pynoddy 1.0: An Experimental Platform for Automated 3D Kinematic and Potential Field Modelling

J. Florian Wellmann[1,2], Sam T. Thiele[3], Mark D. Lindsay[3], and Mark W. Jessell[3]

[1]RWTH Aachen University, Graduate School AICES, Schinkelstr. 2, 52062 Aachen, Germany
[2]ABC/J Geoverbund, www.geoverbund.org
[3]The University of Western Australia, Centre for Exploration Targeting, 35 Stirling Hwy, 6009 Crawley, Australia

Correspondence to: J. Florian Wellmann (wellmann@aices.rwth-aachen.de or florian.wellmann@graduate.uwa.edu.au)

Abstract.

We present a novel methodology for performing experiments with subsurface structural models using a set of flexible and extensible Python modules. We utilise the ability of kinematic modelling techniques to describe major deformational, tectonic, and magmatic events at low computational cost to develop experiments testing the interactions between multiple kinematic events, effect of uncertainty regarding event timing, and kinematic properties. These tests are simple to implement and perform, as they are automated within the Python scripting language, allowing the encapsulation of entire kinematic experiments within high-level class definitions and fully reproducible results. In addition, we provide a link to geophysical potential-field simulations to evaluate the effect of parameter uncertainties on maps of gravity and magnetics.

We provide relevant fundamental information on kinematic modelling and our implementation, and showcase the application of our novel methods to investigate the interaction of multiple tectonic events on a pre-defined stratigraphy, the effect of changing kinematic parameters on simulated geophysical potential-fields, and the distribution of uncertain areas in a full 3-D kinematic model, based on estimated uncertainties in kinematic input parameters. Additional possibilities for linking kinematic modelling to subsequent process simulations are discussed, as well as additional aspects of future research. Our modules are freely available on github, including documentation and tutorial examples, and we encourage the contribution to this project.

1 Introduction

A wide range of methods exists for the computational synthesis of geological models as interpretations about the structure of the subsurface(see, for example, Jessell et al., 2014, for a recent overview of methods). Each modelling method focusses on different aspects of geological data and concepts,

1

but they can be broadly classified in terms of: (1) surface- or volume-based interpolation techniques, (2) pure geophysical inversions, and (3) mechanical or kinematic modelling approaches. We present here a set of open-source Python modules for the efficient, flexible and reproducible construction of kinematic structural models to enable the analysis of uncertainties in geological models.

Structural geological models are generally produced by combining information from direct observations (e.g. measurements in outcrops or boreholes) and indirect data, for example interpreted from geophysical data. Additional aspects of the conceptual geological model or the structural setting are, in the general case, only indirectly taken into account. Computational methods, which are able to capture several or all of the previous considerations, are then used to produce the model.

Regardless of the approach taken, the resulting models always contain uncertainties. These uncertainties are increasingly recognised (Bond et al., 2007; Caers, 2011; Bond, 2015) and addressed with novel methods for uncertainty analysis and visualisation (e.g. Bistacchi et al., 2008; Suzuki et al., 2008; Jessell et al., 2010; Polson and Curtis, 2010; Wellmann et al., 2010; Lindsay et al., 2012; Cherpeau et al., 2012; Lindsay et al., 2013; Laurent et al., 2015). So far, the analysis of uncertainties in kinematic models has been performed for balanced cross sections (e.g. Judge and Allmendinger, 2011) and detailed fault displacement models (Laurent et al., 2013). We contribute here with a methods to analyse and visualise uncertainties in automatically constructed kinematic forward models.

To enable this functionality, we extend the capability of an existing kinematic modelling method, implemented in the software Noddy (Jessell, 1981; Jessell and Valenta, 1996), with a flexible set of dedicated scripting modules developed in the programming language Python. Our aim is to provide high-level access to the underlying model construction methods, enabling: (1) flexible and rapid construction of kinematic models; (2) the definition of fully reproducible modelling experiments, and (3) a framework for automatic model generation, to enable experiments and analyses that require the generation of multiple models, like sensitivity evaluations or Monte Carlo uncertainty analyses (Metropolis and Ulam, 1949).

In the following, we will first describe the concepts of kinematic modelling as implemented in Noddy, outline the limitations of this method, and show how we address these with the newly developed Python modules. We then apply these new methods to several typical modelling scenarios: (1) the construction of a structural geological model on the basis of kinematic considerations, (2) an analysis of the effect of model uncertainty on calculated gravity fields, and (3) a sensitivity study of kinematic parameters in a complex kinematic model of the Gippsland Basin, Australia.

The Python code described here is open-source and freely available online (see Appendix A). All of the examples used in this text are also part of the online repository, and available as executable IPython notebooks.

2 Materials and Methods

Because we extend the functionality of an existing kinematic modelling package, Noddy (Jessell, 1981; Jessell and Valenta, 1996), we briefly describe its functionality here, and then provide de-
60   tails about the implementation of the Python package we have developed, referred to hereafter as pynoddy. Finally, in order to describe the main difference between our approach and other commonly used structural interpolation methods, we also briefly review the relevant approaches in this direction.

2.1 Structural geological modelling concepts

65   Structural geological models can be constructed with different approaches, and the choice of a specific modelling method directly depends on the model applications and the available input information.

The approach that we apply here is based on kinematic modelling concepts. The distinction between interpolation and kinematic methods is most apparent when considering the types of data
70   and geological constraints that are honoured. The most common approach to construct structural models is based on surface and volume interpolation methods (Mallet, 1992; Lajaunie et al., 1997; Sprague et al., 2006; Caumon et al., 2009; Hillier et al., 2014; Jessell et al., 2014). An example of the general interpolation function is presented in Fig. 1a. Structural interpolations focus on honouring parameterised surface contact points (Caumon et al., 2009), although secondary data like orientation
75   measurements can also be taken into account (Lajaunie et al., 1997; Calcagno et al., 2008; Hillier et al., 2014). Constraints on the shape of geological surfaces, or the interaction with other units or faults, are then assigned to different surfaces, according to observations in the field or the expected geological settings. While these considerations are clearly based on geological reasoning, it is not guaranteed that an interpolated structural model matches all the known aspects of the geological
80   evolution of an area. For example, it is easily possible that constraints on thickness of geological units are not consistent, for example across a fault, leading to a violation of mass conservation. Additionally, a wide range of structures observed in multiply deformed terranes, such as complex fault networks or refolded folds, are difficult to construct consistently using current interpolation methods.

Another end member in the evaluation of the structural setting are simulations of physical pro-
85   cesses (e.g. Gerya and Yuen, 2007; Moresi et al., 2007; Kaus et al., 2008; Regenauer-Lieb et al., 2013). Instead of starting with geological observations, these methods are based on mathematical models capturing relevant physical processes that led to the formation of specific structures (Fig. 1b). For realistic simulations, meaningful constitutive models and boundary conditions are required. Multiple different methods exist which capture different aspects of the mechanical deformation, and
90   more and more commonly also the effect of coupled Thermo-, Hydro-, Mechanical- and Chemical-simulations. However, these types of simulations are not yet commonly applied to model the entire
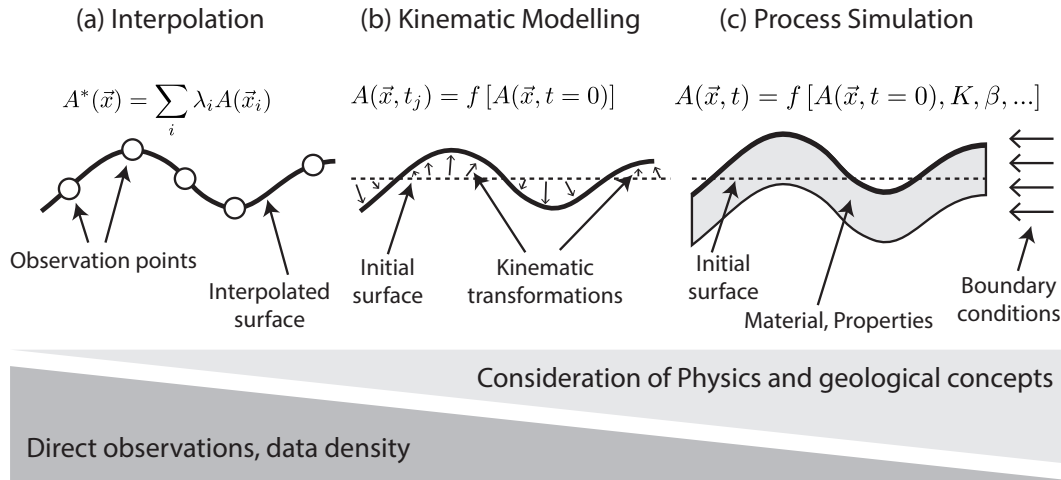
3

(a) Interpolation

$$A^*(\vec{x}) = \sum_i \lambda_i A(\vec{x}_i)$$

Observation points
Interpolated surface

(b) Kinematic Modelling

$$A(\vec{x}, t_j) = f\left[A(\vec{x}, t=0)\right]$$

Initial surface
Kinematic transformations

(c) Process Simulation

$$A(\vec{x}, t) = f\left[A(\vec{x}, t=0), K, \beta, ...\right]$$

Initial surface
Material, Properties
Boundary conditions

Consideration of Physics and geological concepts

Direct observations, data density

Figure 1. Conceptual difference of modelling approaches: (a) interpolation, (b) dynamic process simulations, (c) kinematic models (modified from Jessell and Valenta, 1996)

complexity of multiply deformed geological regions as simulations are computationally demanding and rock properties, and boundary conditions are not always perfectly known. Furthermore, they require an initial distribution of rock properties in space as initial conditions, often determined from an explicit or implicit interpolation approach.

Kinematic modelling methods focus on major tectonic and metamorphic events in geological history (Jessell and Valenta, 1996) and are therefore conceptually located between the previously described end members (Fig. 1c). In these models, the complexity of deformation is greatly reduced and captured in simplified kinematic functions as surrogate models. This means that direct geological observations of surface contacts and orientation measurements are not taken into account in the simulation step. However, the simulations are very fast and enable therefore a quick testing of different deformational scenarios, and the interaction of multiple events in geological history. Furthermore, rapid simulation makes direct (and ideally automated) comparisons between the model and observed structures feasible, allowing the indirect incorporation of geological observations. We will present several examples in which this trade-off between physical realism and geological observations can lead to useful insights into the interaction and relevance of deformational events in geological history.

2.2 Kinematic structural modelling with Noddy

Noddy models begin as a layer cake stratigraphy, for which the heights of the stratigraphic contacts and geophysical rock properties are defined. A history of relevant events that affected the model region is then developed from a predefined set of events, including: folds, faults, and shear zones; unconformities, dykes and igneous plugs; regional tilting and homogeneous strain. In addition to

4

modifying the initial stratigraphy, each event can define (geophysical) alteration halos, penetrative cleavages and lineations.

Each Noddy event is defined by four classes of properties: form, position, orientation, and scale (Jessell, 1981; Jessell and Valenta, 1996). For example, a fault is defined by its dip and dip direction, the pitch and magnitude of the slip vector, and the position of one point on its surface (note that more complex definitions of the fault plane are also possible, c.f. Jessell, 1981). The use of geological descriptions provides a natural and intuitive framework for geologists to build a model. Even though the structural events themselves are relatively simple, complex geometries quickly develop as two or three events are superimposed on one another (see examples in Sec. 3).

Displacement equations are stored as a "history", which provides parameterised definitions of the model kinematics and rock properties. A voxel model of any 3D rectangular volume of interest can be calculated from this history by considering each voxel independently using the Eulerian (inverse) form of the defining Lagrangian displacement equations, and applying them in reverse chronological order (i.e. starting with the most recent deformation event). This operation transforms the x, y, and z position of each voxel into the x, y, z position at the time the associated volume of rock was created. The properties of this voxel can then be calculated directly from the base stratigraphy.

New lithologies can also be created during three specific event types: unconformities, dykes, and plugs. These events are assumed to be instantaneous, and are ordered relative to other events. In order to simplify the underlying kinematic equations, they are all defined in a standard reference frame that is orthogonal to the symmetry of the deformation event. The real world reference frame is rotated into the standard reference frame prior to the calculation of each event, and then subsequently rotated back to the real world reference frame using the variations in the z-values as a continuous implicit field that can be iso-surfaced to produce stratigraphic horizons.

As well as the initial position of the point, a binary "discontinuity code" is stored, that records each time a voxel is affected by an event described by a discontinuous displacement equation (faults, unconformities, dykes, and plugs) but ignores events described by continuous displacement equations (folds, shear zones, strain, rotation, foliations and lineations). This discontinuity code allows the accurate transformation of the voxel data set to a vector data set, since only voxels which have exactly the same sequence of discontinuity codes are part of the same contiguous volume of rock. If two adjacent voxels have different codes, the difference in the discontinuity code that occurred most recently defines the discontinuity which separates them.

The orientations of specific features (bedding, foliation, fault planes, remanence vectors, etc.) are calculated using both the inverse and then the forward displacement equations. Starting with the current 3D location of a point, the position of this point at the time of formation of the structural feature (which may or may not be the time of formation of the rock) is calculated. Three points are defined close to this position which define a plane with the orientation of the feature prior to

5

deformation. The positions of these three points at the final time are then calculated, from which the

150 final orientation of the structural feature can be calculated.

Similarly, the orientation of a linear feature is calculated from the intersection of two planes. Thus both the Eulerian (inverse) and Lagrangian (forward) descriptions of the displacements must be available for a new deformation event to be included in the modelling scheme. For this reason, the displacement equations governing each Noddy deformation event are kept as simple as possible, and

155 superimposed deformation events are combined to produce structural complexity. A full description of the Noddy implementation is presented in Jessell and Valenta (1996).

2.3 Geophysical Potential-field Modelling with Noddy

2.3.1 Basic concept

Noddy has the capability to calculate potential field responses of gravity and magnetics based on the

160 spatial distribution of rock properties in the subsurface, and this functionality is exposed in pynoddy. The petrophysical rock properties of a specific volume are defined by their original stratigraphic value, unless a specific deformation event (faults, unconformities, plugs and dykes) has an associated alteration/metamorphic character, with allows the modification or replacement of pre-existing properties based on that locations distance from the structural feature at the time of the activity of

165 the event. A further complication is possible if a model with an anisotropic magnetic susceptibility (a tensor property) or magnetic remanence (a vector property) is defined, in which case there is the possibility of calculating the voxel-level reorientation of these properties as a result of deformation (for example having a remanence vector deformed during a folding event).

For all surveys the rock property of a cube is defined as the value at the centre of the cube, and

170 for grid surveys (that is, not arbitrary surveys or borehole surveys) the field strength is calculated at the x,y location above the centre of each cube. The Total Magnetic Intensity value calculated for all schemes is actually the value projected onto the Earth's field, following the convention of many modelling schemes. The gravity field calculated is for the z component only.

Three geophysical computational schemes are available in Noddy to calculate magnetic and grav-

175 ity potential fields. The criteria as to which scheme should be used depends on required accuracy, speed and the various geological situations being modelled. A brief description of each scheme is provided below.

2.3.2 Spatial convolution scheme

The spatial convolution scheme works by calculating the summed response of all the cubes within

180 a cylinder centred on the sensor, with a radius defined by the spatial range term. The calculation for each cube is based on the analytical solution for a dipping prism presented by Hjelt (1972) and Hjelt (1974). In order to calculate solutions near the edge of a block, extra geology is used to

produce a padding zone around the block equal in width to the spatial range, so that there are no edge effects in this scheme. The scheme only provides exact solutions when the range is larger than the length of the model. For reasonably complex geology this limitation does not result in inaccurate models, however for idealised geometries using a range that is too small results in a kink in resultant profiles. The spatial convolution scheme is slower than the Spectral scheme for medium ranges (10-20 cube ranges), but generally much faster than the Full Spatial Calculation. As long as the range is greater than the spacing between high density/susceptibility features, the inaccuracies associated with truncating the calculation is probably not evident. The draped survey and down-hole surveys have not been implemented for this scheme.

2.3.3 Spectral scheme

This scheme, based on pioneering work by Parker (1972) works by transforming the rock property distributions into the Fourier domain, applying a transformed convolution, and then transforming this result back into the Spatial Domain. The calculation is performed for each horizontal slice through the geology, and the results are summed vertically. The Spectral scheme produces a different result than the other two schemes in terms of absolute numbers for three reasons:

1. The Fourier transform implies that the geology is infinitely repeating outside the calculation area. This produces edge effects when high susceptibility or density bodies are found near the edges of the survey area. This effect can be lessened by the choice of a suitable padding around the block, including over specified areas of interest, however it cannot be totally removed.

2. The calculation loses the absolute base line of the gravity or magnetic field, so even when comparisons are made for well-padded Spectral and large range Spatial models, an overall offset is apparent between the two schemes. When trying to model real data this offset is not a problem as any regional is removed before the modelling process.

3. There is a high frequency component to the calculated field that is of the same wavelength as the cube size and especially apparent when there are steep gradients in the values of the rock properties.

2.3.4 Full spatial scheme

This is similar to the Spatial Convolution scheme except that all the cubes in the model are summed using the Hjelt schemes in order to calculate the response at any point. It generally takes significantly longer to apply this calculation scheme than either of the other schemes. The only exception is when there is a relatively sparse geological model, in which case contiguous blocks with identical petrophysical properties are aggregated to form rectangular blocks, which reduces computation time. In the extreme case where only one cube has non-zero values for both density and susceptibility, any

cubes which have both zero density and susceptibility are ignored. This is the only scheme that can accurately calculate draped surveys, down-hole surveys and arbitrarily located airborne surveys.

2.4 Creating input files for kinematic modelling with Noddy

Noddy histories are stored as ASCII files with a simple keyword-value ordering. These files can be written or adapted with any text editor, and the kinematic modelling result computed with a compiled command line version of the program and results visualised with other software.

A graphical user interface (GUI) has previously been created to simplify this model setup, combining convenient input file generation directly with computation and visualisation of the results. This GUI is freely available (http://tinyurl.com/noddy-site), though currently only runs on Windows operating systems. The GUI version of Noddy is also limited to user-driven workflows, restricting further automation or extension of the methods for scientific experiments.

In order to overcome the problem of either having to work with a direct text input file, or being restricted by the limitations of a GUI, we have developed flexible modules in the programming language Python that enable scripted access the kinematic modelling functionality and to enable the extension to uncertainty estimations.

2.5 Implementation of pynoddy

Python is an object-oriented scripting language that is widely used in scientific computation (e.g. Langtangen, 2008). It is highly flexible language, and contains a variety of programming and visualisation libraries ideal for scientific purposes. Python also runs on virtually every operating system that is available, meaning that python wrappers retain the platform independence of C applications.

The pynoddy module described here contains a set of classes and functions for managing Noddy input files, passing them to the Noddy command-line application, and processing the results. This approach has many advantages, as it allows automatic generation and analysis of kinematic models in a Python environment, while retaining the performance of Noddy itself (which is written in C).

2.5.1 Overall module structure

The package pynoddy contains three main modules: pynoddy.history, pynoddy.output and pynoddy.experiment. The pynoddy.history and pynoddy.output modules provide interfaces for managing Noddy inputs and outputs, while classes defined in pynoddy.experiment provides methods for implementing and performing repeatable modelling experiments. The output of Noddy simulations can be processed and analysed with classes in pynoddy.output, and exported in VTK file formats for 3-D visualisation with VTK viewers.

The relationship between these main modules and the command-line application Noddy is presented in Fig. 2. More details on the implementation and detailed visualisations of the class structure reflecting the current module state are given in the documentation (see Appendix B).
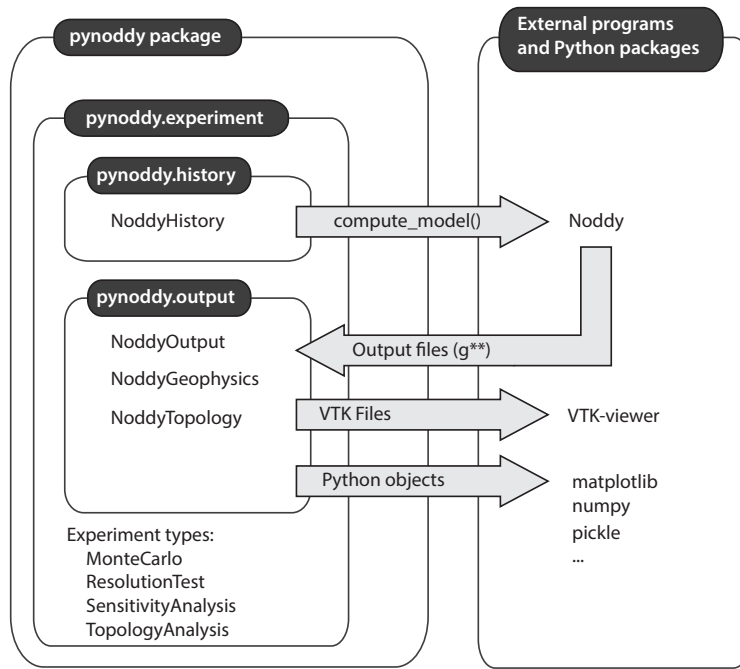
8

Figure 2. High-level structure of main pynoddy modules and relationship to command-line application Noddy and other python packages. Important to note is the concept of high-level classes, defined in the module pynoddy.experiments, to encapsulate history file and output methods.

2.5.2 Noddy histories

250

The NoddyHistory class (defined in the module pynoddy.history) contains methods for generating, opening and manipulating Noddy history files. NoddyHistory instances can be created by: (1) loading existing Noddy history files (including those created using the Noddy GUI), or; (2) programmatically defining an event sequence and all the associated properties. The Noddy events encapsulated by a NoddyHistory instance can easily be modified or reordered, and simulation properties such as voxel size or geophysical properties adjusted. Once a NoddyHistory instance contains the desired properties, it can be written as a Noddy history file (.his) and passed to the Noddy application for processing.

255

2.5.3 Noddy output

260

Noddy writes the results of a model (defined by a .his file) as a series of output files, described individually in Table 1. The NoddyOutput class, (defined in the pynoddy.output module) contains methods for reading, analysing and visualising these outputs, and can be used to create visual representations of sections through the model, or to export a computed model as a 3-D grid to the VTK format for further analysis and visualisation using, for example, the open-source packages Paraview (http://www.paraview.org) or Visit (http://visit.llnl.gov).

265

9

Table 1. Descriptions of the output files produced by the command line version of Noddy. Files for calculated potential fields are calculated when Noddy is called in Geophysics mode (see Sec. 2.3).

File extension	Contents	Details
.g00	Model header file	Information on the dimensions of the model (voxel size etc.), lithology names and associated geophysical properties.
.g01	Density	Spatial distribution of final density in each voxel
.g02	Susceptibility	Spatial distribution of final magnetic susceptibility in each voxel
.g12	Lithology model	Contains the lithology ID of each voxel in the model.
.grv	Gravity field	2-D field data of bouger gravity calculated from Noddy model
.mag	Magnetic field	2-D field data of total magnetic intensity calculated from Noddy model.

2.5.4 Experiments combining Noddy input and output

If all steps of a pynoddy experiment are automated properly, they can be integrated into one script for model set-up and analysis. This method is leading to a possible reproduction of results (as an example: see the scripts that generate the figures in this manuscript, see appendix B for availability).

270 This method is often used successfully to ensure reproducibility. It does, however, have one significant drawback: intermediate results or adapted simulation settings have to be stored in separate files and all of those files have to be available to continue with an experiment at a given state.

In order to overcome this limitation, we follow here the aim of including an entire experiment, from the definition of input parameter of the model, to parameters that are specific to an experiment,

275 to the post-processing of results, within a single Python object. Specific experiments can then be defined as child classes inheriting a set of useful base methods. This object can then be stored (for example with a serialisation using the Python pickle package) and retrieved exactly in the state that it was used and defined for a complete reproduction of results, or the adaptation of model parameters to test different model outputs.

280 The core of the pynoddy.experiment module is the Experiment class, which inherits methods from both the NoddyHistory and NoddyOutput classes, combining and extending their functionality into a single interface that allows a flexible modelling procedure were the Noddy computations are automatically executed when required and outputs directly updated. In addition, methods are provided to encapsulate relevant parameters of an experiment in the most efficient and flexible way.

285 We consider this last point essential to ensure a full reproducibility of scientific experiments with kinematic models.

In order to generate a specific type of experiment, new child classes can then be defined, inheriting from the Experiment base class. Several classes for specific types of experiments are already implemented in the pynoddy package, and we show below the application of one such child class, the UncertaintyAnalysis, applied to a Monte Carlo error propagation experiment.

For more details on the implementation and the structure of the modules in pynoddy, please see the documentation and associated source code at the pynoddy GitHub directory (see appendix **??** and B).

3 Applications

This section outlines the functionality and utility of our pynoddy implementation using a variety of case studies. Firstly, the structural effect of multiple faulting events is investigated, serving mainly as an introduction to the generation of event histories in pynoddy and the visualisation of results. Then, a model from the Atlas of Structural Geophysics is used to evaluate the sensitivities of calculated gravity potential-field values to changes of parameters in kinematic events. Finally, we use the pynoddy framework to evaluate uncertainties in a case study of the Gippsland Basin, Australia.

3.1 Analysis of fault interactions

We start here with an example that is conceptually simple, but can quickly lead to complex structural settings: the interaction of a sequence of fault events on a predefined stratigraphy (Fig. 3a). A more detailed description and interactive version is available as an IPython notebook as part of the repository and as supplementary online material for this manuscript (see Appendix **??** and B).

This model is constructed from a stratigraphic sequence containing five units, each 1000 m thick. We consider a model domain of 10,000 x 7,000 x 5,000 m in x, y, and z-directions. In the following descriptions, we define points with respect to an origin in the model at the top, SW corner (i.e.: the point (0,0,-1000) is at a depth of 1000 m at the SW corner). A representation of the model in a (x,z)-section is given in Fig. 3a.

The second event in the model is a fault that affects the eastern part of the model. We define the fault at the top of the model at position (2000, 3500, 0) dipping 60 → 090 and a fault slip of 1,000 m. The effect of this fault on the previous stratigraphic pile is visualised in Fig. 3b. The third event is also a fault, defined with a surface at position (8000, 3500, 0), dipping 60 → 270 and a slip of 1,000 m (Fig. 3c).

In terms of this definition of kinematic equations, the two fault events are symmetrical. However, the combination of both events leads, as can be expected, to a non-symmetrical interaction pattern, here clearly visible in the central part of the model (Fig. 3d).
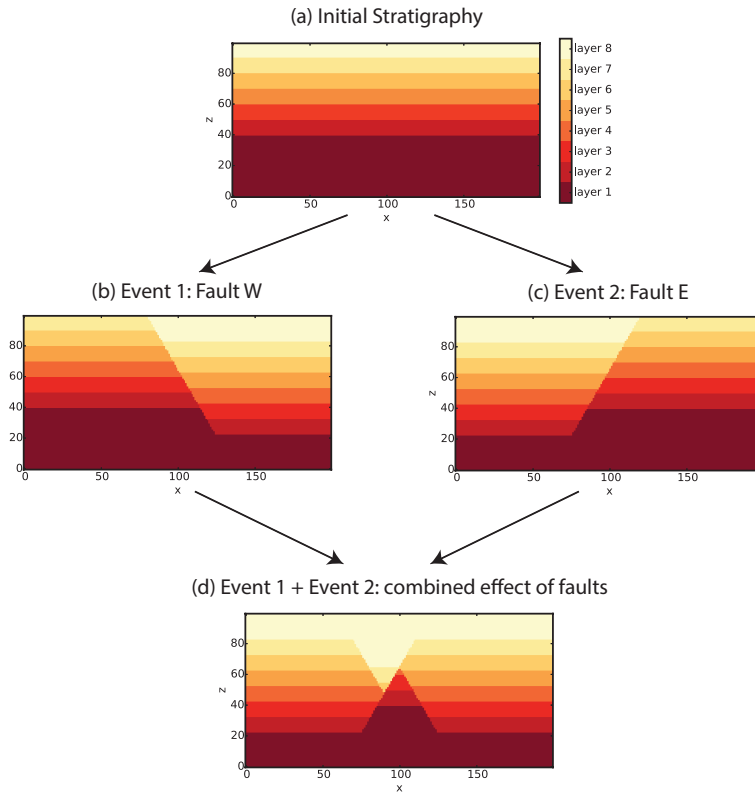
Figure 3. Development of a fault network model with pynoddy: (a) initial stratigraphic pile, (b) effect of the first fault only, (c) effect of the second fault only, and (d) combined effect of both faults.

The previous example is included to present the possibilities for the simple construction of a kinematic model from start. The model itself is mostly interesting from an instruction or teaching perspective and we will move to more complex models in the following.

3.2 Potential field modelling and the Atlas of Structural Geophysics

One motivation for the development of Noddy was to provide a method to explain and teach the effect of subsequent geological events, as we presented an example above. The capability of Noddy to calculate geophysical fields can furthermore be used to provide insights for the interpretation of geophysical potential field data. We can, for example, quickly evaluate how changing the properties of a geological event (for example the dip angle of a fault) influences a simulated potential field.

In fact, this capability of Noddy has been a main driver to develop the "Atlas of Structural Geophysics", an online collection of geological models with their simulated corresponding potential fields for a wide variety of typical structural geological settings (http://tectonique.net/asg).

We provide in pynoddy the functionality to directly load models from this atlas into python objects, for further testing and manipulation. In addition, the pynoddy.output module also contains a class definition to read in the calculated potential field responses (NoddyGeophysics). In combina-

12

tion, these methods enable us to quickly test the effect of different event properties on calculated potential fields.

As an example, we evaluate here how changing properties of deformational events affects the forward calculated gravity field with a model of a fold and thrust belt (Fig. 4). The required commands to download a model from the web page, to adjust cube size (for better representation), to write it to a file, and to run the model, are combined in a tutorial notebook for detailed reference (see Appendix B). The 3-D visualisation in Fig. 3c was generated through the pynoddy export to VTK and visualised in Paraview (see Sec. 2.5.3).
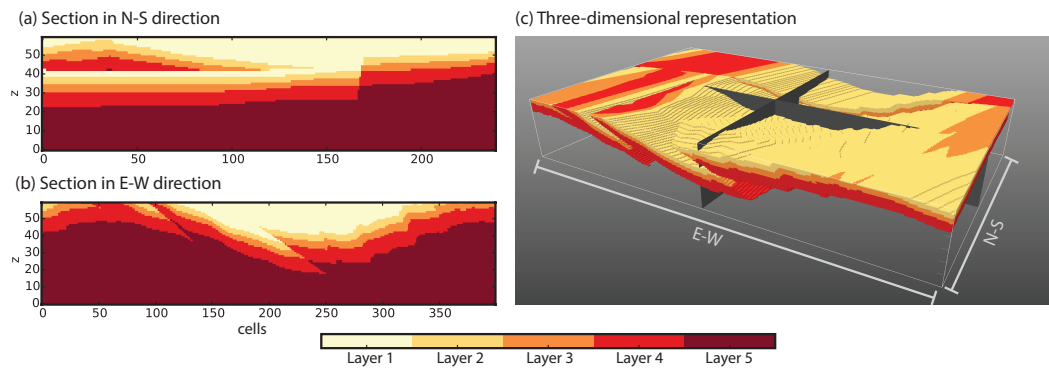

Figure 4. Sections through the fold and thrust belt model in (a) NS-direction, and (b) EW-direction (vertical exaggeration of 1.5) through the centre of the model. (c) Three-dimensional representation for the central three layers of the fold and thrust belt model. The grey surfaces correspond to the location of the sections in the figure above.

We calculate the gravity field for this model with the spectral scheme (Sec. 2.3.3) by calling pynoddy.compute in the geophysics simulation mode. The resulting z-component of the gravity field is visualised in Fig. 5a.

As a next step, we evaluate how the effect of a different wavelength in the folding event, as the latest event in the model history, affects the calculated gravity field. This adaptation, as well as the recalculation and visualisation of the geophysical field (Fig. 5b), can be performed with a few lines of Python code (see tutorial notebook for details). In addition, we use simple Python commands to calculate and visualise the difference between the gravity fields of the original and the changed model (Fig. 5c).

With the previous examples, we showed the application of pynoddy to perform simple kinematical modelling experiments. These types of experiments could also be performed with the already existing GUI of Noddy, or even on the basis of the ASCII input files, only. The use of pynoddy does, however, provide a simple and direct way to adjust models, and to directly perform additional calculations (e.g. for the difference of the gravity fields), and to generate high-quality visualisations with additional Python tools.
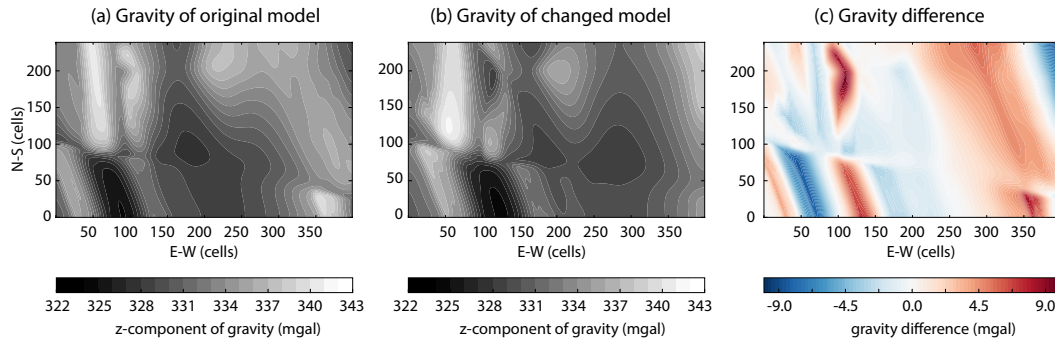
13

Figure 5. Evaluation of the effect of a changing the wavelength in a late folding event on the forward calculated gravity field: (a) gravity field of original model, (b) gravity field of model with changed event parameters, and (c) difference plot of gravity fields.

With the following example, we now want to highlight an essential advantage of our new implementation in pynoddy: the high-level definition of scientific experiments with kinematic models.

3.3 Reproducible Experiments with pynoddy

360 One main motivation for the definition of a python package to access the functionality of kinematic modelling is the increased level of flexibility that it offers when performing scientific studies with kinematic models. Specifically, we can automate the entire model construction processes and can hence easily perform multiple simulations with different parameter settings. This possibility enables a whole new range of applications, from simple scenario testing (as shown above), to the analysis

365 of model uncertainties due to the propagation of errors in input parameter and model settings. In this sense, pynoddy is ideally suited to perform scientific experiments on the basis of kinematic modelling concepts.

If all steps of a pynoddy experiment are automated properly, they can be integrated into one script for model set-up and analysis. If implemented properly, this method enables a complete reproduction

370 of results. As described in section 2.5.4, we provide a high-level object-oriented method for classes of full kinematic experiments, combining Noddy input and output, automatic computation when required, and the additional integration of further methods from external Python packages.

In the following example, we show how we use the pynoddy.experiment methods to investigate error propagation with a Monte Carlo experiment for a complex geological model of the Gippsland

375 Basin. The tectonic history input to Noddy is shown in Fig. 6a. This simplified, but representative geological history has been primarily derived from Rahmanian et al. (1990), Norvik and Smith (2001), Moore and Wong (2002) and Lindsay et al. (2012). Each event shown in Fig. 6a corresponds to an event interpreted from the Gippsland Basin, a Mesozoic to Cenozoic oil and gas field in southeastern Australia (Cook, 2006; Rahmanian et al., 1990). Our model basement is Ordovician rocks and the

14

380   cover sequences include the Oligocene Seaspray and Pliocene Angler sequences. Of particular inter-
est for oil and gas prospectivity is the Paleocene to Late Miocene Latrobe Group, which includes the
Cobia, Golden Beach and Emperor Subgroups (Bernecker et al., 2001). The basin is cross-cut by a
number of transfer and normal faults, however we only model the most pervasive fault sets for this
example. These include the NNE to NE-trending Lucas Point Fault, Spinnaker Fault and Cape Ever-

385   ard Fault System, and the E-W trending Wron Wron/Rosedale Fault Systems. Some large-scale (10s
km wavelength) folding is observed, however the basin retains an overall layer-cake stratigraphy.
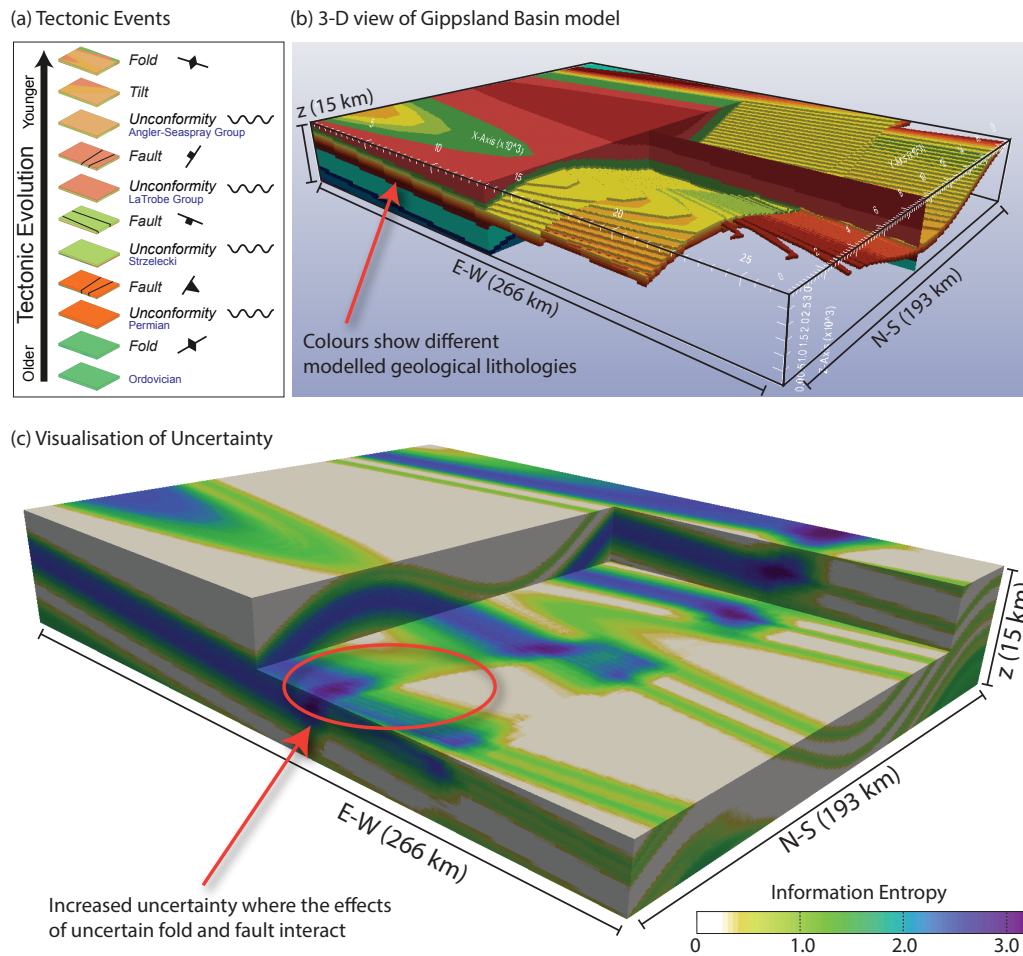

Figure 6. (a) Tectonic events in the kinematic model. Symbols indicate main orientation of events, stratigraphic
units in blue font; (b) 3-D visualisation of simulated block model (transparency for better visualisation of
internal fold), colours indicate geological lithologies; (c) Visualisation of uncertainty with information entropy,
clearly visible are high uncertainties where effects of uncertain fold and fault interact.

We now want to evaluate how uncertainties in the kinematic parameters of the different tectonic
events (Fig. 6a) propagate to the final constructed model (Fig. 6b). The general procedure is briefly

15

outlined here, for more details please see the IPython notebook with the complete example and more
390   thorough descriptions (See documentation and tutorial, Sec. B).

We use here the class UncertaintyAnalysis, which contains methods for Monte Carlo-type error
propagation and subsequent uncertainty analyses. As a first step, we consider relevant kinematic
modelling parameters now as random variables, instead of deterministic variables. The properties of
these random variables can be described as probability distributions in several ways. We use here
395   a simple definition in a table, stored in a comma separated file, that can be loaded directly into the
object.

We assign normal distributions to location points and layer thicknesses, with a mean value accord-
ing to the prior mean, and a standard deviation of 100 m, to reflect the overall uncertainty in defining
representative thickness and location values on the large scale of the model. The wavelength of the
400   late folding event (Fig. 6a) has a mean of 15 km and we assign a standard deviation of 2.5 km,
assuming a high uncertainty in determining a wavelength for this event. Uncertainties in orientation
measures are defined with a von Mises distribution. We provide details on the parameter distributions
in a table in the appendix (C2).

With the parameters of the random variables stored in an external file, we can instantiate the
405   uncertainty analysis object with the history file of the kinematic model and the name of the parameter
file as arguments:

```
ua = UncertaintyAnalysis(history_file, params)
```

We can now directly generate n random samples from this model with:

```
ua.estimate_uncertainty(n)
```

410   The set of results is, by default, saved directly within the object, and can be extracted in the form of
Python numpy arrays for further processing. In addition, A set of standard post-processing methods
and utility functions is already implemented in the class definition. For example, it is directly possible
to generate analyses and visualisations for the probability of outcomes for a specific geological
lithology per voxel (Wellmann et al., 2010; Lindsay et al., 2012), and for the analysis of voxel-based
415   information entropy measures (Wellmann and Regenauer-Lieb, 2012; Wellmann, 2013).

In this example of the Gippsland Basin, we perform Monte Carlo error propagation for a set of 32
parameters of all kinematic events in the model, and generate 100 random realisations of the model
(see tutorial notebook in documentation). For post-processing, we analyse and visualise results in
a 3-D plot of cell information entropies (Fig. 6c). The estimated uncertain areas in the model are
420   clearly visible, and highest uncertainties exist in areas where the effect of uncertainties in different
events overlaps (see Fig. 6c).

The previous experiment is a typical example of Monte Carlo sampling methods (Metropolis and
Ulam, 1949). One characteristic of the sampling is that all realisations are drawn independently.
Therefore, a parallel implementation of the sampling is directly possible. As one possibility, we pro-

425   vide a parallel sampling scheme implemented in the pynoddy.experiment.monte_carlo.MonteCarlo class, based on the Python threading module, and we used this scheme successfully on a supercomputer. For more information on this possibility, see documentation (Appendix B) and the source code of the monte_carlo.py module.

4   Discussion

430   We have presented a newly developed python module for performing scientific experiments with kinematic models, and provided examples of possible applications for investigating the interaction of tectonic events, assessing the effect of kinematic parameters on simulated geophysical potential fields, and identifying uncertainty within 3D geological models. These examples would not have been possible without the methodology that pynoddy provides for defining, modifying and realising

435   kinematic models in a scripting environment. Our developments therefore provide opportunities for performing scientific experiments with kinematic models that have not been possible before.

    One aspect of the developed code is that entire experiments with kinematic models can be encapsulated in one class definition. We demonstrated this encapsulation with the third example (Sec. 3.3), performing complex analyses within a single python class, and hence allowing full reproducibility.

440   This encapsulation has multiple further advantages, including a simple, but still flexible, way to test effects of uncertainties in kinematic parameters and the direct inclusion of post-processing and analysis methods methods, as shown with the analysis of information entropy (Fig. 6c), to ensure consistency between experiments and subsequent analyses. Several experiment classes in addition to the presented Monte Carlo method are pre-defined, including, for example, methods for local and

445   global sensitivity analysis. The definition of custom classes on the basis of this framework is straightforward. In essence, the combination of input and output generation with on-demand computation allows a high flexibility, as well as an integration of essential aspects of entire kinematic experiments in a single object. As the random state is stored, this encapsulation facilitates easy reproduction of entire scientific experiments with kinematic models.

450   Limitations of modelling approach presented here are related to (a) the specific kinematic functions available in Noddy, (b) the conceptual simplification of representing complex dynamical evolutions with purely kinematic functions in general (see Fig. 1), and (c) the limited consideration of surface contact information and measurements.

    The first limitation is based on the fact that we did not extend the basic functionality of the kine-

455   matic equations implemented in Noddy, and these equations may not be complex enough for specific modelling requirements. For example, the fold model in Noddy is based on a simple fold concept, and this may be a limitation when other fold mechanisms need to be modelled. For a full definition of possibilities and limitations, please see Jessell (1981) and Jessell and Valenta (1996). However, as presented in this manuscript (Fig. 6), in the examples of the Atlas of Structural Geophysics (Sec. A6),

17

460 or even in recent publications (Armit et al., 2012), complex models can easily evolve from the inter-action of multiple kinematic events.

The second limitation, that is the use of kinematic equations instead of a full dynamic simulation, is a significant conceptual simplification and has to be kept in mind when constructing and inter-preting results of kinematic modelling, to ensure that the methods are used in the scope where they

465 are valid. With the examples presented in this manuscript, we wanted to highlight such applications; in addition to the instructive aspect of using kinematic models to teach and visualise the effect of interacting deformational and magmatic events, we believe that main advantages come from the po-tential to automatically generate multiple model realisations. These methods are facilitated by the fact that the generation of a single kinematic model is typically very fast (in the order of seconds to

470 minutes on a single core) compared to full dynamic simulations. This possibility therefore enables investigation of interaction between simplified deformational events, but with the consideration of uncertainties in event parameters, orders, and types.

The final limitation is that kinematic modelling only allows indirect consideration of actual ob-servations and measurements in the models. An encouraging avenue of investigation is the inclusion

475 of observations facilitated by combining kinematic modelling with interpolation methods (Fig. 1a). We note at this point the similarity between the kinematic modelling methods described in our work, and object modelling methods in geostatistics (Pyrcz and Deutsch, 2014), which are widely and successfully use in reservoir modelling. We envisage that experience from applications of these ob-ject modelling methods can be transferred to kinematic modelling concepts based on the flexible

480 methods presented in this work.

The methods we have implemented are platform independent, as they are completely implemented in Python, and Noddy itself in C. It is therefore possible to port developed experiments and code easily to other computational environments. We have, for example, tested numerical experiments on supercomputers, a possibility that is especially important for the generation of multiple (i.e. thou-

485 sands or more) high-resolution model realisations, or the combination with complex post-processing methods. In addition, the platform independence circumvents a limitation of the current GUI for Noddy which is restricted to one operating system. One of the main motivations for the original development of Noddy, for use a teaching tool, is therefore also ensured.

Geological modelling is most often not an end in itself, but the input to further modelling and

490 simulation methods. For example, structural geological models are often used as an input for sub-sequent flow simulation studies, or for wave propagation experiments. This combination is directly possible with our developed methods, as the distribution and properties of lithological units in space are stored in numpy arrays, that can easily be exported to other modelling methods in Python or similar frameworks. One example would be using the generated models as input for property distri-

495 bution in hydrothermal experiments with the widely used flow simulation code TOUGH2, through the use of the Python package PyTOUGH, https://github.com/acroucher/PyTOUGH (see Wellmann

18

et al., 2011), or to the generation of synthetic seismic sections and simulations of wave propagation with Madagascar, www.ahay.org

Future extensions of the developed code will include an optimised application in parallel environments, including a better storage of results (e.g. in HDF5 formats), and a better link to geological data sets and parameters (e.g. through the use of GeoSciML, see Sen and Duffy, 2005; Simons et al., 2006). In addition, we are actively working on developments of additional experiment classes, for example for detailed topological analyses of structural models, and further post-processing and uncertainty quantification methods. Another path of future research is to investigate the possibility to integrate kinematic modelling with Noddy into inference frameworks, for example to test the possible inversion of kinematic parameters from observations and geophysical measurements. We hope to include functionality developed by other external users into the main package, and encourage an active participation with successfully developed extensions.

5 Code Availability

The information provided here is relfecting the current state of the repository at the time of manuscript preparation. In case you find information outdated, please contact the corresponding author.

– pynoddy is free open-source software. For detailed information on the license, see the agreement in the LICENSE file of the repository.

– pynoddy is free open-source software. For detailed information on the license, see the agreement in the LICENSE file of the repository.

– Documentation is available as part of the package and online, currently hosted on http://pynoddy.readthedocs.org/

Appendix A: pynoddy package information

A1   Notes on installation

A successful installation of pynoddy requires two steps:

1. An installation of the python modules in the package pynoddy;

2. The existance of an executable Noddy(.exe) program.

530   Currently, pynoddy and Noddy can be installed in two alternative ways: (a) directly from the source code with the full repository, or (b) with a direct installation from the Python Package Index and pre-compiled executables. We suggest to use option (a) for the most recent and most complete version of the code. Version (b) is suggested for less experienced users who would like to quickly test and apply kinematic modelling methods. We describe the installation the alternatives in the

535 following.

    Note: for clarity, we denote command line prompts with a > symbol below:

```
> command to be executed
```

A2   Installation of pynoddy

A2.1   Installing pynoddy from the github repository

540 As a first step, we suggest to clone the current repository to your local machine. This step can be done with a github front-end, or simply with the usual git command in a terminal:

```
> git clone https://github.com/flohorovicic/pynoddy
```

    Note: if you do not have a running version of git installed, then you can also simply download the entire repository as a zip file from the github page. However, you then do not have the full flexibility

545 of the entire repository, and therefore we recommend using git.

    Once the repository is cloned (or downloaded), simply change to the main directory of pynoddy and install the Python package with the installation script:

```
> python setup.py install
```

    Note that this command adds pynoddy to your global Python installation. If you plan to develop

550 parts of pynoddy further yourself, then installation in development mode is suggested:

```
> python setup.py develop
```

    In this mode, modifications in the cloned repository are directly considered when importing the modules in your Python scripts.

A2.2   Installation of pynoddy from Python Package Index

555 pynoddy is hosted on the Python Package Index (https://pypi.python.org/pypi/pynoddy/) and the typical methods can be used to install the Python packages.

    If pip is installed on your system, then the most straight-forward installation is directly though executing in a terminal:

```
> pip install pynoddy
```

560   Alternatively, the package source can be downloaded from the index page, as well as an installation program for Windows systems.

Please note that the Python package on the index is not always the newest version, but in a state that reflects the latest stable developments. For the most current state, we suggest an installation from the repository (Sec. A2.1).

A3 Installation of the Noddy command line program

A3.1 Using a pre-compiled version of Noddy

The easy way to obtain a executable version of Noddy is simply to download the appropriate version for your operating system. Currently, these executables versions are also stored on github (check the up-to-date online documentation if this should not anymore be the case) in the directory:

https://github.com/flohorovicic/pynoddy/tree/master/noddyapp

Furthermore, the executables for Windows are also available for download on the webpage:

http://www.tectonique.net/pynoddy

Download the appropriate app, rename it to noddy or noddy.exe and place it into a folder that is in your local environment path variable. If you are not sure if a folder is in the PATH or would like to add new one, see Sec. A3.3.

A3.2 Compiling Noddy from source files (recommended)

The source code for the executable Noddy is located in the repository directory noddy. In order to perform the installation, a gcc compiler is required. This compiler should be available on Linux and MacOSX operating systems. On Windows, one possibility is to install MinGW. Otherwise, the code requires no specific libraries.

Note for MacOSX users: some header files have to be adapted to avoid conflicts with local libraries. The required adaptations are executed when running the script:

```
> adjust_for_MacOSX.sh
```

The compilation is then performed (in a Linux, MacOSX, or Windows MinGW terminal) with the command:

```
> compile.sh
```

Compilation usually produces multiple warnings, but should otherwise proceed successfully.

The repository is in a state of active further development. We identified the current state of the repository at the time of manuscript submission with a git tag to ensure consistency of examples and descriptions presented in this manuscript.

A3.3 Placing Noddy in the Path

For the most general installation, the executable of Noddy should be placed in a folder that can be located from any terminal application in the system. This (usually) means that the folder with the

21

executable has to be in the `PATH` environment variable. On Linux and MacOSX, a path can simply be added by:

```
> export PATH="path/to/executable/:$PATH"
```

Note that this command should be placed into your .bash_profile file to ensure that the path is added whenever you start a new Python script.

On Windows, adding a folder to the local environment variable Path is usually done through the System Control Panel (Start - Settings - Control Panel - System). in Advanced mode, open the Environment Variables sub-menu, and find the variable Path. Click to edit the variable, and add the location of your folder to this path.

A3.4 Specifying path during **pynoddy** execution

Another option is to tell pynoddy.compute_model the exact path to the Noddy executable:

```
pynoddy.compute_model(history, output_name,
                      noddy_path = 'path/to/program')
```

However, this method should only be used as the fall-back option if adding the executeable to a path (Sec. A3.3) does not work. Also, in this case, the tests (Sec. A4) will most likely fail.

A4   Testing the installation

A4.1   Testing **Noddy**

Simply test the installation by running the generated (or downloaded) executable in a terminal window (on Windows: cmd):

```
> noddy
```

or (depending on your compilation or naming convention):

```
> noddy.exe
```

Which should produce the general output:

```
Arguments <historyfile> <outputfile> <calc_mode>:
BLOCK
GEOPHYSICS
SURFACES
BLOCK_GEOPHYS
BLOCK_SURFACES
TOPOLOGY
ANOM_FROM_BLOCK
ALL
```

Note: if the executable is correctly placed in a folder which is recognised by the (Environment) path variable, then you should be able to run Noddy from any directory. If this is not the case, please see Sec. A3.3.

A4.2 Testing **pynoddy**

630 The pynoddy package contains a set of tests which can be executed in the standard Python testing environment. If you cloned or downloaded the repository, then these tests can directly be performed through the setup script:

```
> python setup.py test
```

Of specific relevance is the test that determines if the noddy(.exe) executable is correctly accessi-
635 ble from pynoddy. If this is the case, then the compute_model test should return:

```
test_compute_model (test.TestHistory) ... ok
```

If this test is not ok, then please check carefully the installation of the noddy(.exe) executable (see either A3.1 or A3.2).

If all tests are successful, you are ready to go!

640 ## A5 Noddy executable and GUI

The original graphical user interface for Noddy and the compiled executable program for Windows can be obtained from http://tinyurl.com/noddy-site. This site also contains the source code, as well as extensive documentation and tutorial material concerning the original implementation of the software, as well as more technical details on the modelling method itself.

645 ## A6 Atlas of Structural Geophysics

The Atlas of Structural Geophysics contains a collection of structural models, together with their expression as geophysical potential fields (gravity and magnetics), with a focus on guiding the interpretation of observed features in potential-field maps.

The atlas is currently available on: **http://tectonique.net/asg**. The structural models are created
650 with Noddy and the history files can be downloaded from the atlas. Together with the Python package pynoddy, which is presented in this manuscript, these models can easily be adjusted and recomputed to reflect different settings, as shown in the example in section 3.2.

Appendix B: Documentation

An up-to-date documentation is available as part of the pynoddy repository, including all source files,
655 a compiled LATEX pdf version (in docs/_build/latex), and a version in html (in docs/_build/html).

In addition, the documentation is hosted on the readthedocs webpage for quick online reference on: http://pynoddy.readthedocs.org/.

The most convenient way to get started with pynoddy is to experiment with the interactive IPython notebooks, for example to reproduce and adapt the examples given in this manuscript. These notebooks are a part of the repository. The only requirement is to have a running Jupyter installation, see http://jupyter.org for more information. We furthermore plan to have these interactive notebooks available for web-based experiments with pynoddy in the future.

Appendix C: Additional information on Models and Results in this Publication

C1 Example models in this manuscript

All example models presented in this manuscript, respectively the python and pynoddy code to generate them, are available as part of the repository. The experiments are directly accessible as Jupyter notebooks to re-generate the presented experiments, or to test different parameters (in docs/notebooks).

C2 Gippsland Basin uncertainty study

The Gippsland Basin model was inspired by previous work of the authors in this region (Lindsay et al., 2012), and further references to the geological setting can be found there. For the purpose of this work, the kinematic parameters for the geological events, as well as the probability distributions consideration of these parameters as random variables, are given in table 2.

Event	Parameter	Distribution Type	Mean	Shape Parameter	Event Name
2	Amplitude	Normal	500	100	Early fold
2	Wavelength	Normal	15000	2500	Early fold
2	X	Normal	0	500	Early fold
2	Z	Normal	0	500	Early fold
3	Z	Normal	250	100	Permian Seds
4|11	Dip	von Mises	70	10	Cape Howe Fault
4|11	Dip Direction	von Mises	270	5	Cape Howe Fault
4|11	X	Normal	23000	100	Cape Howe Fault
4|11	Z	Normal	5000	100	Cape Howe Fault
4	Slip	Normal	-100	100	Cape Howe Fault
5|10	Dip	von Mises	70	10	Cape Everard Fault
5|10	Dip Direction	von Mises	286	5	Cape Everard Fault
5|10	X	Normal	18000	100	Cape Everard Fault
5|10	Y	Normal	0	100	Cape Everard Fault
5|10	Z	Normal	5000	100	Cape Everard Fault
5	Slip	Normal	-100	100	Cape Everard Fault
6	Z	Normal	750	100	Strzelecki Seds
7	Dip	von Mises	70	10	Lake Wellington Thrust
7	Dip Direction	von Mises	180	5	Lake Wellington
7	Y	Normal	13000	100	Lake Wellington
7	Z	Normal	5000	100	Lake Wellington
7	Slip	Normal	500	100	Lake Wellington
8	Dip	von Mises	45	10	Foster Thrust
8	Dip Direction	von Mises	10	5	Foster Thrust
8	X	Normal	8730	100	Foster Thrust
8	Y	Normal	0	100	Foster Thrust
8	Z	Normal	5000	100	Foster Thrust
8	Slip	Normal	500	100	Foster Thrust
9	Z	Normal	750	100	La Trobe Seds
10	Slip	Normal	200	100	Cape Everard
11	Slip	Normal	200	100	Cape Howe
12	Z	Normal	1000	100	Angler-Seaspray seds

Table 2. Distributions and parameters for Gippsland Basin study

References

675   Armit, R. J., Betts, P. G., Schaefer, B. F., and Ailleres, L.: Constraints on long-lived Mesoproterozoic and Palaeozoic deformational events and crustal architecture in the northern Mount Painter Province, Australia, Gondwana Research, 22, 207–226, 2012.

Bernecker, T., Woollands, M., Wong, D., Moore, D., and Smith, M.: Hydrocarbon prospectivity of the deep water Gippsland Basin, Victoria, Australia, APPEA Journal, 41, 91–113, 2001.

680   Bistacchi, A., Massironi, M., Dal Piaz, V. G., Monopoli, B., Schiavo, A., and Toffolon, G.: 3D fold and fault reconstruction with an uncertainty model: An example from an Alpine tunnel case study, Computers & Geosciences, 34, 351–372, 2008.

Bond, C. E.: Uncertainty in structural interpretation: Lessons to be learnt, Journal of Structural Geology, 74, 185 – 200, 2015.

685   Bond, E. C., Shipton, K. Z., Jones, R. R., Butler, W. R., and Gibbs, D. A.: Knowledge transfer in a digital world: Field data acquisition, uncertainty, visualization, and data management, Geosphere, 3, 568, 2007.

Caers, J.: Modeling Uncertainty in the Earth Sciences, John Wiley & Sons, Ltd, Chichester, UK, 2011.

Calcagno, P., Chiles, J.-P., Courrioux, G., and Guillen, A.: Geological modelling from field data and geological knowledge: Part I. Modelling method coupling 3D potential-field interpolation and geological rules: Recent
690   Advances in Computational Geodynamics: Theory, Numerics and Applications, Physics of the Earth and Planetary Interiors, 171, 147–157, 2008.

Caumon, G., Collon-Drouaillet, P., Le Carlier de Veslud, C., Viseur, S., and Sausse, J.: Surface-Based 3D Modeling of Geological Structures, Mathematical Geosciences, 41, 927–945, 2009.

Cherpeau, N., Caumon, G., Caers, J., and Levy, B.: Method for Stochastic Inverse Modeling of Fault Geometry
695   and Connectivity Using Flow Data, Mathematical Geosciences, 44, 147–168, 2012.

Cook, P. J.: Carbon dioxide capture and geological storage: research, development and application in Australia, International Journal of Environmental Studies, 63, 731 – 749, 2006.

Gerya, T. V. and Yuen, D. A.: Robust characteristics method for modelling multiphase visco-elasto-plastic thermo-mechanical problems, Physics of the Earth and Planetary Interiors, 163, 83 – 105, computational
700   Challenges in the Earth Sciences, 2007.

Hillier, M. J., Schetselaar, E. M., de Kemp, E. A., and Perron, G.: Three-Dimensional Modelling of Geological Surfaces Using Generalized Interpolation with Radial Basis Functions, Mathematical Geosciences, pp. 1–23, 2014.

Hjelt, S.-E.: Magnetostatic anomalies of dipping prisms, Geoexploration, 10, 239–254, 1972.
705   Hjelt, S.-E.: The gravity anomaly of a dipping prism, Geoexploration, 12, 29–39, 1974.

Jessell, M.: "Noddy" - An interactive Map creation Package, Master's thesis, Imperial College of Science and Technology, London, 1981.

Jessell, M., Ailleres, L., de Kemp, E., Lindsay, M., Wellmann, J. F., Hillier, M., Laurent, G., Carmichael, T., and Martin, R.: Next Generation Three-Dimensional Geologic Modeling and Inversion, Economic Geology,
710   pp. 261–272, 2014.

Jessell, M. W. and Valenta, R. K.: Structural geophysics: Integrated structural and geophysical modelling, in: Structural Geology and Personal Computers, edited by De Paor, D. G., pp. 303–324, Pergamon, 1996.

Jessell, W. M., Ailleres, L., and Kemp, A. E.: Towards an Integrated Inversion of Geoscientific data: what price of Geology?, Tectonophysics, 490, 294–306, 2010.

715 Judge, P. A. and Allmendinger, R. W.: Assessing uncertainties in balanced cross sections, Journal of Structural Geology, 33, 458–467, 2011.

Kaus, B. J., Gerya, T. V., and Schmid, D. W.: Recent advances in computational geodynamics: Theory, numerics and applications, Physics of the Earth and Planetary Interiors, 171, 2 – 6, recent Advances in Computational Geodynamics: Theory, Numerics and Applications, 2008.

720 Lajaunie, C., Courrioux, G., and Manuel, L.: Foliation fields and 3D cartography in geology: Principles of a method based on potential interpolation, Mathematical Geology, 29, 571–584, 1997.

Langtangen, P. H.: Python scripting for computational science, Springer Verlag, New York, 2008.

Laurent, G., Caumon, G., Bouziat, A., and Jessell, M.: A parametric method to model 3D displacements around faults with volumetric vector fields, Tectonophysics, 590, 83–93, 2013.

725 Laurent, G., Caumon, G., and Jessell, M.: Interactive editing of 3D geological structures and tectonic history sketching via a rigid element method, Computers & Geosciences, 74, 71–86, 2015.

Lindsay, M., Ailleres, L., Jessell, M., de Kemp, E., and Betts, P. G.: Locating and quantifying geological uncertainty in three-dimensional models: Analysis of the Gippsland Basin, southeastern Australia, Tectonophysics, pp. 1–44, 2012.

730 Lindsay, M. D., Perrouty, S., Jessell, M. W., and Ailleres, L.: Making the link between geological and geophysical uncertainty: geodiversity in the Ashanti Greenstone Belt, Geophysical Journal International, 195, 903–922, 2013.

Mallet, J.-L.: Discrete smooth interpolation in geometric modelling, Computer-Aided Design, 24, 178–191, 1992.

735 Metropolis, N. and Ulam, S.: The Monte Carlo Method, Journal of the American Statistical Association, 44, 335–341, 1949.

Moore, D. and Wong, D.: Eastern and Central Gippsland Basin, Southeast Australia; Basement Interpretation and Basin Links, Report, Department of Natural Resources and Environment, 2002.

Moresi, L., Quenette, S., Lemiale, V., Mériaux, C., Appelbe, B., and Mühlhaus, H.-B.: Computational ap-
740 proaches to studying non-linear dynamics of the crust and mantle, Physics of the Earth and Planetary Interiors, 163, 69 – 82, computational Challenges in the Earth Sciences, 2007.

Norvik, M. and Smith, M.: Mapping the plate tectonic reconstruction of southern and southeastern Australia and implications for petroleum systems, APPEA Journal, 41, 15–35, 2001.

Parker, R.: The rapid calculation of potential anomalies, Geophys. JR Astron. Soc, 31, 447–455, 1972.

745 Polson, D. and Curtis, A.: Dynamics of uncertainty in geological interpretation, Journal of the Geological Society, 167, 5–10, 2010.

Pyrcz, M. J. and Deutsch, C. V.: Geostatistical reservoir modeling, Oxford university press, 2014.

Rahmanian, V. D., Moore, P. S., Mudge, W. J., and Spring, D. E.: Sequence stratigraphy and the habitat of hydrocarbons, Gippsland Basin, Australia, Geological Society, London, Special Publications, 50, 525–544,
750 1990.

Regenauer-Lieb, K., Veveakis, M., Poulet, T., Wellmann, F., Karrech, A., Liu, J., Hauser, J., Schrank, C., Gaede, O., and Trefry, M.: Multiscale coupling and multiphysics approaches in earth sciences: Theory, Journal of Coupled Systems and Multiscale Dynamics, 1, 49–73, 2013.

Sen, M. and Duffy, T.: GeoSciML: development of a generic geoscience markup language, Computers & Geosciences, 31, 1095–1103, 2005.

Simons, B., Boisvert, E., Brodaric, B., Cox, S., Duffy, T. R., Johnson, B. R., Laxton, J. L., and Richard, S.: GeoSciML: enabling the exchange of geological map data, ASEG Extended Abstracts, 2006, 1–4, 2006.

Sprague, K., Kemp, E., Wong, W., McGaughey, J., Perron, G., and Barrie, T.: Spatial targeting using queries in a 3-D GIS environment with application to mineral exploration, Computers & Geosciences, 32, 396–418, 2006.

Suzuki, S., Caumon, G., and Caers, J.: Dynamic data integration for structural modeling: model screening approach using a distance-based model parameterization, Computational Geosciences, 12, 105–119, 2008.

Wellmann, J. F.: Information Theory for Correlation Analysis and Estimation of Uncertainty Reduction in Maps and Models, Entropy, 15, 1464–1485, 2013.

Wellmann, J. F. and Regenauer-Lieb, K.: Uncertainties have a meaning: Information entropy as a quality measure for 3-D geological models, Tectonophysics, 526-529, 207–216, 2012.

Wellmann, J. F., Horowitz, F. G., Schill, E., and Regenauer-Lieb, K.: Towards incorporating uncertainty of structural data in 3D geological inversion, Tectonophysics, 490, 141–151, 2010.

Wellmann, J. F., Croucher, A., and Regenauer-Lieb, K.: Python scripting libraries for subsurface fluid and heat flow simulations with TOUGH2 and SHEMAT, Computers & Geosciences, 43, 197–206, 2011.

28