MTfit: A Bayesian approach to seismic moment tensor inversion

D.J. Pugh[1,2,3*], R.S. White [1]

[1]Bullard Laboratories, Department of Earth Sciences, University of Cambridge, CB3 0EZ, UK

[2]Schlumberger Gould Research Centre, High Cross, Cambridge, UK

[3]Now at McLaren Applied Technologies, McLaren Technology Centre, Chertsey Road, Woking, UK

*Email:david.j.pugh@cantab.net

For publication in SRL: Electronic Seismologist.

In original form 13th December 2017

Abstract

MTfit is a Python module for Bayesian moment tensor source inversion of earthquake seismic data using polarities, amplitudes or amplitude ratios. It can solve for double couple or full moment tensor solutions, taking into account uncertainties in polarities, take-off angles of the rays from the source to the receiver, and amplitudes. It provides an easily accessible and extendable approach to earthquake source inversion which is particularly useful for local and regional events.

Introduction

Earthquake source inversion is carried out at many seismological observatories and research facilities around the world. Pugh *et al.* (2016b) introduced a Bayesian approach to estimating the moment tensor of the source using polarities and amplitude ratios, which was extended to include automated Bayesian polarity probability estimates by Pugh *et al.* (2016a). This approach differs from existing approaches, such as FPFIT (Reasenberg & Oppenheimer, 1985), HASH (Hardebeck & Shearer, 2002, 2003) and FOCMEC (Snoke, 2003), because it uses polarities and amplitude ratios in a Bayesian framework to estimate the full source probability density function (PDF) for the double-couple and full moment tensor model spaces. The approach can include location and velocity model uncertainties, as well as marginalizing over measurement uncertainties in the data.

The approach of Pugh *et al.* (2016b) has been developed into MTfit, a Python package for source inversion. Python is a common programming and scripting language with many scientific modules available, both for mathematical calculation such as NumPy (https://www.numpy.org) and SciPy (https://www.scipy.org/), and for seismological applications such as ObsPy (https://www.obspy.org) (Beyreuther *et al.*, 2010).

Python and many of its modules are open source, allowing easy code development and removing licensing restrictions. Moreover, Python is platform independent, intuitive, and accessible, with a good shell interface in the form of iPython (https://ipython.org/). It is used in many fields and is easy to install on almost any computer platform. Python can also interface easily with C and Fortran libraries, and can call functions from compiled C modules, such as those generated with Cython (http://cython.org/), with no difference from normal Python functions. Note that earlier versions of the code were referred to as MTINV, but the name has been changed to MTFfit to avoid a clash with a previous use of the name MTINV. MTfit has already been used in several studies, including these reported by Wilks *et al.* (2015), Greenfield & White (2015), Pugh *et al.* (2016b), Schuler *et al.* (2016), Mildon *et al.* (2016), Smith *et al.* (2017) and Hudson *et al.* (2017).

In this paper, the functionality of MTfit is introduced, and examples of the approach are shown. The model probability estimates derived from the Bayesian evidence are explored, and methods of extending MTfit are presented. Lastly, two examples of plotting the results from MTfit are shown. A flow diagram outlining the main modules of MTfit is shown in Figure 1.

Moment Tensor Inversion

MTfit uses the Bayesian source inversion approach from Pugh *et al.* (2016b). The solutions are estimated using polarities and amplitude ratio data, although the code is extendible, so it is possible to include other data types in this framework. MTfit incorporates uncertainty estimates both in the data, such as those arising due to noise, and due to the model (and location), in the resultant posterior PDF. We have developed three sampling approaches, each with different advantages and disadvantages (Pugh 2015). MTfit can also be used for relative amplitude

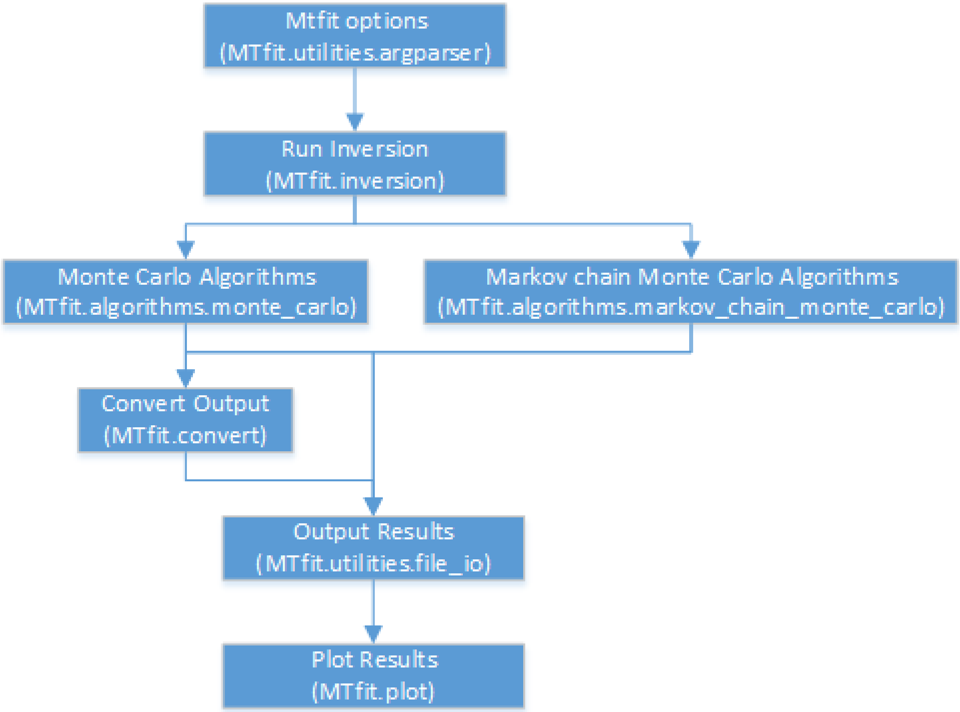

Fig. 1: Flow diagram outlining the main steps in the moment tensor inversion package.

inversions (Pugh 2015).

The MTfit approach evaluates the data likelihood ($p\,(data|model)$) for the observations and measurement uncertainties at each receiver over a range of random moment tensor samples. These likelihoods are combined to produce the likelihood for all the receivers. Location and model uncertainties are included by generating samples of locations of the receivers on the focal sphere, corresponding to the distribution of possible locations of the earthquake, which are marginalized over to produce the location marginalized likelihood. The resultant likelihood is then saved. If a Markov chain algorithm is used, the moment tensor samples are generated and saved using the Markov chain algorithm.

MTfit can be called both from the command line and from within the Python interpreter. On the command line:

```
$ MTfit event_data.inv
```

is equivalent to

```
>>> import MTfit
>>> MTfit.MTfit(data_file="event_data.inv")
```

in the Python interpreter.

Three search algorithms have been implemented. The simplest is a Monte Carlo (MC) random sampling algorithm, which can be limited either by the number of samples or by the elapsed time (in seconds):

```
$ MTfit --algorithm=iterate --max-samples=100000 event_data.inv
$ MTfit --algorithm=time --max-time=600 event_data.inv
```

The other two algorithms are Markov chain Monte Carlo (McMC) approaches: Metropolis-Hastings McMC and reversible jump McMC. These are described in detail in Pugh (2015). The two McMC algorithms can be selected on the command line:

```
$ MTfit --algorithm=mcmc --chain-length=100000 event_data.inv
$ MTfit --algorithm=transdmcmc --chain-length=100000 event_data.inv
```

MTfit can be constrained to the double-couple space or allowed to explore the full moment tensor space. This also allows comparisons to be made between the different models and can be used to evaluate the model probabilities. Additional sampling algorithms can be added using entry points. The prior distribution for generating the source models can also be changed, either to select specific submodels or to change the prior distribution on the source model. An example of the former is the strike-slip example in MTfit.extensions.model_sampling_strike_slip, which generates only strike-slip sources rather than full double-couple sources.

The full moment tensor space used in the calculation has 5 free parameters (the 6 parameters from the symmetrical moment tensor normalised to 1 because the data types cannot constrain the seismic moment).

There are several different output formats, including a MATLAB® format and a format based on the .hyp format of NonLinLoc (Lomax *et al.*, 2000, 2009), with a binary structure for the moment tensor samples, and it is easy to extend the output formats using the entry points described below.

A Simple Example

This example shown in Figure 2, using real data collected from the Krafla volcano in northern Iceland can be found at https://github.com/djpugh/MTfit/tree/master/examples/SRL_examples/krafla.py. It is a strongly non-double-couple event, with manually picked P- and S-wave arrival times and P-wave polarities, located using NonLinLoc (Lomax *et al.* 2000, 2009). In this case, it is difficult to measure the amplitudes of the S-wave arrivals, so amplitude ratios are ignored. Instead, polarities and polarity probabilities (Pugh *et al.*, 2016a) are used separately to constrain the source, along with the location data. This event is shown in Pugh *et al.* (2016b) and investigated in more detail in Mildon *et al.* (2016), and has large location uncertainty, especially in the take-off angle of the source-to-receiver arrays (Figure 2). The script used for generating Figure 2 is equivalent to outputting the data file and location uncertainty from Python:

```
>>> from MTfit.examples.example_data import krafla_event,krafla_location
>>> data = krafla_event()
>>> open('krafla_event.scatangle', 'w').write(krafla_location())
>>> import pickle
>>> pickle.dump(data, open('krafla_event.inv','wb'))
```

and calling MTfit with the command line options:

```
$ MTfit --location_pdf_file_path=krafla_event.scatangle --algorithm=iterate
```

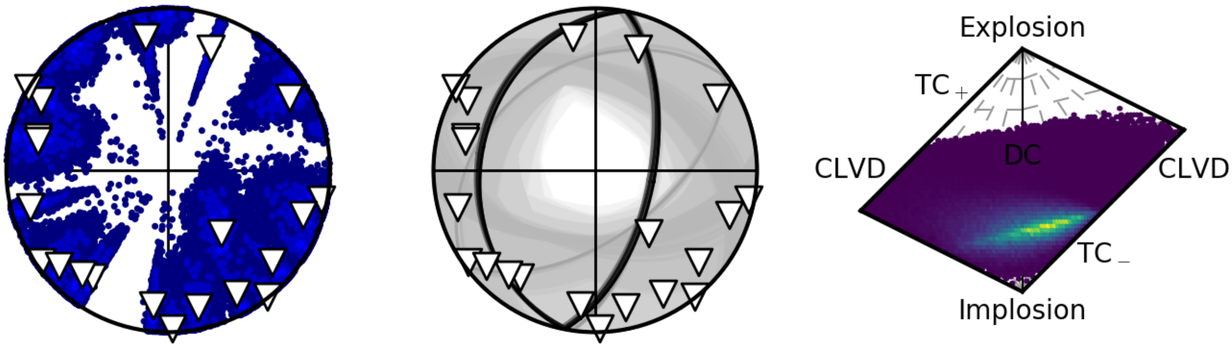

Fig. 2: Krafla example results from the script at https://github.com/djpugh/MTfit/tree/master/examples/SRL_examples/krafla.py (run with 1,000,000 samples). The first plot shows the station distribution of observed receivers on the focal sphere, all with negative polarity, determined from the NonLinLoc estimate of the location PDF. The lighter points correspond to more likely receiver locations, and the maximum likelihood station locations with observed polarities are shown as triangles. The second plot shows the fault plane distribution for the double-couple constrained solution, with darker fault planes more likely. The last plot shows the Hudson type plot of the marginalized source-type PDF from the full moment tensor solution, with dark regions corresponding to low-probability source-types and lighter areas to higher probability types.

```
    --pmem=1 --double-couple --max-samples=
    --inversion-options=PPolarity --convert --bin-scatangle krafla_event.inv
$ MTfit --location_pdf_file_path=krafla_event.scatangle --algorithm=iterate
    --pmem=1 --max-samples=10000000  --inversion-options=PPolarity --convert
    --bin-scatangle krafla_event.inv
```

It is possible to run these inversions using other algorithms, such as those described in Pugh (2015), as described in the MTfit documentation.

The inversion also produces distributions of the moment tensor parameters which can be plotted using the MTplot command to show the distribution of individual parameters (Figure 3).

Model Probabilities

Pugh *et al.* (2016b) introduced a method of estimating the model probabilities using the Bayesian evidence. MTfit can include the Bayesian evidence estimation required for this

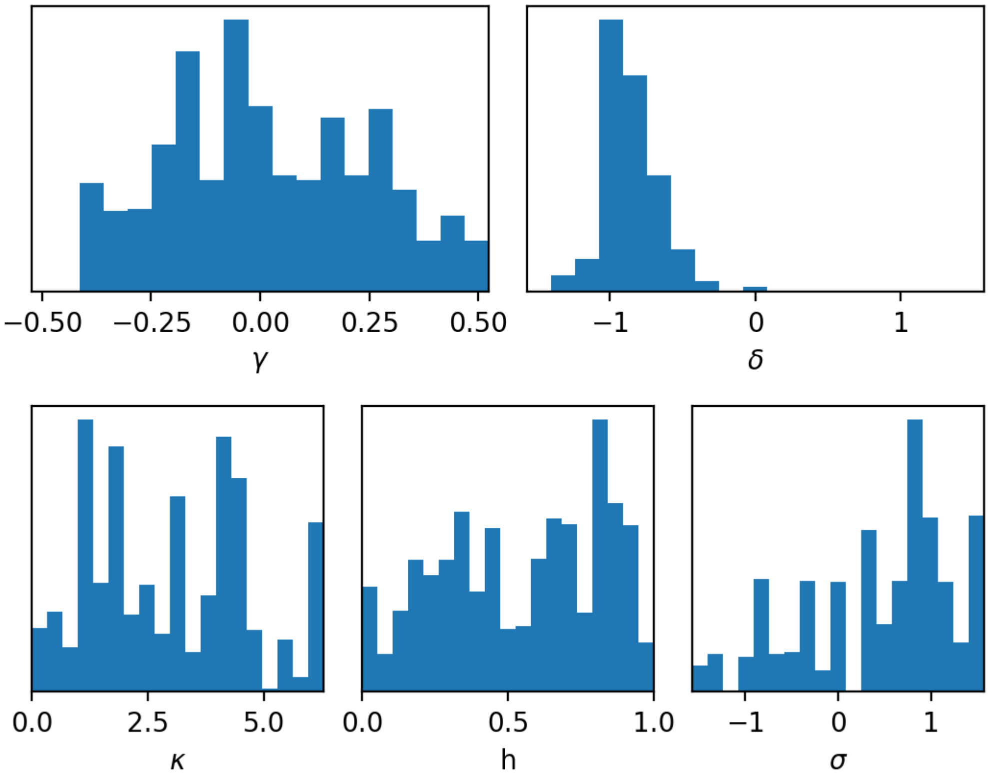

Fig. 3: Marginalised posterior parameter distribution histogram for the five parameters described in Tape & Tape (2012) for the event shown in Fig. 2. γ and δ describe the moment tensor pattern, while κ (strike angle), h (cosine of dip) and σ (rake) describe the orientation. All parameters are dimensionless except κ and σ, which are in radians. This shows that the distributions are well constrained for the δ component, but are less well constrained for the fault plane orientation and γ component.

calculation in its results. To estimate the model probabilities for the double-couple and full moment tensor models, it is necessary to run the inversions in both the model spaces. The –double-couple command line flag will constrain the model to the double-couple space; otherwise the full moment tensor space is used. The Bayesian evidence values generated by each inversion can be combined and normalized to produce the model probabilities

$$
\begin{aligned}
\ln\left(\mathcal{B}_{\max}\right) &= \max\left(\ln\left(\mathcal{B}_{\mathrm{DC}}\right), \ln\left(\mathcal{B}_{\mathrm{MT}}\right)\right), && (1)\\
p_{\mathrm{DC}} &= \frac{\mathrm{e}^{\ln(\mathcal{B}_{\mathrm{DC}})-\ln(\mathcal{B}_{\max})}}{\mathrm{e}^{\ln(\mathcal{B}_{\mathrm{DC}})-\ln(\mathcal{B}_{\max})}+\mathrm{e}^{\ln(\mathcal{B}_{\mathrm{MT}})-\ln(\mathcal{B}_{\max})}}, && (2)\\
p_{\mathrm{MT}} &= \frac{\mathrm{e}^{\ln(\mathcal{B}_{\mathrm{MT}})-\ln(\mathcal{B}_{\max})}}{\mathrm{e}^{\ln(\mathcal{B}_{\mathrm{DC}})-\ln(\mathcal{B}_{\max})}+\mathrm{e}^{\ln(\mathcal{B}_{\mathrm{MT}})-\ln(\mathcal{B}_{\max})}}, && (3)
\end{aligned}
$$

where $\mathcal{B}$ corresponds to a Bayesian evidence estimate (MTfit outputs the logarithm of the Bayesian evidence estimate) and p_{DC} and p_{MT} correspond to the double-couple and full moment tensor model probabilities respectively. As MTfit can be extended (see below), it is possible to introduce new model constraints, and the model probabilities can be extended using a similar logic to that in Eqs 1 – 3. For the example shown in Figure 2, the p_{DC} estimate is 0.0008, and the p_{MT} estimate is 0.9992. This can be calculated using the MTfit.probability.model_probabilities() function, which takes the calculated logarithm of the Bayesian evidence estimates as arguments.

Alternatively, the model probability can be estimated using the transdimensional (reversible jump) McMC algorithm, selected using –algorithm = transdmcmc. This algorithm uses the reversible-jump approach described in Pugh (2015). The model probability estimates from this algorithm are consistent with those from the Bayesian evidence estimators (Pugh, 2015), and both estimates can be used as a hypothesis test for whether or not the source is double-couple.

Figure 4 shows inversions for a synthetic double-couple source with a range of different signal to noise ratios (SNR) and polarity picks. As the SNR decreases, fewer picks can be made on arrivals, thus reducing the constraints available for fitting. The two left hand columns show the results using only polarity picks, while the two right hand columns include constraints from polarity and amplitude data. We show the solutions if they are constrained to be double-couple in the first and third columns. The constraints also allowed full moment tensor solutions to be calculated, and these are shown in the second and fourth columns. It is clear that, as expected, the solutions are constrained better for the higher SNR cases. But there is a marked improvement in the constraints if amplitude ratios as well as polarity data are also taken into account (third and fourth columns in Figure 4). Indeed, for the better SNR cases, down to SNR of 3, the moment tensor solutions that include amplitude ratios still return a double-couple solution as the best fit, and even with a SNR of 2, the best solution is close to a double couple: these full moment tensor solutions also faithfully reproduce the strikes and dips of the nodal planes of the synthetic example we used (top row, Figure 4), at least down to SNR as low as 3.

Computer Run Times

Typical run times depend on the sampling size and the chosen algorithm as well as details of the particular moment tensor solution. Figure 5 shows processor elapsed time for calculation of a typical double couple source mechanism using a relatively slow single core computer. The random sampling and McMC algorithms produce comparable results, but the McMC calculation takes about 5 times longer to achieve similar resolution. Random sampling requires typically 50 million samples to produce a good sampling of the PDF, though the peak is sharpened if the number of samples is increased to 500 million. The McMC approach requires far fewer samples than random sampling, with a chain length of 50,000 for the McMC approach giving comparable results to 100 million random samples. However, the calculation of the likelihood for a large number of samples is much faster with the random sampling algorithm because the McMC

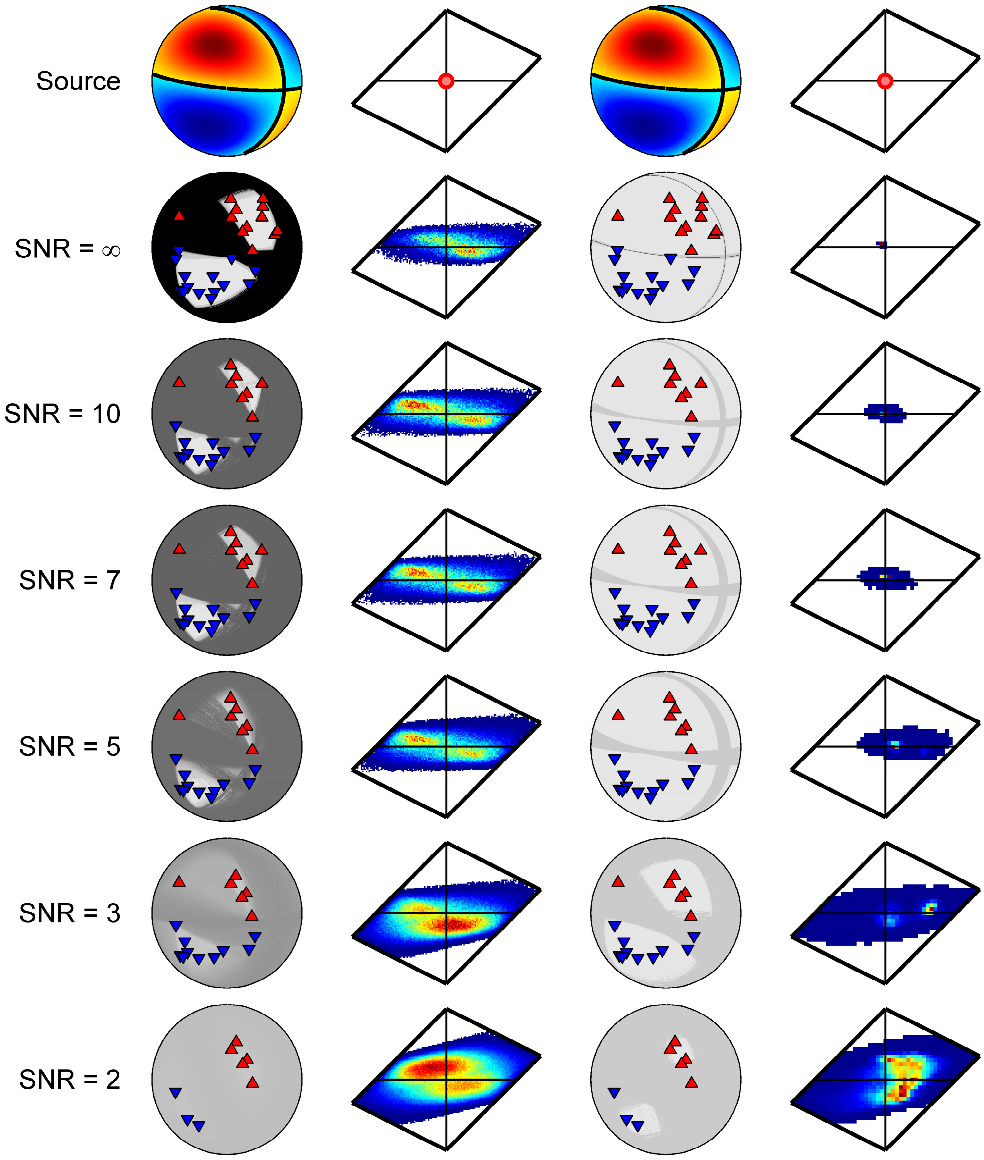

Fig. 4: Lower hemisphere equal area projections and Hudson plots of the source PDF for a synthetic double-couple source for a range of data uncertainties, corresponding to SNR = infinity, SNR = 10, SNR = 7, SNR = 5, SNR = 3 and SNR = 2. The first and third columns show the source PDF for the solution constrained to be double-couple only. The second and fourth columns show the source PDF for the full moment tensor solution. The first two columns show the solutions for inversions using only polarity data, and the second two columns show the solutions using polarity and amplitude ratio data. Manually picked station first motions are given by upward red or downward blue triangles. For the focal sphere plots, possible fault planes are given by dark lines. The most likely fault planes are given by the darkest lines. For the Hudson plots, high probability is red and low probability is in blue.

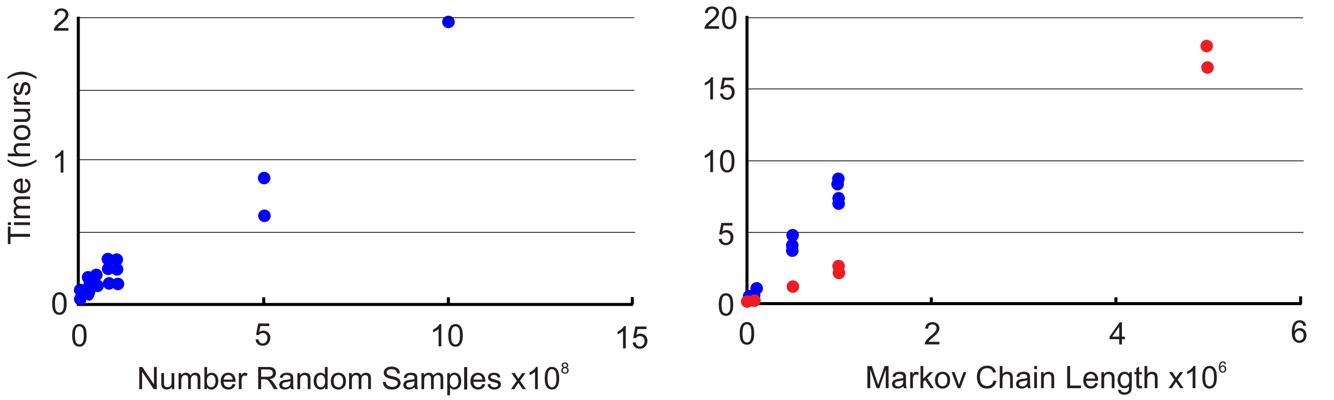

Fig. 5: Elapsed time on a single core computer for different sample sizes of the random sampling (left plot) and for the McMC algorithms with different chain lengths (right plot) for a double couple source with no uncertainties in the input data. The red dots in the McMC case correspond to the trans-dimensional McMC algorithm and the blue dots correspond to the standard algorithm.

algorithm requires extra computations to obtain new samples. The random sampling algorithm can also readily be parallelised, with n processors reducing the calculation time n-fold. Although there are techniques for sampling multiple Markov chains in parallel, the overall gain in speed is much less than for random sampling.

If location uncertainty and model uncertainty are also included in the forward model, there is a significant increase in the time taken to run the random sampling algorithm before sufficient sampling has been achieved because the algorithm is running a Monte-Carlo test over all the location uncertainties: for m-location samples this is equivalent to calculating m-events (where m is typically 500 to 1000 or more). The additional uncertainties have less effect on the time taken to run the McMC algorithm because it requires fewer samples at each iteration. An example of the elapsed calculation time for inversions including location and model uncertainities is shown in Figure 6.

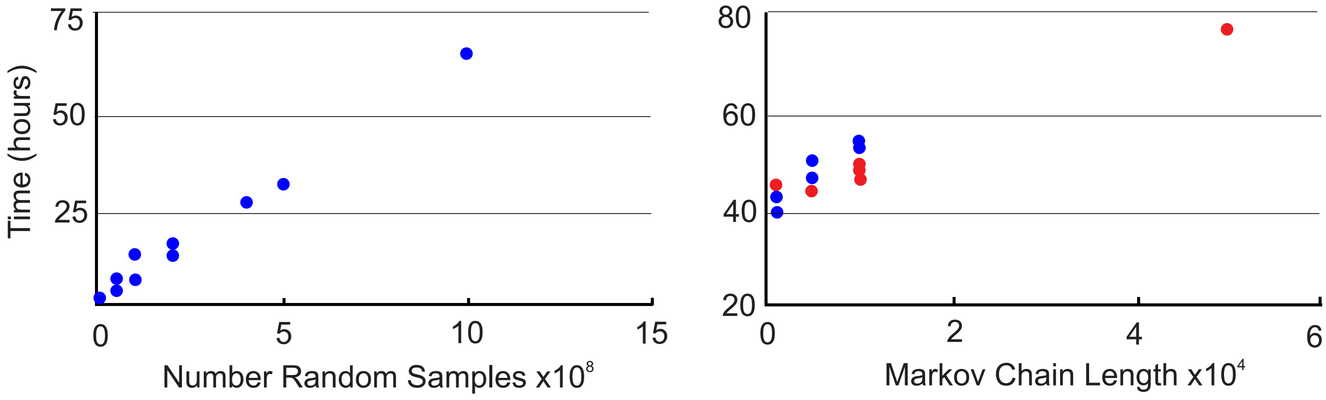

Fig. 6: Elapsed time on a single core computer for different sample sizes of the random sampling algorithm (left plot) and for the McMC algorithms with different chain lengths (right plot) for a double couple source which includes location and velocity model uncertainties. The red dots in the McMC case correspond to the trans-dimensional McMC algorithm and the blue dots correspond to the standard algorithm. The velocity model and location uncertainty in the source was included with a one degree binning, reducing the number of location samples from 50,000 to 5,463.

Extending MTfit

MTfit has been written so that it is easy to extend. This is achieved using the Python setuptools module (https://pythonhosted.org/setuptools/), which provides entry points for a module. These entry points enable a module to check for other functions in different modules that have been advertised at this entry point, and can call them without any changes to the source code of either module. The MTfit documentation provides a more comprehensive description of the entry points, and how to call them, but a small overview is provided here.

Table 1 shows the list of entry points for MTfit. This section presents a step-by-step guide for installing an example data parser entry point.

First, the parser code must be written, which requires understanding the format of the input data, and parsing the required observations to be used in MTfit. The return data format is a Python dictionary of data per event, with the results for multiple events corresponding to a list of dictionaries.

https://github.com/djpugh/MTfit/tree/master/examples/SRL_examples/simple_parser.py

Entry Point	Description
MTfit.cmd_opts	Command line options
MTfit.cmd_defaults	Default parameters for the command line options
MTfit.tests	Test functions for the extensions
MTfit.pre_inversion	Function to be called with all kwargs before the inversion object is initialised
MTfit.post_inversion	Function to be called with all available kwargs after the inversion has occurred
MTfit.extensions	Functions that replace the call to the inversion using all the kwargs
MTfit.parsers	Functions that return the data dictionary from an input filename
MTfit.location_pdf_parsers	Functions that return the location PDF samples from an input filename
MTfit.output_data_formats	Functions that format the output data into a given type, often linked to the output format
MTfit.output_formats	Functions that output the results from the output_data_formats
MTfit.process_data_types	Functions to convert input data into correct format for new data types in forward model
MTfit.data_types	Functions to evaluate the forward model for new data types
MTfit.parallel_algorithms	Search algorithms that can be run (in parallel) like MC random sampling
MTfit.directed_algorithms	Search algorithms that are dependent on the previous value (e.g., McMC)
MTfit.sampling	Function that generates new moment tensor samples in the MC random sampling algorithm
MTfit.sampling_prior	Function that calculates the prior probability distribution either in the McMC algorithm or the MC Bayesian evidence estimate
MTfit.sample_models	Function that generates random samples according to some source model
MTfit.plot	Callable class for source plotting using matplotlib
MTfit.plot_read	Function that reads the data from a file for the MTplot class
MTfit.documentation	Installs the documentation for the extension
MTfit.source_code	Installs the source code documentation for the extension

Tab. 1: List of MTfit entry points and their short descriptions. For details see the MTfit documentation.

shows an example parser for a simple data format of

ReceiverName\tPolarity\tError\tAzimuth\tTakeOffAngle.

This parser can be installed using the MTfit.parsers entry point, which requires a setuptools setup.py file for the parser, which should contain the entry point definition:

```
kwargs['entry_points'] = {'MTfit.parsers': ['.sim = example:simple_parser']}
```

With the parser installed, input files that end in .sim can be read by MTfit.

Similar approaches for the other entry points allow further extension of MTfit.

Plotting Results

MTfit also has a plotting submodule, MTfit.plot, which uses matplotlib (https://www.matplotlib.org) to plot the results. It can handle several different plot types, including beachball plots, fault plane plots, Riedesel-Jordan plots (Riedesel & Jordan, 1989), radiation pattern plots, lune plots (Tape & Tape, 2012), and Hudson plots (Hudson *et al.*, 1989). These are shown in Figure 7, which also shows several representations of the source PDF on the fault plane, lune, and Hudson plots. The MTfit.plot entry point allows other plot types to be added easily.

An example script for generating the plots in Figures 2 and 7 is shown in https://github.com/djpugh/MTfit/tree/master/examples/SRL_examples/plot_krafla.py

There is a similar MATLAB® module, MTplot, available from https://github.com/djpugh/MTplot, which can produce similar plot types and also several additional ones.

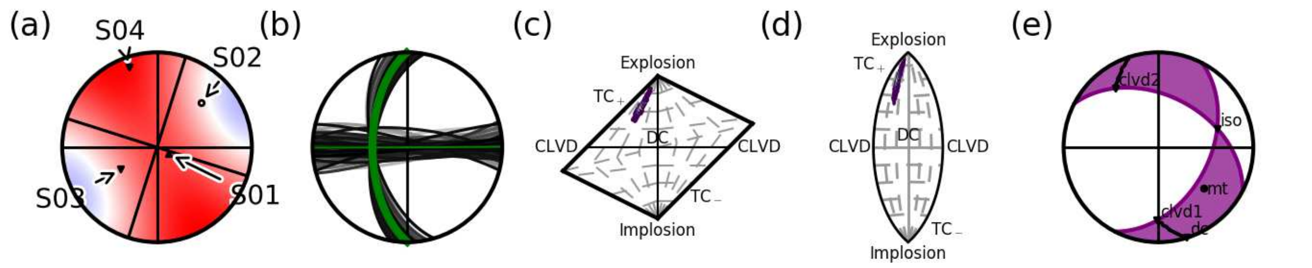

Fig. 7: MTplot examples showing (a) an equal area projection of a beachball for an example moment tensor source, (b) fault plane distribution showing the mean orientation in green, (c) Hudson and (d) lune type plots of a full moment tensor PDF, and (e) a Riedesel-Jordan type plot of an example moment tensor source.

Conclusion

MTfit is a Python module for Bayesian source inversion using different data types. It has been written to allow easy extension using Python and C modules. It has an in-built test suite, which allows changes to the code base to be tested, and it is platform independent, requiring only Python. It has been written to take advantage of parallel computation, both on a single machine and over a larger cluster, using MPI and multiprocessing.

MTfit provides an easily accessible and extendable updated approach to source inversion. The detailed documentation and package can be accessed at https://github.com/djpugh/MTfit.

Data and Resources

The example data used here are included in the MTfit package and have been published in Mildon *et al.* (2016). The MTfit package and detailed documentation is available from https://github.com/djpugh/MTfit for research and teaching i.e. for non-commercial use only. The methods incorporated into the MTfit package are patents-pending, protected, and licensed intellectual property. Applications for commercial use of the MTfit package and/or its underlying methodologies should be made to either Schlumberger or Cambridge Enterprise Limited.

Acknowledgments

The MTfit code was developed as part of a PhD project (Pugh 2015), funded under a Natural Environment Research Council (NERC) studentship at the University of Cambridge, as a CASE award with Schlumberger. Seismometers were borrowed from the NERC SEIS-UK (loan 842). We thank those who have helped, advised, and tested releases of MTfit during its development, including Mike Williams, Phil Christie, Matt Wilks, Tim Greenfield, Rob Green, Jenny Woods, and Thorbjorg Agustsdottir, among others. Department of Earth Sciences, Cambridge contribution number esc.4083.

References

Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., & Wassermann, J., 2010. ObsPy: a Python toolbox for seismology, *SRL: Electronic Seismologist*, **81**, 530–533.

Greenfield, T. & White, R. S., 2015. Building Icelandic igneous crust by repeated melt injections, *Journal of Geophysical Research*, **120**, doi: 10.1002/2015JB012009.

Hardebeck, J. L. & Shearer, P. M., 2002. A new method for determining first-motion focal mechanisms, *Bulletin of the Seismological Society of America*, **92**(6), 2264–2276.

Hardebeck, J. L. & Shearer, P. M., 2003. Using S / P amplitude ratios to constrain the focal mechanisms of small earthquakes, *Bulletin of the Seismological Society of America*, **93**(6), 2434–2444.

Hudson, J. A., Pearce, R. G., & Rogers, R. M., 1989. Source type plot for inversion of the moment tensor, *Journal of Geophysical Research*, **94**(B1), 765–774.

Hudson, T. S., White, R. S., Greenfield, T., Agustsdottir, T., Brisbourne, A., & Green, R. G., 2017. Deep crustal melt plumbing of Bardarbunga volcano, Iceland, *Geophysical Research Letters*, **44**, 3710–3718, doi:10.1002/2017GL074749.

Lomax, A., Virieux, J., Volant, P., & Berge, C., 2000. *Probabilistic earthquake location in 3D and layered models: Introduction of a Metropolis-Gibbs method and comparison with linear locations*, pp. 101–134, Advances in Seismic Location, Kluwer.

Lomax, A., Michelini, A., & Curtis, A., 2009. *Earthquake Location, Direct, Global-Search Methods*, pp. 2449–2473, Encyclopedia of Complexity and System Science, Part 5, Springer.

Mildon, Z. K., Pugh, D. J., Tarasewicz, J., White, R. S., & Brandsdóttir, B., 2016. Closing crack earthquakes within the Krafla caldera, North Iceland, *Geophysical Journal International*, **207**, 1137–1141.

Pugh, D. J., 2015. *Bayesian Source Inversion of Microseismic Events*, Ph.D. thesis, University of Cambridge.

Pugh, D. J., White, R. S., & Christie, P. A. F., 2016a. Automatic Bayesian polarity determination, *Geophysical Journal International*, **206**(1), 275–291.

Pugh, D. J., White, R. S., & Christie, P. A. F., 2016b. A Bayesian method for microseismic source inversion, *Geophysical Journal International*, **206**(2), 1009–1038.

Reasenberg, P. A. & Oppenheimer, D., 1985. FPFIT, FPPLOT and FPPAGE: Fortran computer programs for calculating and displaying earthquake fault-plane solutions - OFR 85-739, Tech. rep., USGS.

Riedesel, M. A. & Jordan, T. H., 1989. Display and assessment of seismic moment tensors, *Bulletin of the Seismological Society of America*, **79**(1), 85–100.

Schuler, J., Pugh, D. J., Hauksson, E., White, R. S., Stock, J. M., & Brandsdóttir, B., 2016. Focal mechanisms and size distribution of earthquakes beneath the Krafla central volcano, NE Iceland, *Journal of Geophysical Research: Solid Earth*, **121**(7), 5152–5168.

Smith, E. C., Baird, A., Kendall, J. M., Martin, C., White, R. S., Brisbourne, A. M., & Smith, A. M., 2017. Ice Fabric in an Antarctic ice stream interpreted from seismic anisotropy, *Geophysical Research Letters*, **44**, 3710–3718, doi:10.1002/2016GL072093.

Snoke, J. A., 2003. FOCMEC: FOCal MEChanism determinations, Tech. rep., Virginia Tech.

Tape, W. & Tape, C., 2012. A geometric setting for moment tensors, *Geophysical Journal International*, **190**(1), 476–498.

Wilks, M., Bradford, I., Williams, M., Rodriguez, I. V., & Pugh, D., 2015. Combined use of a deep monitoring array and shallow borehole arrays for moment tensor inversion: Th N108 09, in *77th EAGE Conference and Exhibition*.

Figure Captions

Figure 1. Flow diagram outlining the main steps in the moment tensor inversion package.

Figure 2. Krafla example results from the script at https://github.com/djpugh/MTfit/tree/master/examples/SRL_examples/krafla.py (run with 1,000,000 samples). The first plot shows the station distribution of observed receivers on the focal sphere, all with negative polarity, determined from the NonLinLoc estimate of the location PDF. The lighter points correspond to more likely receiver locations, and the maximum likelihood station locations with observed polarities are shown as triangles. The second plot shows the fault plane distribution for the double-couple constrained solution, with darker fault planes more likely. The last plot shows the Hudson type plot of the marginalized source-type PDF from the full moment tensor solution, with dark regions corresponding to low-probability source-types and lighter areas to higher probability types.

Figure 3. Marginalised posterior parameter distribution histogram for the five parameters described in Tape & Tape (2012) for the event shown in Fig. 2. γ and δ describe the moment tensor pattern, while κ (strike angle), h (cosine of dip) and σ (rake) describe the orientation. All parameters are dimensionless except κ and σ, which are in radians. This shows that the distributions are well constrained for the δ component, but are less well constrained for the fault plane orientation and γ components.

Figure 4. Lower hemisphere equal area projections and Hudson plots of the source PDF for a synthetic double-couple source for a range of data uncertainties, corresponding to SNR = infinity, SNR = 10, SNR = 7, SNR = 5, SNR = 3 and SNR = 2. The first and third columns

show the source PDF for the solution constrained to be double-couple only. The second and fourth columns show the source PDF for the full moment tensor solution. The first two columns show the solutions for inversions using only polarity data, and the second two columns show the solutions using polarity and amplitude ratio data. Manually picked station first motions are given by upward red or downward blue triangles. For the focal sphere plots, possible fault planes are given by dark lines. The most likely fault planes are given by the darkest lines. For the Hudson plots, high probability is red and low probability is in blue.

Figure 5. Elapsed time on a single core computer for different sample sizes of the random sampling (left plot) and for the McMC algorithms with different chain lengths (right plot) for a double couple source with no uncertainties in the input data. The red dots in the McMC case correspond to the trans-dimensional McMC algorithm and the blue dots correspond to the standard algorithm.

Figure 6. Elapsed time on a single core computer for different sample sizes of the random sampling algorithm (left plot) and for the McMC algorithms with different chain lengths (right plot) for a double couple source which includes location and velocity model uncertainties. The red dots in the McMC case correspond to the trans-dimensional McMC algorithm and the blue dots correspond to the standard algorithm. The velocity model and location uncertainty in the source was included with a one degree binning, reducing the number of location samples from 50,000 to 5,463.

Figure 7. MTplot examples showing (a) an equal area projection of a beachball for an example moment tensor source, (b) fault plane distribution showing the mean orientation in green, (c)

Hudson and (d) lune type plots of a full moment tensor PDF, and (e) a Riedesel-Jordan type plot of an example moment tensor source.