DFN Generator standalone version Installation and Use guide

Michael Welch, Mikael Lüthje and Simon Oldfield

14/09/2021

Installation and launch

The standalone version of DFN Generator is designed to be run "as is", without requiring any installation, specialist libraries or third-party software. It is entirely self-contained within the executable file *DFNGenerator.exe*. This file can be obtained in one of two ways:

- If you do not have a C Sharp compiler or do not wish to make any changes in the code, you can download a precompiled copy of *DFNGenerator.exe* from the same repository as the source code. It is contained in the zip file *DFNGenerator_StandaloneProgram.zip*.
- If you have made changes to the DFN Generator code, you should recompile the code using a standard C Sharp compiler. The code should be compiled in release mode with the `READINPUTFROMFILE` preprocessor flag switched on (uncomment the line `#define READINPUTFROMFILE` in the Program.cs file) and all other preprocessor flags switched off (comment out other define lines, e.g. `//#define DEBUG_FRACS`). A new copy of *DFNGenerator.exe* will be generated in *DFNGenerator_Code**DFNGenerator_Standalone**bin**Release*.

To run a model, the *DFNGenerator.exe* file should simply be copied into the folder in which you wish to save the model output. The standalone version of DFN Generator is run from the command line, so you will need to open a Command Prompt window and navigate to this folder. If you have already prepared an input text file (or files), the model can be run by typing

```
C:\>DFNGenerator inputfilename
```

A new subfolder called *inputfilename* will be created, and all output files will be written to this. Depending on the specifications in the input file, the output files will include:

- One file for each cell in the model containing implicit fracture data, with a name such as *ImplicitData_X0_Y0.txt*. These files are identified by the coordinates of the southeastern corner of the cell.
- Two files containing the layer-bound macrofractures and the microfracture of the final explicit DFN, named *Macrofractures_final.txt* and *Microfractures_final.txt* respectively. If microfractures are not included in the explicit DFN, the *Microfractures_final.txt* file will be empty.
- Files containing the layer-bound macrofractures and the microfracture of each intermediate stage of the explicit DFN, if intermediate stages are to be output. These will be named *Macrofractures_StageN_TimeT.txt* and *Microfractures_StageN_TimeT.txt* respectively.

If the input file has not yet been prepared, you can launch the DFN Generator without specifying an input file by typing

```
C:\>DFNGenerator
```

A template input file called *DFNGenerator_configuration.txt* will be created in the folder containing the DFN Generator executable. You can then edit this with a text editor to set the required values, as

described below, before pressing Enter to run the model. The output files will be written directly into the folder containing the DFN Generator executable.

Alternatively, a copy of the *DFNGenerator_configuration.txt* template input file is included in the zip file *DFNGenerator_StandaloneProgram.zip*, which can be downloaded from the same repository as the source code. Save a copy of this with the name you wish to give the model, and then edit it with a text editor to set the required values, as described below. The models can then be run from the command prompt by specifying the input file name, as described above.

The zip file *DFNGenerator_StandaloneProgram.zip* also contains configuration files for a selection of example models, illustrating variety of different fracture geometries that can be generated by varying the input parameters. Some of these models are discussed in Welch et al. 2019 and Welch et al. 2020.

Preparing the input

The input data for the standalone version of DFN Generator is mostly contained in a single text file, although there can also be a number of associated Include files. All these files follow the same simple format:

- Lines beginning with the % character are comment lines and are ignored by DFN Generator
- Lines containing data follow the format

```
Keyword datavalue [datavalue 2] [datavalue 3]…
```

- Only one data item can be specified per line
- All data must be specified in SI units except strain rates and strain relaxation time constants, which can be specified in seconds, years or ma.

Specifying the deformation and mechanical properties

By default, models comprise a planar horizontal brittle layer of uniform thickness, which is split into a rectangular grid of cells (or gridblocks) with identical geometry and properties. The geometry, applied deformation, mechanical properties and initial stress state of these gridblocks are specified in the main body of the input file, using the following keywords:

- **NoRows, NoCols:** specify the grid size.
- **Width_EW ,Length_NS:** specify the gridblock size in metres.
- **LayerThickness:** specifies the layer thickness in metres.
- **OriginXOffset, OriginYOffset:** set the absolute XY coordinates of the SW corner of the bottom left gridblock.
- **Depth:** specifies the current depth of burial in metres, positive downwards. This is also assumed to be the depth of burial at the time of deformation, unless **DepthAtDeformation** is specified (see below).
- **EhminAzi:** azimuth of minimum applied horizontal strain (i.e. direction of maximum extension). Must be specified in radians, clockwise from North.
- **VariableStrainOrientation, EhminCurvature:** Set **VariableStrainOrientation** to *false* to have the same minimum strain orientatation in all cells. Set **VariableStrainOrientation** to *true* to have laterally variable strain orientation controlled by **EhminCurvature**, which gives the difference in strain orientation between adjacent blocks in radians
- **EhminRate, EhmaxRate:** specify minimum and maximum applied horizontal strain rates. Note that, by convention, extensional strain is represented by negative values, and **EhminRate** is the most negative (i.e. most tensile) horizontal strain rate. **EhminRate** must therefore be negative in order to generate fractures. **EhmaxRate** should be set to 0 for uniaxial strain, to between 0 and EhminRate for an anisotropic fracture network, and equal to EhminRate for an isotropic fracture network. The units are determined by the specified **ModelTimeUnits** (i.e. strain/s, strain/year or strain/ma).
- **VariableStrainMagnitude:** Set to true to add random variation to the input strain rates for individual gridblocks. Strain rates for each gridblock will vary randomly from 0 to 2x specified values.
- **ModelTimeUnits:** Set time units to ma, year or second.
- **YoungsMod:** Young's Modulus; must be specified in Pa (note NOT GPa).
- **VariableYoungsMod:** Set to true to add random variation to the Young's Modulus for individual gridblocks.
- **PoissonsRatio:** specifies Poisson's ratio.

- **BiotCoefficient:** specifies the Biot coefficient.
- **FrictionCoefficient:** specifies the coefficient of internal friction.
- **VariableFriction:** Set to true to add random variation to the friction coefficient for individual gridblocks.
- **CrackSurfaceEnergy:** Specifies the crack surface energy required to propagate fractures in J/m^2.
- **VariableCSE:** Set to true to add random variation to the crack surface energy for individual gridblocks.
- **RockStrainRelaxation:** Time constant controlling the rate of viscoelastic strain relaxation in the rock matrix; elastic strain will reduce to 1/e of its initial value during this time (time units are determined by **ModelTimeUnits** setting). Set to 0 to turn off viscoelastic strain relaxation. Strain relaxation controls the rate of stress accumulation: with no strain relaxation, the horizontal stress will increase at a constant rate proportional to the horizontal strain rate; with strain relaxation, the horizontal stress will increase initially but then level off at a magnitude proportional to the ratio of strain rate to strain relaxation time constant.
- **FractureRelaxation:** Time constant controlling the rate of viscoelastic strain relaxation around the fractures; elastic strain around the fractures only will reduce to 1/e of its initial value during this time (time units are determined by **ModelTimeUnits** setting). To apply viscoelastic strain relaxation around the fractures only, set **RockStrainRelaxation** to 0 and **FractureRelaxation** to >0.
- **InitialMicrofractureDensity, InitialMicrofractureSizeDistribution:** The initial seed microfractures are assumed to follow a fractal distribution. **InitialMicrofractureDensity** specifies the density of initial seed microfractures, and **InitialMicrofractureSizeDistribution** specifies the size distribution of initial microfractures - increase this for larger ratio of small:large initial microfractures
- **SubcriticalPropIndex:** Specifies the fracture propagation rate. Set to <5 for slow subcritical propagation, 5-15 for intermediate, >15 for rapid critical propagation.
- **CriticalPropagationRate:** Specifies the critical fracture propagation rate in m/s. This is typically near the velocity of sound in the rock.
- **StressDistributionScenario:** Use this to turn on or off stress shadow effects. Options are *EvenlyDistributedStress* (stress shadows off) or *StressShadow* (stress shadows on). Do not use *DuctileBoundary* as this is not yet implemented.
- **DepthAtDeformation:** Depth of burial at the time of deformation (in metres, positive downwards) - used to calculate vertical effective stress. If **DepthAtDeformation** is not specified or is set to <=0, the current depth of burial will be used to calculate vertical effective stress.
- **MeanOverlyingSedimentDensity:** Mean bulk density of overlying sediments and fluid in kg/m^3. Used to calculate vertical effective stress.
- **FluidDensity:** Mean pore fluid density in kg/m^3. Used to calculate vertical effective stress.
- **InitialOverpressure:** Fluid overpressure in Pa. Used to calculate vertical effective stress.
- **InitialStressRelaxation:** Controls the initial horizontal stress, prior to the application of horizontal strain. Set **InitialStressRelaxation** to 1 to have initial horizontal stress = vertical stress (viscoelastic equilibrium). Set **InitialStressRelaxation** to 0 to have initial horizontal stress = v/(1-v) * vertical stress (elastic equilibrium). Set **InitialStressRelaxation** to -1 for initial horizontal stress = Mohr-Coulomb failure stress (critical stress state).

Specifying the output and calculation control parameters

This is followed by sections describing the outputs and control parameters for the model as a whole. These can be left at their default values for most models; however it may be useful to adjust these to simulate certain specific scenarios or to generate non-standard outputs. This can be done using the following keywords:

- **WriteImplicitDataFiles, WriteDFNFiles:** Set to true to write implicit fracture data and the explicit DFN data to file. If these are set to false, no output will be generated.
- **OutputDFNFileType:** Output file type for explicit DFN data. This can be in ASCII or FAB format. ASCII files can be read by the data analysis spreadsheets provided with DFN Generator, while FAB files can be loaded directly into Petrel and other geomodelling packages.
- **NoIntermediateOutputs:** Set this to generate DFNs representing intermediate stages of the growth of the fracture network; this specifies the number of intermediate stages to output. If set to 0, only the final DFN will be output.
- **OutputAtEqualTimeIntervals:** Controls the interval between the intermediate stage DFNs specified by **NoIntermediateOutputs**. If set to true, DFNs will be output at equal intervals of time; if set to false, DFNs will be output at approximately regular increments in the total fracture area.
- **OutputCentrepoints:** Set this to true to output the macrofracture centrelines as polylines. These will be written to a separate ASCII format text file.
- **OutputComplianceTensor:** Set this to true to output the bulk rock compliance tensor for the final fracture network. This data will be added to the implicit output file for each gridblock. Note that if stress shadows are active (i.e. **StressDistributionScenario** is set to *StressShadow*), the bulk rock compliance tensor will be equal to the isotropic intact rock compliance tensor; to see the effect of fractures on the bulk rock compliance tensor, it is necessary to switch off the stress shadows (i.e. set **StressDistributionScenario** to *EvenlyDistributedStress*).
- **CalculateFracturePorosity:** Set to true to calculate and output fracture porosity. This data will be added to the implicit output file for each gridblock. Aperture values will also be calculated and set for the fractures in the explicit DFN.
- **FractureApertureControl:** Use this to determine the method used to determine fracture aperture, used in porosity and permeability calculation. It can be set to *Uniform*, *SizeDependent*, *Dynamic*, or *BartonBandis*. Depending on the aperture control setting, the following parameters are used to calculate fracture aperture:
  - *Uniform* fracture aperture: All fractures are assigned an arbitrary user-specified aperture. Different apertures can be specified for fractures with different orientations.
    - **Mode1HMin_UniformAperture:** Specifies the aperture for Mode 1 fractures striking perpendicular to the minimum horizontal strain, in metres.
    - **Mode2HMin_UniformAperture:** Specifies the aperture for Mode 2 fractures striking perpendicular to the minimum horizontal strain, in metres.
    - **Mode1HMax_UniformAperture:** Specifies the aperture for Mode 1 fractures striking perpendicular to the maximum horizontal strain, in metres.
    - **Mode2HMax_UniformAperture:** Specifies the aperture for Mode 2 fractures striking perpendicular to the maximum horizontal strain, in metres.
  - *SizeDependent* fracture aperture: Fracture aperture is proportional to the minimum fracture dimension (fracture diameter for microfractures, layer thickness for layer-bound fractures). Different scaling factors can be defined for fractures with different orientations.
    - **Mode1HMin_SizeDependentApertureMultiplier:** Size-dependent aperture multiplier for Mode 1 fractures striking perpendicular to the minimum horizontal strain. The layer-bound fracture aperture is given by layer thickness times this multiplier.
    - **Mode2HMin_SizeDependentApertureMultiplier:** Size-dependent aperture multiplier for Mode 2 fractures striking perpendicular to the minimum horizontal strain. The layer-bound fracture aperture is given by layer thickness times this multiplier.

    - **Mode1HMax_SizeDependentApertureMultiplier:** Size-dependent aperture multiplier for Mode 1 fractures striking perpendicular to the maximum horizontal strain. The layer-bound fracture aperture is given by layer thickness times this multiplier.
    - **Mode2HMax_SizeDependentApertureMultiplier:** Size-dependent aperture multiplier for Mode 2 fractures striking perpendicular to the maximum horizontal strain. The layer-bound fracture aperture is given by layer thickness times this multiplier
  - *Dynamic* fracture aperture: Calculates the equilibrium elastic aperture for dilatant fractures subject to a tensile normal stress. An arbitrary user-defined multiplier can also be applied. The calculation is based on the in situ stress state at the end of deformation; NB if this is compressive (e.g. for Mode 2 shear fractures), the resulting fracture porosity will be 0.
    - **DynamicApertureMultiplier:** Specifies a multiplier for dynamic aperture.
  - *BartonBandis* model for fracture aperture: Calculates the aperture for shear fractures subject to a compressive normal stress using the Barton-Bandis formula (Bandis et al. 1983, Int J Rock Mech, Min Sci & Geomech Abs 20, 249-268). The calculation is based on the in situ stress state at the end of deformation and various parameters related to the fracture morphology, defined in the paper.
    - **JRC:** Joint Roughness Coefficient.
    - **UCSRatio:** Ratio of unconfined compressive strength of unfractured rock to fractured rock.
    - **InitialNormalStress:** Reference value for the normal strength on fracture, at which the fracture normal stiffness was measured, in Pa.
    - **FractureNormalStiffness:** Stiffness normal to the fracture, at initial normal stress, in Pa m.
    - **MaximumClosure:** Maximum fracture closure in metres.
- **NoFractureSets:** Specifies the number of fracture sets in the model. Set to 2 to generate two orthogonal fracture sets, perpendicular to the minimum and maximum horizontal strain directions; this is typical of a single stage of tectonic deformation in intact rock. Increase this to model polygonal or strike-slip fractures, or multiple deformation episodes (where there are pre-existing fractures oblique to the principal horizontal stresses); 6 fracture sets (oriented at 30° intervals from the minimum horizontal strain direction) generally allows accurate representation of polygonal and strike-slip fractures with manageable runtimes.
- **Mode1Only, Mode2Only:** Set these to *true* to force only Mode 1 (dilatant) or only Mode 2 (shear) fractures; otherwise model will include both, depending on which is energetically optimal.
- **CheckAlluFStressShadows:** Controls whether to check microfractures against stress shadows of all macrofractures, or only macrofractures in the same set. Can be set to *None*, *All* or *Automatic*.
- **AnisotropyCutoff:** Specifies cutoff value to use the isotropic method for calculating cross-fracture set stress shadow and exclusion zone volumes.
- **AllowReverseFractures:** Controls whether reverse fractures are allowed in the fracture network. If set to false, fracture dipsets with a reverse displacement vector will not be allowed to accumulate displacement or grow.
- **MaxTimestepDuration:** Specifies maximum duration for individual timesteps; set to -1 for no maximum timestep duration. Time units are determined by **ModelTimeUnits** setting.
- **MaxTimestepMFP33Increase:** Specifies maximum increase in fracture density allowed in each timestep (defined by $_{MF}P_{33}$). This controls the optimal timestep duration; increase this to run the calculation faster, with fewer but longer timesteps.
- **MinImplicitMicrofractureRadius:** Specifies the minimum radius for microfractures to be included in implicit fracture density and porosity calculations (in metres). If this is set to 0 (i.e. include all microfractures) then it will not be possible to calculate the volumetric microfracture density $_{\mu F}P_{30}$

as this will be infinite. If this is set to -1 the maximum radius of the smallest bin will be used (i.e. exclude the smallest bin from the microfracture population).

- **No_r_bins:** Specifies the number of bins used in numerical integration of $_{\mu F}P_{32}$. This controls accuracy of numerical calculation of microfracture populations; increase this to increase accuracy of the numerical integration at expense of runtime.
- **CalculatePopulationDistribution:** Set to true to calculate implicit fracture population distribution functions. This data will be added to the bottom of the implicit output file for each gridblock
- **No_l_indexPoints:** Specifies the number of macrofracture length values to calculate for each of the implicit fracture population distribution functions.
- **MaxHMinLength, MaxHMaxLength:** Specify the range of macrofracture lengths to calculate for the implicit fracture population distribution functions for fractures striking perpendicular to minimum and maximum horizontal strain direction respectively. Set these values to the approximate maximum length of fractures generated (in metres); if this is not known, set these to 0 to default to the maximum potential fracture length; however this may be much greater than actual maximum fracture length.
- Calculation termination controls: The calculation is set to stop automatically when fractures stop growing. This can be defined in one of three ways:
  - When the total volumetric ratio of active (propagating) half-macrofractures ($^{a}_{MF}P_{33}$) drops below a specified proportion of the peak historic value
  - When the total volumetric density of active (propagating) half-macrofractures ($^{a}\underline{_{MF}}P_{30}$) drops below a specified proportion of the total (propagating and non-propagating) volumetric density ($\underline{_{MF}}P_{30}$)
  - When the total clear zone volume (the volume in which fractures can nucleate without falling within or overlapping a stress shadow) drops below a specified proportion of the total volume

  Increase these cutoffs to reduce the sensitivity and stop the calculation earlier. This prevents a long calculation tail - i.e. late timesteps where fractures have stopped growing so they have no impact on fracture populations, just increase runtime. To stop calculation while fractures are still growing reduce the deformation stage duration or maximum number of timesteps.

  - **Current_HistoricMFP33TerminationRatio:** Specifies the ratio of current to peak active macrofracture volumetric ratio at which fracture sets are considered inactive. Set to a negative value to switch off this control.
  - **Active_TotalMFP30TerminationRatio:** Specifies the ratio of active to total macrofracture volumetric density at which fracture sets are considered inactive. Set to a negative value to switch off this control.
  - **MinimumClearZoneVolume:** Specifies the minimum required clear zone volume in which fractures can nucleate without stress shadow interactions (as a proportion of total volume). If the clear zone volume falls below this value, the fracture set will be deactivated.
  - **DeformationStageDuration:** Specifies the total deformation stage duration. Time units are determined by **ModelTimeUnits** setting. Set to -1 to continue running the model until fracture saturation is reached.
  - **MaxTimesteps:** Specifies the maximum number of timesteps; if this is exceeded the calculation will stop. Use this to prevent excessively long runtimes.
- **GenerateExplicitDFN:** Set to true to generate explicit DFN; set to false to generate only implicit fracture data.
- **CropAtBoundary:** Set to true to crop explicit fractures at the outer boundary of the grid; set to false to allow fractures to propagate outside of the outer grid boundary.

- **LinkStressShadows:** Set to true to link fractures that terminate due to stress shadow interaction into one long fracture, via a relay segment.
- **MaxConsistencyAngle:** Specifies the maximum variation in fracture propagation azimuth allowed across a gridblock boundary (in radians). If the orientation of the fracture set varies across the gridblock boundary by more than this, the algorithm will seek a better matching set.
- **MinimumLayerThickness:** Specifies the minimum layer thickness cutoff (in metres); the explicit DFN will not be calculated for gridblocks thinner than this. Use this to prevent the generation of excessive numbers of fractures in very thin gridblocks where there is geometric pinch-out of the layers.
- **ProbabilisticFractureNucleationLimit:** Specifies the minimum limit for fracture nucleation to be controlled probabilistically. By default, fractures nucleate deterministically at regular intervals determined by the microfracture density and growth rate. However if the gridblocks are small relative to the fracture spacing, so the average number of fractures per gridblock is less than 1, no fractures will nucleate. This control allows the timing fracture nucleation to be controlled probabilistically, if the number of fractures nucleating per timestep is less than the specified value. This will allow some fractures to nucleate even when gridblocks are small. Set to 0 to disable probabilistic fracture nucleation. Set to -1 for automatic: probabilistic fracture nucleation will be activated whenever searching neighbouring gridblocks is also active, and if **SearchNeighbouringGridblocks** is also set to automatic, this will be determined independently for each gridblock based on the gridblock geometry. NB If probabilistic fracture nucleation is required to generate fractures, it is often because the height:width ratio of the gridblocks is too high. Try using horizontal upscaling to reduce this instead.
- **PropagateFracturesInNucleationOrder:** Controls the order in which fractures are propagated within each timestep. Set to true to propagate fractures in order of nucleation time regardless of fracture set; set to false to propagate fractures in order of fracture set. Propagating in strict order of nucleation time removes bias in fracture lengths between sets, but will add a small overhead to calculation time.
- **SearchNeighbouringGridblocks:** Controls whether to search adjacent gridblocks for stress shadow interaction. It can be set to *All*, *None* or *Automatic*; if set to *Automatic*, then searching adjacent gridblocks will be determined independently for each gridblock based on the gridblock geometry
- **MinExplicitMicrofractureRadius:** Specifies the minimum radius for microfractures to be included in explicit DFN (in metres). Set this to 0 to exclude microfractures from DFN; set to between 0 and half layer thickness to include larger microfractures in the DFN
- **Number_uF_Points:** Specifies the number of cornerpoints defining the microfracture polygons in the explicit DFN. Set to 0 to output microfractures as just a centrepoint and radius; set to 3 or greater to output microfractures as polygons defined by a list of cornerpoints.

Specifying more complex models

More complex models, with more realistic geometry and laterally variable properties, can be defined using gridblock overrides or include files:

- **Overrides** for individual gridblocks are added at the end of the main input file, and modify the geometry, applied deformation and/or mechanical properties of the gridblocks. They comprise a block of text for each gridblock that is to be modified. This block should be sandwiched between the keywords **Gridblock column_index row_index** and **End Gridblock**, each of which should occupy a separate line. Within the block, it is possible to specify overrides for specific properties which will apply only to the gridblock in question, using the line

```
Keyword datavalue
```

Only the following properties can be overridden: **EhminAzi**, **EhminRate**, **EhmaxRate**, **YoungsMod**, **PoissonsRatio**, **BiotCoefficient**, **CrackSurfaceEnergy**, **FrictionCoefficient**, **SubcriticalPropIndex**, **RockStrainRelaxation**, **FractureRelaxation**, **InitialMicrofractureDensity**, **InitialMicrofractureSizeDistribution**.

It is also possible to override the default position of the cornerpoints of the gridblocks using the line

```
Cornerpoint XCoord YCoord Zcoord
```

The cornerpoints are specified using the keywords **SETopCorner**, **SEBottomCorner**, **NETopCorner**, **NEBottomCorner**, **NWTopCorner**, **NWBottomCorner**, **SWTopCorner**, **SWBottomCorner**. Z coordinates should be specified positive downwards. NB the cornerpoints of adjacent gridblocks will automatically be adjusted when a gridblock cornerpoint is overridden, but there is no sanity check to ensure the resulting grid geometry is consistent (e.g. checking for negative gridblock volumes, etc.); this must be done by the user.

A typical override block will therefore be as follows:

```
Gridblock 1 2
  EhminAzi 0.112
  EhminRate -0.0154
  EhmaxRate 0.00025
  YoungsMod 5670000000
  SETopCorner 11.23 5.43 2005.3
  SEBottomCorner 11.23 5.43 2011.2
End Gridblock
```

Include files allow values for specific properties to be set individually for every gridblock. They also allow the model geometry to be defined in a grid format. To use an include file, simply add the line

```
Include Filename
```

at the end of the main input file. It is possible to call multiple include files, but each must be called from a separate **Include** statement in the main input file.

Include files to set specific property values should follow the format

```
#PropertyA
Gridblock_1_1_value Gridblock_2_1_value Gridblock_3_1_value
Gridblock_1_2_value Gridblock_2_2_value Gridblock_3_2_value
Gridblock_1_3_value Gridblock_2_3_value Gridblock_3_3_value
#PropertyB
Gridblock_1_1_value Gridblock_1_2_value Gridblock_1_3_value
Gridblock_1_2_value Gridblock_2_2_value Gridblock_3_2_value
Gridblock_1_3_value NA Gridblock_3_3_value
```

Include files can include values for multiple properties, in separate blocks. Each new property must start on a new line, with # followed by the property name. Within each property block, the values can be separated by either spaces or line returns. However a value must be specified for each gridblock, and the values must be given in order shown above, looping first through the grid rows and then through the grid columns (moving from west to east and from south to north). Use *NA* instead of a specifying a value to revert to the default value for a specific gridblock (as has been done for **PropertyB** in Gridblock 3,2 in the example above). The following properties can be set from an Include file: **EhminAzi**, **EhminRate**, **EhmaxRate**, **YoungsMod**, **PoissonsRatio**, **BiotCoefficient**, **CrackSurfaceEnergy**, **FrictionCoefficient**, **SubcriticalPropIndex**, **RockStrainRelaxation**, **FractureRelaxation**, **InitialMicrofractureDensity**, **InitialMicrofractureSizeDistribution**. Include files can also include comment lines, beginning with the % character; these are ignored by DFN Generator.

The format for an Include file specifying the model geometry is slightly different:

```
#Geometry

Gridblock_1_1_SWTopCornerpoint_X
Gridblock_1_1_SWTopCornerpoint_Y
Gridblock_1_1_SWTopCornerpoint_Z
Gridblock_1_1_SWBottomCornerpoint_X
Gridblock_1_1_SWBottomCornerpoint_Y
Gridblock_1_1_SWBottomCornerpoint_Z

Gridblock_2_1_SWTopCornerpoint_X
Gridblock_2_1_SWTopCornerpoint_Y
Gridblock_2_1_SWTopCornerpoint_Z
Gridblock_2_1_SWBottomCornerpoint_X
Gridblock_2_1_SWBottomCornerpoint_Y
Gridblock_2_1_SWBottomCornerpoint_Z

Gridblock_3_1_SWTopCornerpoint_X
Gridblock_3_1_SWTopCornerpoint_Y
Gridblock_3_1_SWTopCornerpoint_Z
Gridblock_3_1_SWBottomCornerpoint_X
Gridblock_3_1_SWBottomCornerpoint_Y
Gridblock_3_1_SWBottomCornerpoint_Z

Gridblock_3_1_SETopCornerpoint_X
Gridblock_3_1_SETopCornerpoint_Y
Gridblock_3_1_SETopCornerpoint_Z
Gridblock_3_1_SEBottomCornerpoint_X
Gridblock_3_1_SEBottomCornerpoint_Y
Gridblock_3_1_SEBottomCornerpoint_Z

Gridblock_1_2_SWTopCornerpoint_X
Gridblock_1_2_SWTopCornerpoint_Y
Gridblock_1_2_SWTopCornerpoint_Z
Gridblock_1_2_SWBottomCornerpoint_X
Gridblock_1_2_SWBottomCornerpoint_Y
Gridblock_1_2_SWBottomCornerpoint_Z

Gridblock_2_2_SWTopCornerpoint_X
Gridblock_2_2_SWTopCornerpoint_Y
Gridblock_2_2_SWTopCornerpoint_Z
```

Gridblock_2_2_SWBottomCornerpoint_X
Gridblock_2_2_SWBottomCornerpoint_Y
Gridblock_2_2_SWBottomCornerpoint_Z

Gridblock_3_2_SWTopCornerpoint_X
Gridblock_3_2_SWTopCornerpoint_Y
Gridblock_3_2_SWTopCornerpoint_Z
Gridblock_3_2_SWBottomCornerpoint_X
Gridblock_3_2_SWBottomCornerpoint_Y
Gridblock_3_2_SWBottomCornerpoint_Z

Gridblock_3_2_SETopCornerpoint_X
Gridblock_3_2_SETopCornerpoint_Y
Gridblock_3_2_SETopCornerpoint_Z
Gridblock_3_2_SEBottomCornerpoint_X
Gridblock_3_2_SEBottomCornerpoint_Y
Gridblock_3_2_SEBottomCornerpoint_Z

Gridblock_1_3_SWTopCornerpoint_X
Gridblock_1_3_SWTopCornerpoint_Y
Gridblock_1_3_SWTopCornerpoint_Z
Gridblock_1_3_SWBottomCornerpoint_X
Gridblock_1_3_SWBottomCornerpoint_Y
Gridblock_1_3_SWBottomCornerpoint_Z

Gridblock_2_3_SWTopCornerpoint_X
Gridblock_2_3_SWTopCornerpoint_Y
Gridblock_2_3_SWTopCornerpoint_Z
Gridblock_2_3_SWBottomCornerpoint_X
Gridblock_2_3_SWBottomCornerpoint_Y
Gridblock_2_3_SWBottomCornerpoint_Z

Gridblock_3_3_SWTopCornerpoint_X
Gridblock_3_3_SWTopCornerpoint_Y
Gridblock_3_3_SWTopCornerpoint_Z
Gridblock_3_3_SWBottomCornerpoint_X
Gridblock_3_3_SWBottomCornerpoint_Y
Gridblock_3_3_SWBottomCornerpoint_Z

Gridblock_3_3_SETopCornerpoint_X
Gridblock_3_3_SETopCornerpoint_Y
Gridblock_3_3_SETopCornerpoint_Z
Gridblock_3_3_SEBottomCornerpoint_X
Gridblock_3_3_SEBottomCornerpoint_Y
Gridblock_3_3_SEBottomCornerpoint_Z

Gridblock_1_3_NWTopCornerpoint_X
Gridblock_1_3_NWTopCornerpoint_Y
Gridblock_1_3_NWTopCornerpoint_Z
Gridblock_1_3_NWBottomCornerpoint_X
Gridblock_1_3_NWBottomCornerpoint_Y
Gridblock_1_3_NWBottomCornerpoint_Z

Gridblock_2_3_NWTopCornerpoint_X
Gridblock_2_3_NWTopCornerpoint_Y

Gridblock_2_3_NWTopCornerpoint_Z
Gridblock_2_3_NWBottomCornerpoint_X
Gridblock_2_3_NWBottomCornerpoint_Y
Gridblock_2_3_NWBottomCornerpoint_Z

Gridblock_3_3_NWTopCornerpoint_X
Gridblock_3_3_NWTopCornerpoint_Y
Gridblock_3_3_NWTopCornerpoint_Z
Gridblock_3_3_NWBottomCornerpoint_X
Gridblock_3_3_NWBottomCornerpoint_Y
Gridblock_3_3_NWBottomCornerpoint_Z

Gridblock_3_3_NETopCornerpoint_X
Gridblock_3_3_NETopCornerpoint_Y
Gridblock_3_3_NETopCornerpoint_Z
Gridblock_3_3_NEBottomCornerpoint_X
Gridblock_3_3_NEBottomCornerpoint_Y
Gridblock_3_3_NEBottomCornerpoint_Z

i.e. the grid is defined by pillars that comprise the vertical sides of each gridblock, and the X, Y and Z coordinates of the top and bottom of each pillar are specified in order, looping first from west to east and then from south to north. Z coordinates should be specified positive downwards. The values can be separated by either spaces or line returns. An example of a simple geometry file is given below:

```
#Geometry

0 0 1995 0 0 1997.5

10 0 1996 10 0 1998.5

20 0 1996.5 20 0 1998.8

30 0 1997 30 0 1999.3

0 10 1997 0 10 1999.5

10 10 1998 10 10 2000.5

20 10 1998.5 20 10 2000.8

30 10 1999 30 10 2001.3

0 20 1997.7 0 20 2000.2

10 20 1998.7 10 20 2001.1

20 20 1999.2 20 20 2001.8

30 20 1999.8 30 20 2002.3

0 30 1998.7 0 30 2001.1

10 30 1999.7 10 30 2002.1

20 30 2000.1 20 30 2002.9

30 30 2000.5 30 30 2003.4
```

NB there is no sanity check to ensure the resulting grid geometry is consistent (e.g. checking for negative gridblock volumes, etc.); this must be done by the user.

Processing and interpreting the output data

The exact format of the output files will depend on the output options selected, but they will all be ASCII format text files.

Implicit data output

The implicit data files show the evolution of the fracture network through time, by outputting implicit fracture data at each timestep. One output file is generated for each gridblock, with a name such as *ImplicitData_X0_Y0.txt*, identified by the coordinates of the southeastern corner of the cell. The files are in tab-delimited text format so can be easily loaded into most spreadsheets and databases.

The main part of each file comprises a table containing the key output parameters in columns, with each row representing a timestep. The first 7 columns contain data relating to the model as a whole (timestep number, time, total elastic and inelastic strain). These are followed by several sets of columns giving parameters relating to individual fracture sets (fracture status, driving stress, sense of displacement, active and static mean volumetric and mean linear microfracture densities, active and static mean volumetric and mean linear half-macrofracture densities, and clear zone volume). Then, if specified, will be 8 columns containing the total microfracture and macrofracture porosity, calculated using different methods. Finally, if specified, will be 36 columns containing each of the elements of the bulk rock compliance tensor. Note that if stress shadows are active (i.e. **StressDistributionScenario** is set to *StressShadow*), the bulk rock compliance tensor will be equal to the isotropic intact rock compliance tensor; to see the effect of fractures on the bulk rock compliance tensor, it is necessary to switch off the stress shadows (i.e. set **StressDistributionScenario** to *EvenlyDistributedStress*).

Underneath this, if specified, will be a series of tables giving the final cumulative density distribution functions for active and static mean volumetric and mean linear densities of each fracture set. These tables comprise a series of columns, each representing a specified microfracture radius or half-macrofracture length, with rows giving the cumulative mean volumetric and mean linear densities of active and static microfractures or half-macrofractures for each set, for the final fracture network. Note that this data is not available for the intermediate stages of fracture growth.

Explicit data output

The explicit data files describe a realisation of the simulated fracture network at specific points in time, in the form of a Discrete Fracture Network (DFN). In a DFN, individual fractures are represented explicitly by a series of planar geometric objects. The explicit data files contain the coordinates of the cornerpoints of these planar objects, as well as information about their geometry (e.g. azimuth, dip, size etc.) and properties (e.g. aperture, displacement).

The layer-bound macrofractures and the microfracture of the final fracture network will be saved in two separate files, named *Macrofractures_final.txt* and *Microfractures_final.txt* respectively. If microfractures are not included in the explicit DFN, the *Microfractures_final.txt* file will be empty. If the user has selected to output intermediate stages of the fracture network, two additional explicit data files will be generated for each of these, named *Macrofractures_StageN_TimeT.txt* and *Microfractures_StageN_TimeT.txt* respectively. The format of this output must be specified by the user from two possible formats:

- **ASCII format** is a format specific to DFNGenerator, and is readable by the data analysis spreadsheets described below. Each fracture is described in a separate block, which starts with a

header line of information relating to the fracture as a whole, including a unique ID number, the fracture set, and dip. For circular microfractures, this information also includes the coodinates of the centre of the fracture, its radius, azimuth and whether it is active. For layer-bound macrofractures, this information also includes the number of cornerpoints, the time of nucleation, the fracture length, and the fracture tip relationships (whether they are active or terminate due to stress shadow interaction or intersection, and if so the ID number of the fracture they terminate against). This is then followed by a block of text listing the coordinates of the fracture cornerpoints, moving around the fracture clockwise from the top middle point. Each cornerpoint is output on a separate row, and the entire block is sandwiched between rows with labels `Start Points` and `End Points` respectively. Note that for layer-bound macrofractures, all fracture segments will be included in the same block, so the cornerpoints may not all be coplanar. For microfractures, if the specified number of cornerpoints to output (**Number_uF_Points**) is less than 3, only the header line will be given for each fracture and there will be no cornerpoint blocks.

- **FAB format** is a text format originally specified by Golder Associates for use in the Fracman software, but it is also compatible with many other geomodelling packages such as Petrel. This format is therefore recommended if the explicit data will be loaded data into external software packages. It is described fully in the Fracman manual, but one important point to note is that each fracture segment is described in a separate block; it is therefore not possible to automatically link all segments in a single layer-bound macrofracture in this format.

Data analysis spreadsheets

The DFNGenerator_StandaloneProgram.zip zip file also contains several Excel spreadsheets to help display and analyse the output data from the DFN Generator program. These spreadsheets contain macros to help load, plot and extract data from the implicit and explicit output files described above. Each spreadsheet comprises two worksheets, one (*Implicit_Fracs*) for implicit data and the other (*Explicit_DFN*) for explicit data; each spreadsheet can therefore load data from one implicit data file (representing a single gridblock) and one explicit DFN file (representing an entire grid).

Four spreadsheets are provided:

- **DFN_Plotter_macrofractures.xlsm:** Used to display and analyse layer-bound macrofracture data from DFNs comprising two orthogonal fracture sets with both Mode 1 and Mode 2 fractures.
- **DFN_Plotter_microfractures.xlsm:** Used to display and analyse microfracture data from DFNs comprising two orthogonal fracture sets with both Mode 1 and Mode 2 fractures.
- **DFN_Plotter_6set.xlsm:** Used to display and analyse layer-bound macrofracture data from DFNs comprising six fracture sets with both Mode 1 and Mode 2 fractures.
- **DFN_Plotter_6set1Mode.xlsm:** Used to display and analyse layer-bound macrofracture data from DFNs comprising six fracture sets with both Mode 1 and Mode 2 fractures.

These spreadsheets can however be easily modified to display and analyse data from other types of DFN.

Each spreadsheet contains three macros:

- The **Load Data** macro automatically loads data from the implicit and explicit output files specified by the user, and then copy this to the appropriate parts of the spreadsheet. It then gives the user the option of launching the **Plot** and **Extract Explicit Data** macros (see below), before saving the spreadsheet under a new, user-specified name. This macro can be launched by pressing Ctrl+Shift+L.

- The **Plot** macro generates a 2D horizontal visualisation of the explicit DFN. Before running this macro, the outer bounds of the model must be specified in cells AA2 to AA5. This macro can be launched by pressing Ctrl+p.
- The **Extract Explicit Data** macro calculates cumulative density distribution data for the explicit DFN, and plot this alongside the implicit cumulative density distribution functions, in the charts on the *Implicit_Fracs* worksheet. This allows for easy comparison and validation of the models: the explicit cumulative density distribution data should approximately match the implicit cumulative density distribution functions, although there will be some discrepancies due to random and systematic errors (see Welch et al. 2020 for a more detailed discussion of this). Before running this macro, the area from which to extract the cumulative density distribution data of the DFN must be specified in cells AA12 to AA15, and the layer thickness in cell AA16. It is recommended to data extraction area is smaller than the overall model size to avoid edge effects due to fracture truncation at the grid boundary. This macro can be launched by pressing Ctrl+Shift+M.

The spreadsheets also contain a number of other charts that show the evolution of the fracture network, and the individual fracture sets, through time. These show for example how the ratio of active to static fractures varies through time for the different fracture sets, and how the clear zone volume (the volume available for new fractures to nucleate in) decreases. They can easily be modified to show other data if required.

Example models

The example models provided with the standalone code may be helpful to understand how to design and analyse models to replicate different aspects of fracture network evolution. The following example models are contained in the zip file *DFNGenerator_StandaloneProgram.zip*:

- **Model A** illustrates the growth of circular microfractures through time. It also shows how to output multiple DFNs representing different stages in the evolution of a fracture network. This is a small (10mx10mx1m) single-gridblock model; the microfracture outputs can be plotted using the **DFN_Plotter_microfractures.xlsm** spreadsheet.
- **Model B** illustrates a fracture network developing in response to a uniaxial extensional strain with stress shadows. It contains just one fracture set but comprises 3x3 gridblocks, all with identical input data. The model is 150x150m in size and 1m thick, and the output can be plotted using the **DFN_Plotter_macrofractures.xlsm** spreadsheet.
- **Model C** illustrates a fracture network developing in response to a uniaxial extensional strain with evenly distributed stress (i.e. without stress shadows). It contains just one fracture set, but since there are no stress shadows it does not reach saturation. The model geometry and other input parameters are identical to Model B, and the output can be plotted using the **DFN_Plotter_macrofractures.xlsm** spreadsheet.
- **Model D** illustrates an isotropic fracture network with two orthogonal fracture sets, developing in response to an isotropic extensional strain with stress shadows. The geometry and other input parameters are identical to Model B, and the output can be plotted using the **DFN_Plotter_macrofractures.xlsm** spreadsheet.
- **Model E** illustrates an anisotropic fracture network with two orthogonal fracture sets, developing in response to an anisotropic extensional strain with stress shadows. The resulting fracture network comprises a primary set containing long fractures, and a secondary set containing shorter fractures that terminate against the primary set. The model geometry and other input parameters are identical to Model B, and the output can be plotted using the **DFN_Plotter_macrofractures.xlsm** spreadsheet.

- **Model F** illustrates a polygonal fracture network comprising 6 fracture sets, striking at 30° intervals, developing in response to an isotropic extensional strain with stress shadows. The model geometry and other input parameters are identical to Model B, but the output is plotted using the **DFN_Plotter_6set.xlsm** spreadsheet.
- **Model G** illustrates a conjugate strike-slip fracture network, developing in response to a shear strain (i.e. $\varepsilon_{hmax} = -\varepsilon_{hmin}$) with stress shadows. The fracture network comprises 6 fracture sets in total, striking at 30° intervals, but the strain is mostly accommodated on two conjugate sets striking 60° apart. The model geometry and other input parameters are identical to Model B, but the output is plotted using the **DFN_Plotter_6set.xlsm** spreadsheet.
- **Model H** illustrates an isotropic network of Mode 1 dilatant fractures, comprising two orthogonal fracture sets, developing in response to an isotropic extensional strain with stress shadows and high fluid overpressure. The model is 30x30m in size and 0.1m thick, comprising 3x3 gridblocks, all with identical input data. The output can be plotted using the **DFN_Plotter_macrofractures.xlsm** spreadsheet, but this requires some manual editing as only one fracture dipset is generated for each set.

Note that *DFNGenerator_StandaloneProgram.zip* contains only the configuration files for the example models; however the spreadsheets showing the output from these models can be downloaded directly from the Test_models folder.

Further information and contact details

More details of the algorithm used in the DFN Generator software, as well as analysis of the key controls on the development of fracture networks, can be found in

> Welch, M. J., Lüthje, M., & Glad, A. C. 2019. Influence of fracture nucleation and propagation rates on fracture geometry: insights from geomechanical modelling. Petroleum Geoscience, 25(4), 470-489.

and in more detail, in the book

> M. Welch, M. Lüthje and S. Oldfield. Modelling the Evolution of Natural Fracture Networks - Methods for Simulating the Nucleation, Propagation and Interaction of Layer-Bound Fractures. Springer. 2020

The latter also contains examples of the application of this software to outcrop and subsurface examples of fractured horizons. It can be ordered direct from the publisher, at https://www.springer.com/gp/book/9783030524135.

DFN Generator has been developed with funding from the Danish Hydrocarbon Research and Technology Centre (DHRTC) under the Advanced Water Flooding programme. Please note therefore that DFN Generator comes with no warranty and DHRTC and the authors accept no liability for any consequence arising from its use. There is also no formal support or service level agreement for the software. However if you encounter any problems, or have any comments or suggestions, please contact Michael Welch (mwelch@dtu.dk) or Mikael Lüthje (mikael@dtu.dk) and we will try to help you. Please also report any bugs that you encounter or requests for functionality enhancements in the same way.