the lower left corner of the search area, corresponding to lower values for fFric and tauMax. This plot was generated with ShellSet's included Python scatter plotter.

5 Performance analysis

ShellSet causes no appreciable delay when individual components are compared to the three original programs as the underlying software is not changed. The linkage within ShellSet offers two opportunities for a performance increase when compared to the original program. Firstly, the total simulation time of each model is made faster by removing the need for a user interface during simulations. Secondly, the MPI framework allows numerous models to be run in parallel. We now provide analysis of these two enhancements.

Removing the need for a user interface between the formerly independent program units decreases the time required to complete any model or set of models. The links, which would otherwise require a user to feed output from one program into the next, are now performed automatically by ShellSet. Comparing the time taken by ShellSet to form these links to a user is a tricky task as it depends on the user and their abilities so it is not something which we will show, however we report the following comparison from our experience.

Table 2 of Bird et al. (2008) shows 18 models in which the fFric and tauMax variables were varied. Each model required one run of OrbData, 4-5 iterations (as stated in the text) of Shells and one run of OrbScore. With all file IO controlled manually by the user these 18 models (72-90 Shells runs) took four days to one week when performed in a serial manor. As reported in Table 3, ShellSet is able to perform 16 models with 5 Shells iterations in less than 65 minutes and 32 models in less than 130 minutes. Although the results shown in Table 3 were performed on more recent hardware and software, a large portion of the time taken for the 18 models was lost to user interface and times where the user was not present - these are not issues with ShellSet.

The use of an MPI framework provides the second opportunity for a performance improvement by allowing multiple models to be run in parallel. This can be tested in a more traditional fashion by comparing the time taken to complete a given number of models using a given number of threads. We also calculate the ratio between the serial and parallel times, known as the speedup.

For simplicity the test was performed over a single search level of a 1D parameter space, varying the number of models and the number of worker threads. We do not report the results where the number of threads is greater than models as ShellSet contains a check on the number of models and worker threads to prevent an over-allocation of compute resources, in reality it would likely have a negative effect by unnecessarily tying up compute resources.

At each number of models the models run are equal, for example the 8 models run by 1 worker are the same as the 8 models run by 8 workers, however altering the number of models will alter the models generated by the grid search. This limits our ability to comment in detail on ShellSet's performance between tests with a different number of models as the underlying models are different, however we are still able to compare the performance of tests when altering the number of workers, as was the aim of ShellSet.

Black entries of Table 3 report time in minutes and seconds, red entries are the speedup compared to a single worker. We calculate the speedup (SU) using:

$$\mathrm{SU} = \frac{T_s}{T_p}$$

where T_s is the time taken by a single worker and T_p is the time taken by multiple workers.

Following the approach outlined in Bird et al. (2008) each timed model consists of: using OrbData to perform an update to the finite element grid; four Shells iterations; a fifth Shells iteration (using updated boundary conditions); and a single OrbScore run to score the final run.

We generate models that we know complete fully, meaning they converge within the user defined MKL iteration limit and pass all input variable criteria.

Table 3 shows that, although clearly sub-linear, ShellSet's MPI framework allows the user to appreciate a performance speedup through using multiple workers to perform models in parallel. The speedup values reported (bracketed red) in Table 3 are around 1.5-2.3.

The ideal speedup would be equal to the number of workers used, i.e., running two models in parallel would be twice as fast as in serial, however due to the relative sizes of the hardware and problem we immediately require more MKL threads than cores that are present in the system for optimal timing. This immediately blunted the performance of the parallel tests and limited their potential speedup. The number of MKL threads used is calculated based on the problem size. We set an upper

Table 3. Performance test performed using an Intel Core i9-12900 CPU at 2.4GHz desktop with 64GB RAM, 16 physical cores (8 performance, 8 efficient) and 24 threads. Times are reported to the nearest second, in mm:ss, speedup is calculated using seconds to 5 decimal places but reported at 2. All results are an average of 3 tests.

Workers	Models						
	1	2	4	8	16	32	64
1	3:50	7:47	16:06	32:22	64:26	128:39	262:09
2	×	5:12 (1.50)	10:19 (1.56)	21:13 (1.52)	42:06 (1.53)	84:59 (1.51)	172:33 (1.52)
4	×	×	7:59 (2.02)	15.45 (2.05)	32:05 (2.01)	62:12 (2.07)	125:49 (2.08)
8	×	×	×	15:02 (2.15)	29:35 (2.18)	58:21 (2.21)	113:03 (2.32)
16	×	×	×	×	29:31 (2.18)	57:22 (2.24)	111:15 (2.36)

1 worker uses 6 MKL threads, 2 workers uses 4 MKL threads, 4 workers uses 2 MKL threads, 8 & 16 workers use 1 MKL thread. Grid search.

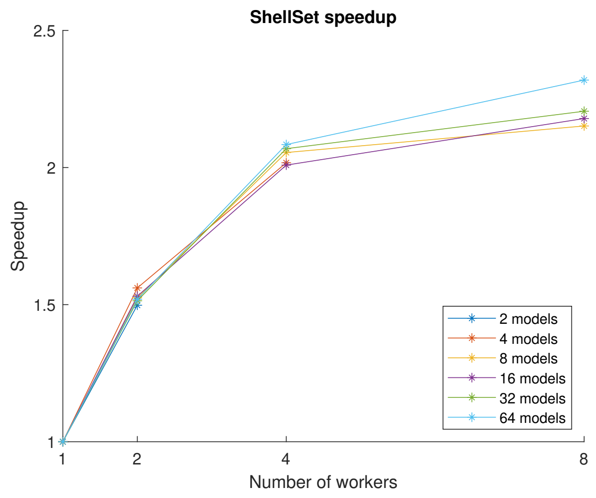

Figure 3. Speedup plot for 1-8 workers, 16 workers excluded as results are comparable to 8 workers.

limit on the number of MKL threads using the number of cores available and the number of MPI threads used. Further testing (not reported) has shown that ShellSet was able to automatically select the optimal number of MKL threads for the model size and number of CPU cores available. Performance is therefore always constrained by the number of MPI threads and the number of machine cores not utilised, as these cores are available to parallelise the MKL subroutines.

Despite this limitation, Table 3 shows that performing models in parallel, even with fewer MKL threads, can yield a good speedup performance. This indicates that, for this problem size, the time lost through reducing the number of MKL threads is more than compensated for by running models in parallel. For the tests with enough models there is shown to be minimal difference between using 8 and 16 workers in parallel, both cases have fully serial MKL regions and so this is likely due the machine having 8 performance and 8 efficient cores, and therefore reaching the limit of its performance.

6 Conclusions and future work

This article has outlined a new MPI parallel dynamic neotectonic modelling software, ShellSet, which has been designed for use by the wider geodynamic modelling community. ShellSet was therefore developed and tested on a computer with widely available capabilities. Despite this we have demonstrated that ShellSet achieves an improved performance, relative to the original software, and speedup when running multiple models in parallel, see Sect. 5.

We have shown, see Sect. 4, two examples of ShellSet's abilities within the target field of study by first improving on an existing global model (developed in Bird et al. (2008)), then further improving on that with the addition of a new data set. Both improvements were completed in a fraction of the time taken to generate the existing global model.

The simplified *hands-off* nature of ShellSet reduces user-program interactions to a minimum, while the addition of a GUI further simplifies those remaining interactions. These improvements open ShellSet to a less experienced user, while also reducing setup errors for all users.

Within the INGV PEACE project *PEricolosità sismica in Appennino CEntrale*, funded within the ongoing *Pianeta Dinamico* project, ShellSet will be used to optimise the rheology of the lithosphere in the Apennine mountain region for better dynamical simulations of neotectonics and associated seismicity.

Future updates include altering the search algorithm to allow ShellSet to better utilise high performance computing (HPC) parallel structures for larger tests and/or model sets. Efficient HPC machine use would enable searching the parameter space to a finer level, faster results, and the use of much larger (in terms of memory) models.

Code and data availability. The current version of ShellSet is available from the project website: https://github.com/JonBMay/ShellSet under the GPL-3.0 license. The exact version of ShellSet used to produce the results shown in this paper is archived on Zenodo, see May et al. (2023), along with all input files, scripts to compile and run ShellSet, the GUI and the scatter plotting routine.

320 ## Appendix A: Grid search

Grid search is a searching algorithm used to tune parameters towards optimum values. It works by partitioning a parameter space into a grid, each cell of which is represented by a single central model whose location defines the value for each of the tested parameters. The upper left side image of Fig. A1 shows an initial 2×2 grid in a 2-dimensional space, with each of the 4 models represented by a coloured point.

325 After a full grid of models has been tested the algorithm will select a number of best models to continue onto the next level. Each of these models has its cell divided in the same fashion as the first level and representative models at the centre of each cell. Figure A1 shows the transition from a single best model to a new level. The red point in the upper left image represents the best model, which is selected for the next level shown in the upper right image of the image. The lower image of Fig. A1 shows an overview of these two levels.

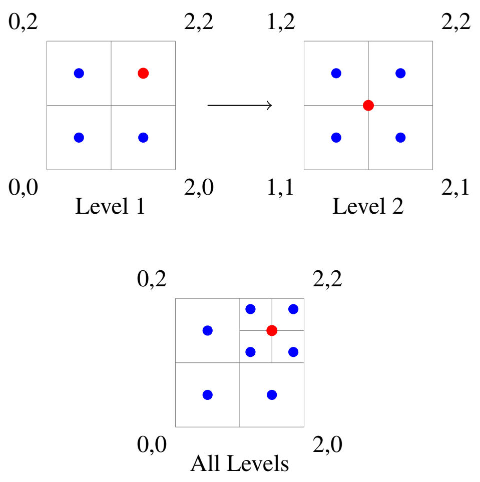

Figure A1. Example 2 level grid search that selects a best cell (red point) and divides that cell into 4 new cells.

330 A typical algorithm will continue in this way until a defined stopping criteria is met, such as a fixed number of levels, a desired model score, a total number of models, etc. ShellSet requires a fixed number of levels, whereupon the program will stop, and optionally allows an early stop should a defined model score be reached.

The base grid search algorithm was adapted from the Fortran grid search version available from May (2022). Like the original, the grid search used in ShellSet has no limit on the number of dimensions it is possible to use.

335 ## Appendix B: Fault slip rate (FSR) data set

We now outline the process followed to create a usable fault slip rate data set, as used in example 2 of Sect. 4, for our simulations.

In Styron and Pagani (2020) the authors report a global active fault database of approximately 13,500 faults collected through the combination of regional data sets. The authors note that approximately 77% of the faults have slip rate information, this provided us with an estimated initial 10395 faults for potential use in our work.

Due to the requirements of our program the database needed some cleaning. Of the approximately 13,500 initial fault traces, around 9,500 faults either have no offset rate or lack upper and lower limits which identify a rate based on a dated offset feature (unbounded rates are model rates, which we do not consider to be data.) A further 684 lack rake info and 1022 fault traces lack necessary dip information. After discounting these fault traces, we were left with 2487 traces which might be used in OrbScore if each aligns with a fault element of the current finite element grid.

Those fault traces remaining are mostly located in one of four areas: the Mediterranean/Tethyan orogenic belt, well covered from Portugal to Pakistan; Japan; New Zealand; and California.

Many of the almost 2,500 fault traces (e.g., those in Japan) did not correspond in position (and orientation) to any of the fault elements of the existing finite element grid as used in Bird et al. (2008). Therefore, they were also removed from our data set.

Remaining fault traces were subjected to the following two comparison filters with respect to the finite element grid of 2008: firstly, the overall azimuth (endpoint-to-endpoint) of the fault trace must be within a $\pm$ 15 degrees tolerance of the grid fault element azimuth; secondly, the shortest distance from the fault trace to the grid fault element must be less than 1/8 of the length of that fault element.

After applying these conditions to the fault traces, we were left with 931 associations. Accounting for the fault traces of the database which match with multiple fault elements left us with 572 faults.

The fault elements of the finite element grid file that have multiple associations within the new data set needed to be controlled to avoid over-weighting of those elements with more associations. This was done by summing the product of “adjacent” fault length (limited to the length of the fault-element) with the offset rate for all the associated traces, then dividing the sum by the length of the fault element to get the aggregated offset rate that the fault element should match.

Appendix C: Figures and tables related to demonstrated real world examples

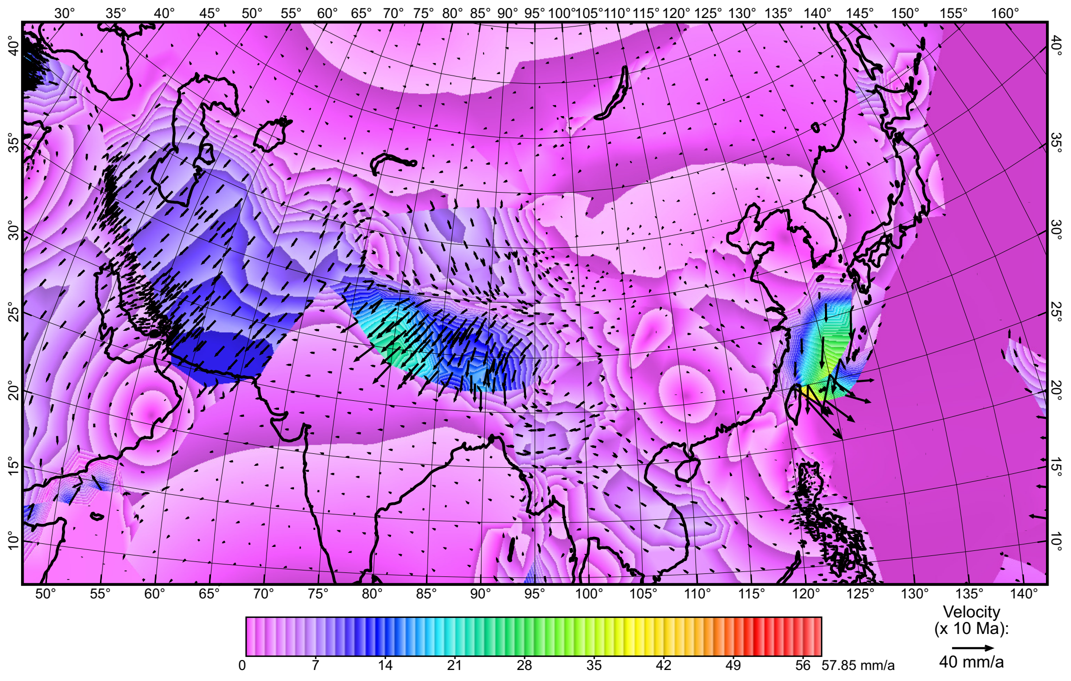

Figure C1. Map of velocity differences, calculated between the best model of Bird et al. (2008) and the first example shown in 4. We focus on Asia as the most striking differences occur here.

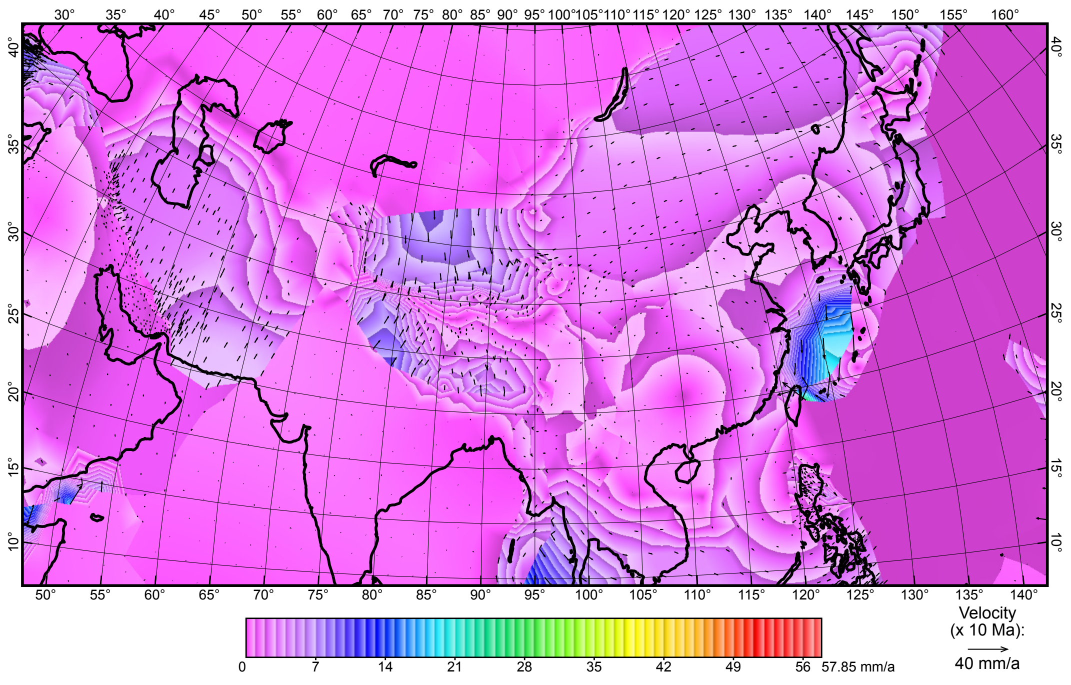

Figure C2. Map of velocity differences, calculated between the best model of Bird et al. (2008) and the second example shown in 4.

Table C1. Example 1 grid search results, excluding the first level which is shown in Table 1.

Global Model Number	fFric	tauMax	SSR	GV	SD	SA	Geometric Mean
10	0.06806	1.50×10^{12}	7.14	12.04	31.28	27.52	16.50
11	0.15417	1.50×10^{12}	9.43	12.38	32.09	25.69	17.61
12	0.24028	1.50×10^{12}	11.75	13.45	32.80	26.23	19.20
13	0.06806	2.50×10^{12}	8.06	13.19	30.49	26.12	17.06
14	0.15417	2.50×10^{12}	9.97	13.02	31.10	24.93	17.81
15	0.24028	2.50×10^{12}	12.18	14.17	32.00	24.83	19.24
16	0.06806	3.50×10^{12}	8.47	14.02	30.24	25.26	17.35
17	0.15417	3.50×10^{12}	10.79	14.80	30.25	24.13	18.48
18	0.24028	3.50×10^{12}	13.09	16.28	31.59	24.09	20.07
19	0.06806	4.50×10^{12}	10.11	19.12	30.36	24.49	19.47
20	0.15417	4.50×10^{12}	12.31	19.80	30.61	23.39	20.44
21	0.24028	4.50×10^{12}	14.52	20.56	31.61	23.25	21.64
22	0.06806	5.50×10^{12}	11.35	21.05	30.87	23.89	20.49
23	0.15417	5.50×10^{12}	13.65	21.85	31.04	23.04	21.49
24	0.24028	5.50×10^{12}	16.02	22.64	31.57	22.67	22.57
25	0.06806	6.50×10^{12}	11.84	21.57	31.41	23.35	20.80
26	0.15417	6.50×10^{12}	14.39	22.75	31.40	22.90	22.03
27	0.24028	6.50×10^{12}	17.19	23.80	32.12	22.34	23.27
28	0.03935	1.17×10^{12}	6.57	12.27	32.00	27.65	16.34
29	0.06806	1.17×10^{12}	7.03	11.70	31.57	27.10	16.29
30	0.09676	1.17×10^{12}	7.59	11.64	31.49	26.94	16.55
31	0.03935	1.50×10^{12}	6.69	12.68	31.73	27.50	16.49
32	0.06806	1.50×10^{12}	7.14	12.04	31.28	27.52	16.50
33	0.09676	1.50×10^{12}	7.74	11.96	31.03	26.35	16.58
34	0.03935	1.83×10^{12}	6.89	13.04	31.46	27.23	16.65
35	0.06806	1.83×10^{12}	7.31	12.27	30.99	27.01	16.55
36	0.09676	1.83×10^{12}	7.86	12.09	30.87	25.66	16.56
37	0.03935	2.17×10^{12}	7.17	13.43	31.21	27.00	16.88
38	0.06806	2.17×10^{12}	7.56	12.64	30.65	26.58	16.70
39	0.09676	2.17×10^{12}	8.10	12.44	30.48	25.29	16.69
40	0.03935	2.50×10^{12}	7.56	13.95	31.04	26.82	17.22
41	0.06806	2.50×10^{12}	8.06	13.19	30.49	26.12	17.06
42	0.09676	2.50×10^{12}	8.63	12.96	30.33	24.88	17.05
43	0.03935	2.83×10^{12}	7.60	14.23	31.07	27.35	17.41
44	0.06806	2.83×10^{12}	8.21	13.48	30.40	25.65	17.14
45	0.09676	2.83×10^{12}	8.95	13.41	30.19	24.58	17.28

Table C2. Example 2 grid search results, excluding the first level which is shown in Table 2.

Model	fFric	tauMax	SSR	GV	SD	SA	FSR	Geometric Mean
10	0.06806	1.50×10^{12}	7.14	12.04	31.28	27.52	4.80	12.89
11	0.15417	1.50×10^{12}	9.43	12.38	32.09	25.69	3.63	12.85
12	0.24028	1.50×10^{12}	11.75	13.45	32.80	26.23	3.20	13.42
13	0.06806	2.50×10^{12}	8.06	13.19	30.49	26.12	4.98	13.33
14	0.15417	2.50×10^{12}	9.97	13.02	31.10	24.93	3.84	13.10
15	0.24028	2.50×10^{12}	12.18	14.17	32.00	24.83	3.33	13.55
16	0.06806	3.50×10^{12}	8.47	14.02	30.24	25.26	5.68	13.88
17	0.15417	3.50×10^{12}	10.79	14.80	30.25	24.13	4.40	13.87
18	0.24028	3.50×10^{12}	13.09	16.28	31.59	24.09	3.62	14.24
19	0.32639	1.50×10^{12}	14.61	14.66	33.78	26.32	3.16	14.32
20	0.4125	1.50×10^{12}	18.22	15.84	34.67	26.27	3.27	15.37
21	0.49861	1.50×10^{12}	21.68	17.40	35.14	26.01	3.25	16.21
22	0.32639	2.50×10^{12}	15.25	16.03	33.10	24.99	3.11	14.45
23	0.4125	2.50×10^{12}	18.84	17.43	33.82	24.90	3.15	15.41
24	0.49861	2.50×10^{12}	22.18	18.94	34.28	24.85	3.17	16.25
25	0.32639	3.50×10^{12}	16.32	18.42	32.50	23.78	3.32	15.05
26	0.4125	3.50×10^{12}	19.90	19.88	33.14	23.59	3.10	15.72
27	0.49861	3.50×10^{12}	23.11	21.20	33.84	23.79	3.08	16.48
28	0.12546	1.17×10^{12}	8.46	11.85	32.06	25.78	3.93	12.66
29	0.15417	1.17×10^{12}	9.28	12.16	32.36	25.93	3.60	12.78
30	0.18287	1.17×10^{12}	10.03	12.51	32.58	26.09	3.38	12.93
31	0.12546	1.50×10^{12}	8.63	12.12	31.74	25.46	3.97	12.74
32	0.15417	1.50×10^{12}	9.43	12.38	32.09	25.69	3.63	12.85
33	0.18287	1.50×10^{12}	10.17	12.71	32.28	25.84	3.41	12.97
34	0.12546	1.83×10^{12}	8.84	12.36	31.29	24.99	4.02	12.80
35	0.15417	1.83×10^{12}	9.57	12.59	31.76	25.35	3.69	12.90
36	0.18287	1.83×10^{12}	10.35	12.97	31.93	25.53	3.45	13.04
37	0.03935	1.17×10^{12}	6.57	12.27	32.00	27.65	5.31	13.05
38	0.06806	1.17×10^{12}	7.03	11.70	31.57	27.10	4.78	12.74
39	0.09676	1.17×10^{12}	7.59	11.64	31.49	26.94	4.35	12.67
40	0.03935	1.50×10^{12}	6.69	12.68	31.73	27.50	5.32	13.16
41	0.06806	1.50×10^{12}	7.14	12.04	31.28	27.52	4.80	12.89
42	0.09676	1.50×10^{12}	7.74	11.96	31.03	26.35	4.39	12.72
43	0.03935	1.83×10^{12}	6.89	13.04	31.46	27.23	5.34	13.26
44	0.06806	1.83×10^{12}	7.31	12.27	30.99	27.01	4.83	12.94
45	0.09676	1.83×10^{12}	7.86	12.09	30.87	25.66	4.44	12.73

Author contributions. JBM wrote the ShellSet code and additional Python routines, designed and ran the examples and performance tests, wrote the user guide, and wrote the first draft of the article after which all authors contributed. PB and MMCC supervised the correct combination of software into the final program. PB generated the new fault slip rate data set, and generated map figures. MMCC devised the work.

Competing interests. The authors declare that they do not have any competing interests.

References

Austermann, J., Ben-Avraham, Z., Bird, P., Heidbach, O., Schubert, G., and Stock, J. M.: Quantifying the forces needed for the rapid change of Pacific plate motion at 6 Ma, Earth and Planetary Science Letters, 307, 289–297, 2011.

Bird, P.: Testing hypotheses on plate-driving mechanisms with global lithosphere models including topography, thermal structure, and faults, Journal of Geophysical Research: Solid Earth, 103, 10 115–10 129, 1998.

Bird, P., Ben-Avraham, Z., Schubert, G., Andreoli, M., and Viola, G.: Patterns of stress and strain rate in southern Africa, Journal of Geophysical Research: Solid Earth, 111, 2006.

Bird, P., Liu, Z., and Rucker, W. K.: Stresses that drive the plates from below: Definitions, computational path, model optimization, and error analysis, Journal of Geophysical Research: Solid Earth, 113, 2008.

Carafa, M., Barba, S., and Bird, P.: Neotectonics and long-term seismicity in Europe and the Mediterranean region, Journal of Geophysical Research: Solid Earth, 120, 5311–5342, 2015.

Jian-jian, Z., Feng-li, Y., and Wen-fang, Z.: Tectonic characteristics and numerical stress field simulation in Indosinian-early Yanshanian stage, Lower Yangtze Region, Geological Journal of China Universities, 16, 475, 2010.

Kalbas, J. L., Freed, A. M., Ridgway, K. D., Freymueller, J., Haeussler, P., Wesson, R., and Ekström, G.: Contemporary fault mechanics in southern Alaska, 2008.

Kong, X. and Bird, P.: SHELLS: A thin-shell program for modeling neotectonics of regional or global lithosphere with faults, Journal of Geophysical Research: Solid Earth, 100, 22 129–22 131, 1995.

Liu, Z. and Bird, P.: Finite element modeling of neotectonics in New Zealand, Journal of Geophysical Research: Solid Earth, 107, ETG–1, 2002a.

Liu, Z. and Bird, P.: North America plate is driven westward by lower mantle flow, Geophysical Research Letters, 29, 17–1, 2002b.

May, J.: Fortran Grid Search v1.0.0 [software], https://doi.org/10.5281/zenodo.6940088, 2022.

May, J., Bird, P., and Carafa, M.: ShellSet v1.1.0 [software], https://doi.org/10.5281/zenodo.7986808, 2023.

Stamps, D., Flesch, L., and Calais, E.: Lithospheric buoyancy forces in Africa from a thin sheet approach, International Journal of Earth Sciences, 99, 1525–1533, 2010.

Styron, R. and Pagani, M.: The GEM global active faults database, Earthquake Spectra, 36, 160–180, 2020.

Tunini, L., Jiménez-Munt, I., Fernandez, M., Vergés, J., and Bird, P.: Neotectonic deformation in central Eurasia: A geodynamic model approach, Journal of Geophysical Research: Solid Earth, 122, 9461–9484, 2017.