https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

Geoscientific Model Development
Discussions

DFN Generator v2.0: A new tool to model the growth of large-scale natural fracture networks using fundamental geomechanics

Michael J. Welch[1], Mikael Lüthje[1], Simon J Oldfield[1]

[1]Danish Hydrocarbon Research and Technology Centre, Danish Technical University, Kgs. Lyngby, 2800, Denmark

Correspondence to: Michael J. Welch (mwelch@dtu.dk)

Abstract. In this paper we present a new code to build geologically realistic models of natural fracture networks in rock formations, by simulating fracture nucleation, growth and interaction, guided by fundamental geomechanical controls and the local geological history. This code implements the fracture modelling algorithm described by Welch et al. (2020), developed to generate more accurate, better constrained models of large fracture networks than current stochastic techniques. It efficiently builds fracture modesl as either implicit (structured grid) and explicit (discrete fracture network) representations, across large (km-scale) geological structures such as folds, major faults or salt diapirs. It will thus have applications in engineering and fluid flow modelling, including CO_2 sequestration and geothermal energy, as well as in understanding the controls on the evolution of fracture networks.

The code is written in C Sharp and is provided with two interfaces: a standalone interface with text file input and output, that can be compiled in standard C Sharp to run simple models, and a plug-in interface for Schlumberger's Petrel geomodelling package, that can run more complex models of real geological structures. The standalone version has been used to run extensive sensitivity analyses, which studied the influence of various mechanical and physical parameters (e.g. layer thickness, applied strain, Young's Modulus, etc.) on the fracture evolution and geometry, by varying the parameters individually in simple models. The Petrel plug-in has been used to evaluate the code applicability by running simulations of actual fractured layers in outcrops and in the subsurface, and comparing the results with observed fracture patterns.

1 Introduction

This paper presents a new code that simulates the processes of fracture nucleation, growth and interaction in rock to build geologically realistic models of natural fracture networks. The derivation of the algorithm used to do this has been previously published in Welch et al. 2020, so only a brief outline will be presented here. The purpose of this paper is to describe the structure of the code, and explain in detail how it implements the algorithm described in Welch et al. (2020).

The code has been used to run sensitivity analyses, which study the influence of various mechanical and physical parameters on the fracture evolution and geometry, and to run simulations of actual fractured layers in outcrops and in the subsurface, which can be used to evaluate the method by comparing the results with observed fracture patterns. These models are also

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

described in detail in previous publications (Welch et al. 2019, 2020) and are summarised here to support code verification and evaluation.

1.1 Motivation

Natural fracture networks in geological formations can be important controls on various physical properties and processes, such as mechanical strength, elastic stiffness, seismic response, and subsurface fluid flow. These fracture networks may be related to larger geological structures including folds, major faults and salt diapirs. An ability to predict and model such networks is therefore important in a wide range of applications ranging from engineering (e.g. mine or tunnel construction), geological hazard prediction, monitoring pollutant dispersion, geothermal energy, oil and gas production and CO_2 sequestration.

These fracture models may be presented in two styles (Fig. 1):

- Implicitly – representing fracture network properties (e.g. fracture density, or primary azimuth) in a laterally continuous grid.
- Explicitly – representing individual fractures geometric objects.

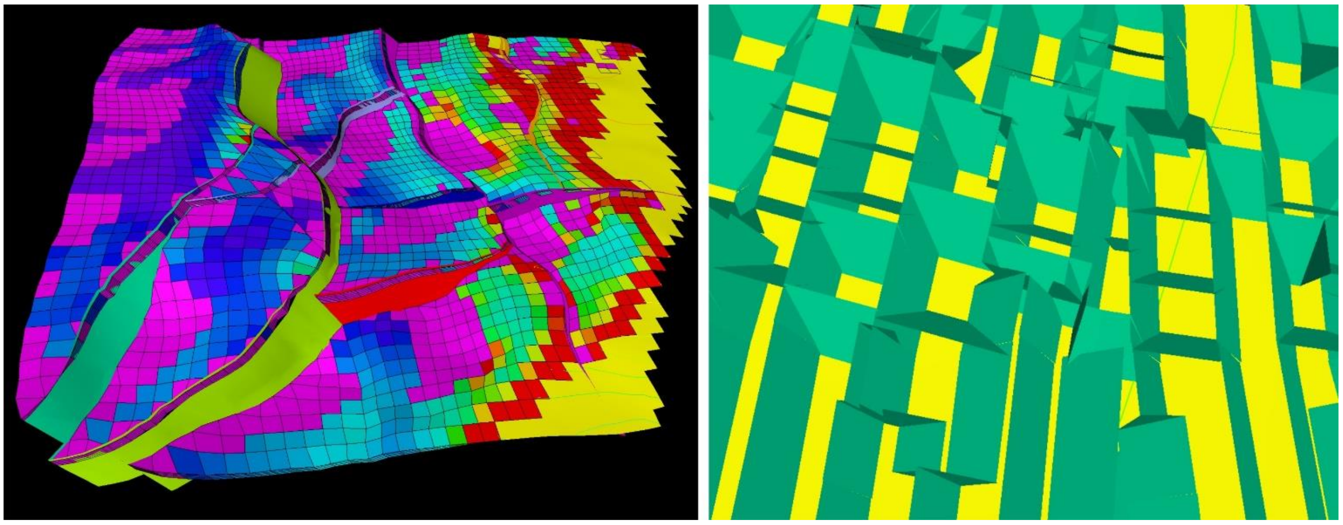


Figure 1: Comparison of an implicit fracture model (left) and an explicit DFN model (right). In the implicit fracture model, fracture density (defined as total fracture area per unit volume, in m^{-1}) is defined as a continuum property in a grid of discrete cells. In the explicit DFN, individual fractures are represented as geometric objects. While the fracture density can be calculated from the DFN, it also contains other valuable information such as mean fracture length, fracture orientation and connectivity, and the anisotropy of the overall fracture network.

Unfortunately it is not usually possible to map fractures directly, as they are generally below the resolution of geophysical imaging techniques. This problem is typically addressed using stochastic modelling of fracture distribution, placing fractures at random locations with arbitrary size, constrained to population distributions observed in boreholes. This takes no account of the geology or the geomechanical processes of fracture formation, and thus often results in inaccurate, poorly constrained and geologically unrealistic models.

To solve these problems, we developed a new method of modelling realistic models of fracture networks in a layer-bound setting, by simulating the processes of fracture nucleation, growth and interaction, based on geomechanical principles and an understanding of the geological history of the structure Welch et al. (2019, 2020). This algorithm combines linear elastic fracture mechanics theory (LEFM) (after Griffith 1921, Sneddon 1946, Sack 1946) with subcritical fracture propagation theory (after Atkinson 1984, Swanson 1984) to calculate the propagation rate of fractures in response to an applied strain (Fig. 2). Unlike conventional numerical modelling techniques (e.g. finite element methods), the new method can efficiently model large fracture networks covering major geological structures. It is capable of building explicit DFNs containing hundreds of thousands of fractures, but it can also build implicit fracture models directly, if the fracture network contains too many fractures to model explicitly (for example in the case of a thin fractured layer extending across a salt diapir).

Building a fracture model dynamically in this way provides a number of benefits over stochastic methods, including; additional insight into the likely constraints of the uncertainty range of fracturing and additional ease of comparison between observations and predictions by providing coupled implicit and explcit outputs. The resulting model will more accurately reflect the underlying geology, and will predict properties such as connected fracture volume and permeability anisotropy, that must be supplied as input properties in stochastic models. Furthermore, deterministic models are much quicker and easier to build than stochastic models, and it is much easier to model multiple realisations, providing input to uncertainty analysis considering both natural variation within a set of models and completely different interpretation scenarios based on geological constrainst such as fracture density and relationships.

1.2 Overview of the algorithm

The basis of the Welch et al (2020) algorithm is the coupling of linear elastic fracture mechanics theory (LEFM) with subcritical fracture propagation theory (SFP). LEFM describes the stress concentration around a fracture tip in a homogeneous elastic medium, for various simple fracture geometries (e.g. radially symmetrical fractures, or fractures of infinite length under plane strain conditions), while SFP gives an expression for the rate of propagation of a fracture tip as a function of the stress concentration at the fracture tip. Combining the two theories yields a differential equation for the growth of the fracture, which can be solved analytically to give the fracture size through time, if the in situ stress state and mechanical properties are known. This is described in Sections 3.1, 3.2 and 4.1 of Welch et al. (2020).

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

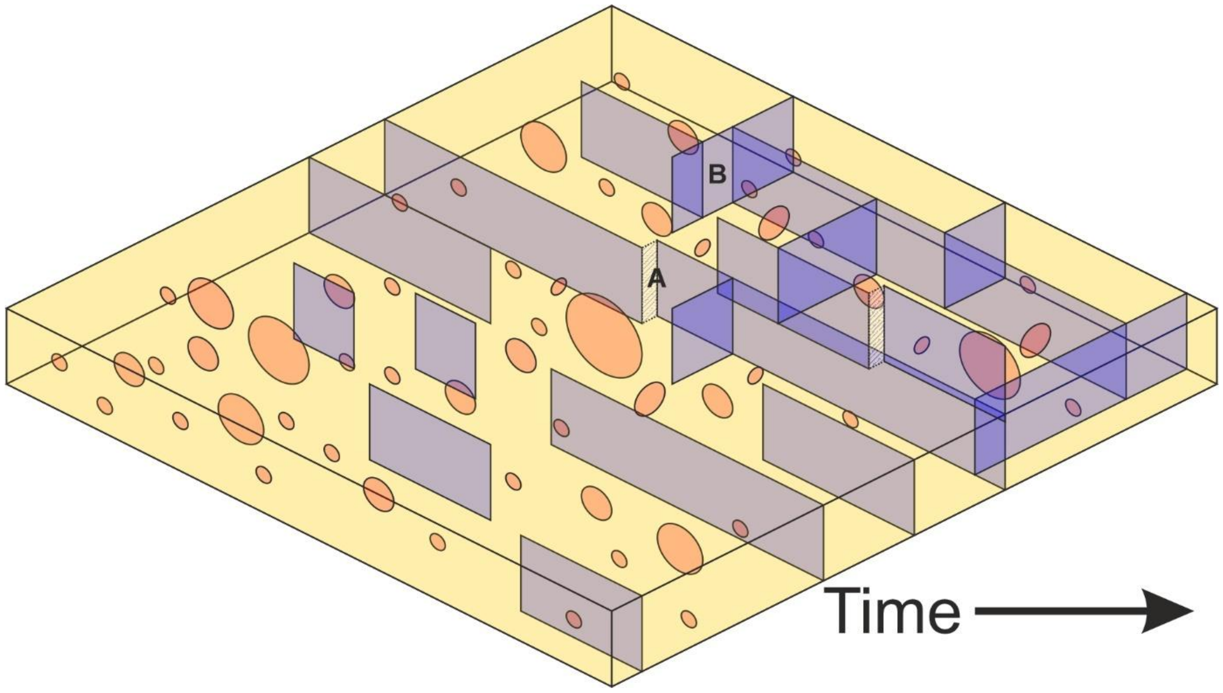


Figure 2: Conceptual illustration of the growth of a layer-bound fracture network. Initially, small circular microfractures develop within the layer (shown in red) and grow until they reach the upper and lower bounds of the layer, whereupon they become layer-bound macrofractures, propagating laterally (shown in blue). These continue to grow until they terminate, either because they propagate into the stress shadow of a parallel microfracture (A) or because they intersect a perpendicular or oblique microfracture (B). Reproduced with kind permission from Fig. 2.2 of Welch et al. (2020).

This analytical expression can model the growth of a fracture much more efficiently than the numerical methods that are more commonly used (e.g. finite element methods), as it does not need to calculate the complex local stress pattern that develops around the fracture. It is therefore possible to simulate the growth of very large fracture networks, containing hundreds of thousands of individual fractures, within a reasonable runtime. The drawback is that it can only model simplified fracture geometries – typically either circular fractures in a homogeneous medium (e.g. the red microfractures of Fig. 2) or layer-bound fractures with vertical extent much greater than their height (the blue macrofractures of Fig. 2). It is therefore not able to model more complex fracture geometries, such as splays linking initially independent fracture segments (as described in Nicol et al. 1995 and Walsh et al. 2003), and it cannot model fracture growth in a highly heterogeneous medium. However these problems can be circumvented: relay segments can be added to the fracture network post-simulation to represent linking splay fractures (see Section 2.2.3), and lateral heterogeneities in the mechanical properties or the in situ stress can be modelled by discretising the layer into polyhedral gridblocks and calculating fracture growth separately in each gridblock (see Section 2.1).

The real power of this technique is that, the expressions for fracture growth can be applied directly to the cumulative density distribution functions (CDDFs) instead of modelling how individual fractures grow, hence the fracture network can be described as an implicit fracture model. CDDFs describe the fracture density of a given size or greater. They are often approximated by power law functions, indicating a fractal distribution of fracture sizes (Westaway 1994). Coupling the derived expression for fracture growth rate with a power law function for the initial fracture density yields an expression for the evolution of the CDDFs through time; this is described in Sections 3.3 and 4.2 of Welch et al. (2020). The runtime required to calculate these evolving functions is dependent only on the number of cells in the spatial discretisation of the fractured layer, and the number of timesteps in the temporal discretisation, but not on the quantity of fractures in the layer. This enables simulation of the growth of fracture networks containing many millions of fractures within a reasonable runtime.

To simulate the growth of the circular microfractures, we must assume an initial "seed" population of microfractures. It is not necessary to seed the layer-bound macrofractures, as these develop from the microfractures when they reach a sufficient size to span the entire layer. Thus the density of macrofractures at any time can be calculated from the microfracture CDDF at that time, and the layer thickness; this is described in Section 4.2 of Welch et al. (2020).

Since the microfracture growth rate is highly non-linear with respect to time, the nucleation rate of the macrofractures will also vary considerably through time. This, in turn, controls the macrofracture size distribution. However the stress concentration at the propagating tips of the macrofractures is proportional to the macrofracture height, which is equal to the layer thickness (and therefore does not change). Therefore all macrofractures propagate at a constant rate, regardless of length. This vastly simplifies the calculation of the CCDs through time. Nevertheless, the macrofracture propagation rate may still vary through time if the in situ stress changes, so it is necessary to discretise the model temporally into a series of timesteps, and recalculate the macrofracture propagation rate at each timestep.

The fracture count per unit volume is known as the volumetric or P_{30} fracture density (where the subscript 3 refers to the dimensionality of the measurement space and the subscript 0 refers to the dimensionality of the measured quantity). The P_{32} and P_{33} fracture densities, describing the total fracture area or the total fracture volume per unit volume respectively, can be derived by differentiating the P_{30} CDDF with respect to fracture size, multiplying by the fracture area or volume, and re-integrating through the required range of fracture sizes. For microfractures, this integration must be carried out numerically (see Section 3.4 of Welch et al. 2020), but for macrofractures, it is possible to derive analytical expressions for both the P_{32} and P_{33} fracture densities, as described in Sections 4.3 and 4.4 of Welch et al. (2020). The P_{32} (linear fracture count per unit length) and P_{33} (fracture volume per unit rock volume) CDDFs are often used as indices for calculating the influence of fractures on permeability and bulk mechanical properties, due to their ease of comparison to linear transect counts (from outcrops or boreholes) and fracture porosity, respectively.

With continuing growth, fractures in a network will begin to interact, preventing further propagation. Two types of fracture interaction are included in this model: fracture intersection and stress shadow interaction. The rates of fracture interaction will depend on the existing fracture density – fracture interaction will become more frequent as the fracture network grows and the fracture density increases. This is another reason why it is necessary to discretise the model temporally into timesteps: we can thus recalculate the fracture interaction rates at each timestep. Since the two tips of the layer-bound macrofractures propagate and become deactivated independently of each other, we must also split each macrofracture into two half-macrofractures, propagating in opposite directions, and model the CDDFs for half-macrofractures rather than the full macrofractures (as such the half-macrofracture P_{30} density will be double the macrofracture P_{30} density, while the half-macrofracture P_{32} density will

be the same as the macrofracture P_{32} density). Expressions for the rates of fracture interaction are derived in Section 5.1 of Welch et al. 2020.
The fracture population can therefore be divided into active (still propagating) fractures and static (non-propagating) fractures. The CDDFs for the active fractures at any time can be calculated by multiplying the total fracture densities with the inverse of the fracture interaction rate (i.e. the probability that a given fracture will not intersect another fracture or propagate into a stress shadow zone within a certain time). Calculating the CDDFs for the static fractures is more difficult: it requires multiplying the active fracture density by the fracture interaction rates, to obtain an expression for the fracture deactivation rate through time; this function is then integrated from the start of deformation, to obtain the total density of deactivated fractures. The resulting expressions for the active and static fracture CDDFs are described fully in Sections 5.2, 5.3 and 5.4 of Welch et al. 2020.

1.3 Overview of the DFN Generator code

The DFN Generator code has been developed to implement this new algorithm in order to build fracture models of idealised or real geological structures. It can build either explicit DFN models, in which individual fractures are represented explicitly, as discrete geometric objects, or implicitly as fracture models, in which the population is represented only by CDDFs. This allows efficienct modelling of fracture network development over very large geological structures incorporating the impact of multiple structural features, which may be many kilometres in size and comprise of many millions of fractures.
To model fracture growth in simple idealised structures, DFN Generator is released with a standalone interface that receives model configuration instructions and delivers its output as text files. This interface uses only C Sharp code and can be compiled using a standard C Sharp compiler.
To model fracture growth in real geological structures, we recommend first building a static geomodel of the structure using a dedicated geomodelling package, and then linking DFN Generator to that package. Various commercial or open source geomodelling packages are available, and it is not our intention to replicate their functionality in DFN Generator. Links between DFN Generator and static geomodelling packages may take various forms, such as

- a “file converter” script to write the static geomodel as a text file or files that can be read by the current DFN Generator standalone interface,
- a standalone interface that can read files output by the geomodelling package and write the output from DFN Generator in a format that can be read back into the geomodelling package, or
- a plug-in module for the geomodelling package, written using an API specific to that package, that can access the geomodel data directly and then call the DFN Generator classes to run the calculation.

DFN Generator is already provided with one example of the latter type of interface, in the form of a plug-in for Schlumberger’s Petrel software, a commercial geomodelling package. This interface is compiled using Schlumberger’s Ocean API. Compiling this code creates a plug-in module that can be launched from within Petrel, and can build implicit or explicit fracture models directly from static geomodels in Petrel; the implicit models are output as properties in the Petrel geomodel grid and the explicit models are output as Petrel DFN objects.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

We would welcome any efforts to provide similar interfaces with other geomodelling packages. The code is designed such that it should be possible to write such interfaces without modifying the underlying DFN Generator calculation classes.

2 Implementation of the algorithm

A key innovation in the new method is the ability to model the evolution of the CDDFs describing the fracture network as a whole, without simulating the fractures on an individual basis that comprise the network. This allows the traditional fracture modelling workflow to be reversed: rather than building an explicit DFN first, then extracting the CDDFs, and finally (if at all) calculating the rates of fracture nucleation, propagation and interaction, we instead use fundamental geomechanical principles to calculate the fracture nucleation, propagation and interaction rates, then we derive the CDDFs, and finally we use this data to build an explicit representation of the fracture network. However the final step is optional and if the explicit DFN is not required, we can save time by not building it.

We have already shown how the fracture network can be described implicitly by a set of CDDFs describing the volumetric fracture density (P_{30}), the total area of fracture planes per unit volume (P_{32}), or the fracture porosity (P_{33}), with each considered with respect to fracture size. The CDDFs are calculated independently for circular microfractures (abbreviated μF) and layer-bound half-macrofractures (abbreviated $\underline{\text{MF}}$), for different fracture sets with different orientations, and for active (still propagating) and static (non-propagating) fractures. Thus ${}^{a1}{}_{\mu F}P_{30}(r)$ represents the count of circular active microfractures of set 1 with radius greater than r, per unit volume, and ${}^{s2}{}_{\underline{MF}}P_{32}(\ell)$ represents the total area of static layer-bound half-macrofractures of set 2 with length greater than ℓ, per unit volume.

We can sum the CDDFs of all fracture types and sets to obtain the total volumetric density, area or porosity of the entire fracture network. The ratio of densities for the different fracture sets can be used as indices for the overall fracture network anisotropy, and the ratio of active to static fractures (those deactivated by intersection or stress shadow interaction), can be used as indices for the network connectivity (see Section 8.5 of Welch et al. 2020).

The derivation of the algorithms for calculating the CDDFs and indices is described in detail in Welch et al. (2020), and the reader is referred to that publication for more information. In this paper, we describe how these algorithms are implemented in the DFN Generator code.

2.1 Workflow to generate the implicit fracture model

The simplest models, for example those used in sensitivity studies, assume a layercake model in which the fractured layer is flat, has uniform thickness and mechanical properties, and the in situ stress is homogeneous. However, in order to simulate fracture growth across more complex geological structures, we need to recreate the complex geometry of the geological layers within these structures, including dip, folding, thickness changes, and offset due to faulting. The models must also include lateral variation in the physical properties of rock and the local stress state.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

Static geomodels typically do this by discretising the layers into polyhedral gridblocks, with defined cornerpoints, and internally homogeneous properties. Once the fractured layer has been discretised into gridblocks and these have been assigned properties (upscaling if necessary and taking an average value of thickness and depth if these are not uniform), we can calculate the CDDFs for each gridblock independently. These calculations can be done sequentially or in parallel. A workflow to do this is illustrated in Fig. 3.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022
© Author(s) 2022. CC BY 4.0 License.

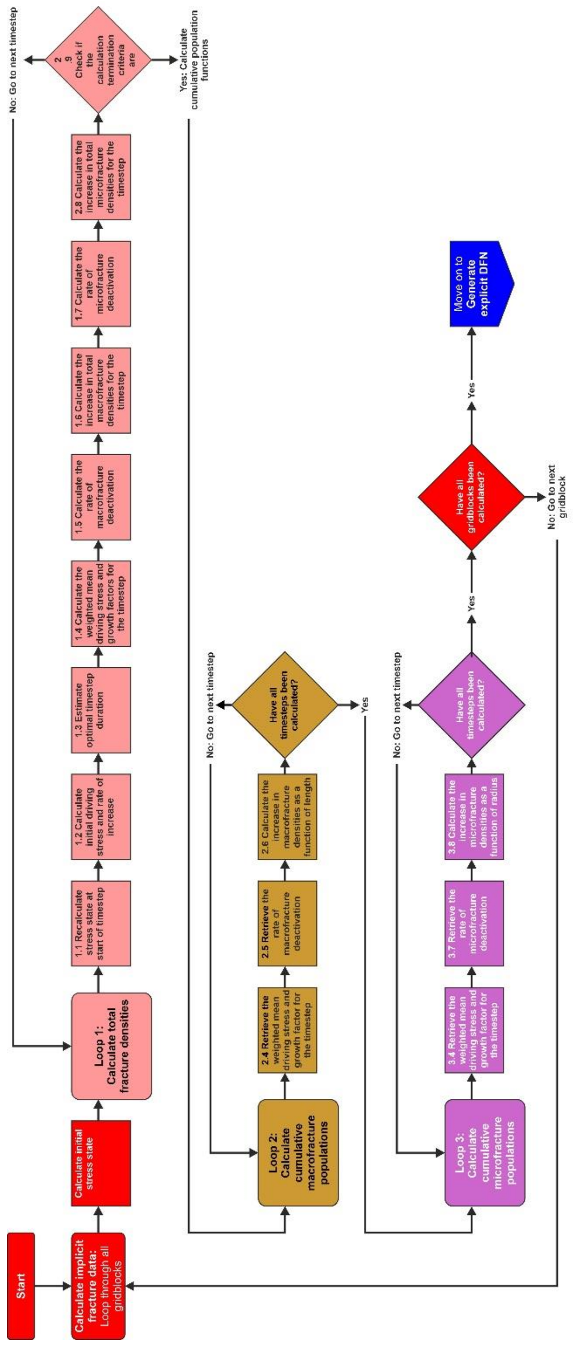

Start
Calculate implicit fracture data: Loop through all gridblocks
Calculate initial stress state
Loop 1: Calculate total fracture densities
1.1 Recalculate stress state at start of timestep
1.2 Calculate initial driving stress and rate of increase
1.3 Estimate optimal timestep duration
1.4 Calculate the weighted mean driving stress and growth factors for the timestep
1.5 Calculate the rate of macrofracture deactivation
1.6 Calculate the increase in total macrofracture densities for the timestep
1.7 Calculate the rate of microfracture deactivation
2.8 Calculate the increase in total microfracture densities for the timestep
2.9 Check if the calculation termination criteria are
No: Go to next timestep
Yes: Calculate cumulative population functions
Loop 2: Calculate cumulative macrofracture populations
2.4 Retrieve the weighted mean driving stress and growth factor for the timestep
2.5 Retrieve the rate of macrofracture deactivation
2.6 Calculate the increase in macrofracture densities as a function of length
Have all timesteps been calculated?
No: Go to next timestep
Yes
Loop 3: Calculate cumulative microfracture populations
3.4 Retrieve the mean weighted driving stress and growth factor for the timestep
3.7 Retrieve the rate of microfracture deactivation
3.8 Calculate the increase in microfracture densities as a function of radius
Have all timesteps been calculated?
No: Go to next timestep
Yes
Have all gridblocks been calculated?
No: Go to next gridblock
Yes
Move on to Generate explicit DFN

Figure 3: A flow diagram summarising the workflow elements of the methodology to calculate an implicit model from a structured grid representing a fractured layer. Based on Fig. 7.1 of Welch et al. (2020).

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

2.1.1 Workflow steps

Before modelling the fracture growth, it is necessary to calculate the initial in situ stress in gridblock, prior to applying external horizontal strain. This depends on the burial depth and fluid pressure, and includes compaction-related strain. The workflow then loops three times through the sequence of timesteps:

- The first loop calculates the total P_{30}, P_{32} P_{33} for all the microfractures and half-macrofractures in each fracture set. The fractures are subdivided into active and static fractures. The number and duration of the timesteps are determined dynamically, in order to optimise both the accuracy and runtime of the calculation. The timestep durations, the mean (time-weighted) fracture driving stress, the half-macrofracture propagation rate, the fracture interaction rates and some other key parameters, are all cached for use in the later loops.
- The second loop calculates an array of values for the half-macrofracture CDDFs, i.e. values of $_{\underline{MF}}P_{30}(\ell)$, $_{\underline{MF}}P_{32}(\ell)$ and $_{\underline{MF}}P_{33}(\ell)$ for a specified array of lengths ℓ. The results are output at the end of the final timestep; however it is necessary to loop through each timestep in order to obtain the final values, as the static half-macrofracture density data is calculated incrementally. The timestep durations, mean fracture driving stress, propagation rate and fracture deactivation probabilities are taken from the data cached during the first loop.
- The third loop calculates an array of values for the microfracture CDDFs, i.e. $_{\mu F}P_{30}(r)$, $_{\mu F}P_{32}(r)$ and $_{\mu F}P_{33}(r)$ for a specified array of radii r. The results are output at the end of the final timestep.

The calculation for each timestep in the first loop proceeds as follows:

1. The in situ stress is calculated at the start of the timestep (as described in Chapter 6 of Welch et al. 2020). This is calculated from the fluid pressure, lithostatic stress, and total horizontal strain. The total horizontal strain is a function of the total cumulative applied strain up until this point, but also takes into account viscoelastic strain relaxation, and the strain accommodated by previous fracture displacement, which is calculated using fracture density data from the previous timestep. If the horizontal strain is not balanced (i.e. the strain rate applied is not equal to the rate of strain relaxation and strain accommodation by fracture displacement), then the in situ stress will not be constant, and thus it is also necessary that the rate of change of the in situ stress is calculated.
2. Driving stress for each fracture set at the start of the timestep is calculated. The driving stress represents the resolved stress acting on the fracture set that drives fracture displacement and growth (Section 3.1 of Welch et al. 2020), and is a function of the situ stress tensor, the fracture set orientation, and the mode of fracture displacement. Therefore if the in situ stress tensor is not constant, it is also necessary to determine the rate of change of the fracture driving stress.
3. Based on the fracture driving stress, the optimal duration for the timestep is estimated as the time taken to increase the volume of stress shadows surrounding the fractures by 0.2% of the total volume, if there is no fracture deactivation (Appendix C of Welch et al. 2020). This will be different for each fracture set, so we must calculate this time for each fracture set and use the smallest value. By limiting the fracture growth in each single timestep in this way, we minimise

the discretisation error caused by carrying the calculated fracture densities from the last timestep to calculate the rate of deactivation when fractures are growing rapidly, while minimising runtime by reducing the number of timesteps when the fractures are not growing.

4. The weighted mean driving stress is calculated during the timestep for each fracture set (see section 2.2.2 and Chapter 6 of Welch et al. 2020). If the fracture driving stress is not constant. Then use the mean driving stress and timestep duration to calculate the fracture growth parameters γ and Γ (Section 3.2 of Welch et al. 2020) and the half-macrofracture propagation distance $\alpha_{MF}\Lambda$ (Section 4.2 of Welch et al. 2020) for each fracture set. The fracture growth factors represent the microfracture growth rate, and are used to calculate the macrofracture nucleation rate.
5. The rates of half-macrofracture interaction caused by intersection F_{ij} or interaction of stress shadow F_{ii} are calculated and combined to obtain the probability that an initially active half-macrofracture will remain active at the end of the timestep (the inverse deactivation probablility Φ). The half-macrofracture interaction rates are a function of the half-macrofracture propagation rate and the half-macrofracture densities at the end of the previous timestep (Section 5.1 of Welch et al. 2020).
6. At the end of the timestep the densities of the total active half-macrofracture densities ${}^{a}{}_{\underline{MF}}P_{30}(0)$, ${}^{a}{}_{\underline{MF}}P_{32}(0)$ and ${}^{a}{}_{\underline{MF}}P_{33}(0)$, while the static half-macrofracture density increments $\Delta^{s}{}_{\underline{MF}}P_{30}(0)$, $\Delta^{s}{}_{\underline{MF}}P_{32}(0)$ and $\Delta^{s}{}_{\underline{MF}}P_{33}(0)$ are calculated during the timestep. These are calculated from the the mean driving stress, timestep duration, the half-macrofracture nucleation and propagation rates, and the inverse of the deactivation probability for half-macrofractures of each fracture set (Section 5.3 of Welch et al. 2020). If the half-macrofracture densities and interaction rates are high, we must use the residual macrofracture populations formulae described in Section 5.4 of Welch et al. (2020). Finally, the total stress shadow volume ψ is calculated at the end of the timestep; this is simply total macrofracture area (${}_{\underline{MF}}P_{32}$) multiplied by the width (W) of the stress shadow.
7. Calculate the microfracture interaction rate, q, the probability that a microfracture active at the beginning of a timestep, will remain active throughout the timestep and into the next (the inverse microfracture deactivation probablility θ; Section 5.1 of Welch et al. 2020). Microfracture deactivation occurs as a result of the stress shadow interaction with parallel macrofractures, and is thus proportional to the rate of growth of the macrofracture stress shadow volume in the timestep. Therefore the half-macrofracture growth rate calculated in Step 6 must be calculated before the microfracture interaction rate.
8. The total active microfracture densities ${}^{a}{}_{\mu F}P_{30}(0)$ and ${}^{a}{}_{\mu F}P_{32}(0)$ are calculated at the end of the timestep, following calculation of the static microfracture density increments $\Delta^{s}{}_{\mu F}P_{30}(0)$ and $\Delta^{s}{}_{\mu F}P_{32}(0)$ during the timestep. The microfracture densities are calculated from the mean driving stress, the microfracture interaction rate for each fracture set, the timestep duration, and the microfracture growth parameters; ${}_{\mu F}P_{30}$ can be calculated analytically while ${}_{\mu F}P_{32}$ must be calculated numerically (Section 5.2 of Welch et al. 2020).

9. Check the updated fracture and timestep data to determine whether the calculation termination criteria are met (section 2.1.2) and continue to the next timestep if not.

Reusing the data cached in the first loop can optimise the second and third loops: the weighted mean driving stress, timestep duration, , fracture growth parameters γ and Γ, fracture interaction rates probabilities q, θ, F and Φ and macrofracture propagation distance $\alpha_{MF}\Lambda$. Therefore only step 6 in the third loop and step 8 in the second loopneed be calculated directly, although in these loops the CDDFs must be calculated for an array of half-macrofracture lengths ℓ and microfracture radii r.

2.1.2 Terminating the calculation

When an intact layer of brittle rock is subject to an external applied strain, fractures will often nucleate and propagate rapidly in the initial early stages of deformation, on a timescale of thousands of years, rather than the millions of years that may be commonly associated with broader tectonic regimes (Welch et al. 2019). However more of the fractures will become deactivated and stop propagating due to intersection with perpendicular or oblique fractures or stress shadow interactions as the fracture network grows, and the overall rate of fracture growth drops. Eventually the fracture network becomes saturated, i.e. the entire rock volume becomes occupied by fracture stress shadows. At this point no new fractures can nucleate and no fractures can propagate, so the fracture network stops growing, and displacement on existing fractures accommodates all subsequent strain.

The workflow should therefore include termination criteria that determine when to stop the calculation. Many different termination criteria could be applied, but they can all be classified into two types: static criteria, which are determined in advance, and dynamic criteria, which depend on the values of parameters calculated during the simulation.

The static criteria apply to the entire model, and will stop the calculation regardless of the state of the individual fracture sets. Two static termination criteria are used in this workflow:

- Duration: If we know the duration (in geological time) of the deformation episode during which the fractures developed, then the simulation can be set to terminate after this time has elapsed. However, while this criterion may be a limiting factor for some slow-growing fracture networks, most fracture networks develop over much shorter timescales than tectonic deformation episodes (10^3 to 10^4, rather than 10^6 years, Welch et al. 2019). Duration will not be a limiting criterion in these cases.
- Number of timesteps: The purpose of this termination criterion is to prevent excessive calculation times, especially in cases where none of the other termination criteria are met, or where the timestep durations are very short. However this criterion must be used with care, as it will cause the calculation to stop at an arbitrary time which has no geological or physical basis. Generally, if this termination criterion is triggered, it is an indication that there is a problem with the model that should be resolved before the results are used.

The dynamic criteria are used to determine when individual fracture sets have stopped growing. The calculation will only stop when all fracture sets have met one or more dynamic termination criteria. There are many permutations of fracture set parameters that could be used to define when a set has stopped growing, but three are used in this workflow:

- Proportion of active fractures: A fracture set can be considered to have stopped growing when the ratio of active to total fractures falls below a specified value. However care should be taken when applying this criterion, as it may exclude the slow development of late residual fractures. In some cases (e.g. secondary fracture sets), a significant proportion of the final fracture network may grow as late residual fractures.
- Ratio of current to peak historic active fractures: Typically there is an initial rapid rise in the density of active macrofractures within a fracture set, as new macrofractures nucleate; however as the macrofracture deactivation rate starts to rise, the density of active macrofractures reaches a peak before declining to negligible levels (see Fig. 4). We can therefore consider that the fracture set has stopped growing when the density of active fractures drops below a specified proportion of its peak value. Again, care should be taken when applying this criterion as it may exclude the slow development of late residual fractures.
- Total clear zone volume: Defined as the proportion of the total rock volume lying outside of the stress shadow exclusion zones surrounding fractures from that fracture set. The clear zone volume will decrease as the fracture set grows, and when it reaches zero, there is no more space for fractures to nucleate or grow. In practice, we often define a minimum clear zone volume slightly greater than zero since some of the equations can give unstable results when the clear zone volume is very low. Using this termination criteria to determine when a fracture set is inactive has the advantage that it will not exclude the development of late residual fractures. However there are two drawbacks: it cannot be used in situations where there are no stress shadows, and it can be difficult to apply when a fracture set contains both vertical Mode 1 and inclined Mode 2 (dilatant and shear, respectively) fractures, as these will feature differing stress shadow widths.

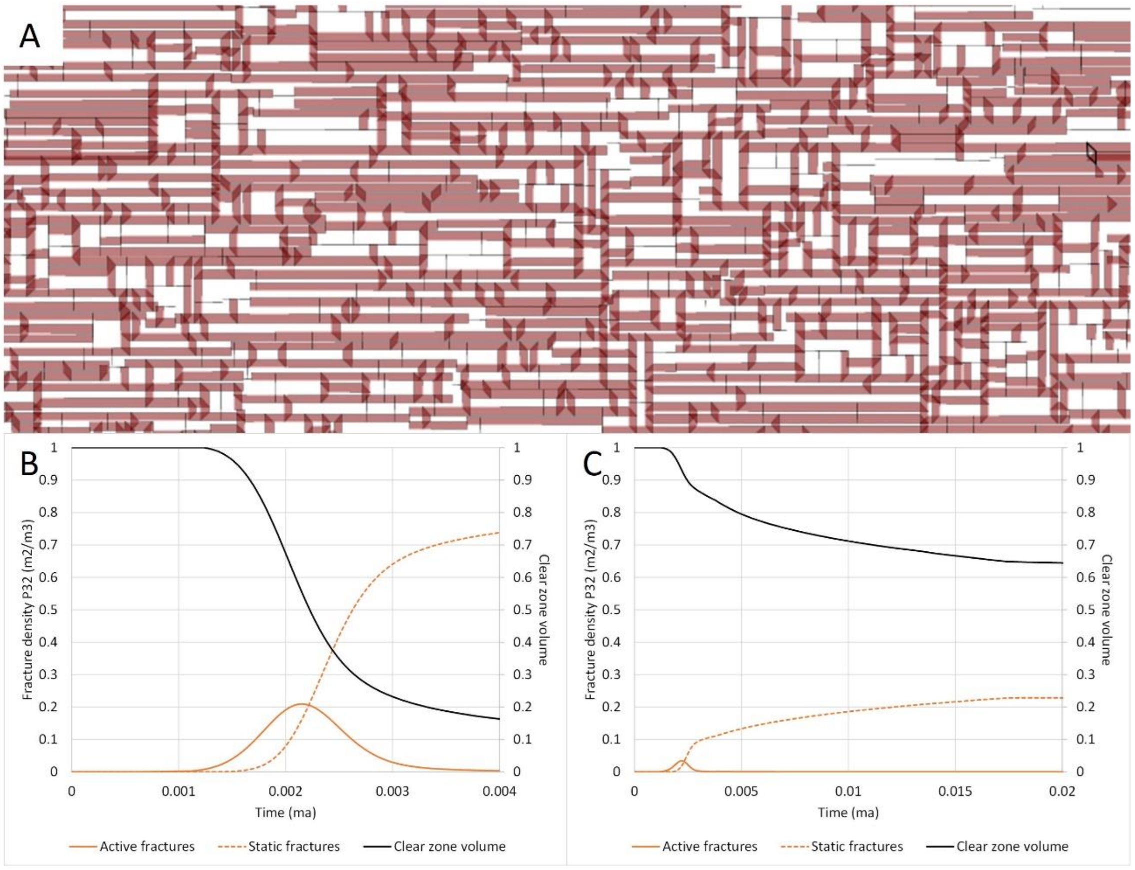


Figure 4: Typical evolution of macrofracture populations through time for an anisotropic fracture network (shown in A): (B) shows the growth of the primary fracture set (east-west fractures in A) and (C) shows the secondary fracture set (north-south fractures in A). In the primary fracture set, we see an initial rapid rise in the number of active macrofractures as new macrofractures nucleate; however as macrofracture deactivation rate starts to rise, the density of active macrofractures reaches a peak before declining to negligible levels, while the density of static macrofractures continues to rise. However for the secondary set, macrofracture deactivation rates are much higher from the outset because of the high probability of intersecting primary macrofractures. Therefore the active secondary macrofracture density never reaches the level of the active primary macrofracture density, and many of the secondary macrofractures develop during the residual fracture stage.

Even with these termination criteria, it can be difficult to correctly identify when the fracture network stops growing. Often the geometry of a fracture network may change significantly during the late stages of evolution, for example the growth of large numbers of short residual fractures, or the development of a late secondary fracture set connecting the previously isolated primary fractures. In some cases, there may be a long "dormant" period between the primary fracture set growth and the secondary fracture set growth (see Welch et al. 2019). Some trial and error may therefore be required to determine the optimal termination criteria for a specific model. However we have generally found good results are achieved with a maximum limit of 1000 timesteps, a minimum clear zone volume of 0.01 for all fracture sets, and no limit to the proportion of active fractures or ratio of current to peak historic active fractures. If the timing of fracturing is well constrained, this can be used to set a maximum limit to the model duration, however, as discussed this is not usually the limiting criterion.

2.2 Workflow to generate the explicit DFN

The conventional method of creating a Discrete Fracture Network (DFN) model is to place fractures in random locations, and assign them a size and orientation using stochastic distribution functions based on observations from well data, seismic attribute analysis or outcrop analogues. However, as we have discussed, this method has two main drawbacks:

- The statistical properties of the actual fracture network are often poorly constrained: especially size, geometry and connectivity of the fractures, but also their density and orientation away from the wellbores.
- Even where the statistical properties of the actual fracture network are well constrained, the stochastic method takes no account of the physical mechanisms of fracture nucleation and growth, so can result in a geologically unrealistic fracture network.

We can generate more accurate and realistic DFNs using the results of the implicit fracture modelling. These results include the rates of fracture nucleation and propagation for each fracture set, through time. We therefore simply need to place new seed fractures at random locations, according to the calculated fracture nucleation rate at that time, and allow them to grow at the calculated propagation rate at that time. As the fractures grow, we must also check for interaction between them, and deactivate them where appropriate.

This technique can be applied to model both microfractures and macrofractures explicitly. Since the algorithms used to determine propagation and interaction rates as well as fracture nucleation, are consistent with the physical mechanisms of fracture nucleation and growth, the result will be a DFN that honours the geological and geomechanical data, even in areas where there is no direct data to condition the DFN.

When building a DFN of a real geological layer, we will therefore use the same gridblocks and timesteps that were used for the implicit model. This means that we can simulate the network development through nucleation, propagation, and interaction of individual fractures independently within each gridblock (although allowing for propagation of individual fractures between gridblocks), and only at the end of the workflow must we combine these independent simulations to create a global fracture network extending across the entire fractured layer. When calculating the nucleation, propagation and interaction of fractures within the gridblocks it is helpful to use local spatial and temporal coordinate systems, defined relative to the orientation and growth rates of the fractures themselves, as this will simplify the calculations required. However these must be converted these back into global coordinates to generate the global fracture network.

2.2.1 Spatial coordinate systems

We can define three spatial coordinate systems: a global system, a local system relative to the gridblocks, and a local system relative to each fracture set. The global coordinate system is required to construct the global DFN, but the local system relative to the gridblock is more useful for various geometric calculations, while the local system relative to the fracture set is more useful for calculating fracture interactions.

Global spatial coordinate system (the XYZ coordinate system)

The global coordinate system is defined to be consistent across the entire static geomodel, so that it can be used to define the location of any point in the model. It can therefore be used to define the geometry of the entire fractured layer (as top, centre and bottom surfaces), and the global fracture network. It can also be used to define the location and size of the gridblocks themselves. It is common in static geomodelling to align the axes of the global coordinate system with the compass directions, so that the X coordinate represents the distance to the east (negative X values represent the distance to the west) and the Y coordinate represents the distance to the north (negative Y values represent the distance to the south). However, there is sometimes an advantage in using a different alignment for the global coordinate axes: for example to align them with a linear geological structure (e.g. a fold or a basin axis), a preferred flow direction (e.g. a well alignment or waterflood direction), or even a data acquisition orientation (e.g. seismic inlines and crosslines). The workflow described here is valid for any alignment of the global coordinate axes, as long as the X and Y axes are horizontal and the Y axis is oriented 90° anticlockwise from the X axis.

The Z axis is always vertical in static geomodels, but in some models it is positive downwards (so the Z coordinate represents depth) whereas in others it is positive upwards (so the Z coordinate represents height, and subsurface depths are represented by negative Z values). Care should therefore be taken to check the convention used in any particular geomodel. The equations given here assume that Z coordinates are positive downwards (so subsurface depth values will be positive); however it is easy to adapt the equations to the opposite convention if required, by reversing the polarity of the Z coordinate terms.

The origin of the global coordinate system for static geomodels is often chosen to be at the nearmost left hand corner of the model, relative to the X and Y axes. This ensures that the X and Y coordinates for all points in the model are positive, and have a large range (i.e. the length and width of the model is of the same order of magnitude as the absolute values of the coordinates). However, it is also common to choose an origin to be consistent with some universal standard coordinate system (e.g. Universal Transverse Mercator or UTM coordinates). In this case, the X and Y coordinates of points in the model may have a narrow range – i.e. the length and width of the model may be much smaller than the absolute values of the X and Y coordinates of points within the model. In this situation, there is a risk that errors may be introduced by rounding of the coordinate values during calculation, and it may be necessary to "rebase" the coordinates to a local origin before running the workflow. The Z coordinate origin is generally taken as either mean sea level (MSL) or ground level (GL) for onshore geomodels, and is usually selected to be consistent with the seismic reference datum (SRD), where applicable.

Spatial coordinates relative to the gridblock (the uvw coordinate system)

It is also possible to define a local coordinate system relative to the boundaries of a specific hexahedral gridblock, as shown in Fig. 5. We call this the uvw coordinate system:

- u represents the relative distance from the left hand gridblock boundary to the right hand gridblock boundary. u=0 for a point lying on the left hand boundary, u=1 for a point lying on the right hand boundary, and u=0.5 for a point halfway between the two boundaries.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

- v represents the relative distance from the near gridblock boundary to the far gridblock boundary. v=0 for a point lying on the near boundary, v=1 for a point lying on the far boundary, and v=0.5 for a point halfway between the two boundaries.
- w represents the relative distance from the top gridblock boundary to the bottom gridblock boundary. w=0 for a point lying on the top boundary, w=1 for a point lying on the bottom boundary, and w=0.5 for a point halfway between the two boundaries. This assumes the global Z axis is positive downwards, and must be swapped if the Z axis is positive upwards.

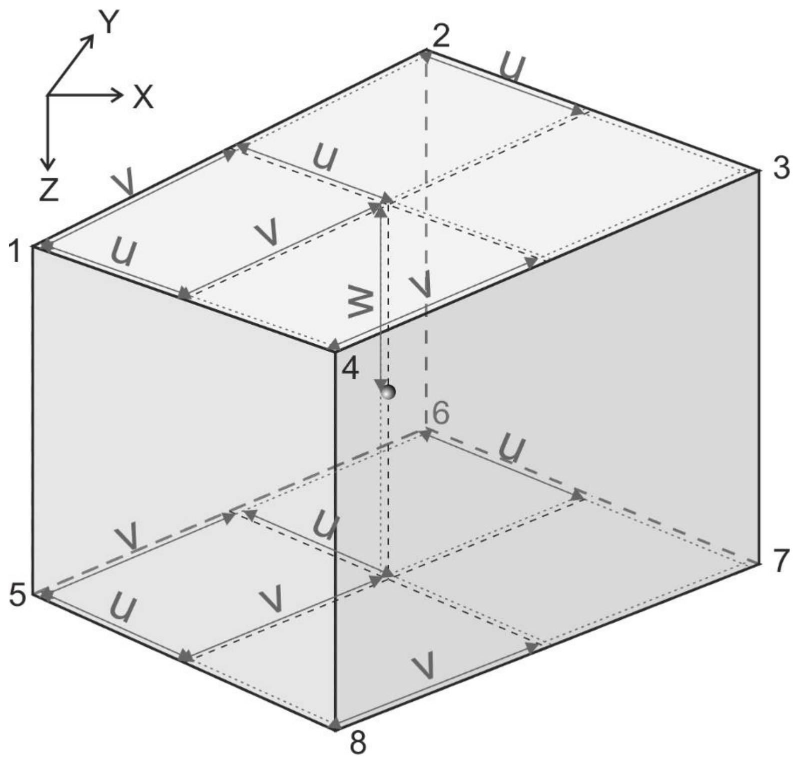


Figure 5: Schematic illustration showing the local uvw coordinate system relative to the gridblock, and to the global XYZ coordinate system, for a vertically aligned gridblock.

Note that the axes of the uvw coordinate system will generally not be orthogonal, and furthermore their orientation will vary spatially within the gridblock, unless the sides of the gridblock are all orthogonal (i.e. the gridblock is a rectangular prism). Common uses of the uvw coordinate system include:

- To interpolate the height of the gridblock at any point p within the gridblock.
- To interpolate the value of a spatially variable property at any point within a gridblock, if we know its values at the gridblock cornerpoints.

- To determine whether a specified point lies within the gridblock or not. It is possible to refer to some (but not all) points outside the gridblock using this coordinate system, but only points inside the gridblock will have uvw coordinates within the range 0 to 1.
- To generate a point at a random location within a gridblock.

Spatial coordinates relative to the fracture set (the IJK coordinate system)

In the IJK coordinate system, the I-axis is parallel to fracture strike, and the J-axis is 90° clockwise from the I-axis (see Fig. 6). Note that the I axis can be chosen in one of two opposite directions. The choice is arbitrary, but it is advisable (although not essential) to ensure consistency between adjacent gridblocks – i.e. if the I axis of a particular fracture set is oriented approximately north in one gridblock, it should be chosen to be approximately north rather than approximately south in the adjacent gridblocks, assuming that the variation fracture strike between the adjacent gridblocks is not too large.

The K axis is again assumed to be vertical and positive downwards; however the datum is taken as the centre surface of the gridblock. In most gridblocks the centre surface will not be horizontal, so the IJK coordinate system has the effect of shearing the gridblock vertically to bring the centre surface to the horizontal. Lengths and angles measured in the IJK coordinate system will therefore not correspond exactly with lengths and angles measured in the XYZ coordinate system, although if the dip of the layer is relatively low (<c.20°) the discrepancies should be small.

Because they are defined relative to the orientation of the fracture sets, we can use the IJK coordinates to define the location and geometry of individual fractures much more efficiently than we can using XYZ coordinates. Since, by definition, the layer-bound macrofractures extend from the top surface to the bottom surface, we can define each fracture segment in terms of just two nodes located on the centre surface of the gridblock (where K=0). Furthermore, since (again by definition) the segment must strike parallel to the I axis, the J coordinates of each node will be the same, so we need specify only three coordinates to define the location and geometry of the segment: a minimum I coordinate (I_{MF-}), a maximum I coordinate (I_{MF+}) and a J coordinate J_{MF}. We must of course specify the fracture dip and dip direction (either J^+ or J^-), and we must also specify whether the segment is still actively propagating, and if so in which direction. We can do this by specifying an active node a, which may be either the I^+ node, the I^- node, or neither. Microfractures can be defined by simply specifying the coordinates of the fracture centrepoint $I_{\mu F}$, $J_{\mu F}$, $K_{\mu F}$ and the fracture radius r, as well as the dip direction (either J^+ or J^-), and whether the segment is still actively growing. As we will see later, modelling fracture propagation and interaction is also easy in the IJK coordinate system.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

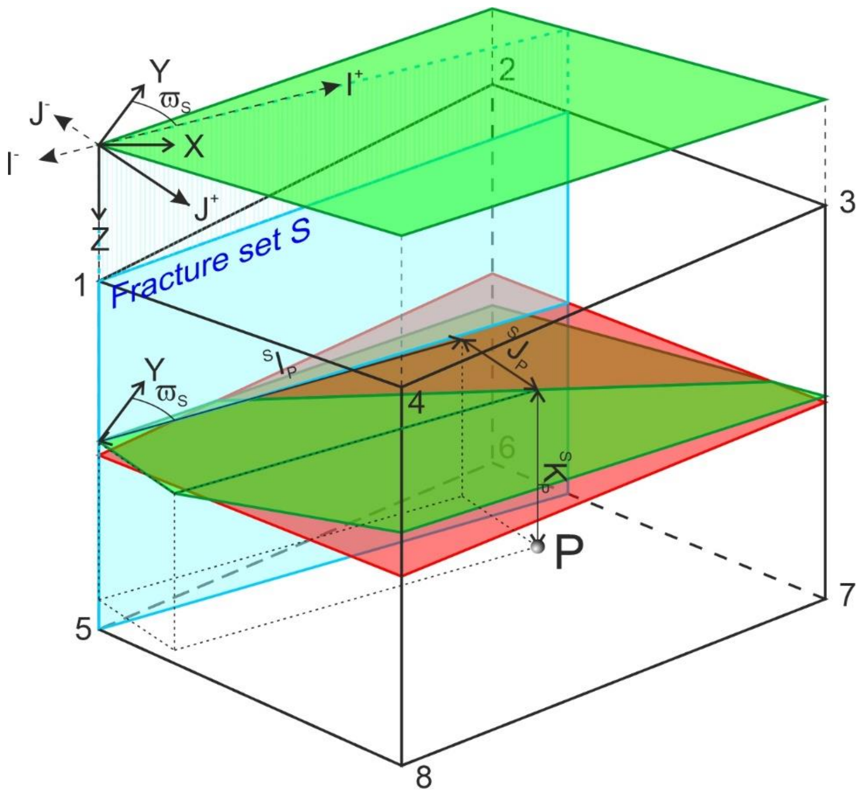


Figure 6: Schematic illustration showing the local IJK coordinate system relative to the fracture set, the gridblock, and to the global XYZ coordinate system. Note that, while the I and J coordinates are measured on a horizontal plane (shown in green), the K coordinate (depth) is measured from the centre surface of the gridblock (shown in red), which is not horizontal but dips gently towards the near corner (cornerpoints 4 and 8).

2.2.2 Temporal coordinate systems

Time may also be specified using either a universal temporal coordinate system (i.e. real time) or a local weighted temporal coordinate system. Real time must be used to compare the sequence of events in different locations and determine absolute rates of growth, but local weighted time can often simplify calculations of fracture nucleation and propagation within a gridblock.

Real time

The real time simply represents physical time elapsed since the start of the model. It is therefore consistent across all gridblocks, timesteps and fracture sets, and can be used to ascertain the relative order of events occurring in different gridblocks, including the start and end times of each timestep. It is advisable, although not essential, to set the origin (t=0) at the start of the deformation episode rather than at the present time, so that the real time coordinates of events in the model are all positive.

One problem that arises using real time coordinates is the extreme range of timescales between different processes when simulating fracture propagation: critical fracture propagation occurs at such a rapid rate (c.2000m/s) that a fracture may propagate across an entire gridblock in less than a second, whereas geological processes such as strain accumulation may occur over millions of years. Care must therefore be taken that rounding errors do not lead to inaccurate results when calculating times and rates.

Weighted time

Often the fracture driving stress is not constant but increases as more external strain is applied. Since the fracture nucleation and propagation rates are highly nonlinear with respect to driving stress, this can make it difficult to calculate the number of new fractures that will nucleate and the distance that each one will propagate within a given timestep.

We can resolve this problem by defining a "weighted" time, such that a macrofracture will always propagate the same distance during the same interval of weighted time, regardless of whether this is near the start or the end of the timestep. This can make it easier to calculate and compare the nucleation time and maximum propagation distances of different fractures within the same gridblock.

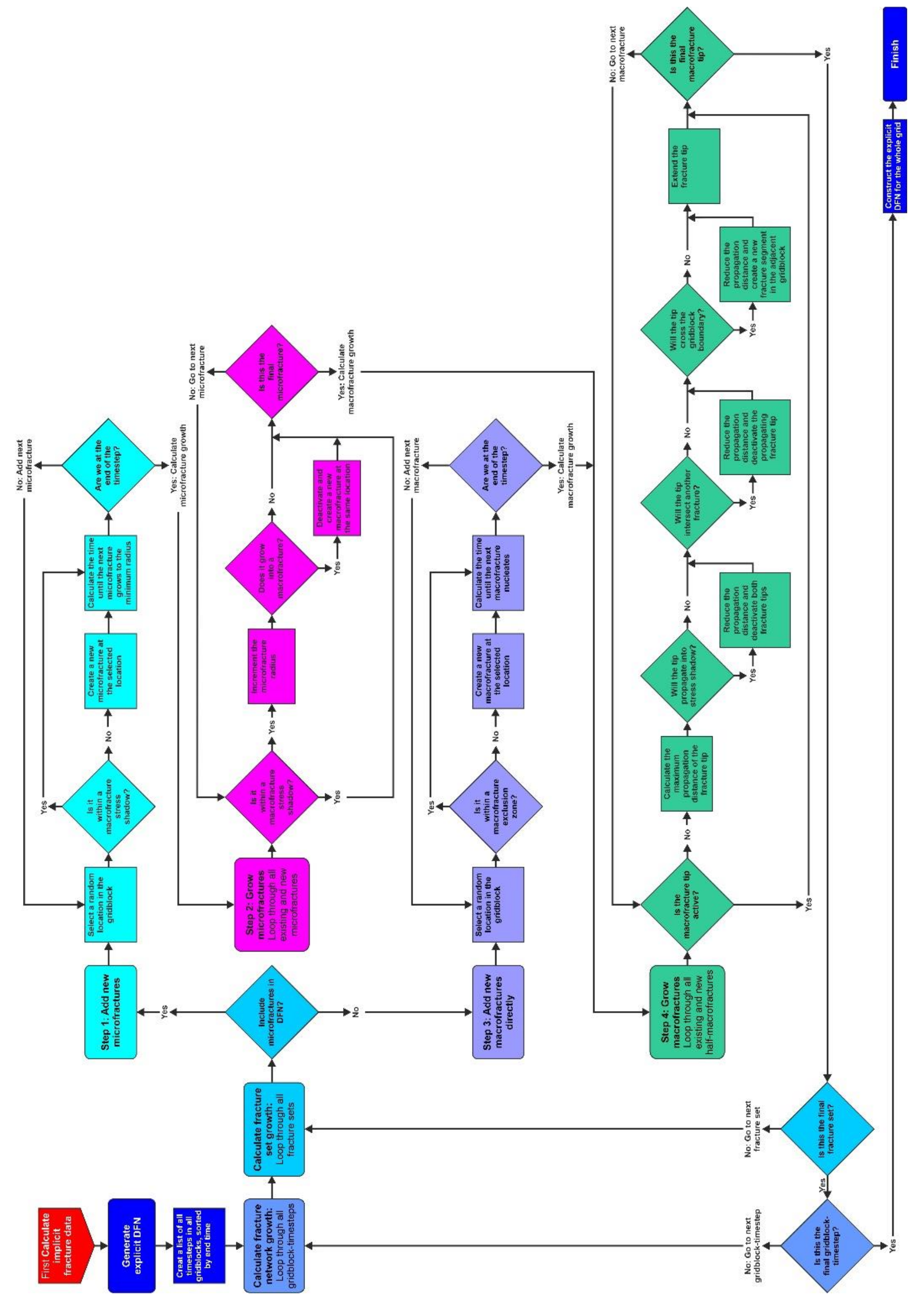
Note, however, that the timestep durations are determined dynamically, in response to the growth of the fracture network within each gridblock. The timesteps within different gridblocks will therefore not be contemporaneous. We must therefore use real time to correlate events that involve more than one gridblock, such as the propagation of fractures across gridblock boundaries, or the interaction of fractures in different gridblocks.

2.2.3 Workflow steps

A workflow for generating an explicit DFN model for an entire fractured layer, based on data from an implicit fracture model is presented in Figure 7. It can generate a DFN that comprises only macrofractures, or a DFN that comprises both microfractures (above a specified minimum radius) and macrofractures.

The growth of the DFN is calculated independently for each gridblock. This allows the use of local spatial and temporal coordinate systems to simplify the calculations of fracture propagation, interaction and nucleation. After calculation, the local DFNs are combined into a single global DFN referenced in global coordinates.

However, as previously discussed, the timesteps within different gridblocks are not contemporaneous. We therefore start by generating a list of each timestep for each gridblock, and sort them into chronological order. We then calculate the growth of the DFN within each "gridblock-timestep" sequentially, allowing fractures to propagate between adjacent gridblocks in real time (Section 2.2.2). The calculation for each "gridblock-timestep" comprises up to 4 steps: microfracture nucleation, microfracture growth, macrofracture nucleation, and macrofracture propagation and interaction. In this paper, we give only a brief overview of the equations used to perform these calculations; for a full description, see the "DFN Generator Technical Notes" document accompanying the source code.


First Calculate implicit fracture data
Generate explicit DFN
Creat a list of all timesteps in all gridblocks, sorted by end time
Calculate fracture network growth: Loop through all gridblock-timesteps
Calculate fracture set growth: Loop through all fracture sets
Include microfractures in DFN?
Yes
No
Step 1: Add new microfractures
Select a random location in the gridblock
Is it within a macrofracture stress shadow?
Yes
No
Create a new microfracture at the selected location
Calculate the time until the next microfracture grows to the minimum radius
Are we at the end of the timestep?
No: Add next microfracture
Yes: Calculate microfracture growth
Step 2: Grow microfractures Loop through all existing and new microfractures
Is it within a macrofracture stress shadow?
Yes
Yes
Increment the microfracture radius
Does it grow into a macrofracture?
Yes
Deactivate and create a new macrofracture at the same location
No
Is this the final microfracture?
No: Go to next microfracture
Yes: Calculate macrofracture growth
Step 3: Add new macrofractures directly
Select a random location in the gridblock
Is it within a macrofracture exclusion zone?
Yes
No
Create a new macrofracture at the selected location
Calculate the time until the next macrofracture nucleates
Are we at the end of the timestep?
No: Add next macrofracture
Yes: Calculate macrofracture growth
Step 4: Grow macrofractures Loop through all existing and new half-macrofractures
Is the macrofracture tip active?
Yes
No
Calculate the maximum propagation distance of the fracture tip
Will the tip propagate into stress shadow?
Yes
Reduce the propagation distance and deactivate both fracture tips
No
Will the tip intersect another fracture?
Yes
Reduce the propagation distance and deactivate the propagating fracture tip
No
Will the tip cross the gridblock boundary?
Yes
Reduce the propagation distance and create a new fracture segment in the adjacent gridblock
No
Extend the fracture tip
Is this the final macrofracture tip?
No: Go to next macrofracture
Yes
Is this the final fracture set?
No: Go to next fracture set
Yes
Is this the final gridblock-timestep?
No: Go to next gridblock-timestep
Yes
Construct the explicit DFN for the whole grid
Finish

495 **Figure 7: Flow diagram for a workflow to generate an explicit DFN model for a fractured layer discretised into gridblocks. Based on Fig. 7.2 of Welch et al. (2020).**

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

Step 1: Microfracture nucleation

The total fracture density of any microfracture population that follows a power law size distribution is theoretically infinite , since ${}_{\mu F}P_{30}(0)$ is infinite for a power law CDDF. If we are including microfractures in the DFN, we must therefore specify a minimum cutoff radius. New microfractures will only be added to the DFN when they exceed this radius; the results from the implicit model will give the rate at which this happens in each gridblock and timestep.

If there are no macrofracture stress shadows (meaning stress is evenly distributed), then microfractures will attain the minimum radius cutoff at predictable intervals. The interval until the next microfracture reaches the minimum cutoff is a function of the microfracture growth parameter, the microfracture CDDF and the gridblock volume, and it can be calculated easily using weighted time, from equation 3.14 from Welch et al. (2020). In reality, of course, there will be some random scatter in the initial microfracture size distribution, and hence in the rate at which microfractures reach the minimum cutoff; however this calculation should give a good approximation of the average rate throughout the timestep, as long as the gridblock is large enough and the timestep long enough.

The new microfractures should be placed located randomly locations in gridblocks, and their dip direction should also be assigned randomly.

However if microfractures are eveloped by stress shadows originating from the same fracture set, they will be unable to grow further . The actual rate at which microfractures reach the minimum radius limit will thus be lower than that given by equation 3.14 from Welch et al. (2020). Therefore we must check whether the randomly allocated location of each new microfracture lies within the stress shadow of a macrofracture of the same set; if it does, the new microfracture should be discarded. In this way, the number of new microfractures added will be reduced in proportion to the total macrofracture stress shadow volume, and will hence be consistent with the implicit microfracture density calculated as described in Section 5.2.1 of Welch et al. (2020).

We can easily determine whether a microfracture is directly influenced by a stress shadow of a macrofracture segment of the same fracture set by comparing their IJK coordinates. If the centre of the microfracture has coordinates ($I_{\mu F}$, $J_{\mu F}$), the I^- macrofracture segment node has coordinates (I_{MF-}, J_{MF}), and the I^+ macrofracture segment node has coordinates (I_{MF+}, J_{MF}), then the microfracture will lie within the macrofracture segment stress shadow if

$$I_{MF-} \leq I_{\mu F} \leq I_{MF+} \tag{1}$$

and

$$J_{MF} - \frac{W}{2} \leq J_{\mu F} \leq J_{MF} + \frac{W}{2} \tag{2}$$

where W is the microfracture stress shadow width.

Step 2: Microfracture growth

We can simulate the growth of active microfractures during any timestep, by using equation 3.8 of Welch et al. (2020) to calculate the increment in radius for each microfracture. This will be a function of current radius, microfracture growth factor and timestep duration (or time since nucleation, if this is shorter). Before this, however, we must recheck whether the macrofracture stress shadow covers and influences the microfracture that lies within, since the macrofractures may have grown during the previous timestep; any microfractures that is influenced by a stress shadow should be flagged as deactivated. After calculating the increment in radius for each microfracture, we must check whether the new radius is greater than half the layer thickness. If this is the case, the microfracture will cover the entire layer and will thus evolve into a layer-bound macrofracture. In this case we must deactivate the microfracture and add a new macrofracture seed at the microfracture location.

Step 3: Macrofracture nucleation

If microfractures are included in the DFN, then whenever a microfracture grows to radius greater than half the layer thickness, new macrofractures will nucleate, as described above. The new macrofracture will be a member of the same fracture set as the original microfracture, and will have the same dip and dip direction. It will comprise two segments, representing the two half-macrofractures propagating in opposite directions, each of which will have zero length.

If microfractures are not included explicitly in the DFN, then new macrofractures must be nucleated directly, by adding paired zero-length macrofracture segments at a random location in the gridblock. In this case, the interval until the next macrofracture nucleates is a function of the microfracture growth parameter, the microfracture CDDF, the gridblock volume and the layer thickness, and can be calculated easily using weighted time, from equation 4.10 of Welch et al. (2020).

Before adding the new macrofracture, we check that it is not situated within the stress shadow exclusion zone of a pre-existing macrofracture of its own set. Note that the exclusion zone is twice the width of the stress shadow, as the stress shadows of both the existing and the nucleating macrofractures are considered. Therefore, if the macrofracture nucleation point of the macrofracture has coordinates ($I_{\mu F}$, $J_{\mu F}$), then it will then lie within the exclusion zone of the existing macrofracture segment if

$$I_{MF-} \leq I_{\mu F} \leq I_{MF+} \tag{3}$$

and

$$J_{MF} - \mathrm{W} \leq J_{\mu F} \leq J_{MF} + \mathrm{W} \tag{4}$$

Step 4: Macrofracture propagation and interaction

Within an individual gridblock, the tips of all macrofractures of a given fracture set and mode will propagate at the same rate. Using weighted time, this rate will be constant during each timestep, and the maximum propagation distance during the timestep can be calculated from the driving stress and timestep duration (or time since nucleation, if this is shorter). The actual distance, however, is that any tip propagates may be less than thisf it becomes deactivated due to intersection with another fracture, stress shadow interaction, or if it crosses a gridblock boundary. Once we have determined the actual propagation distance for a fracture tip, we can easily calculate the new position of the propagating tip in IJK coordinates by adding or subtracting this to the I coordinate; the J coordinate will not change.

We have already seen that checking for microfracture stress shadow interaction is much simpler using IJK coordinates. The same is true for macrofracture stress shadow interaction: a macrofracture tip propagating in the I^+ direction (labelled MF1+) will be deactivated due to stress shadow interaction if there is an I^- tip of another macrofracture from the same set (labelled MF2-) lying within the space defined by

$$I_{MF2-} - \Delta\ell \leq I_{MF1+} \leq I_{MF2-} \tag{5}$$

and

$$J_{MF2} - \mathrm{W} \leq J_{MF1} \leq J_{MF2} + \mathrm{W} \tag{6}$$

where $\Delta\ell$ is the maximum propagation distance. When this occurs, the MF2- fracture tip will also be deactivated.

Checking for macrofracture intersection is also simpler if we use IJK coordinates. It is especially simple if the two macrofracture sets (labelled S1 and S2) have perpendicular strike; then we can take advantage of the orthogonality of the S1 and S2 IJK coordinates to avoid explicitly converting between the two coordinate systems. In this case, an S1 macrofracture tip propagating in the I^{S1+} direction will intersect an S2 macrofracture segment if

$$-J_{MF2} - \Delta\ell \leq I_{MF1+} \leq -J_{MF2} \tag{7}$$

and

$$I_{MF2-} \leq J_{MF1} \leq I_{MF2+} \tag{8}$$

When this occurs, only the propagating S1 macrofracture tip will be deactivated.

If a propagating macrofracture tip crosses a gridblock boundary, it will not be stop propagating, but further propagation will be controlled by the stress orientation and magnitude in the adjacent gridblock. We must therefore deactivate the currently propagating macrofracture segment and create a new zero-length macrofracture segment in the adjacent gridblock at the crossing-point of the shared boundary edge.

Checking for fractures crossing gridblock boundaries is more complex than checking for fracture interaction, as the gridblock boundaries may not be aligned with the fractures. The easiest way to check if a propagating macrofracture tip of set S will cross a gridblock boundary is to convert the locations of the cornerpoints of the gridblock to IJK coordinates, and then calculate the I coordinate of the intersection of the projected fracture propagation path and the gridblock boundary.

Two other complications may also arise when fractures cross gridblock boundaries:

- Occasionally the stress orientation in the adjacent gridblocks may be convergent, so the fracture propagates towards the boundary from both sides. In this case, the fracture will continue to propagate backwards and forwards between the two gridblocks indefinitely. This problem can be resolved by propagating the fracture along the gridblock boundary.
- Since the timesteps in each gridblock are generally not contemporaneous, it is possible that fracture propagation in the adjacent gridblock has already been calculated beyond the time the fracture crosses the boundary. In this case, the

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

new segment will need to be extended immediately, at the appropriate propagation rate, and checking for stress shadow interaction and intersection.

Populating the global DFN

Once we have generated the local DFNs for each gridblock, we must combine them into a single global DFN that extends across the entire fractured layer. The fractures in this global DFN must of course be referenced in global XYZ coordinates. Since each microfracture is confined to a single gridblock, we can simply copy them across from each local DFN into the global DFN. However the microfractures in the local DFNs are described in a minimal form using the local IJK coordinate system. In the global DFN, the microfractures must be defined in a more complete form using the global XYZ coordinate system. This can be done in one of two ways:

- Each microfracture can be defined as a circle. To do this, we must define a centrepoint, radius, dip and azimuth.
- Each microfracture can be defined as a planar polygon. To do this, we can calculate the positions (in XYZ coordinates) of a series of equally spaced points around the circumference of the fracture, starting at the top. Although this is a less precise and more cumbersome form for defining the geometry of a microfracture than the circular form, it may provide better compatibility with some geological modelling software packages that can only handle polygonal fractures.

Populating the global DFN with macrofractures is more complicated for two reasons:

- Macrofractures typically comprise multiple segments which may not be coplanar.
- Individual macrofractures may span multiple gridblocks, although each segment is confined to a single gridblock.

We must therefore combine the individual macrofracture segments from each gridblock into complete macrofractures, that may span multiple gridblocks. As with the microfractures, the macrofracture segments in the local DFNs are described in a minimal form using the local IJK coordinate system, which must be converted to a more complete form in the global XYZ coordinate system. We can calculate the positions of the cornerpoints of the macrofracture segments in XYZ coordinates, by extending each of the segments vertically to the top and bottom surfaces of the appropriate gridblock, based on the dip and dip direction of the segment. However if two adjacent segments are non-parallel (e.g. if they are in different gridblocks where the fracture strikes are different), the top and bottom cornerpoints of the extended segments will not be coincident, even though the segment nodes are. In this ca segmentsse we must calculate the intersection points between the top and bottom edges of the two segments, and use these as the respective cornerpoints for both segments (see Fig. 8).

It has been suggested that in nature, large fractures and faults often form by coalescing of multiple, independently nucleated dislocation planes that propagate towards each other, joining to form a single large fault (e.g. Huggins et al. 1995, Ferrill and Morris 2001, Walsh et al. 2003). As discussed in Section 1.2, this phenomenon is not replicated in the implicit models by the method described here. However in the explicit DFN, it is possible to combine independent macrofractures that due to stress shadow interaction terminate against each other into a single large macrofracture, by adding an additional relay segment between the two fracture tips.

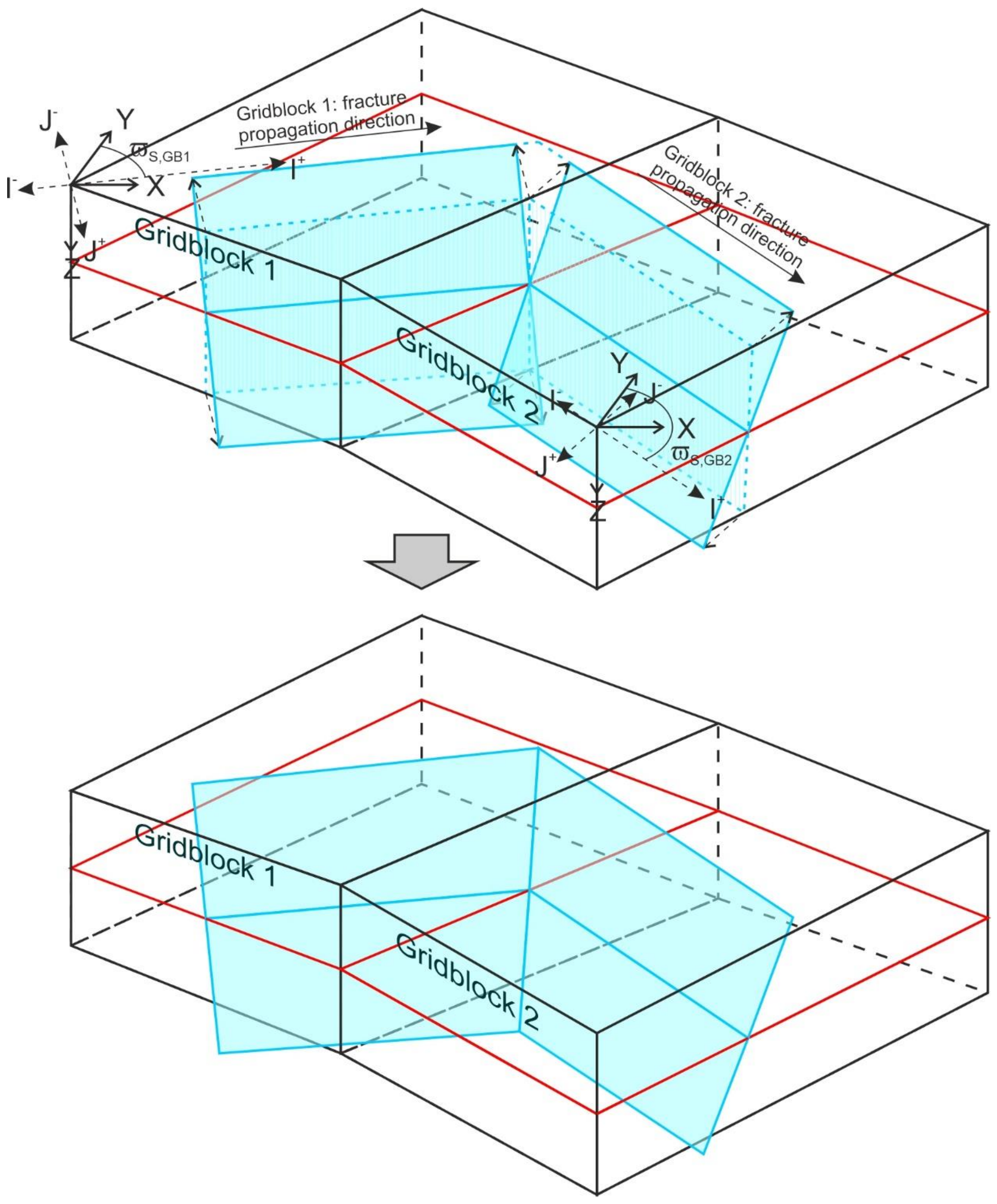

Gridblock 1: fracture propagation direction
Gridblock 2: fracture propagation direction
J-
Y
ϖS,GB1
I-
X
I+
Z
J+
Gridblock 1
Gridblock 2
ϖS,GB2
Gridblock 1
Gridblock 2

Figure 8: Schematic illustration showing bevelling of the cornerpoints between two non-parallel segments of a layer-bound macrofracture in adjacent gridblocks. 630

Two points to bear in mind when linking macrofractures in this way are:

- The two macrofractures may dip in opposite directions. This will generate a combined macrofracture that switches polarity along the strike. However this is consistent with observations of linked fault segments in outcrop and subsurface data (Nicol et al. 1995).
- Since the calculation of the implicit CDDFs does not take account of macrofracture linkage, this will introduce a discrepancy between the explicit and implicit CDDFs: specifically, the implicit macrofracture CDDFs will underestimate the density and area of large macrofractures and overestimate the density of small macrofractures. This is examined further in Section 8.3.5 of Welch et al. (2020). However the total macrofracture area ${}_{MF}P_{32}$ will be unaffected by macrofracture linkage, so this should still be consistent for both explicit and implicit fracture models.

Nevertheless, large faults and fractures formed by connected segments can provide important conduits for flow in the subsurface, so including them in the DFNs by this method is likely to give more accurate results from fluid flow modelling.

3 Code structure

The workflows described in the previous section have been implemented in object-oriented code (mostly C Sharp), packaged in a Microsoft Visual Studio solution. This is available for download from Github (https://github.com/JointFlow/DFN-Generator) as open source software under the Apache v2 license, and can be freely modified or enhanced under the terms of the license.

The Github package contains code both to carry out the calculation and to provide a user interface. We envisage two likely motives for modifying or enhancing the code:

- To provide a new interface or to integrate the code with another software package (e.g. as a plug-in to a geomodelling package).
- To provide additional functionality or to modify existing functionality (including bug fixing).

To simplify this, the Visual Studio solution has been structured so that the interface and calculation components of the code are kept in separate projects:

- The interface code reads the input data, handles the model building and property population, launches the calculation and then displays or outputs the results in an appropriate format. The DFN Generator software can have multiple interfaces, for example to handle different data formats or to interact with different geomodelling packages. Each interface is stored in a dedicated project within the Visual Studio solution, and can be compiled independently.
- The calculation code runs the calculations to implement the workflows described in Section 2, based on the algorithms described in Welch et al. (2020). The same calculation code is used by all interfaces, and is stored in the Shared Code project within the Visual Studio solution. It cannot be compiled directly, but only in combination with one or more interfaces.

Therefore, without modifying the calculation code, it is possible to modify or add interfaces, and similarly, to modify or add functionality with minimal changes to the interface code.

3.1 Interface code

At present two interfaces are provided:

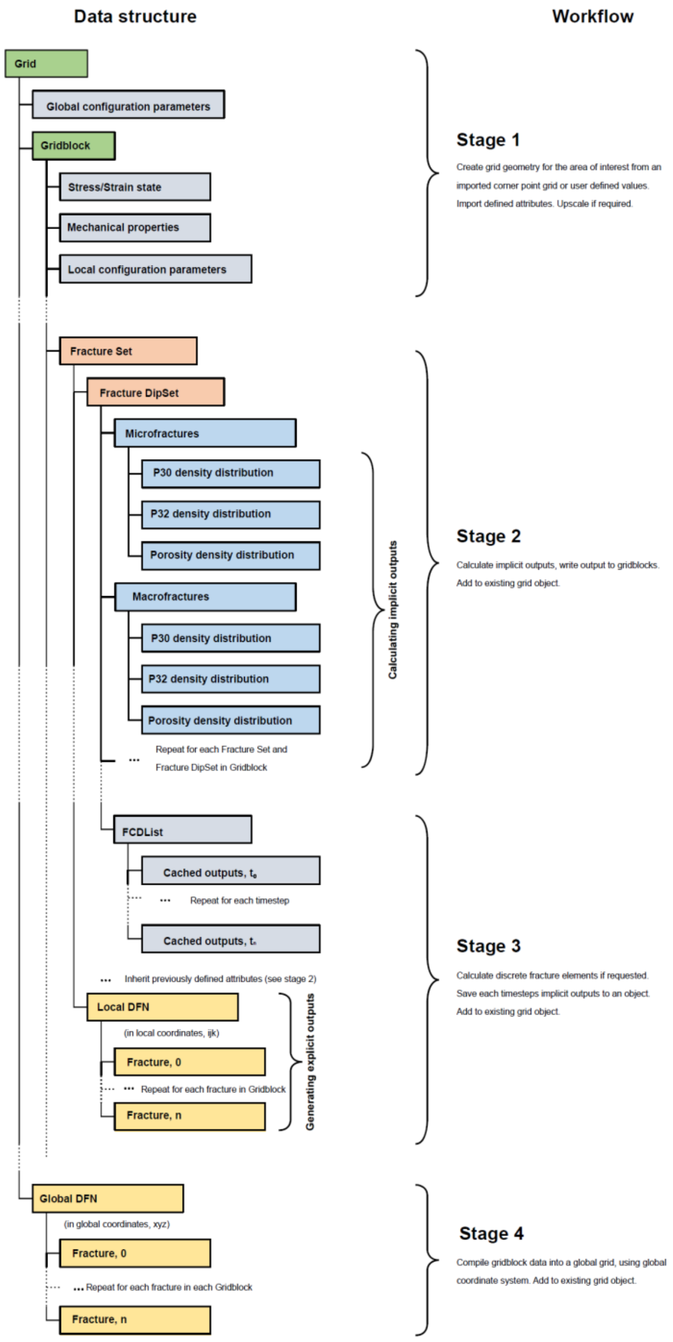
- A standalone interface which reads input data from and writes output data to text files. This is written in standard C Sharp, and can be compiled by any C Sharp compiler without additional software.
- A plug-in for Schlumberger's Petrel geomodelling package, written using the Ocean API. Note that this will require an Ocean licence to compile and a Petrel license to run, both of which are available from Schlumberger. Also included in the Visual Studio solution is a separate project to generate Petrel installers for the compiled Ocean code. These installers will allow users to run the compiled plug-in without an Ocean license, although a Petrel license is still required.

There are many other static geomodelling packages available, either as commercial or open source software, that users may wish to interface with DFN Generator. This would allow these geomodelling packages to be used to define the model geometry and input properties (including in situ stress and strain), as well as to display the output, while the DFN Generator code is used to run the algorithms and generate the fracture models (as illustrated in Fig. 9). The integration of the DFN Generator with the static geomodelling packages may be a soft integration (i.e. as two stand-alone software packages transferring data by reading and writing text files in a standardised format), or a hard integration (e.g. using interface code to create a plug-in or module for the static geomodelling package).

3.2 Calculation code

The calculation code is illustrated schematically in Fig. 9. It comprises a series of classes representing geological and geometric entities in hierarchical order, from the grid class representing the entire fractured layer, down through the gridblock and fracture set classes, to classes representing individual fractures and fracture segments. These classes contain methods that are triggered to run various aspects of the simulations, such as the generation of the implicit fracture model within each gridblock, and the propagation and nucleation of fractures in the explicit DFN. In addition there are a number of classes used to store specific data, for example the mechanical properties, stress and strain state, and CDDFs. Note that Fig. 9 is not comprehensive; the calculation code also defines basic classes, such as points (in XYZ or IJK coordinates), vectors and tensors which are used to construct the more complex classes, and numerous methods to perform elements of the calculations, or to store or retrieve data.

Detailed guidance, both for adding new interfaces and for modifying the calculation code, is provided in Section 5 of the "DFN Generator Technical Notes" document accompanying the source code.


Data structure
Workflow
Grid
Global configuration parameters
Gridblock
Stress/Strain state
Mechanical properties
Local configuration parameters
Stage 1
Create grid geometry for the area of interest from an imported corner point grid or user defined values. Import defined attributes. Upscale if required.
Fracture Set
Fracture DipSet
Microfractures
P30 density distribution
P32 density distribution
Porosity density distribution
Macrofractures
P30 density distribution
P32 density distribution
Porosity density distribution
... Repeat for each Fracture Set and Fracture DipSet in Gridblock
Calculating implicit outputs
Stage 2
Calculate implicit outputs, write output to gridblocks. Add to existing grid object.
FCDList
Cached outputs, t_0
... Repeat for each timestep
Cached outputs, t_n
... Inherit previously defined attributes (see stage 2)
Local DFN
(in local coordinates, ijk)
Fracture, 0
... Repeat for each fracture in Gridblock
Fracture, n
Generating explicit outputs
Stage 3
Calculate discrete fracture elements if requested. Save each timesteps implicit outputs to an object. Add to existing grid object.
Global DFN
(in global coordinates, xyz)
Fracture, 0
... Repeat for each fracture in each Gridblock
Fracture, n
Stage 4
Compile gridblock data into a global grid, using global coordinate system. Add to existing grid object.

695 **Figure 9: Schematic diagram showing the object hierarchy of the DFN Generator code. The right hand side shows the objects involved at different stages of the calculations.**

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022

4 Verification and evaluation

Validation of the DFN Generator code can take two forms:

- Verification involves checking that the code correctly implements the algorithm described in Welch et al. (2020), and
- Evaluation involves checking whether this algorithm can accurately replicate the geometry of observed natural fracture networks.

Both verification and evaluation of the DFN Generator output are addressed in Welch et al. (2020), but are summarised here.

4.1 Verification by comparison of implicit and explicit fracture models

The best way to test whether the code is correctly implementing the algorithm, and that this algorithm correctly represents the underlying processes and assumptions described in Welch et al (2020), is to compare the implicit and explicit fracture models for consistency. We can calculate CDDFs empirically from the explicit DFN by measuring the radius of each microfracture and the length of each half-macrofracture, and compare these with the calculated CDDFs of the corresponding implicit model. Since the fracture density and size distribution are emergent properties in the explicit DFN, controlled by the rates of fracture nucleation, propagation and interaction, a good match between the implicit and explicit CDDFs will confirm that the fracture propagation, nucleation, and interaction rates calculated from the fracture densities for the implicit model, are valid for the underlying assumptions.

A validation exercise along these lines is described in Section 7.4 of Welch et al. 2020, in which implicit and explicit CDDFs for six fracture sets from five different planar, homogeneous models were calculated and compared (see Table 1 for details). The key results are summarised in Figure 10 and below:

- A good match is observed between the implicit and explicit CDDFs for all fracture sets. This confirms that the equations describing the implicit fracture model, derived in Chapters 3, 4 and 5 of Welch et al. 2020, are valid and correctly implemented for microfractures and macrofractures, with or without stress shadows, and for fracture sets generated by uniaxial, isotropic or anisotropic horizontal strains.
- There are, however, some minor discrepancies between the implicit and explicit CDDFs. These discrepancies are partly due to the random location of fracture nucleation points in the explicit DFN; however, as described in Section 8.5 of Welch et al. (2020), they are partly caused by positive feedback processes which cause small local heterogeneities in the early stages of the explicit DFN to become amplified as the fracture network grows.
- There are also systematic biases resulting from the simplifications and discretisation inherent in the models. These may affect both the implicit and explicit CDDFs. Causes of these systematic biases are discussed in detail in Section 7.4 of Welch et al. (2020), but may include the numerical solutions employed for some differential equations, discretisation into timesteps, and errors caused when fractures approach or cross gridblock boundaries in the explicit DFN.

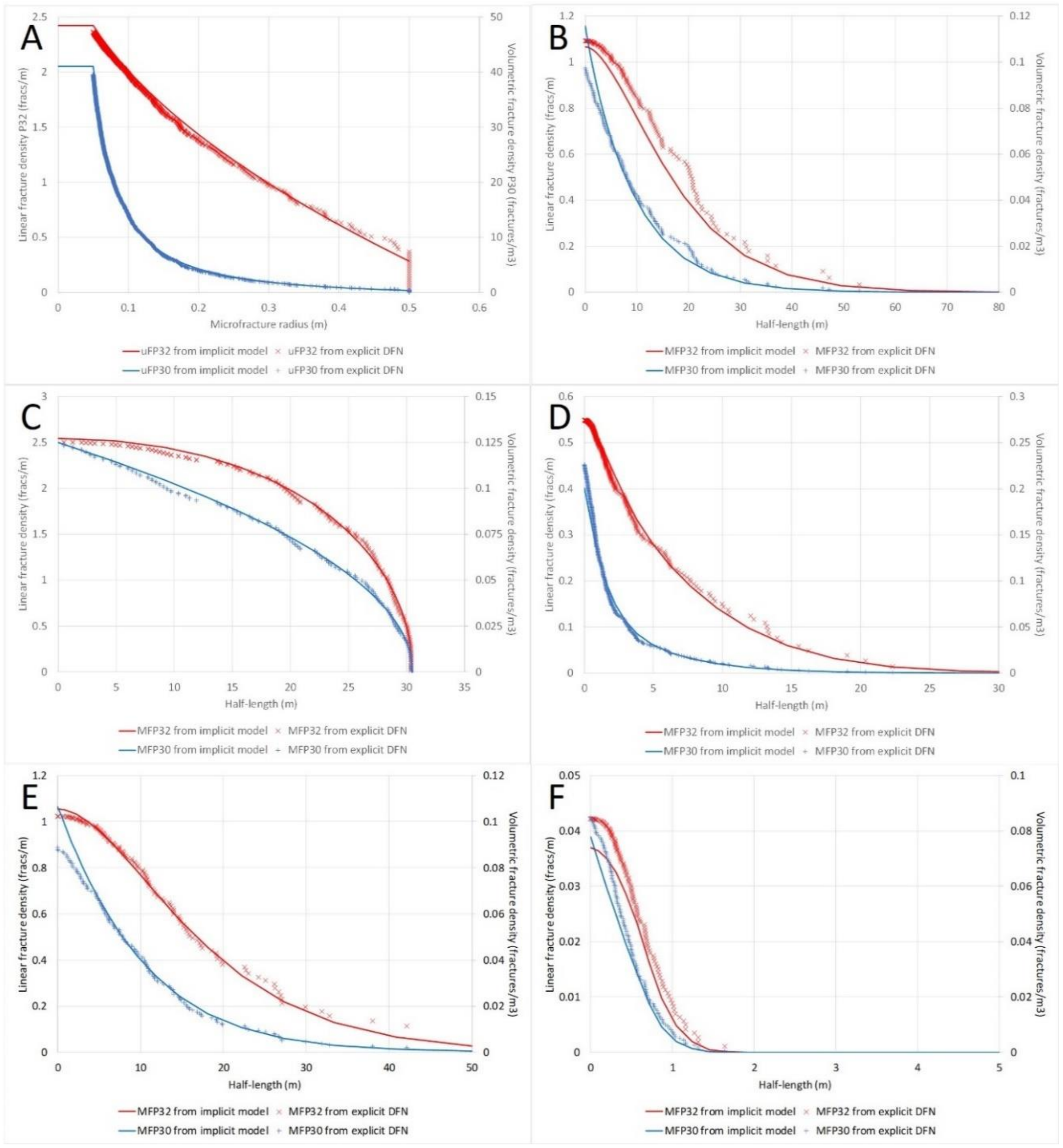


Figure 10: Comparison of implicit and explicit CDDFs for different scenarios: a) single microfracture set with uniaxial horizontal strain and stress shadows, b) single macrofracture set with uniaxial horizontal strain and stress shadows, c) single macrofracture set with uniaxial horizontal strain and evenly distributed stress, d) primary macrofracture set with isotropic horizontal strain ($\varepsilon_{hmin} = \varepsilon_{hmax}$) and stress shadows, e) and f) primary and secondary macrofracture sets respectively, from a model with anisotropic horizontal strain ($\varepsilon_{hmin} = 0.5\varepsilon_{hmax}$) and stress shadows. $_{\mu F}P_{30}(r)$ and $_{\underline{MF}}P_{30}(\ell)$ are shown in blue, $_{\mu F}P_{32}(r)$ and $_{\underline{MF}}P_{32}(\ell)$ are shown in red; solid lines indicate implicit CDDFs, crosses indicate the CDDFs calculated empirically for the explicit DFN. Reproduced with kind permission from Fig. 7.4 of Welch et al. (2020).

Property	**Fig. 10A**	**Fig. 10B**	**Fig. 10C**	**Fig. 10D**	**Fig. 10E&F**
Thickness h (m)	1	1	1	1	1
Model size (m)	10	150	150	150	150
Effective vertical stress σ_v' (MPa)	29.43	29.43	29.43	29.43	29.43
Min horizontal strain rate $\dot{\varepsilon}_{hmin}$ (/ma)	-0.01	-0.01	-0.01	-0.01	-0.01
Max horizontal strain rate $\dot{\varepsilon}_{hmax}$ (/ma)	0	0	0	-0.01	-0.008
Strain relaxation time constant (ma)	0.2	0.2	0.2	0.2	0.2
Stress shadows?	Yes	Yes	No	Yes	Yes
Subcritical index b	2	10	10	10	10
Initial μF density coefficient B	0.001	0.001	0.001	0.001	0.001
Initial μF distribution coefficient c	1.5	2	2	2	2
Young's Modulus E (GPa)	10	10	10	10	10
Poisson's ratio ν	0.25	0.25	0.25	0.25	0.25
Friction coefficient μ_{fr}	0.5	0.5	0.5	0.5	0.5
Crack surface energy (J/m^2)	1000	1000	1000	1000	1000
Initial compaction factor υ	1	1	1	1	1

Table 1: Input parameters for the models illustrated in Fig. 10. An effective vertical stress of 29.43MPa represents deformation at 2000m depth with hydrostatic fluid pressure. Reproduced with kind permission from Table 7.1 of Welch et al. (2020).

If the timestep duration is reduced, systematic biases can in general be minimised, so that the macrofracture density increase within each timestep is minimised, or by increasing the gridblock volume, so that, compared with the size of individual fractures, the gridblocks are large. The random variability will also be reduced by increasing the gridblock volume, since local heterogeneities have less impact on the overall fracture density distribution functions in a large gridblock. However increasing the gridblock volume comes at the expense of lateral resolution in defining variations in situ stress state and in the mechanical properties , which may cause problems when modelling complex geological structures.

4.2 Evaluation by comparison with observed natural fracture networks

Ultimately, the value of the model in predicting actual fracture networks can only be established by comparing the model results with natural fracture networks observed in subsurface data such as borehole images and core, or from outcrop. This has been carried out for three fractured outcrops (Robin Hood's Bay in northeast England, Nash Point in south Wales, and Pegwell

Bay in southeast England) and two fractured subsurface reservoirs (above the Anloo salt diapir, onshore Netherlands and the Kraka oilfield in the Danish North Sea). The results are presented in detail in Chapters 9 and 10 of Welch et al. (2020), but the key results will be highlighted here.

In all cases, the DFN Generator was able to accurately reproduce the observed fracture geometry and was able to make predictive models with limited input data. However these examples did demonstrate the importance of an accurate understanding of the geological deformation that caused the fracturing, and in particular an accurate representation of the applied elastic strain field, to successfully replicate the observed fracture networks. It also demonstrated the importance of understanding the mechanical stratigraphy, and accurately identifying the brittle layers prone to fracturing.

4.2.1 Nash Point

The Nash Point study aimed to simulate a fracture network developed in a c.0.5m thick limestone bed, in response to local stress anomalies developed around larger (m-scale throw) faults (as mapped by Maerten et a. 2016). Figure 11 shows a comparison of the outcrop and modelled fracture network. This study highlighted three key points:

- The layer thickness is a key control on fracture spacing and hence the P_{32} fracture density, since it controls the stress shadow widths around the fractures. Therefore it is necessary understand the mechanical stratigraphy, and to correctly identify the extent of the brittle layers within that stratigraphy, in order to accurately replicate the observed fracture pattern. This is easy in outcrops such as Nash Point which comprise clearly distinguishable interbeds of brittle limestone and ductile clay, but may be more difficult in layers with more subtle lithology contrasts.
- The subcritical fracture propagation index and the initial microfractures density are key controls on the fracture length. This is consistent with the findings from similar modelling studies by Olson (1993, 2004). Estimates for both parameters can be taken from the literature (e.g. Atkinson 1984 and Swanson 1984 for subcritical fracture propagation index, Westaway 1994 for initial fracture populations) or by calibration against observed fracture patterns in the outcrop.
- Accurate modelling of the local stress and strain makes it possible for the DFN Generator to replicate the complex fracture patterns that can develop around larger geological structures. In this study, we used a boundary element simulator build into the Petrel geomodelling package to calculate the local elastic strain in the rockmass around the faults, and supplied this as input to generate the DFN. The grid resolution was set sufficiently high to replicate the geometry of the observed fracture network (see Fig. 11).

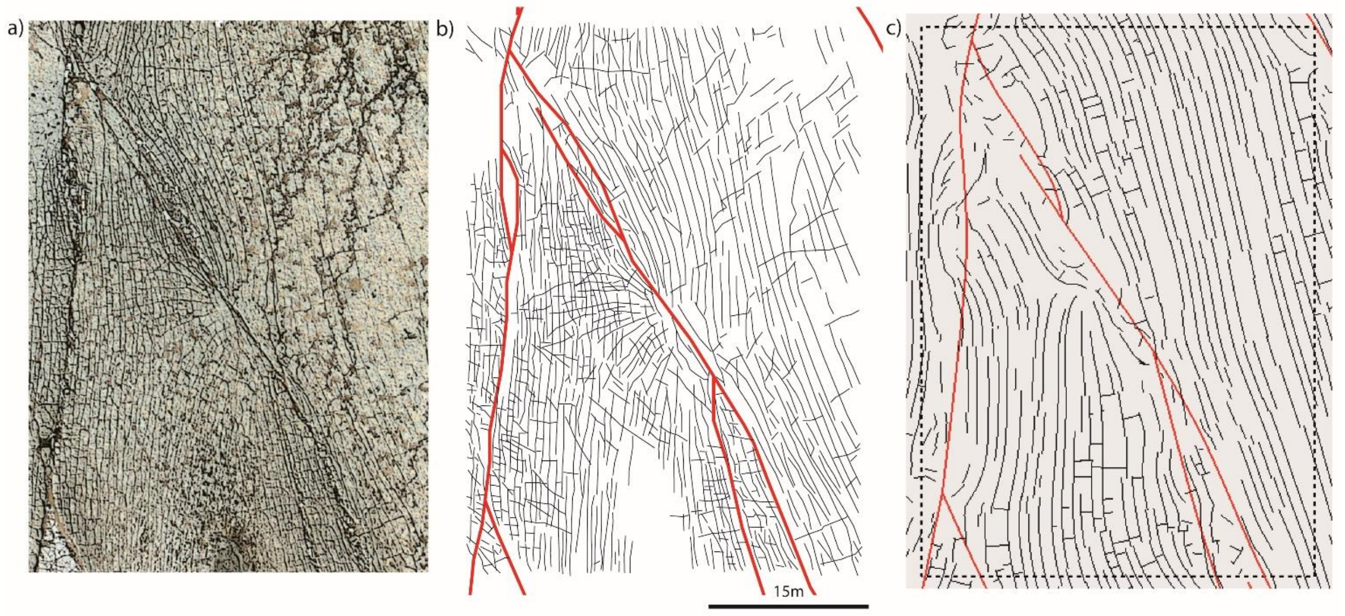


Figure 11: Comparison of the orthorectified aerial photo of the Nash Point outcrop from Maerten et al. (2016) (A), our own interpretation of the fracture patterns in this image (B), and the results of our best fit fracture model, with 0.5m bed thickness, a subcritical propagation index 24 and an initial microfracture density of 0.015r^{-2} fractures/m^3. Reproduced with kind permission from Figs 9.3 and 9.4 of Welch et al. (2020).

4.2.2 Robin Hood's Bay

The Robin Hood's Bay study also aimed to simulate the fracture networks developed in thin limestone beds surrounded by ductile shale, but in this case the strain was generated by a much larger normal fault, with c.150m throw (Rawnsley et al. 1992). Boundary element modelling showed lateral variability in the strain anisotropy as well as in the strain orientation, and this was reflected in both the modelled fracture networks and the fractures observed in outcrop: in areas where the horizontal strain was nearly isotropic, the fracture network was also isotropic, comprising two orthogonal fracture sets with a similar fracture density and size distribution; but in areas where the horizontal strain was strongly anisotropic, the fracture network was also anisotropic, comprising a primary set of long fractures connected by a secondary set of much shorter fractures that are terminated by the primary set (Fig. 12). The anisotropy of the fracture network is most probably an essential factor controlling fluid flow through a fractured layer in the subsurface. The controls on fracture network anisotropy are studied further in Welch et al. (2019).

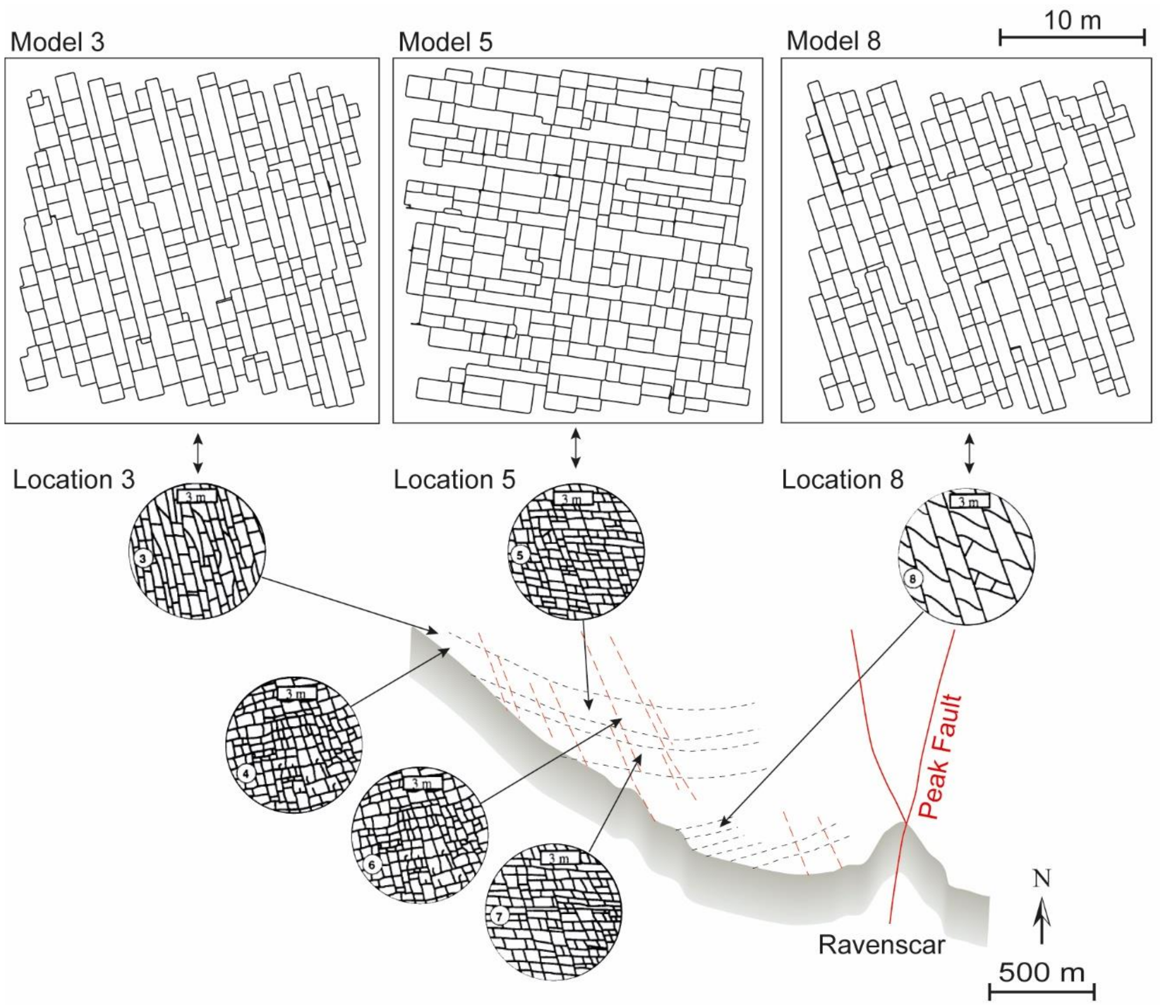
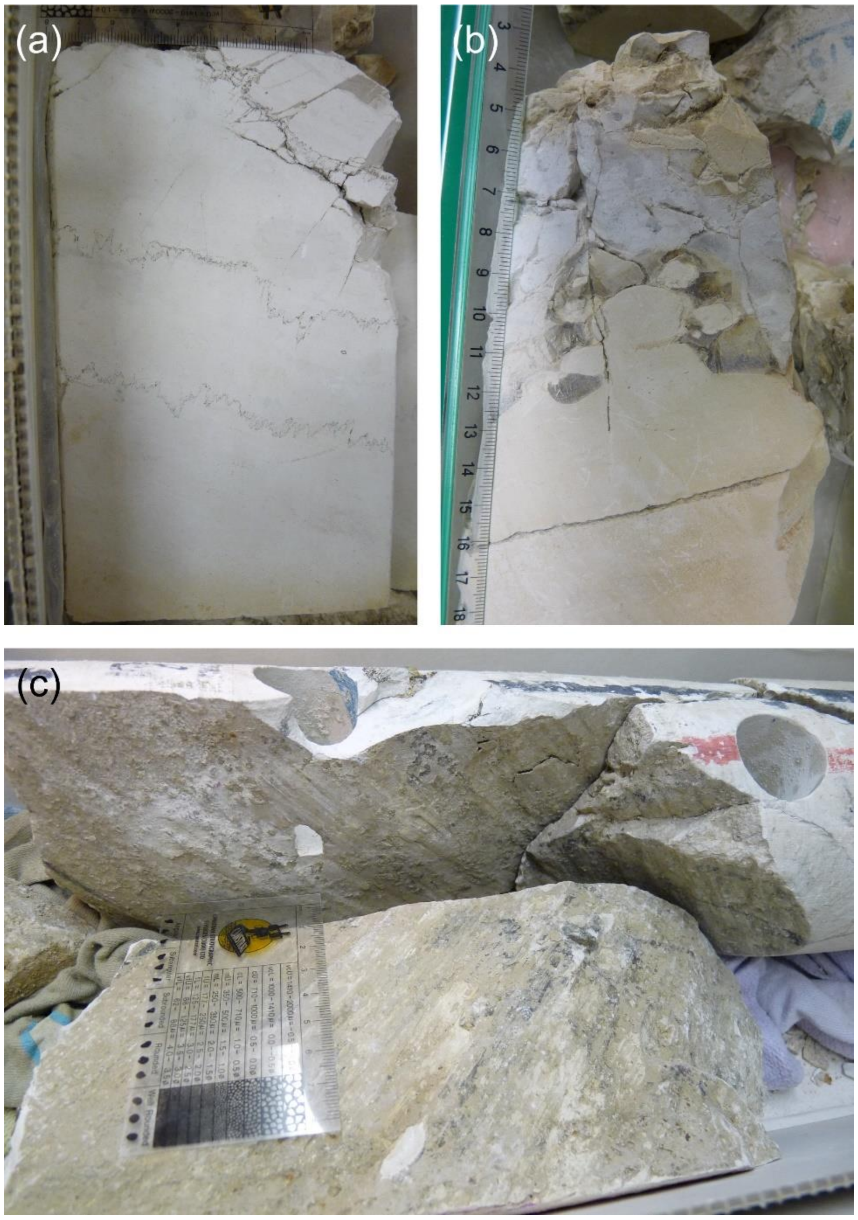


Figure 12: Comparison of three modelled fracture outputs compared to the fracture patterns observed by Rawnsley et al. (1992). Reproduced with kind permission from Fig. 9.9 of Welch et al. (2020).

4.2.3 Kraka field

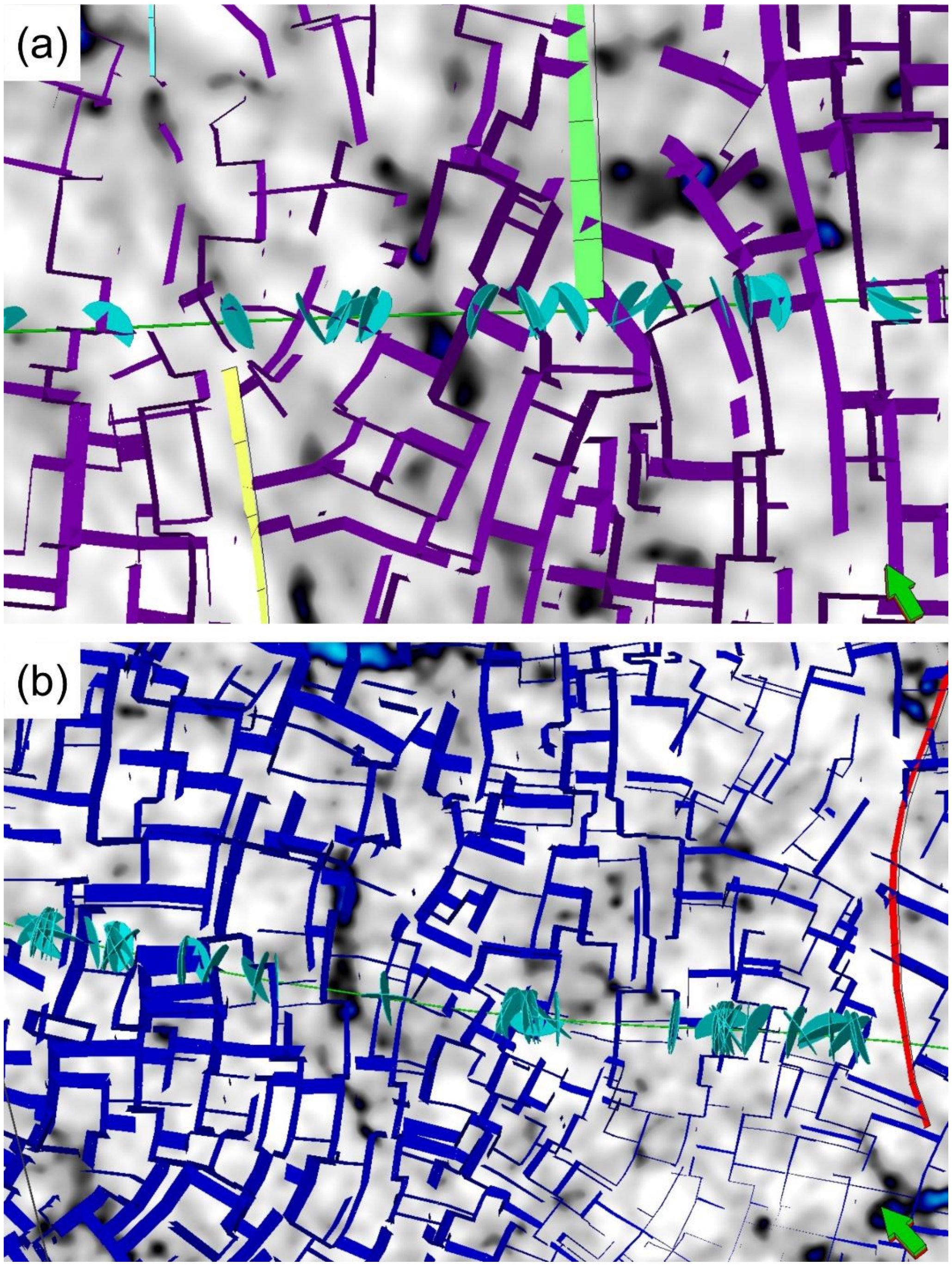
In the Kraka field, offshore Denmark, oil is produced from a fractured chalk layer (the Ekofisk Formation) overlying a large (km scale) salt pillow. Core from the Kraka field shows a population of small factures, typically c.2-20cm in size and often clustered around stylolites or chert nodules, and a few much larger open fractures with well-developed cement and/or slickensides, interpreted as major shear fractures that may be several hundred metres long (Fig. 13). Borehole image interpretation by Aabø et al. (2020) revealed two fracture sets: one striking NNE-SSW, parallel to a series of faults interpreted on seismic data, and the other striking parallel to the circumference of the salt diapir. The first set was inferred to be the result of ESE-WNW tectonic extension that also formed the seismic-scale faults, while the second set was inferred to be the result of strain induced by growth of the salt pillow.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022
© Author(s) 2022. CC BY 4.0 License.



810 **Figure 13: Small (cm-scale) microfractures developed around stylolites (a) and chert nodules (b) and large through-cutting open fracture with well-developed slickensides (c), observed in core from the Kraka oilfield, offshore Denmark. Reproduced with kind permission from Fig. 10.1 of Welch et al. (2020).**

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022
© Author(s) 2022. CC BY 4.0 License.



815 **Figure 14: Comparison of the modelled fracture network (dark blue and purple rectangles) with the resistive fractures observed on borehole images (light blue disks), from the Kraka oilfield, offshore Denmark. In (A) the fracture network was generated by a regional ESE-WNW extensional strain with local strain anomalies around the seismic-scale faults, which gives a better match to the borehole image data close to the faults (the yellow and green rectangles); in (B) the fracture network was generated by flexural strain developed as a result of salt pillow growth, which gives a better match to the borehole image data away from the faults. Reproduced with kind permission from Figs 10.6 and 10.7 of Welch et al. (2020).**

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022
© Author(s) 2022. CC BY 4.0 License.

Fracture models were therefore generated based on two strain inputs: a regional ESE-WNW extensional strain with local strain anomalies developed around the seismic-scale faults, modelled using the boundary element simulator build into the Petrel geomodelling package, and the flexural strain developed as a result of salt pillow growth, modelled using a backstripping algorithm developed by K. Petersen of Aarhus University (pers. comm.). The resulting fractures networks gave a good match with the resistive fractures and fracture clusters observed on borehole images, which were interpreted to correspond with the large shear fractures observed in core. Close to the seismic-scale faults, the best match was obtained with the fracture network based on the local strain anomalies around the faults, but away from the seismic-scale faults, the best match was obtained with the fracture network based on flexural strain due to salt pillow growth (Fig. 14).

4.2.4 Drenthe structure

The Drenthe structure, onshore NE Netherlands, is also a large salt diapir overlain by fractured chalk. It does not contain hydrocarbons, but has been considered as a potential site for CCS or for geothermal energy production. However much less data is available than for the Kraka field: the stratigraphy is only known from a few regional wells and there is no core or borehole image data to calibrate the modelled fracture network.

We therefore treated this as a pilot study for preliminary fracture modelling in an environment with extremely limited data. There is clearly insufficient data here to build a stochastic fracture model. However the DFN Generator can use the data that is available (the geometry of the structure and the stratigraphy) to quickly build geologically realistic fracture models. In this case, we identified potential brittle layers (chalk and sandstone) from the stratigraphic data, and assigned mechanical properties based on published data from similar lithologies. We used the curvature of the top surfaces of the fractured layers as a rough proxy for the flexural strain induced by growth of the salt diapir: areas of higher curvature are likely to correspond to areas of higher strain. We were able to generate multiple geologically consistent fracture models to be used in uncertainty analysis by outputting the results at intermediary stages in the simulation of fracture growth. Furthermore this could be done very rapidly, as is often required in preliminary assessments, and when more data becomes available the models can easily be rerun.

We now plan calibrating the resulting models by running simulations of groundwater flow and chemical diffusion through the fracture networks, to compare the results with observed surface hydrology data from the area.

5 Discussion and conclusions

The DFN Generator was developed for two purposes:

- As a research tool, to study the processes of natural fracture evolution, and to understand how the geometry of natural fracture networks is controlled by the geology.
- For practical application, to model and predict natural fracture networks that are important in various industrial settings, such as hydrogeology, geothermal energy, hydrocarbon production, carbon sequestration, and mine and tunnel engineering.

Some of the results from our research into natural fracture evolution using the DFN Generator code are described in Welch et al. (2019) and Welch et al. (2020), and we plan to publish further work using the code to study the growth of strike-slip and polygonal fracture networks in the near future. However, we believe that there is scope for further research along these lines, and we would welcome input from other researchers interested in using the code.

We also believe there is scope for the DFN Generator to be used widely for industrial applications. This may require the creation of additional interfaces to allow the code to interact with other geomodelling packages, for example creating a plug-in for a GIS-based package. It might also require enhancements to the algorithm or additional functionality to generate specific output data, or to tackle specific problems or scenarios. We would welcome feedback and guidance on this from end users.

DFN Generator is an ongoing project, and we are planning several major developments to the code. These involve significant effort to develop new modelling algorithms and to evaluate them against outcrop or subsurface data. Proposed developments include:

- Modelling of fracture sets striking oblique to the principal horizontal stress directions: This will allow the simulation of strike-slip and polygonal fracture sets, and also allow calculation of full 3D bulk rock stiffness and compliance tensors. This functionality is in fact already implemented experimentally in v2.0.0 of the DFN Generator code, but has yet to be fully verified and evaluated against observed fracture networks. When validation work is complete, we will publish a full description of this functionality in a separate paper.
- Improved calculation of fracture permeability tensors: To obtain maximum value from the DFN Generator in subsurface flow modelling applications, better methods of calculating the fracture permeability tensor are required, using all the data available from the implicit or explicit fracture models. Current methods such as Oda's method (Oda 1984, Dershowitz et al. 2000) do not fully account for the effects of fracture length, anisotropy and connectivity on the permeability tensor. Previously this has not been a major problem as such information is not usually available in the subsurface; however it can be obtained from the DFN Generator simulations. One method of incorporating this information is to use flow simulation on small-scale or idealised DFNs to derive an empirical relationship between these various geometric parameters and the fracture permeability tensor (e.g. Andrianov 2021, Andrianov & Nick 2021).
- Improved calculation of bulk rock stiffness and compliance tensors: Fracture networks can significantly impact the elastic behaviour of host rock layers, and lead to a significant modification of the stiffness and compliance tensors, which may become much more anisotropic as well as weaker. This will be important in many geomechanical applications such as mine or tunnel stability, surface subsidence and natural hazard prediction. The additional information on fracture length, anisotropy and connectivity available from the DFN Generator should enable better prediction of the bulk rock stiffness and compliance tensors for fractured layers.
- Modelling the propagation of fluid-driven fractures through impermeable layers: This would have applications in seal integrity analysis for hydrocarbon prospects and potential CCS reservoirs, as well as in hydrogeology and pollutant monitoring. Fracture propagation driven by fluid overpressure is a key risk to seal integrity in all these cases.

Currently, the DFN Generator algorithm is optimised to model the growth of fractures nucleating within a layer in response to a tectonic strain over geological time, and it would require some minor adaptations to model the growth of fluid-driven fractures nucleating at the base of the layer over much shorter timescales (hours to decades).

- Modelling the propagation of fractures through mechanically layered sections: Currently, the DFN Generator algorithm models only layer-bound fracture networks, and the user must define the brittle layers prior to running the models. This is not a problem in many scenarios, where the majority of fractures are layer-bound and the brittle layers are clearly defined. However in some scenarios, through-cutting fractures also develop which can provide fluid conduits between otherwise isolated reservoir layers. Moreover, studies of the early development of large fault zones in mechanically layered strata suggest that these initially develop as independent fractures in the more brittle layers, which then propagate across the ductile layers and link up to form incipient fault zones (Childs et al. 1996, Wilkins and Gross 2002). An ability to model the nucleation and growth of fractures across mechanical boundaries would therefore help in understanding the evolution of major faults. To do this, we need to extend LEFM theory to heterogeneous materials; this will typically require some approximation but it is possible to derive empirical or numerical expressions to model fracture propagation across mechanical layer boundaries (see e.g. Rijken & Cooke 2001, Welch et al 2009). We are currently working on this and we plan to publish our results in the near future.

Code availability

The source code, user manuals and technical notes for the DFN Generator software are hosted on Github at https://github.com/JointFlow/DFNGenerator. The Github project also includes files for the validation models described in Section 4.1 and illustrated in Fig. 10. DFN Generator v2.0.0.1, used in this work, is archived on Zenodo with doi 10.5281/zenodo.5862827.

Author contributions

MW and ML developed the DFN Generator algorithm and source code, and SO carried out much of the outcrop and subsurface evaluation. MW wrote this manuscript with contributions from ML and SO.

Competing interests

The authors declare they have no competing interests.

Acknowledgements

The developers kindly acknowledge the Danish Underground Consortium (TotalEnergies E&P Denmark, Noreco & Nordsøfonden) for granting the permission to publish this work. This research has received funding from the Danish Hydrocarbon Research and Technology Centre (DHRTC) under the AWF Improved Recovery programme.

References

Aabø, T.M., Dramsch, J.S., Würtzen, C.L., Seyum, S., Welch, M.: An integrated workflow for fracture characterization in chalk reservoirs, applied to the Kraka Field. Marine and Petroleum Geology, 112, 104065, doi: 10.1016/j.marpetgeo.2019.104065, 2019.

Andrianov, N.: Upscaling of Realistic Discrete Fracture Simulations Using Machine Learning. SPE203962, presented at SPE Reservoir Simulation Conference, On Demand, October 2021. doi: 10.2118/203962-MS, 2021.

Andrianov, N., Nick, H.M. Machine learning of dual porosity model closures from discrete fracture simulations. Advances in Water Resources, 147, 103810. doi: 10.1016/j.advwatres.2020.103810, 2021.

Atkinson, B.K.: Subcritical crack growth in geological materials. Journal of Geophysical Research, 89, B6, 4077-4114, doi: 10.1029/jb089ib06p04077, 1984.

Childs, C., Nicol, A., Walsh, J.J., Watterson, J.: Growth of vertically segmented normal faults. Journal of Structural Geology, 18, 1389-1397, doi: 10.1016/s0191-8141(96)00060-0, 1996.

Dershowitz, B., LaPointe, P., Eiben, T., Wei, L.: Integration of discrete feature network methods with conventional simulator approaches. SPE49069, presented at SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, doi: 10.2118/62498-pa, 2000.

Ferrill, D.A., Morris, A.P.: Displacement gradient and deformation in normal fault systems. Journal of Structural Geology, 23, 619-638, doi: 10.1016/s0191-8141(00)00139-5, 2001.

Griffith, A.A.: The phenomena of rupture and flow in solids. Philosophical Transactions of the Royal Society, A221, 163-197, doi: 0.1098/rsta.1921.0006, 1921.

Huggins, P., Watterson, J., Walsh, J.J., Childs, C.: Relay zone geometry and displacement transfer between normal faults recorded in coal-mine plans. Journal of Structural Geology, 17, 1741-1755, doi: 10.1016/0191-8141(95)00071-k,1995.

Maerten, L., F. Maerten, M. Lejri, and Gillespie, P.: Geomechanical paleostress inversion using fracture data. Journal of Structural Geology, 89, 197–213, doi: 10.1016/j.jsg.2016.06.007, 2016.

Nicol, A., Walsh, J.J., Watterson, J., Bretan, P.G.: Three-dimensional geometry and growth of normal faults. Journal of Structural Geology, 17, 847-862, doi: 10.1016/0191-8141(94)00109-d, 1995.

Oda, M.: Permeability Tensor for Discontinuous Rock Masses. Geotechnique, 35, 483-495, doi: 10.1680/geot.1985.35.4.483, 1984.

https://doi.org/10.5194/gmd-2022-22
Preprint. Discussion started: 19 May 2022
© Author(s) 2022. CC BY 4.0 License.

Olson, J.E.: Joint pattern development: Effects of subcritical crack growth and mechanical crack interaction. Journal of Geophysical Research, B98, 12251-12265, doi: 10.1029/93jb00779, 1993.

Olson, J.E., Predicting fracture swarms - the influence of subcritical crack growth and the crack-tip process zone on joint spacing in rock. In: Cosgrove, J.W. & Engelder, T. (eds), The Initiation, Propagation and Arrest of Joints and Other Fractures. Geological Society, London, Special Publications, 231, 73-88, doi: 10.1144/gsl.sp.2004.231.01.05, 2004.

Rawnsley, K.D., Rives, T., Petit, J.-P., Hencher, S.R., Lumsden, A.C.: Joint development in perturbed stress fields near faults. Journal of Structural Geology, 14, 939-951, doi: 10.1016/0191-8141(92)90025-r, 1992.

Rijken, P., Cooke, M.L.: Role of shale thickness on vertical connectivity of fractures: application of crack-bridging theory to the Austin Chalk, Texas. Tectonophysics, 337, 117-133, doi: 10.1016/s0040-1951(01)00107-x, 2001.

Sack, R.A.: Extension of Griffith's theory to three dimensions. Proceedings of the Physical Society, London, 58, 729-736, doi: 10.1088/0959-5309/58/6/312, 1946.

Sneddon, I.N.: The distribution of stress in the neighbourhood of a crack in an elastic solid. Proceedings of the Royal Society, A187, 229-260, doi: 10.1016/0020-7225(65)90028-5, 1946.

Souque, C., Knipe, R.J., Davies, R.K., Jones, P., Welch, M.J., Lorenz, J.: Fracture corridors and fault reactivation: Example from the Chalk, Isle of Thanet, Kent, England. Journal of Structural Geology, 122, 11-26, doi: 10.1016/j.jsg.2018.12.004, 2019.

Swanson, P.L.: Subcritical crack growth and other time- and environment-dependent behaviour in crustal rocks. Journal of Geophysical Research, 89, 4137–4152, doi: 10.1029/jb089ib06p04137, 1984.

Walsh, J.J., Bailey, W.R., Childs, C., Nicol, A., Bonson, C.G.: Formation of segmented normal faults: a 3-D perspective. Journal of Structural Geology, 25, 1251-1262, doi: 10.1016/s0191-8141(02)00161-x, 2003.

Welch, M.J., Davies, R.K., Knipe, R.J., Tueckmantel, C.: A dynamic model for fault nucleation and propagation in a mechanically layered section. Tectonophysics, 474, 473-492, doi: 10.1016/j.tecto.2009.04.025, 2009.

Welch, M.J., Lüthje, M., Glad, A.C.: Influence of fracture nucleation and propagation rates on fracture geometry: Insights from geomechanical modelling. Petroleum Geoscience, 25, 470-489, doi:10.1144/petgeo2018-161, 2019.

Welch, M.J., Lüthje, M., Oldfield, S.J.: Modelling the Evolution of Natural Fracture Networks. Springer Nature Switzerland AG., doi:10.1007/978-3-030-52414-2, 2020.

Welch, M.J., Lüthje, M.: DFN Generator: Technical Notes. Documentation provided with the DFN Generator software on Github, https://github.com/JointFlow/DFNGenerator/blob/main/Documentation/DFNGenerator_TechNotes.pdf, 2021.

Westaway, R.: Quanititative analysis of population of small faults. Journal of Structural Geology, 16, 1259-1273, doi: 10.1016/0191-8141(94)90068-x, 1994.

Wilkins, S.J., Gross, M.R.: Normal fault growth in layered rocks at Split Mointain, Utah: influence of mechanical stratigraphy on dip linkage, fault restriction and fault scaling. Journal of Structural Geology, 24, 1413-1429, doi: 10.1016/s0191-8141(01)00154-7, 2002.