Lecture Notes in Earth Sciences 106

Paul D. D. Bons · Daniel Koehn ·
Mark W. Jessell (Eds.)

Microdynamics Simulation

With 185 Figures

Springer

Paul D. D. Bons
Eberhard Karls Universität
Institut für Geowissenschaften
Fachbereich Mineralogie und
Geodynamik
10 Sigwartstrasse
Tübingen 72076
Germany
paul.bons@uni-tuebingen.de

Mark W. Jessell
IRD LMTG UMR 5563
14 avenue Edouard Belin
Toulouse, France 31400
mjessell@lmtg.obs-mip.fr

Daniel Koehn
Universität Mainz
FB Geowissenschaften
Geographisches Institut
21 Johann-Joachim-Becher-Weg
Mainz 55099
Germany
koehn@uni-mainz.de

"For all Lecture Notes in Earth Sciences published till now please see final pages of the book"

ISBN: 978-3-540-25522-2 e-ISBN: 978-3-540-44793-1

Lecture Notes in Earth Sciences ISSN: 0930-0317

Library of Congress Control Number: 2007937514

Cover design: WMXDesign GmbH

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Appendices

These Appendices describe the steps necessary to run arbitrary single-processes or multi-process *Elle* experiments. Currently *Elle* runs under Windows and Linux, and as the code is continuously being updated, both to fix bugs and add features, **it is recommended that you visit the *Elle* web site to download the latest releases of the *Elle* platform (http://www.microstructure.info/elle)**; however the description here applies specifically to the code released on the CD which accompanies this book.

In the appendices we use the following formats to distinguish menu items, programs and files:

• **File/Open**	Menu item
• *Program*	Name of software or modules
• <u>File</u>	Name of a file or URL

READERS WHO ONLY WISH TO RUN THE EXAMPLE EXPERIMENTS DESCRIBED IN THIS BOOK SHOULD INITIALLY REFER TO APPENDICES A, B & C.

Appendix A Installing *Elle*

Mark W. Jessell

A.1 Windows

To install the *Elle* platform on your Windows computer, open the CD and double click on the **ElleWinSetup** icon. There are two installation levels for the platform: User and Developer.

1. Users are those people who wish to run or modify example experiments, and who will have access to the high-level executables, scripts and input files.
2. Developers are those people who wish to alter the codes to describe new processes, who will have also have a complete GCC compiler, a simple Linux-like environment (MSys) and the complete *Elle* source code installed on their system. The Linux-like environment runs under Windows and does not interfere with the normal operation of the operating system. If you choose this option, you can recompile the *Elle* platform at any time by opening an MSys shell, then change to the directory elle and type in `./install.win wx > debug.txt&`

For both Users and Developers, *Sybil*, the graphics postprocessor for the *Basil* Finite Element program, will also be installed at this stage, which uses an Open Source X Windows system, running under *Cygwin*.

After successful installation, all example experiments (Appendix B) are copied over to your harddisk, so you will not need the CD again after installation. On your harddisk a folder elle/ is created containing the following files and folders:

- elle/, a folder containing all code and other elle-related material
- experiments/, a folder containing all examples of Appendix B
- extras/, a folder containg some colour maps, elle input files, etc.
- COPYING.txt, a general public licence
- LICENCE.txt, the *Elle* licence
- Internet shortcut, a link to the *Elle* website
- unins000.exe & unins00.dat, files to un-install *Elle*

A.1.1. Installing *Basil* and *Sybil*

Basil **-** The Finite Element deformation program, *Basil*, will be installed with the *Elle* binaries when you run ElleWinSetup.exe.

Sybil - This is an X11 application -(Appendix F). For instructions on how to install this program, which requires the *Cygwin* system to also be installed on your PC, please go to the *Elle* Website (http://www.microstructure.info/elle) and select the menu "**Sybil**". If you already have *Cygwin* installed, or once you have it installed, you may install *Sybil* by running SybSetup.exe. The *Experiment Launcher* expects sybil.exe and sybilps.exe to be on the same drive as Cygwin and in the directory DRIVE:\cygwin\home\sybil\. If you install it in a different location, you will have to edit sybil.bat and sybilps.bat in the binwx directory of the Elle installation.

If you have a firewall or something similar installed under Windows, you have to allow *Cygwin* access to the internet. Internally it uses the network (a so-called loopback-device) to send messages. This is safe to do, as no data will be downloaded to your computer or sent to the internet.

A.2 Linux

To install the *Elle* platform on your Linux computer (for both **Users** and **Developers**):

1. open the CD;
2. copy the directory called elle to your hard disk;
3. open up a shell window, and then change to the directory elle
   type in ./install.sh wx > debug.txt&

This will compile the complete *Elle* platform on your computer (and the *Basil* Finite Element package), assuming that all the correct libraries have been installed on your system.

To install *Elle* you will require the following libraries to be installed

- WXWidgets installed using the --with-gtk option (http://www.wxwidgets.org and comes with all standard Linux distributions).
- GTK+ (http://www.gtk.org and comes with all standard Linux distributions)
- GCC with Fortran (comes with all standard Linux distributions)

If the compilation does not run to completion (you do not get a directory created called binwx full of files), you should check the debug.txt to see what the errors are, and refer to the Installation Forum at the *Elle* Website: http://www.microstructure.info/elle.

A.2.1. Installing *Basil* and *Sybil*

Pre-built binaries can be found at the *Elle* website, www.microstructure.info/Elle. Contact basil@earthsci.unimelb.edu.au or one of the authors if you are unable to find an appropriate package.

Sybil (Appendix F) uses X11 and Motif libraries and you may need to install the Motif libraries on your system. These are available on the web e.g www.lesstif.org or www.openmotif.org. These applications do not need to be installed in a particular location but your environment variable BASILPATH should be set such that $BASILPATH/bin is the location of the binaries (*Basil*, *Sybil* and *sybilps*) and $BASILPATH/bin should be added to your $PATH variable.

Appendix B Example experiments

Edited by Mark W. Jessell

After installation, all experiment files are found within the elle/experiments directory on your hard disk. This is necessary, because Elle cannot write output files onto the CD. The input files for the experiments described in this Appendix (see Table B.1), for example Experiment 3, are contained in a directory whose name starts with experiment_03... Where there are more than one set of input files for an experiment, there will be a series of subdirectories, e.g. **a, b, c** etc. To launch an example experiment without modification, use the *Experiment Launcher*, (elle/elle/binwx/experiment_launcher.exe under Windows).

The *Experiment Launcher*

The *Experiment Launcher* is a simple program that allows you to launch any of the example experiments in this Appendix. Once started up, the Launcher offers you a single menu: the **File menu**, and two panels the **Experiments panel**, the **Utilities panel.** The **Experiment panel** and its submenus list all the experiments in this Appendix. To run a particular example, select it from the **Experiment panel** and you will be provided with a brief overview of the experiment. If you wish to start the experiment, simply click on the **Go!** button.

Single-Process Experiments: If a single-process was started, an *Elle* window will open up showing the starting microstructure. To enact the experiment select **Run→Run** from within the *Elle* Window. See Appendix C for details on the graphical user interface for single processes.

Important: to kill a single process experiment under windows, use the task manager (in Windows press CTRL-ALT-DEL), rather than clicking on the $\boxed{x}$-icon, as otherwise the calculation will continue in the background!

Multi-Process Experiments: If a multi-process was started, the experiment will be started automatically in the background, and to view any of the output *Elle* files, select *showelle* from the **Utilities panel** of the Launcher, and load in the desired file. See Appendix C for details on the graphical user interface for *showelle.* All output files will be created in the same directory as the experiment, and have a naming convention described in Appendix E.

Important: to kill a multi-process experiment under windows, select the command-line window and type CTRL-C.

Starting Single Processes or Utilities: The *Experiment Launcher* may also be used to launch utilities such as *showelle*, without any input file pre-loaded.

The duration of individual experiments is highly variable and denoted with icons for “fast” (▶), “medium” (☕), “slow” (🚶), to “very slow” (🛏).

Table B.1. List of example experiments, and the sections in the book that describe them in more detail.

Experiment number	Section in book	Title
1	2.2	Diffusion *
2	2.2	Fluid flow in a porous medium*
3	2.6	Crystal growth from melt
4	3.12	Fracturing in granular aggregates*
5	3.2	Cation exchange reactions
6	3.3	Subgrain growth
7	3.5	Grain growth
8	3.7	Evolution of a partial melt
9	3.8	Rigid porphyroblast growing in a deforming matrix
10	3.9	Lattice rotations
11	3.11	Boudinage*
12	3.12	Dissolution grooves*
13	3.12	Stylolites*
14	4.8	Strain-rate partitioning during porphyroblast growth
15	4.1	Anisotropic grain growth
16	4.2	Dynamic recrystallisation
17	4.3	Deformation localisation
18	4.4	Expanding inclusions*
19	4.5	Mud cracks*
20	4.6	Visco-elastic deformation and fractures*
21	4.7	Strain localization and rigid object kinematics

* Note: Experiments that use *elle_latte* will not show the starting microstructure until the experiment is actually started (with **run→run**). The stage number shown on the display does not necessarily equal the number of time steps. Also note that *elle_latte* experiments cannot be re-run.

Although we have done our best to test and debug these examples, there may still be some bugs. If you encounter problems, please check the *Elle* website (http://microstructure.info/elle).

Experiment 1 - Diffusion

In this example we demonstrate diffusion through a fluid trapped in a granular medium using a simple Lattice-Gas automaton that uses the FHP rules as its basis and is mapped on a triangular grid (see Chap. 2.2). The algorithm is included in *elle_latte*. These experiments look "noisy" because we are looking at very small areas of material. Hence the random fluctuations are much more evident than in larger-scale types of models, such as those that use Finite Difference calculations.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you can use the higher-resolution files since the algorithm is relatively fast. Possible sub-experiments, which simply vary the density of unodes are:

- **a** diffusion in a system with 2850 particles
- **b** diffusion in a system with 11500 particles
- **c** diffusion in a system with 46000 particles

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

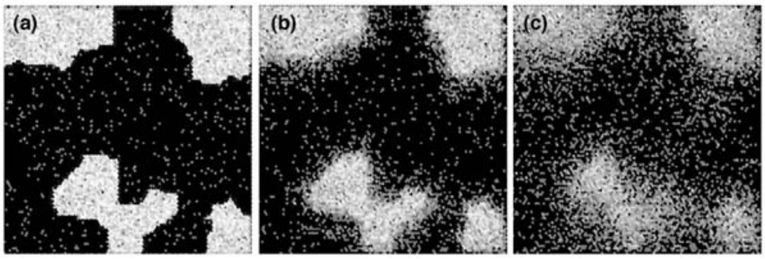


Fig. B1. Modelling diffusion with a Lattice-Gas automaton (experiment **b**). Time steps are **(a)** 1, **(b)** 10, **(c)** 40. High concentration is bright and low concentration black

Interface: These files will each load their appropriate preferences file automatically. The resulting plots show you the concentration of fluid particles per unode. Unodes that are part of a grain, and which are therefore not permeable, have a concentration of 6 to visualize the pore space.

Examples: Figure B1 shows plots of an example run that illustrates how diffusion can be modelled with a Lattice-Gas automaton. The input file is res100.elle. Plotted are the concentrations of fluid particles where the background is dark, and the brightness increses with concentration of fluid particles (U_DENSITY). The initial configuration is of four regions with a high concentration embedded in a background with a zero concentration.

Functions and parameters for users wishing to use the Experiment Interface to modify parameters: In the initialization function (Experiment::Init(), case 12) the following functions contain parameters that can be changed:

- SetFluidLatticeGasRandom(0.005): This function is used to set randomly distributed fluid particles in the background. The input variable specifies the concentration of fluid particles in the whole *Elle* box where 0.005 is 0.5%.
- SetFluidLatticeGasRandomGrain(0.7,j*10): This function specifies the fluid particle concentration in a specific grain. The first input value determines the fluid concentration (0.7 is 70%) and the second variable the specific grain (in this case a loop where every tenth grain is chosen).
- SetWallsLatticeGas(j*5): This function is used for the creation of porosity in the 2d fluid flow. All grains (j*5) are set to be impermeable so that a complex permeability structure evolves.

In the run function (Experiment::Run(), case 12) the following functions contain parameters that can be changed:

- UpdateFluidLatticeGas(): simply does a Lattice-Gas step where particles are transported and collisions are handled.
- InsertFluidLatticeGas(0.03, 0.9): inserts fluid particles at the left hand side of the box, every lattice particle with *x* position smaller than 0.03 becomes each time step a fluid concentration of 90 percent (second input value, 0.9).
- RemoveFluidLatticeGas(0.97): removes all fluid particles from lattice particles with an *x* position that is larger than 0.97.

Experiment 2 - Fluid flow in a porous medium

In this example we demonstrate fluid flow in porous medium using a simple Lattice-Gas automata that uses the FHP rules as basis and is mapped on a triangular grid (see Chap. 2.2). The algorithm is included in ***elle_latte***.
Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you can use the higher-resolution files since the algorithm is relatively fast. Possible sub-experiments, which simply vary the density of unodes are:

a fluid flow in a system with 2850 particles ▶
b fluid flow in a system with 11500 particles ▶
c fluid flow in a system with 46000 particles ▶

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve. It is a good idea to start with experiment **a**, as this gives a detailed expression of how the system behaves.

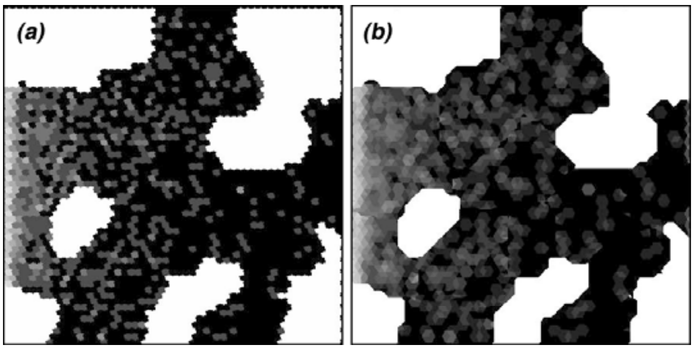


Fig. B2. Simulation of fluid flow through a porous medium. Fluid comes in at the left hand side and exits at the right hand side. Background is black, wall rock is white and fluid particles are bright. **(a)** Unodes are shown assmall hexagons whose brightness is a function of concentration. **(b)** Concentrations are interpolated between unodes when the option **triangulate unodes** is turned on in the **Graphics→Preferences→Unode** menu

Interface: These files will each load their appropriate preferences file automatically. The resulting plots show you the concentration of fluid particles per unode. Unodes that are part of a grain, and which are therefore not permeable, have a concentration of 6 to visualize the pore space.

Examples: This example is shown in Fig. B2. Here 2D porous flow is modelled with a Lattice-Gas automaton. The input file is res50.elle (experiment **a**) with a low resolution. Particle concentration on the left boundary is kept constant, while all particles reaching the right boundary of the system are removed. This effectively models fluid that is injected from the left hand side and that can escape on the right hand side of the model.

Functions and parameters for users wishing to use the Experiment Interface to modify parameters: In the initialization function (Experiment::Init(), case 13) the following functions contain parameters that can be changed:

- SetFluidLatticeGasRandom(0.005): This function is used to set randomly distributed fluid particles in the background. The input variable specifies the concentration of fluid particles in the whole *Elle* box where 0.005 is 0.5%.
- SetFluidLatticeGasRandomGrain(0.7,j*10): This function specifies the fluid particle concentration in a specific grain. The first input value determines the fluid concentration (0.7 is 70%) and the second variable the specific grain (in this case a loop where every tenth grain is chosen).
- SetWallsLatticeGas(j*5): This function is used for the creation of porosity in the 2d fluid flow. All grains (j*5) are set to be impermeable so that a complex permeability structure evolves.

In the run function (Experiment::Run(), case 13): the following functions contain parameters that can be changed:

- UpdateFluidLatticeGas(): simply does a Lattice-Gas step where particles are transported and collisions are handled.
- InsertFluidLatticeGas(0.03, 0.9): inserts fluid particles at the left hand side of the box. Each time step every lattice particle with an *x* coordinate smaller than 0.03 gets a fluid concentration of 90% (second input value, 0.9).
- RemoveFluidLatticeGas(0.97): removes all fluid particles from lattice particles with an *x* position that is larger than 0.97.

Experiment 3 - Crystal growth from melt

This example illustrates the growth of a single crystal from a melt, using a Phase Field approach. The background of the software (from Biben 2005) is explained in Chap. 2. In the following examples we can vary the crystal symmetry and the latent heat of crystallisation.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. Possible sub-experiments, which vary the crystal symmetry 6-fold or (the physically extremely rare!) 5-fold symmetry; or the latent heat, are:

a 6-fold symmetry, relatively high latent heat
b 5-fold symmetry, relatively high latent heat
c 6-fold symmetry, relatively low latent heat
d 5-fold symmetry, relatively low latent heat

Interface: These files will each load their appropriate preferences file automatically. Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve. You can either display the concentration (CONC_A) of the unodes (the default) or the temperature (U-ATTRIB-A). In order to speed up the experiment, only every 10-th time step is displayed.

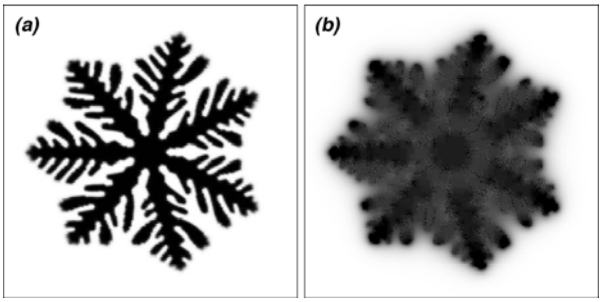


Fig. B3. Phase Field simulation of the growth of a crystal from a melt after 2000 time steps. **(a)** Shape of crystal (CONC_A) and **(b)** temperature (U_ATTRIB_A) field. The simulation shown is based on experiment **a**, but the Anisotropy was changed to achieve an unusual 7-fold symmetry (Select **Run->Run Options** then edit the **Anisotropy** field in the **UserData** area)

Functions and parameters for users wishing to modify runtime parameters: The default userdata settings for *elle_phasefield* are shown below:

- userdata[0] default= 0 (reset unode values if we want to create a seed crystal from a file that has no seed already defined)
- userdata[1] default = 1.8 (high latent heat) or 0.18 (low latent heat)
- userdata[2] default = 0.01 (interfacial width)

- userdata[3] default = 0.02 (modulation of the interfacial width)
- userdata[4] default = 1.57 (orientation of the anisotropy axis)
- userdata[5] default = 6.0 (6-fold symmetry) or 5.0 (5-fold symmetry)
- userdata[6] default= 0.9
- userdata[7] default = 10
  (where m(T)=ALPHA/PI*atan(GAMMA*(TEQ-T)))

Experiment 4 - Fracturing in granular aggregates

This example illustrates the development of fractures in a granular aggregate during compaction and pure shear deformation. The background of the software is explained in Chap. 3.12 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you should use the lower-resolution files since the algorithm is relatively slow. Possible sub-experiments, which simply vary the density of unodes are:

a 2850 particles
b 11500 particles
c 46000 particles

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You can now choose to view the Young's moduli (U_YOUNGSMODULUS) of different grains, the fractures (U_FRACTURES), the mean stress (U_MEAN_STRESS), or the differential stress (U_DIFFSTRESS, default). In order to view fractures select **Graphics->Preferences** then click on the **Unodes** tab and select and set **Clamp color between** -1.0 and 0.0 to limit the display range. Unodes with fractured bonds will be blue and unfractured unodes red. Differential and mean stress should be scaled during a run, normal values are: differential stress 0.0 to 0.01 and mean stress - 0.01 to 0.01 (negative is compression).

Examples: Figure B3 (a) to (d) shows the four different plotting options. Figure B3a shows the distribution of Young's moduli of different grains where dark grey is high Young's modulus (range 0.0 to 1.7). Figure B3b shows the first fractures after 7 *Elle* steps where dark are fractured particles (range is –1.0 to 0.0). Figure B3c shows the differential stress where dark is high and light colour low stress (range 0 to 0.02) of stage 7 and Fig. B3d shows the mean stress of the same stage where dark is high stress (range –0.01 to 0.005, negative is high compressive stress). Figure B3e shows an example with 5 steps of uniaxial deformation followed by 15 steps of pure shear deformation whereas Fig. B3f shows an example with 15 steps of uniaxial deformation followed by 5 steps of pure shear deformation. Plots show again the differential stress (range 0-0.035). The second example that has experienced a longer uniaxial compression builds up more compressive stress and therefore develops more pronounced shear fractures whereas the first example shows a combination of extension and shear fractures. Comparing Fig. B3a and B3b one can also see that regions with high Young's moduli are fracturing first.

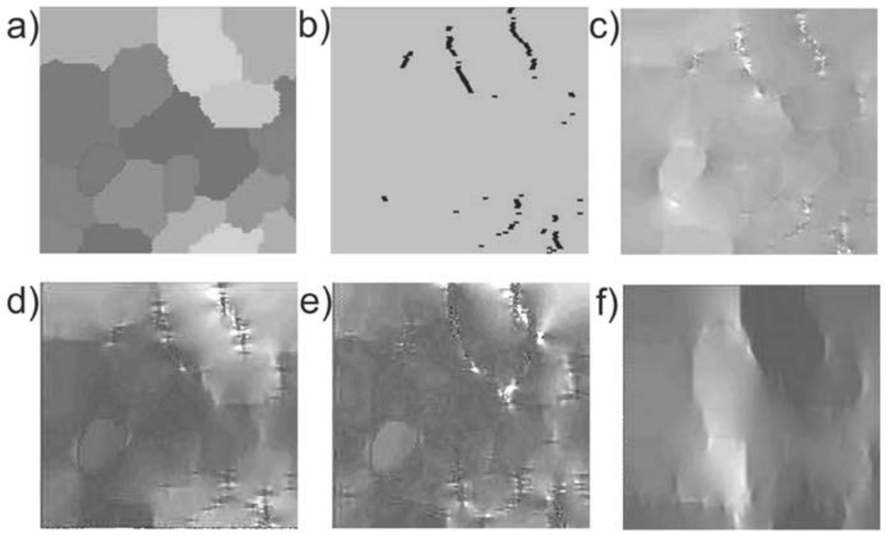


Fig. B3. Examples of fracture development in granular aggregates. **(a)** to **(d)** plotting options showing different unode attributes **(a)** U_YOUNGSMODULUS: Young's moduli of grains, **(b)** U_FRACTURES: fractured particles, **(c)** U_DIFFSTRESS differential stress and **(d)** UMEAN_STRESS: mean stress. **(e)** Differential stress after 5 stages of uniaxial compaction followed by 15 steps of pure shear deformation. **(f)** Differential stress after 15 stages of uniaxial compaction followed by 5 steps of pure shear deformation

Functions and parameters for users wishing to use the Experiment Interface to modify parameters: Parameters used in the example are:

- SetPhase(0.0,0.0,2.0,1.2),
- SetGaussianSpringDistribution(1.0,0.5),
- MakeGrainBoundaries(1.0,0.5),
- DeformLattice(0.001,1), and
- DeformLatticePureShear(0.001,1).

In the initialization function (`Experiment::Init()`, case 1) the following functions contain parameters that can be changed:

- SetPhase(0.0,0.0,2.0,1.2) is used to set a distribution on breaking strengths of springs. The breaking strengths will be distributed randomly between original strength - 0.6 * original strength and original strength +0.6 * original strength (0.6 is value 1.2/2 in function). The whole distribution will be multiplied by 2.0 (first value in function). Note that both values in the function change the behaviour of the model since the breaking strength is a function of the weakest bonds.
- SetGaussianSpringDistribution(1.0,0.5): This function is used to set a distribution on the Young's moduli of grains. In this case the distribution is Gaussian with a mean value 1.0 and a standard deviation of 0.5 (smaller values make the distribution narrower and larger values make the distribution wider). Grains are picked randomly.
- MakeGrainBoundaries(1.0,0.5): This function gives the grain boundaries different properties, in this case 1.0 means that the elastic properties are not changed and 0.5 that the breaking strength of grain boundary springs is on average half that of intragrain springs (This does not affect the distributions that were set before).
- SetFracturePlot(50,1) simply means that we make an extra plot after 50 fractures formed. This is useful in order to view the dynamics of the very non-linear fracture process. Note however that if you call this function your *Elle* stages will be non-linear with time and do not necessarily show the real time steps anymore.

In the run function (Experiment::Run(), case 1) the following functions contain parameters that can be changed:

- if (experiment_time <5) means that the program is applying 5 steps of uniaxial compression and starts with pure shear deformation afterwards.
- DeformLattice(0.001,1): This function is used to apply uniaxial compression with a vertical strain of 0.1 percent per step. The value 1 means

that a picture is taken after the deformation was applied and the lattice relaxed.

- DeformLatticePureShear(0.001,1): This function is used to apply pure shear deformation that is conserving the area of the box. The value 0.001 means vertical strain is 0.1 percent and 1 means a picture is taken after a deformation step.

Experiment 5 - Cation exchange reactions

This section gives details for running the cation exchange reactions between garnet and biotite that are described in Chap. 3.2 and illustrated in Fig. 3.2.6-8.

Execution: In order to run these experiments, start up the *Experiment Launcher* and select the experiment from the **Experiments Menu**. The subexperiments are:

a isothermal experiment with large matrix grain size ☕
b isothermal experiment with small matrix grain size ☕
c cooling experiment ☕

In practice, a shell script (or batch file in Windows) calls the appropriate *Elle* routines and requires an *Elle* input file containing a description of the initial microstructure. To start a simulation, select the desired experiment and select **Go**. Contrary to the previous experiment, you will now not get a new window with the model, but a console window with scrolling lines.

Interface: As these are multiprocess experiments, the easiest way to monitor the progress is to view the generated *Elle* files with *showelle.* (select *showelle* from the **utilities box** on the *Experiment Laucher window*) A preferences file, defaults.zip, will be read by *showelle* so that the look up table is optimised to show Fe concentration variation in the garnet as for Fig. 3.2.8.

Processes: These experiments combine two *Elle* processes, *elle_exchange* (processes/exchange) and *elle_gbdiff* (processes/gbdiff). The exchange process simulates lattice diffusion within the grains and cation exchange between the boundary and lattice, and the second process calculates diffusion along the boundaries.

Examples: The initial file for experiment 5a has one garnet grain and one biotite grain in a matrix of large grains providing few pathways for grain boundary diffusion (see Fig. B4a). In experiment 5b, the matrix consists of smaller grains with more pathways for diffusion between the biotite and garnet (Fig. B4b). In experiments 5a and 5b, the initial Fe mole fraction is 0.8 for the garnet and 0.6 for the biotite and the bnodes are set to 0.05. The temperature is a constant 600 °C. Experiment 5c is a cooling experiment for a microstructure that includes a large garnet and several biotite grains. The initial compositions for the garnet and biotite are assumed to be 0.8 and 0.54 (in Fe mole fraction), respectively. Since these compositions are disequilibrium compositions in the experimental temperature range, garnet and biotite start to exchange cations to approach the equilibrium KD value. There is little change in zonation in the latter part of the experiment as diffusion slows with decreasing temperature (Fig. B4c).

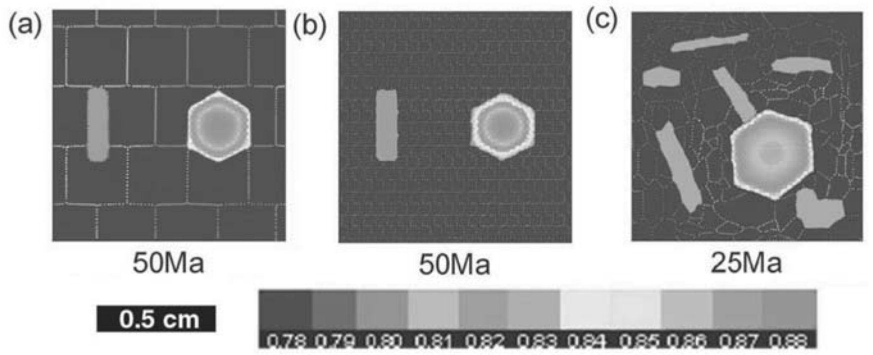


Fig. B4. **(a)** Results from an isothermal experiment with matrix grain size of 0.25cm. The biotite grain is on the left, the garnet grain on the right. **(b)** Same experiment, but with a small grain size. **(c)** Results from a cooling experiment. The garnet grain is the large hexagonal grain near the middle of the model. The CLUT (colour lookup table) of Fe molar fraction is optimized to highlight the garnet composition

Functions and Parameters for users wishing to modify parameters in the Shelle script: Details are given for experiment 5c but the parameters used and naming conventions apply to all the subexperiments. exp5c.elle is the initial elle file in which one grain (grain 84) is garnet, six grains are mica (25, 49, 54, 56, 90, 129) and the matrix is quartz. The unodes in the garnet have an Fe mole fraction of 0.8 and the unodes in the mica have an Fe mole fraction of 0.54. The bnodes have a constant initial Fe concentration of 0.05. The elle file also sets the initial temperature to 750°C.

exp5c.shelle/bat is a batch file generated using the Shelle Script Generator, shelle24.html, to set the following parameters which control the elle_experiment:

1. Experiment Name exp_5_out, the base for naming saved output files
2. *Elle* Input File exp5c.elle, the initial microstructure
3. Last Time **600**, the number of iterations for the experiment loop
4. Save Interval **100**, output files from all processes will be saved every 100 iterations

with the following processes turned on:

1. Grain Boundary Diffusion [20] Stages **500** Kappa **2.0e-9**
2. Exchange Reaction [23] Stages **500**. Kappa **30.90528**. Temperature increment **-0.5**
3. The saved output file has the process list LIST= "20 23"

The parameter, Kappa, is the pre-exponential factor used to calculate the diffusion coefficient D(T) using the Arrhenius equation D(T) = Kappa*exp(-Ea/(RT)). In this code the activation energy for diffusion (Ea) has been set to 239 kJ/mole.

All processes in the list are called once per TIMESTEP, i.e. at the end of this experiment, the exchange process will have performed a total of 30,000 stages and the temperature will have dropped from 750 to 450 °C.

Experiment 6 - Subgrain growth

This example illustrates the process of subgrain growth resulting from two different conceptual models, using a Potts Model approach. The background of the software is explained in Chap. 3.3.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. There are two possible sub-experiments: isotropic and anisotropic:

a isotropic
b anisotropic

Interface: These files will each load their appropriate preferences file automatically. Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve. You can either display any of the three EULER orientations of the unodes.

Examples: In the first example (Fig. B5) a simple subgrain growth experiment has been performed by taking an input file with a highly strained grain of NaCl derived from EBSD measurements, which exhibit high grain lattice distortions. We let the substructure of the grains evolve while the grain boundaries remain stable. The driving force is the reduction of energy where the energy below the critical misorientation (between adjacent subgrains) of 15° is isotropic.

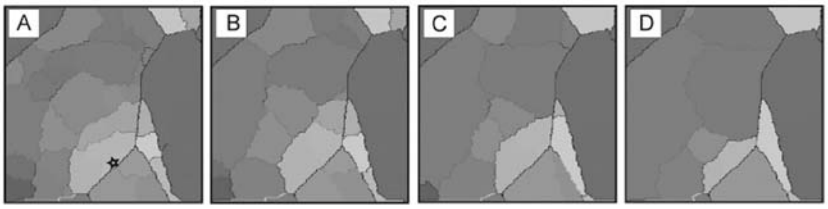


Fig. B5. Example run showing isotropic surface energy driven subgrain growth. Greyscale scheme shows relative misorientation from one crystallographic orientation (marked as black star) in greyscale; black lines signify grain boundaries with >15° misorientation; **(a-d)** results at model time steps 1500, 2500, 4000, 6000 (modified after Piazolo et al., 2004)

Example Elle Run: Subgrain growth – anisotropic In this simulation (Fig. B6) we take into account the anisotropy of surface energy and mobility of subgrain boundaries. The input microstructure is the same as for section 3.3.4, however the calculation of the energy state differs as now the energy between data points below the critical misorientation of 10° is taken to be anisotropic (see above for details). It can be seen that in this case more subgrains remain at the end of the simulation, as the anisotropy has the effect of slowing down the microstructure evolution.

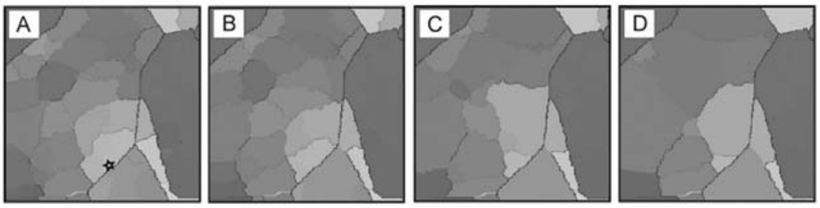


Fig. B6. Example run showing anisotropic surface energy driven subgrain growth. Greyscale scheme shows relative misorientation from one crystallographic orientation (marked as black star) in Greyscale; black lines signify grain boundaries with >10° misorientation; **(a-d)** results at model time steps 0, 20, 40, 60

Functions and parameters for users wishing to modify runtime parameters: This is a process simulates subgrain growth using a Potts model. It only used the unodes in the grains themselves, no migration of flynn_boundaries is taken into account. The user can vary 2 parameters:

1. userdata[0] = max_angle gives the angle at which the maximum energy is reached (e.g. 15° for salt)
2. userdata[1] = slope. A value of slope=1 signifies that all energies are the same, while another value allows differences in energy according to misorientation. Slope=2 signifies that the energy changes linearly, as a fraction of the max_angle. This means that at a misorientation of 10 and max_angle is 15, the energy is 10/15=0.66667. The energy reaches unity when misorient is equal or larger than max-angle. Slope=3 signifies a Shockley equation change between 0 and max_angle misorientation, with:

$$\text{energy} = \left(\frac{\text{orient}}{\text{max\_angle}}\right)\left(1 - \ln\left(\frac{\text{orient}}{\text{max\_angle}}\right)\right)$$

   If the value for slope is not 1, 2, 3 it is set by default to 1. For example: elle_sub_gg -u 15 3 means that the max_angle is 15 (here the energy is 1), and the Shockley equation is used to calculate the energy.

Experiment 7 - Grain growth

The grain growth example shows how the grain size in a grain aggregate increases by grain boundary migration that is solely driven by grain boundary surface energy (see Section 3.5). It also illustrates how a non-equilibrium microstructure evolves towards a foam texture.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. There are two possible sub-experiments, the first simply shows the current position of the grain boundaries, while the second tracks past positions so that the parts of grains previously swept by grain boundaries can be seen:

a without boundary tracking (Fig. B7)

b with boundary tracking (Fig. B8)

Interface: These files will each load their appropriate preferences file automatically. Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

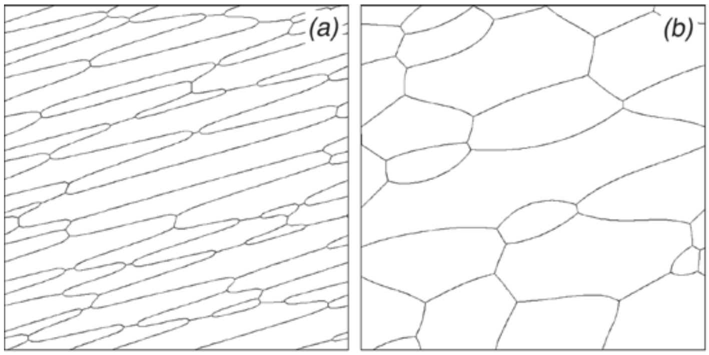


Fig. B7. Example of a grain growth experiment. **(a)** Originally foliated aggregate at *t*=0, and **(b)** at *t*=40000 time steps

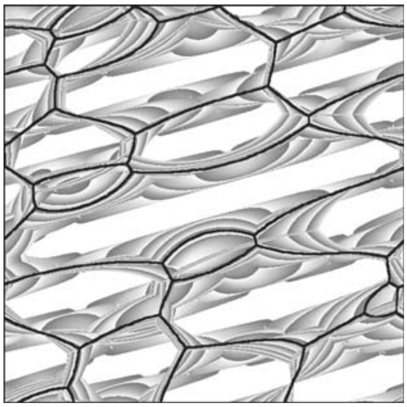


Fig. B8. The successive positions of all grain boundaries at each time step during this time interval are shown in shades of grey that vary cyclically with time. White areas have never been swept by grain boundaries

Examples: In this example (Fig. B7) we will perform a simple grain growth experiment by taking an input file with a microstructure far from equilibrium in terms of grain shapes and evolve the grain boundaries by defining a boundary energy term, which provides the driving force for grain boundary migration. The example can also be seen in Chap. 3.5 (Fig. 3.5.4). Figure B8 is the same experiment, but the past positions of grain boundaries are displayed. This way one can see which parts of the grains have been swept by grain boundaries at least once (Jessell et al., 2003).

Experiment 8 - Evolution of a partial melt

These experiments show how melt pockets evolve for different melt-crystal boundary energies. The example experiment has a solid-solid to solid-liquid surface energy ratio such that the wetting angle is 10°, 60° or 120° at equilibrium. The melt fraction is fixed at 2%. The starting microstructure already shows some disequilibrium features, but most melt pockets have wetting angles of around 10°. During the start of the simulation these melt pockets quickly adjust to an equilibrium shape depending on the wetting angle. This behaviour was predicted by Laporte et al. (1997) for <60° wetting angles in a static system when grains have similar sizes.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. There are three possible sub-experiments, with varying wetting angles for the same starting microstructure:

a 10° wetting angle 🚶
b 60° wetting angle 🚶
c 120° wetting angle 🚶

Interface: These files will each load their appropriate preferences file automatically. Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve. Red areas are melt, blue are crystal. Sometimes two melt pockets may merge, and a boundary will be seen between these pockets, however this does not have any physical significance in our calculations. The image is updated only every 10 timesteps. Note that the simulations take rather a long time.

Example: The example provided here (**c**) has a solid-solid to solid-liquid surface energy ratio such that the wetting angle is 120° at equilibrium (Fig. B9). The melt fraction is fixed at 2%. The starting microstructure is far from equilibrium for a 10° wetting angle. However, during the start of the simulation these melt pockets quickly adjust to an equilibrium shape (convex triangles). This behaviour was predicted by Laporte et al. (1997) for <60° wetting angles in a static system when grains have similar sizes.

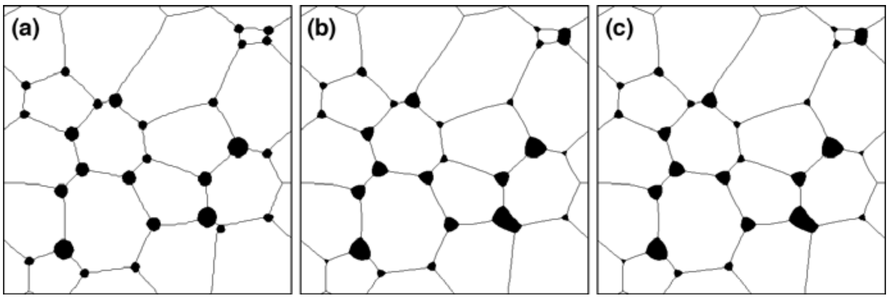


Fig. B9. Three stages from experiment **c**, where melt is black. **(a)** The starting melt pocket shape is strongly out of equilibrium for a 120° wetting angle. **(b)** after 1000 and **(c)** after 5000 steps of boundary movement. Notice the rapid change in melt pocket shape in the first 1000 steps and no visible difference in the next 4000 steps

Experiment 9 - Rigid porphyroblast growing in a deforming matrix

This section describes how to run the *Basil* program in conjunction with the *Elle* software using the specific examples shown in Figs. 3.8.1, 3.8.2 and 3.8.4. We model the development of grain growth during simple shear deformation and a relatively hard porphyroblast rotating within a relatively weak mantle deformed to high strains under simple shear deformation.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments Menu**. The three experiments in this section are:

- **a** Rigid grain in a deforming matrix
- **b** Grain growing in a deforming matrix with viscosity contrast
- **c** High strain experiment with soft mantle

Experiments **b** and **c** will take several hours to run and you can either use *showelle* (see Appendix C) or *Sybil* (see Appendix F) to follow the evolution of the sample. Only *Sybil* can be used to view experiment **a**. This experiment should only take a few minutes and *Sybil* will open automatically if found on your system.

In practice, a shell script (or batch file in Windows) is run, which calls *Basil* and requires an input file containing the parameters needed to run *Basil*, including boundary conditions. For examples **b** and **c**, the script also calls the appropriate *Elle* routines and requires an *Elle* input file containing a description of the initial microstructure (grains, subgrains, etc.).

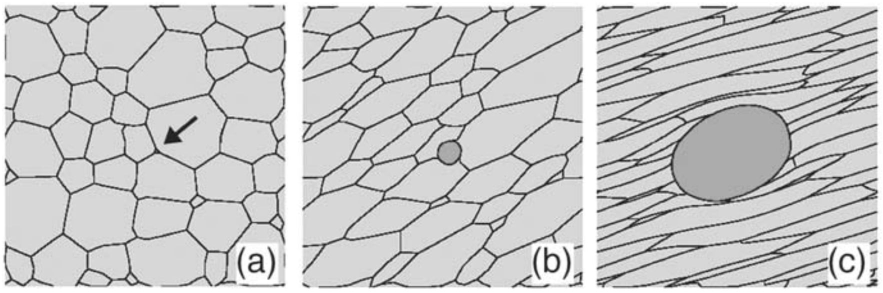


Fig. B10. Experiment 9b in which a central growing grain, with viscosity 5 times that of the matrix, resists deformation. **(a)** Initial microstructure. **(b)** After a strain of 1 and 750 growth steps. **(c)** After a strain of 3 and 2250 growth steps

Examples: In example **a**, we use *Basil* to calculate the velocity around a rigid, circular object in a soft matrix that is subject to simple shear.

In the example shown in Fig. B10, we use *Elle* and *Basil* to study the growth of a porphyroblast in a crystalline matrix that is subject to simple shear. The 2D square domain of unit dimensionless length shows a crystalline matrix with irregular grain size surrounding a small grain of harder material (viscosity coefficient 5, representing garnet). We assume Newtonian constant viscosity (stress exponent n=1) in both materials.

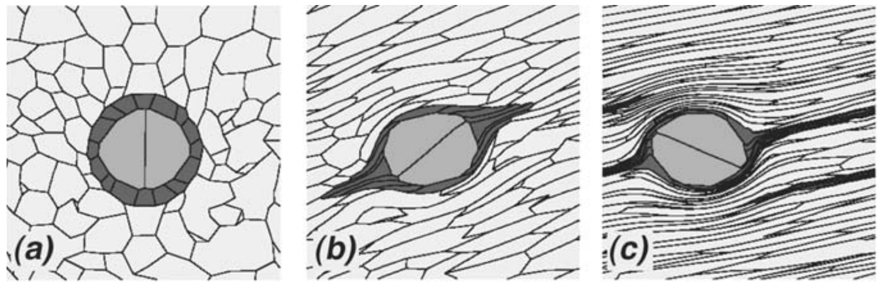


Fig. B11. Experiment 9c. Rotation of a porphyroblast (medium grey: viscosity $50\eta_o$) surrounded by a weak layer of crystals (dark grey: viscosity $0.5\eta_o$) embedded in a matrix (light grey: viscosity η_o) undergoing simple shear. The total finite strain is: **(a)** 0, **(b)** 2, and **(c)** 5. Boundary conditions are as described for Experiment 9b but bnode data is mapped onto a regular *Basil* mesh

In the final example, we show a dynamic shear experiment in which a constant volume porphyroblast that is embedded in a matrix that undergoes a large simple shear strain (Fig. B11). In this case the viscosity of the porphyroblast is 50 times that of the background matrix. A layer of grains whose viscosities are half that of the matrix surrounds the porphyroblast and the mechanical boundary conditions are the same as those for Fig. B11 but there is no grain growth.

Functions and Parameters for users wishing to modify parameters in the Shelle script: The *Elle* shell script file, exp9b.shelle/bat, is generated using the web page shelle24.htm*l,* to set the following parameters that control the *Elle* experiment:

1. Experiment Name exp9b_out, the base for naming saved output files
2. *Elle* Input File exp9b.elle, the *Elle* input file containing the initial microstructure
3. Last Time **150**, the number of time steps for the experiment loop
4. Save Interval **50**, output files from all processes will be saved every 50 time steps with the following processes turned on:
5. *Basil* [1 2 3]
6. *Basil* Input File exp9b.in
7. Reposition [13]
8. Expand [19], Expand Speed **1**, Expand Growth Stages **15**, Expand Max Area **0**
9. The saved output file has the process list: LIST= "**1 2 3 13 19**"

The first three processes in the list are *elle2poly* (1), *Basil* (2), *basil2elle* (3) and are an immutable group included when *Basil* is turned on. Reposition (13) puts the bnodes back into a unit cell and is normally called after any deformation process (e.g. *Basil*, *OOF*, *Manuel*). The last process in the list is *elle_expand* (19) (performed by the routine in the code processes/pblast/expand.elle.cc) and results in growth of any grains with a non-zero EXPAND attribute set in the *Elle* input file. Expand Area is turned off but allows the growth rate to decline exponentially as the grain expands. All processes in the list are called once per time step (the size of which is set by *Basil*), but the Expand process requires a finer discretisation in time than that required by *Basil*, in order to limit the distance a bnode can move in one stage, and thereby avoid instabilities and topological problems due to overstepping. The Expand process is therefore implemented using a number (15 in this example) of sequential Growth Stages in each *Basil* time step (0.02 in the example, as described below). The rate at which a grain boundary moves in the Expand process is the default value of 10^{-4} distance unit per growth stage (0.075 dimensionless velocity units in this example). This rate can be modified by a multiplicative factor set in the Expand Speed line (1 in this example).

Elle input file parameters: The *Elle* input file, exp9b.elle, contains the initial microstructure which includes a grain (grain number 25) with the EXPAND attribute set and a viscosity constant 5 times that of the matrix.

Basil input file parameters: In the *Basil* input file, exp9b.in, the following parameters can be changed to vary the time step and simple shear boundary conditions:

• STEPSIZE	IDT0=50	
• STOP	KEXIT=500	TEXIT=0.02
• SAVE	KSAVE=50	TSAVE=0.02

TEXIT is the dimensionless time level for the *Basil* run. It may be varied but keep TSAVE=TEXIT for the *Elle* experiment so that the *Basil* solutions will contain two records, t=0 and t=TEXIT. IDT0 controls the internal time step for the *Basil* calculations and should be set so that 1/IDT0 <= TSAVE, e.g. if TEXIT and TSAVE are set to 0.01 then set IDT0 to 100.

To change the strain rate for this problem, keep the *y*-component of the velocity, UY, set to zero and vary the *x*-component of the velocity, UX, for the top (Y=YMAX) and bottom (Y=YMIN) boundaries:

```
ON Y = YMIN : UX = -0.5
ON Y = YMIN : UY = 0.0
ON Y = YMAX : UX = 0.5
ON Y = YMAX : UY = 0.0
```

Displaying output: The output of experiment **a** can be viewed using *Sybil*. To reproduce Fig. B10(a), open a terminal or command window, change to the experiment directory, run *Sybil* and open exp9a.log from the File menu.

The output from **b** and **c** can be viewed using the auxiliary program, *showelle*. The graphics preferences will be loaded from the defaults.zip in the corresponding directory when an *Elle* file is opened.

Experiment 10 - Lattice rotations

This section gives details for running the lattice rotation during deformation experiment that is described in section 4.3.

Execution: In order to run one of these experiments (🚶), start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**.

As this is a multi-process experiment, it will start automatically and you can either use *showelle* to view the microstructural evolution (see Appendix C) or *Sybil* (see appendix F) to follow the stress/strain evolution of the sample.

In practice, a shell script (or batch file in Windows) is run, which calls the appropriate *Elle* and *Basil* routines, and requires an *Elle* input file containing a description of the initial microstructure (grains, subgrains, etc.) and a *Basil* input file containing the parameters needed to run *Basil*, including boundary conditions.

Interface: As these are multi-process experiments, they will start automatically and you can either use *showelle* to view the microstructural evolution (see Appendix C) or *Sybil* (see appendix F) to follow the stress/strain evolution of the sample.

To reproduce Fig. B12a-b, use *showelle* to open any *Elle* file generated by this experiment. The Postscript format orientation plots for each *Elle* file may be saved by selecting **Save→Orientation Plots** from the **Graphics menu**.

Example: In this example (Fig. B12) we couple a TBH calculation with the *Basil* FEM code to simulate deformation of a quartz polycrystal, and base the updated viscosities on the work term calculated by the TBH code. In this way grains which are able to deform by glide on slip systems with low Critical Resolved Shear Stress will deform more rapidly than grains which only have 'hard' slip systems available.

Processes: This experiment combines three processes:

1. *Basil*- non-linear viscous flow
2. *ell_tbh*- Taylor-Bishop-Hill lattice reorientation calculations
3. *elle_viscosity*- resetting of each grains viscosity value

***Functions and parameters for users wishing to modify parameters in the Shelle Script*:** exp10.elle is the initial elle file that consists of a simple foam texture with 121 grains. The initial viscosities of all grains is set to 1. exp10.shelle/ba*t* is a batch file generated using Shelle24.html to set the following parameters which control the *Elle* experiment:

1. Experiment Name **tbh**, the base for naming saved output files

2. *Elle* Input File exp10.elle, the initial microstructure
3. Last Time **30**, the number of iterations for the experiment loop
4. Save Interval **1**, output files from all processes will be saved every iteration with the following processes [and their process id] turned on:
5. **Basil** [7 1 2 3 13] The file **exp10.in** controls the *Basil* code, in this series of experiments you can change the stress exponent by altering the value SE on line 10 (this is initially set to 1). The initial time step includes a randomisation [7] of all crystallographic orientations.
6. **elle_tbh** [4] Calculation of lattice rotations and work term
7. **elle_viscosity** [16] ViscosityMode 0 (viscosity is based on work term calculated by TBH code)
8. The saved output file has the process list LIST= "LIST="7 1 2 3 13 4 16""

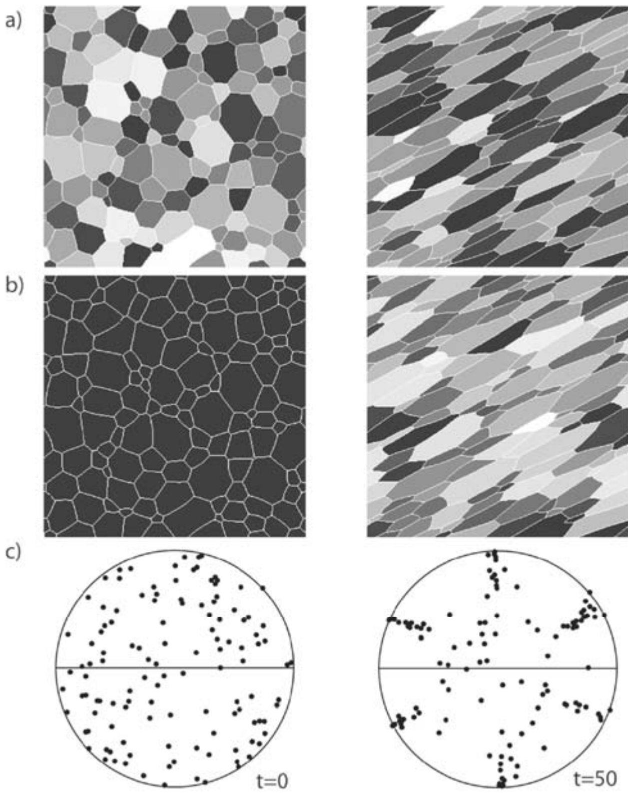


Fig. B12. Simple coupled TBH-FEM calculation. **(a)** Lattice orientations at *t*=0 and *t*=50, grey scale a function of only the Alpha Euler angle. Notice that by *t*=50 many of the grains share a similar orientation. **(b)** Instantaneous viscosities at *t*=0 and *t*=50 (bright=high viscosity) **(c)** c-axes stereograms at *t*=0 and *t*=50, the latter equivalent to a shear strain of 1.5

***Functions and parameters for users wishing to modify parameters in the Shelle Script**:* exp10.elle is the initial elle file that consists of a simple foam texture with 121 grains. The initial viscosities of all grains is set to 1. exp10.shelle/ba*t* is a batch file generated using Shelle24.html to set the following parameters which control the *Elle* experiment:

9. Experiment Name **tbh**, the base for naming saved output files
10. *Elle* Input File exp10.elle, the initial microstructure
11. Last Time **30**, the number of iterations for the experiment loop
12. Save Interval **1**, output files from all processes will be saved every iteration with the following processes [and their process id] turned on:
13. **Basil** [7 1 2 3 13] The file **exp10.in** controls the *Basil* code, in this series of experiments you can change the stress exponent by altering the value SE on line 10 (this is initially set to 1). The initial time step includes a randomisation [7] of all crystallographic orientations.
14. **elle_tbh** [4] Calculation of lattice rotations and work term
15. **elle_viscosity** [16] ViscosityMode 0 (viscosity is based on work term calculated by TBH code)
16. The saved output file has the process list LIST= "LIST="7 1 2 3 13 4 16""

All processes in the list are called once per time step.

quartz.crss This file defines the relative Critical Resolved Shear Stress values for different slip systems. In the example used here they were defined as:

1	5.00	5	∞	9	9.50
2	11.00	6	9.50	10	15.00
3	6.00	7	9.50	11	15.00
4	∞	8	9.50		

where the slip system id numbers, which are defined in quartz.xl are:

1 BASAL A	6 (2-1-1 1) C+A2	10 RHOMB A
2 PRISM C	7 (2-1-1 1) C+A3	11 RHOMB -A
3 PRISM A	8 (2-1-1 1) -C-A2	12 RHOMB C+A
4 PRISM C+A	9 (2-1-1 1) -C-A3	13 RHOMB -C-A
5 PRISM -C-A		

Experiment 11 - Boudinage

This example illustrates the development of boudinage in a granular aggregate during compaction and pure shear deformation. The background of the software is explained in Chap. 3.11 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you should use the lower-resolution files since the algorithm is relatively slow. Possible sub-experiments, which simply vary the density of unodes are:

a 2850 particles
b 11500 particles
c 46000 particles

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

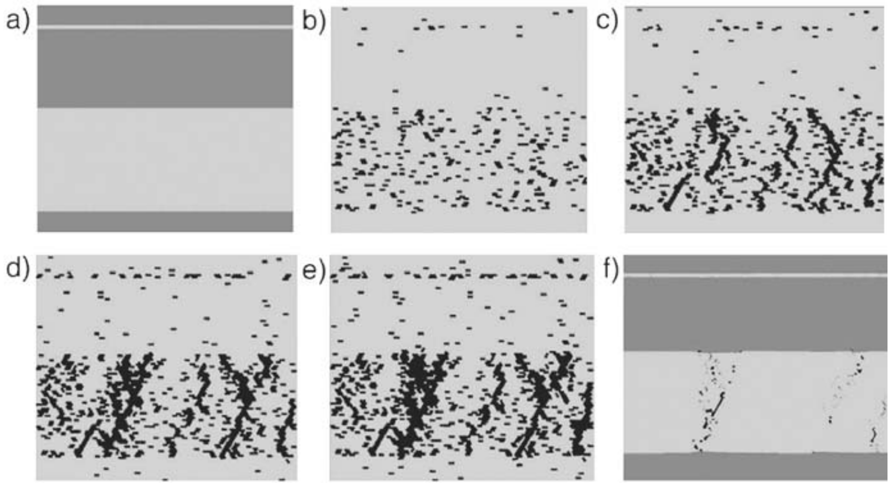


Fig. B13. Fracture development in two competent layers (Young's modulus is 10 times that of the matrix). **(a)** Initial setup plotting the Young's moduli. **(b-e)** fracture pattern after **(b)** 10 steps, **(c)** 20 steps, **(d)** 30 steps and **(e)** 40 steps. **(f)** Final fractures or larger damage zones in the thicker layer (plotting Young's moduli)

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You can now choose to view the Young's moduli of different grains, the fractures and the differential and mean stress. In order to view fractures, choose values between -1.0 and 0.0, unodes with fractured bonds will be blue and unfractured unodes red. Differential and mean stress should be scaled during a run, normal values are: differential stress 0.0 to 0.01 and mean stress - 0.01 to 0.01 (negative is compression).

Examples Figure B13a and B13f show the initial and final geometry with Young's modulus as parameter. Figure B13b-e show the fracture patterns during 40 stages (fractured particles are dark, set fractures to -1 and 0.0). It can be seen that lattice geometry does have an influence on the fracture patterns. The thick layer develops a fracture spacing where two fracture clusters dominate and start to open.
Functions and parameters for users wishing to use the Experiment Interface to modify parameters The parameters used in the example are:

- SetGaussianStrengthDistribution(2.0,0.8),
- WeakenAll(0.1,1.0,1.0,
- WeakenHorizontalParticleLayer(0.9,0.92,10.0,1.0,1.0),
- WeakenHorizontalParticleLayer(0.2,0.6,10.0,1.0,1.0),
- DeformLatticePureShear(0.001,1).

In the initialization function (Experiment::Init(), case 2) the following functions contain parameters that can be changed:

- SetGaussianStrengthDistribution(2.0,0.8) This function is used to set a Gaussian distribution on breaking strengths of all springs. Mean of the distribution is 2.0 and the standard deviation 0.8. The breaking strength of the material is dependent on the lowest breaking strengths of springs so that a variation of the standard deviation will affect the breaking strength of the whole material.
- WeakenAll(0.1,1.0,1.0) This function changes the Young's modulus, viscosity or breaking strength of all particles. In this case the material made softer.
- WeakenHorizontalParticleLayer(0.2,0.6,10.0,1.0, 1.0) This function inserts a horizontal layer of particles that may have different Young's modulus, viscosity and breaking strength than the surrounding matrix. All particles between $x = 0.2$ and $x = 0.6$ are part of the layer. In this example the layer has a Young's modulus that is 10 times that of the surrounding matrix.

- SetFracturePlot(50,1) This function just specifies that a plot is made after 50 bonds are broken. In that case *Elle* stages may not necessarily represent time steps.

In the run function (Experiment::Run(), case 2) the following function contain parameters that can be changed:

- DeformLatticePureShear(0.001,1) Deform the lattice by pure shear deformation with a vertical strain of 0.001.

Experiment 12 - Dissolution grooves

This example illustrates the development of dissolution grooves at a stressed crystal-solute interface. The background of the software is explained in Chap. 3.12 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you should use a medium-resolution file since the algorithm is relatively fast. Possible sub-experiments, which simply vary the density of unodes are:

a	2850 particles ☕
b	11500 particles 🚶
c	46000 particles 🚶

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

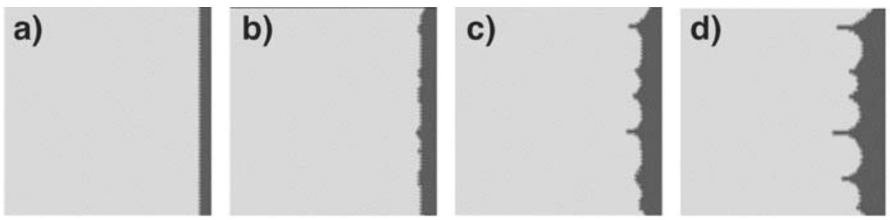


Fig. B14. Development of grooves on a crystal surface after **(a)** 2, **(b)** 15, **(c)** 30 and **(d)** 49 stages. Fluid is black and solid grey. The initial roughness that develops due to the heterogeneous dissolution develops into cusp instabilities with a well-defined wavelength

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You can now choose to view the different phases (from –1 to 1; where the fluid is –1, the interface 0 and the solid 1) or the stress. Differential and mean stress should be scaled during a run, normal values are: differential stress 0.0 to 0.01 and mean stress - 0.01 to 0.01 (negative is compression).

Examples Figure B14 shows the development of grooves on a crystal surface. Input file is res200.elle. Parameters used are the ones given above. The solid in Fig. B14 is grey and the fluid black. The figure shows the roughening of the surface (Fig. B14b) and development of cusps after 30 stages (Fig. B14c), which have developed a well-defined wavelength after 49 stages (Fig. B14d).

Functions and parameters for users wishing to use the Experiment Interface to modify parameters: In the initialization function (Experiment::Init(), case 6) the following functions contain parameters that can be changed:

- SetPhase(0.0,0.0,500.0,0.8*)* This function makes the lattice unbreakable since fractures are not wanted in the dissolution processes (at least in this example).
- SetGaussianRateDistribution(2.0,0.3) Set a Gaussian distribution on rate constants in order to induce the roughening. Mean is 2.0 and standard deviation 0.3.
- WeakenAll(8.0,1.0,1.0) Multiply spring constants by 8.
- Set_Mineral_Parameters(1) Mineral is quartz (set molecular volume and surface free energy)
- Set_Absolute_Box_Size(0.0001) x dimension of the simulation box is 0.0001 m.
- Set_Time(6000.0,4) time of one deformation step is 6000 years.
- DissolveXRow(0.95,1.1) Dissolve particles between $x = 0.95$ and $x = 1.1$.
- In the run function (Experiment::Run(), case 6) the following functions contain parameters that can be changed:
- DeformLattice(0.002,1) Deform the lattice by uniaxial vertical compression with confined walls on the sides and a vertical strain of 0.002.
- Dissolution_Strain(20) Dissolve as a function of elastic and surface energies and plot a picture after 20 particles have dissolved.

Experiment 13 - Stylolites

This example illustrates the development of stylolites at a stressed crystal-solute interface. The background of the software is explained in Chap. 3.12 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you should use the low-resolution files since the algorithm is relatively slow. Possible sub-experiments, which simply vary the density of unodes are:

a 2850 particles ☕
b 11500 particles 🚶
c 46000 particles 🚶

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You should now choose to view the phases, where –1 is the fluid, 0 the interface and 1 the solid.

Examples Figure B15 illustrates an example of stylolite development in a lattice that has a horizontal resolution of 100 particles. Parameters in the functions are the same as above. The initially flat surface roughens during progressive dissolution due to the heterogeneities in dissolution constants. The absolute size of the box is large (0.1 m horizontally) so that surface energy is rather low and elastic effects dominate. Therefore some relatively steep structures develop that are comparable with the characteristic stylolite "teeth".

Figure B16 illustrates roughening of a stylolite with the same properties as the stylolite shown in Fig. B15 except for the dimension of the box, which is only 1 mm horizontally. Therefore surface energy effects are more pronounced so that the structures appear more rounded.

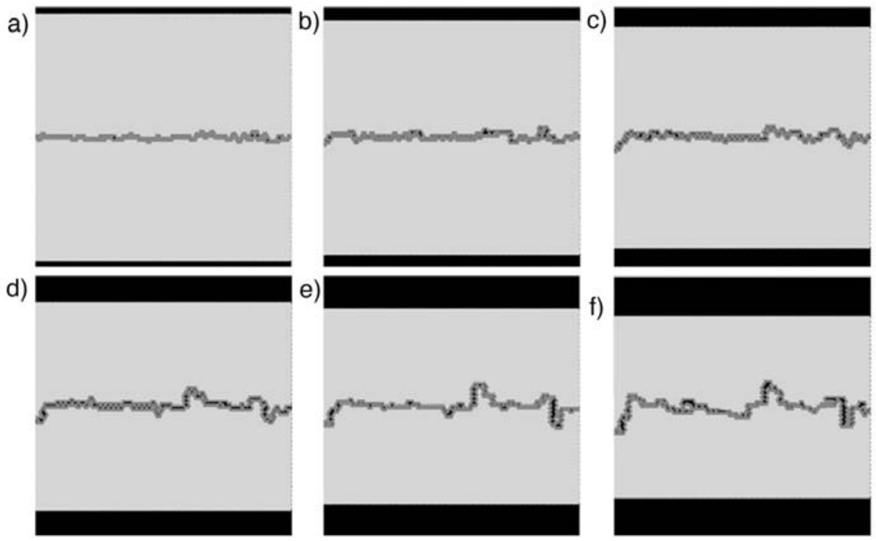


Fig. B15. Development of the roughness of a stylolite. Stages are **(a)** 500, **(b)** 1000, **(c)** 1500, **(d)** 2000, **(e)** 2500 and **(f)** 3000. The horizontal dimension of the box is 0.1 m

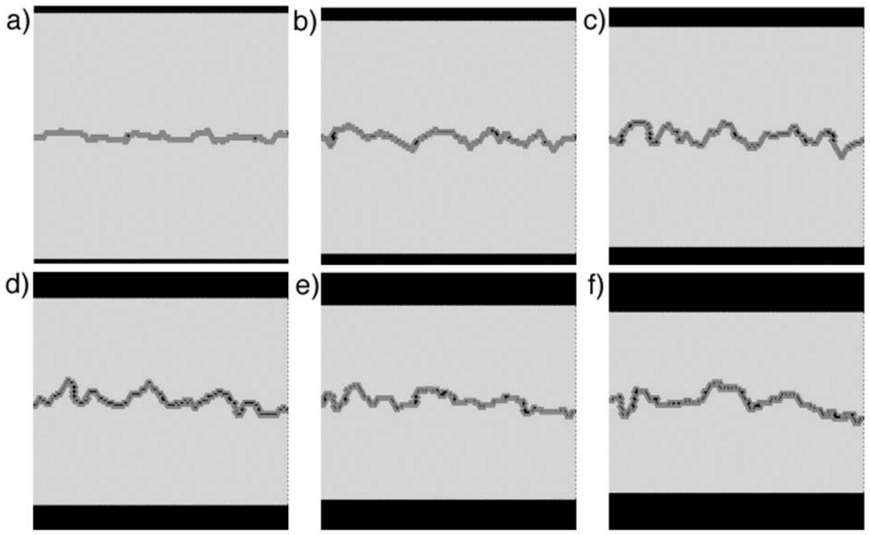


Fig. B16. Development of the roughness of a stylolite where the horizontal dimension (1 mm) of the box is two orders of magnitude smaller than that of Fig. 13.1. Stages are **(a)** 500, **(b)** 1000, **(c)** 1500, **(d)** 2000, **(e)** 2500 and **(f)** 3000

Functions and parameters for users wishing to use the Experiment Interface to modify parameters: In the initialization function (Experiment::Init(), case 7) the following functions contain parameters that can be changed:

- SetPhase(0.0,0.0,500.0,0.8) As in the groove case this turns fracturing off.
- SetGaussianRateDistribution(2.0,0.5) Set a Gaussian distribution on dissolution constants, 2.0 is mean of the distribution and 0.5 the standard deviation.
- Set_Rate_Two_Phase(0.05,0.6,1.0) Set a bimodal distribution on the dissolution constants of particles. 5% of all particles dissolve a factor 0.6 slower than the rest.
- WeakenAll(4.0,1.0,1.0) Multiply the spring constants by 4.
- Set_Mineral_Parameters(1) Mineral is quartz (set molecular volume and surface free energy)
- Set_Absolute_Box_Size(0.1) x dimension of box is 0.1 m.
- Set_Time(40.0,4*)* One time step represents 40 years.
- DissolveYRow(0.49,0.5, true) Dissolve a row of particles in the centre between $y = 0.49$ and $y = 0.5$.True means that the dissolved particles are removed from the picture.

In the run function (Experiment::Run(), case 7) the following functions contain parameters that can be changed:

- DeformLatticeNewAverage2side(0.00005,1) Deform the lattice by pushing upper and lower crystal together.
- Dissolution_StylosII(100000,0,0,0,1) Dissolve as a function of stress and elastic and surface energy.

Experiment 14 - Strain-rate partitioning during porphyroblasts growth

This section explores the evolving strain-rate partitioning behaviour within a layered succession during porphyroblast growth, and explore the feedback between strain-rate partitioning around effectively rigid porphyroblasts and metamorphic reactions, as described in section 4.8.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**.

Possible sub-experiments are:

- **a** porphyroblast growth, no deformation
- **b** deformation, but no porphyroblast growth
- **c** porphyroblast growth, with deformation

As these are multi-process experiments, they will start automatically and you can either use *showelle* to view the microstructural evolution (see Appendix C) or *Sybil* (see appendix F) to follow the stress/strain evolution of the sample.

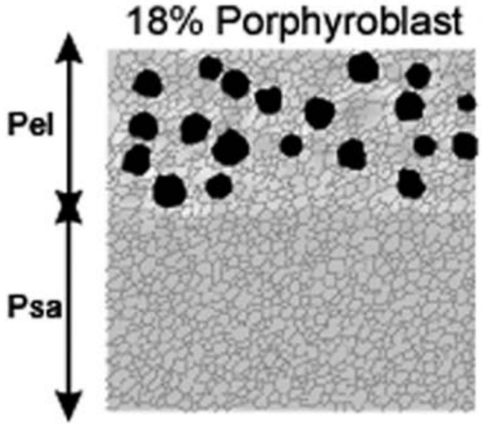


Fig. B17. Porphyroblast growth without deformation (Experiment a)

Examples: These examples compare porphyroblast with and without deformation, and deformation with and without porphyroblasts. The first experiment simply shows the growth from small nuclei of a set of porphyroblasts (Fig. B17). In the following two experiments (Fig. B18), two geometries were deformed in simple shear to bulk strains of approximately 1.5 to illustrate the difference between strain partitioning in a porphyroblast-free layered system and a porphyroblastic layered system. In the absence of porphyroblasts, the pelitic layer in our model has a viscosity 0.4 times that of the psammitic layer. Strain is partitioned in this model such that the pelitic layer records a shear strain of approximately 2.1 and the psammitic layer records a shear strain of approximately 0.8, consistent with the viscosity contrast between the two layers. A second geometry, with approximately 18% porphyroblast in the pelitic layer, was deformed to a similar bulk strain.

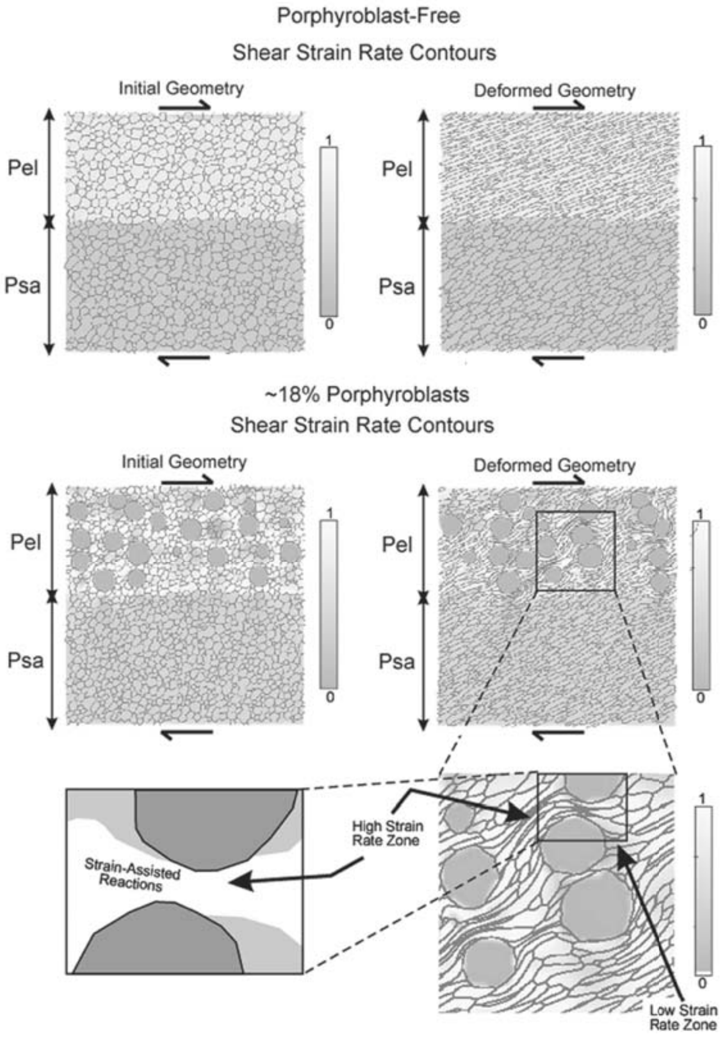


Fig. B18. Top: Shear strain rate contour maps for the porphyroblast-free simple shear deformation experiment b. Note the homogeneous shear strain rate in each layer and the bulk strain rate partitioning into the pelitic layer. Middle: Shear strain rate contour maps for the 18% porphyroblast simple shear deformation experiment c. Note strain rate partitioning within the pelitic layer with high strain rate zones around the porphyroblasts. Also, the strain rate is higher in the psammitic layer than in the porphyroblast-free experiment. Bottom: Close-up images of the shear strain rate distribution around porphyroblasts showing high and low shear strain rate zones. The image on the left has had the grain topology removed for simplification. The location of strain-assisted reaction between two porphyroblasts is indicated

Interface: Strain-rate maps and viscosity profiles are created using *Sybil*, and the evolving topology can be viewed using *showelle*.

To reproduce strain-rate maps, open any solution file generated by the shelle script in *Sybil*. In *Sybil*, select **contour → strain → msst**. To reproduce the figures used in Chap. 4.8, set the options to "min = 0, max = 2. " To produce viscosity profiles through the model, open any solution in *Sybil*. In *Sybil*, select **profile → 2-D → stress → visc**. Set the lower limit to 0 and the upper limit to 1 (in the Y-direction) to generate a profile through both layers in the model. To generate a profile through just the upper pelite layer, set the limits to min=0.5, max=1.0. Set the minimum scale value to 0 and the maximum value to 2 (note the scale is $\log_{10}$ of the viscosity).

Processes: This is a multi-stage experiment using the following processes:

1. ***Basil*:** The models in Chap. 4.8 use linear viscous flow
2. ***elle_gg***: curvature-driven grain boundary migration
3. ***elle_pblast***: selected grains grow as porphyroblasts. To reproduce the syn-deformation growth experiments in Chap. 4.8, the following parameters are used in the shelle script: Lasttime=030, Timestep=001, GGstages=003 (Process [18]), PblastStages=015 (Process [8]), SaveInterval=1, List= "1 2 3 8 18 13".

Modifying the Shelle Script: A single deformation step is modelled using a .shelle script with the following parameters:

- Startfile = the name of the porphyroblast growth stage you want to deform,
- Time = 0,
- LastTime = 1,
- GGStages = 0,
- Saveinterval = 1,
- SaveAll = 13,
- Basinfile = exp14b.in,
- Processes: "1 2 3 13"

Experiment 15 - Anisotropic grain growth

This section gives details for running the anisotropic grain growth experiments that are described in section 4.1. Be warned that these experiments each take several hours to run; overnight experiments often prove to be the least disruptive.

Processes: This experiment use only the *elle_gbm* process, in its earlier version, see Chap. 3.5 for details.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. Possible sub-experiments, which simply vary the amount of anisotropy are:

a No boundary anisotropy
b 10% Anisotropy
c 20% Anisotropy
d 30% Anisotropy
e 40% Anisotropy
f 50% Anisotropy

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

Interface: As this is a single process experiment, the easiest way to monitor the progress is to view the generated *Elle* files directly in *elle_gbm*. In addition, every 1000 time steps an *Elle* file will be saved.

Example: An initially randomly oriented aggregate consisting of 315 grains (and hence an average grain area of $3.17 \cdot 10^{-3}$ (the entire model has unit area) with a foam texture, was allowed to grow for between 100,000 and 300,000 time steps (this may take several hours). Experiment **a**, which has an isotropic surface energy distribution, produces a foam texture with triple junction angles of about 120°. At a first glance, the microstructure in the other experiments looks like a foam texture, but most triple junction angles are in fact far from 120 degrees (Fig. B19).

The rate of grain growth in all experiments with a surface energy anisotropy is vastly decreased relative to an identical experiment where surface energies were assumed to be isotropic (A), and experiments with a high boundary energy anisotropy quickly reach an apparently stagnant grain size. After a considerable time in the experiments with higher surface energy anisotropies, a second phase of grain growth commences, dominated by the exaggerated growth of just one or two grains (Fig. B20).

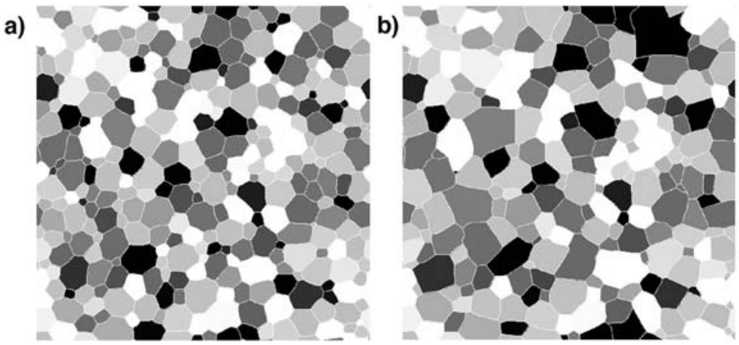


Fig. B19. Experiment F. **(a)** Time step 0, shading reflect lattice orientation. **(b)** Time step 100,000. Most triple junction angles reflect surface energy anisotropy by being far from 120 degrees

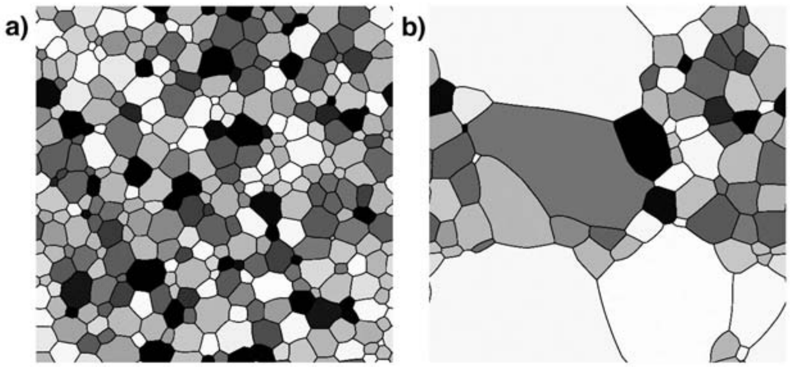


Fig. B20. Experiment C. **(a)** Time step 0, shading reflects lattice orientation **(b)** Time step 200,000

Experiment 16 - Dynamic recrystallisation

In the model presented here, we can simulate dynamic recrystallisation during plane strain simple shear deformation of a polycrystal, involving crystal lattice rotation, formation of subgrains, recrystallisation by nucleation, recrystallisation by subgrain rotation (also called rotational recrystallisation), grain boundary migration and recovery (see Section 4.2 for more details, but the complete list of parameters used and reasoning behind those the reader is referred to Piazolo (2001) and Piazolo et al. (2002). Three coarse-grained starting microstructure files and three fine-grained files are provided, each at three different grain boundary mobilities ($1\cdot10^{-12}$ $20\cdot10^{-12}$ and $40\cdot10^{-12}$).

***Execution*:** In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. Possible sub-experiments are:

a fine grained with low mobility
b fine grained with medium mobility
c fine grained with high mobility
d coarse grained with low mobility
e coarse grained with medium mobility
f coarse grained with high mobility

As these are multi-process experiments, they will start automatically and you can either use *showelle* to view the microstructural evolution (see Appendix C) or *Sybil* (see appendix F) to follow the stress/strain evolution of the sample.

Interface: In practice, a shell script (or batch file in Windows) is run, which calls the appropriate *Elle* and *Basil* routines, and requires an *Elle* input file containing a description of the initial microstructure (grains, subgrains, etc.) and a *Basil* input file containing the parameters needed to run *Basil*, including boundary conditions.

Examples: To investigate the effect of the relative rates of the different microscale processes active during dynamic recrystallisation experiments were performed in which one parameter was varied. The grain boundary mobility (*GBMob*) was chosen to be $1\cdot10^{-12}$, $20\cdot10^{-12}$ or $40\cdot10^{-12}$ $m^2s^{-1}J^{-1}$. The interested reader is referred to Piazolo (2001) and Piazolo et al. (2002) for further simulations the energy threshold value for recrystallisation by nucleation was also varied. Two different starting microstructures were used: one fine-grained and one-coarse grained aggregate Figure B21 shows the development of the microstructure in simple shear at a shear srain of 0.25 and 2, showing both the grains as well as their dislocations densities. A variety of image sequences of developing microstructures during progressive deformation are provided at

www.uni-mainz.de/FB/Geo/Geologie/tecto/elle_movies._

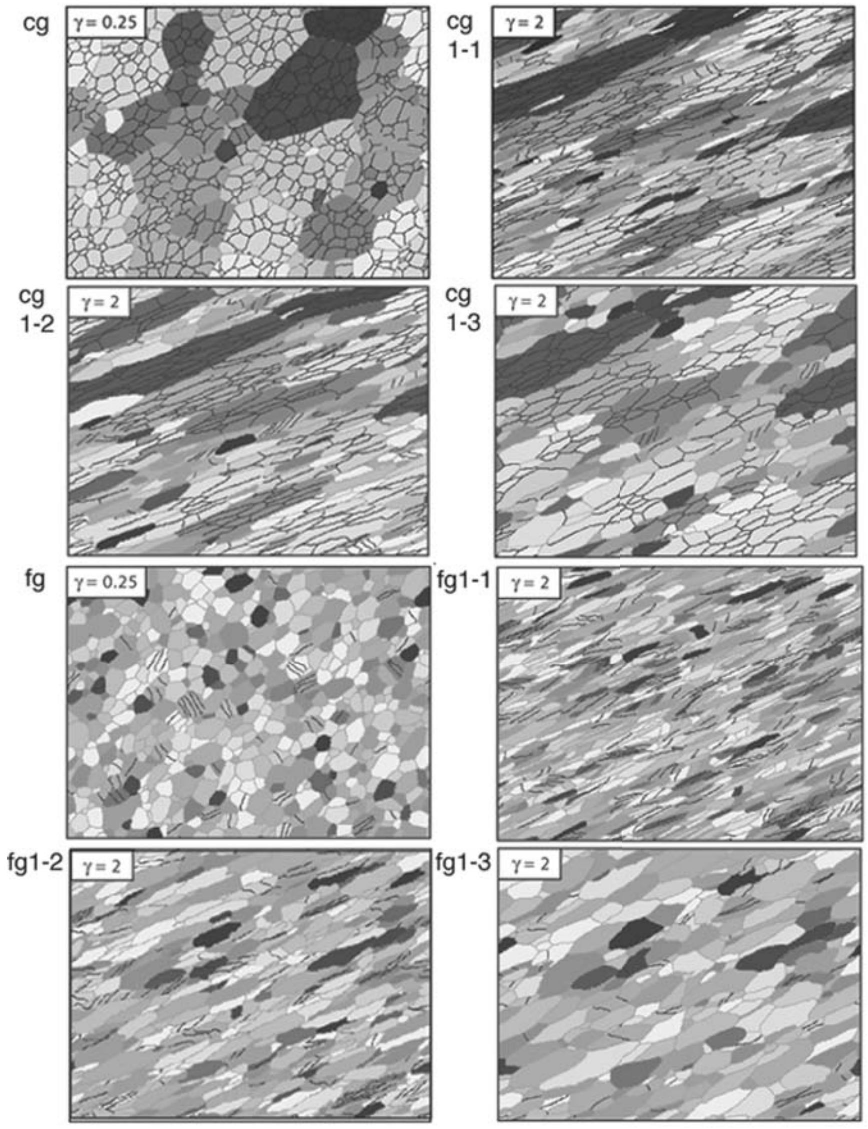


Fig. B21. Results of the 6 simulation runs. Different grey colours signify different grains, dark boundaries low angle boundaries and light grey coloured boundaries high angle boundaries. For the naming of simulation runs refer to Table 4.2.1 (modified from Piazolo 2001)

Processes: This experiment combines several processes:

1. ***Basil***- non-linear viscous flow
2. ***elle_split***- creation of new subgrain boundaries
3. ***elle_gbm***- curvature and defect energy driven grain boundary migration
4. ***elle_tbh***- calculates lattice rotations
5. ***elle_nucl_xx***- nucleation of new grains
6. ***elle_angle_rx***- nucleation of new grains by rotation recrystallisation
7. ***elle_recovery***- reduction in defect energies within grains
8. ***elle_viscosity***- resetting of each grains viscosity value

Experiment 17 - Deformation localisation

This section gives details for running the deformation localisation experiment which is described in section 3.9. As this experiment depends on probabilistic calculations, no two experiments will run the same, and its long-term evolution may be quite different from that shown in the figures.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. The three experiments in this section are:

a Stress Exponent of 1, no Grain Size Modifying Processes ☕
b Stress Exponent of 3, no Grain Size Modifying Processes ☕
c Stress Exponent of 1, with Grain Size Modifying Processes ☕
d Stress Exponent of 3, with Grain Size Modifying Processes 🚶

As these are multi-process experiments, they will start automatically and you can either use *showelle* to view the microstructural evolution (see Appendix C) or *Sybil* (see appendix F) to follow the stress/strain evolution of the sample.

In practice, a shell script (or batch file in Windows) is run, which calls the appropriate *Elle* and *Basil* routines, and requires an *Elle* input file containing a description of the initial microstructure (grains, subgrains, etc.) and a *Basil* input file containing the parameters needed to run *Basil*, including boundary conditions.

Examples: We simulate a simple system in which three processes are active. We stress that these are not supposed to accurately represent the detailed behaviour of any single deformation process, but rather act as proxies for three classes of processes, namely a grain-size and strain dependant rheology, a process that increases the grain size, and a process that reduces the grain size (see Jessell et al. (2005) for a complete set of the experimental parameters).

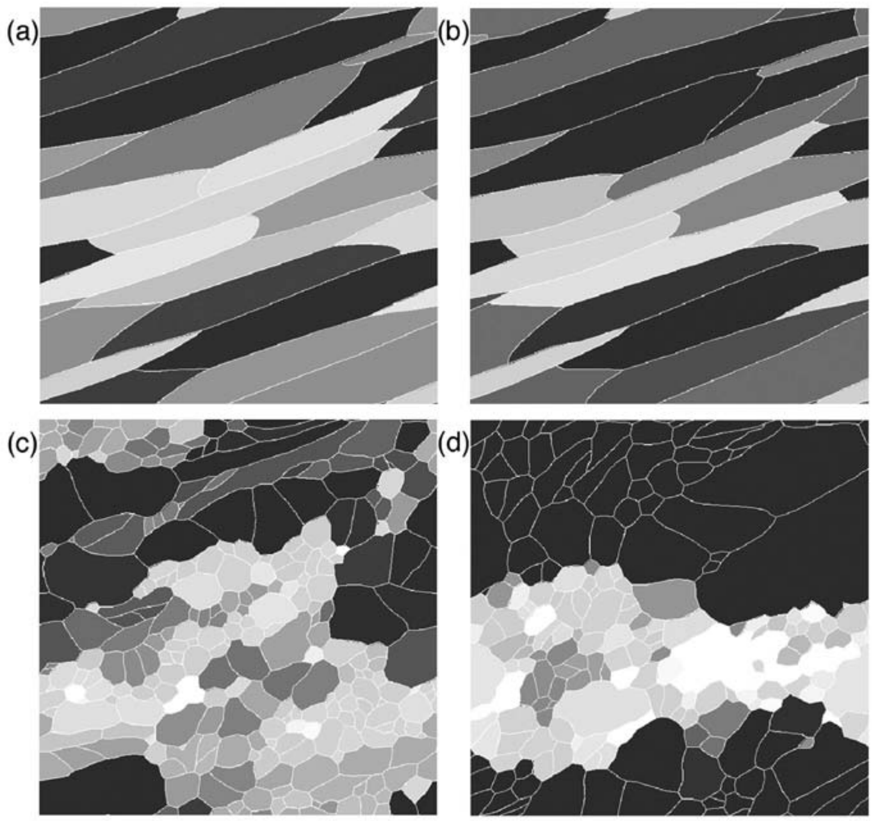


Fig. B22. Maps of the average finite first principal strain for each grain during time step 50 for each of the four experiments. The shading in each map is proportional to accumulated strain, overlaid by the grain boundary network as white lines. The same look up table is used for all four maps. **(a)** Experiment A, showing a relatively homogeneous strain distribution, with a concentration of deformation in the small grain in the top-right. **(b)** Experiment B, showing a relatively homogeneous strain distribution, but with a concentration of deformation in the small grain in the top-right. **(c)** Experiment C, showing a broad zone of higher strain. **(d)** Experiment D, showing a broad sub-horizontal zone of high strain. Notice that the high strain rate zones in Fig. 4.3.2d are not all high finite strain zones in this figure, and that the zone of bulk finite strain is much more clearly defined than the equivalent instantaneous high strain rate zone

A single experiment consists of defining a starting microstructure, then cycling this microstructure through the three processes, so that the microstructure is slightly modified by each process in turn. In this sequence of experiments we chose to change only two parameters. The first of the two parameters is the stress exponent of the grain size dependent flow law, which could have values of 1 or 3. The second variable is the activation or non-activation of the Grain Size Reduction and Grain Size Increase processes. We thus define four Experiments A, B, C and D (Table 4.3.1).

These four simple experiments demonstrate a range of behaviours (Fig. B22), a monotonic evolution for Experiments A and B versus a more complex path for Experiments C & D. As one might expect, in these experiments, the more non-linear the system, either in terms of the rheology or the process coupling, the more complex the behaviour.

Processes: This experiment combines four processes:

1. ***Basil***- non-linear viscous flow
2. ***elle_split***- creation of new high angle grain boundaries
3. ***elle_gg***- curvature driven grain boundary migration and
4. ***elle_viscosity***- resetting of each grains viscosity value

Functions and parameters for users wishing to modify parameters in the Shelle Script: exp17.elle, the initial elle file that consists of a simple foam texture with 16 grains. The initial viscosity of all grains is set to 1. exp17.shelle/bat is a batch file generated using Shelle24.html to set the following parameters which control the Elle_experiment:

1. Experiment Name exp_17a, the base for naming saved output files
2. *Elle* Input File loca.elle, the initial microstructure
3. Last Time **100**, the number of iterations for the experiment loop
4. Save Interval **5**, output files from all processes will be saved every fifth iteration with the following processes [and their process id] turned on:
5. ***Basil*** [1 2 3 13] The file exp17.in controls the Basil code, in this series of experiments you can change the stress exponent by altering the value SE on line 10 (this is initially set to 3 in experiments (b) and (d)).
6. ***elle_split*** [5] SplitMode 1 (split probability is based on bulk strain of each grain)
7. ***elle_gg*** [18] GGStages 010
8. ***elle_viscosity*** [16] ViscosityMode 1 (viscosity is based on grain area and bulk strain of each grain)

9. The saved output file has the process list LIST= "1 2 3 13 5 18 16" to modify this experiment to run without grain size modifying processes, change this to LIST= "1 2 3 13 16".

All processes in the list are called once per time step.

Experiment 18 - Expanding inclusions

This example illustrates single grains expanding in a brittle matrix. The background of the software is explained in Chap. 4.4.3 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. On most computers you should use the medium-resolution files since the algorithm is relatively slow. Possible sub-experiments, which simply vary the density of unodes are:

a 2850 particles
b 11500 particles
c 46000 particles

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You can now choose to view the Young's moduli of different grains, the fractures and the differential and mean stress. In order to view fractures choose values between -1.0 and 0.0, unodes with fractured bonds will be blue and unfractured unodes red. Differential and mean stress should be scaled during a run, normal values are: differential stress 0.0 to 0.01 and mean stress - 0.01 to 0.01 (negative is compression).

Example Figure B23 shows the fracture pattern around the inclusion after 15 and 30 steps, as well as the mean and differential stress. Values for mean stress range from -0.03 to 0.005 and values for differential stress range from 0 to 0.03. Figure (d) shows the fracture pattern after 30 steps and figure (e) and (f) the corresponding mean and differential stress using the same values.

The example shows the fracture and stress pattern around an expanding grain or inclusion. Input file is res100.elle.

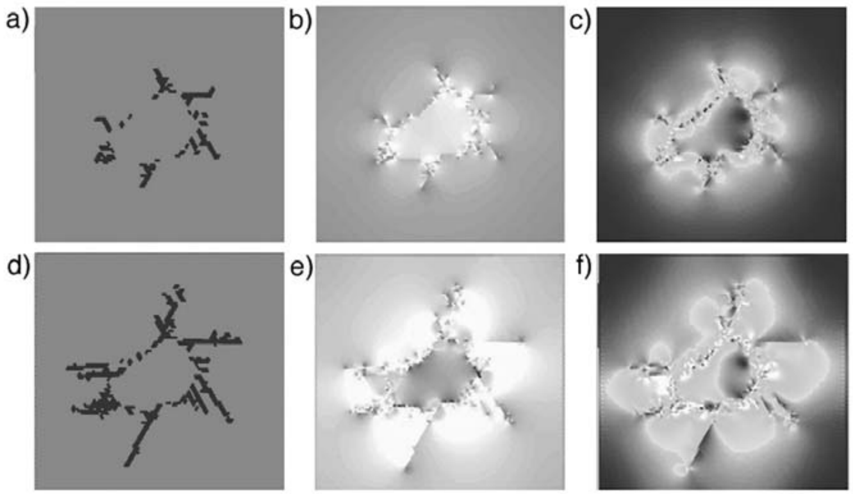


Fig. B23. Expanding inclusions, associated fracture and stress patterns. **(a)** Fracture pattern after 15 steps. **(b)** Mean stress after 15 steps, ranging from -0.03 (bright?) to 0.005 (dark). **(c)** Differential stress after 15 steps, ranging from 0 (dark) to 0.03 (bright). **(d)** Fracture pattern after 30 steps, fractured particles are dark. **(e)** Mean stress after 30 steps and **(f)** differential stress after 30 steps. Mean stress values are from –0.008 to 0.0001 and differential stress values from 0.0 to 0.02

Functions and parameters for users wishing to use the Experiment Interface to modify parameters In the initialization function (Experiment::Init(), case 3) the following functions contain parameters that can be changed:

- SetPhase(0.0,0.0,0.8,1.2) Set a linear distribution on breaking strength of single bonds. Mean is 0.8 times the default value. Distribution will be from default value ± 0.6 times default value.
- SetFracturePlot(20,1) Plot a picture after 20 particles are fractured.

In the run function (Experiment::Run(), case 3) the following functions contain parameters that can be changed:

- ShrinkGrain(5,-0.002,1) Shrink grain number 5 by the amount -0.002 times area of grain and plot a picture.

Experiment 19 - Mud cracks

This example illustrates the development of shrinkage cracks in a brittle medium. The background of the software is explained in Chap. 4.4.3 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. As input file you should use a rather low-resolution file unless you have a very fast computer (**a** & **b** work fine, **c** is ok, **d** becomes problematic and **e** takes a long time even on a fast computer), **c** gives good results in reasonable time (several hours). Possible sub-experiments, which simply vary the density of unodes are :

a 2850 particles ☕
b 11500 particles 🚶
c 46000 particles 🚶

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve.

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You can now choose to view the Young's moduli of different grains, the fractures and the differential and mean stress. In order to view fractures, choose values between -1.0 and 0.0, unodes with fractured bonds will be blue and unfractured unodes red. Differential and mean stress should be scaled during a run, normal values are: differential stress 0.0 to 0.01 and mean stress - 0.01 to 0.01 (negative is compression).

Examples: Several stages in the development of a mud crack are shown in Fig. B24. After nucleation, the crack propagates through the whole model (Fig. B24a-c), when a regular spacing develops. This spacing subsequently decreases (Fig. B24d-f). The typically hexagonal fracture pattern develops that is well known from mud-cracks or columnar basalts.

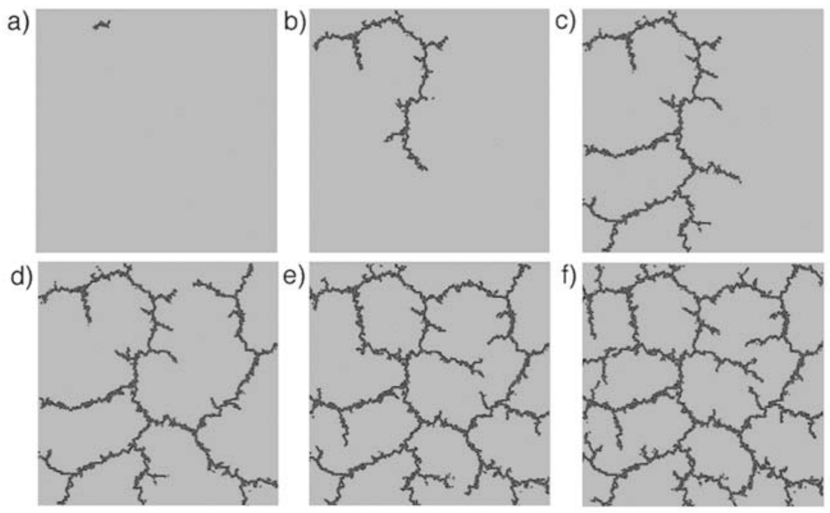


Fig. B24. Mud cracks after stages **(a)** 10, **(b)** 50, **(c)** 100, **(d)** 150, **(e)** 200 and **(f)** 240. Cracks develop a regular spacing with a hexagonal symmetry

The example in Fig. B 19.1 shows the progressive development of fractures in a shrinking layer (for example mud-cracks). The input file is res200.elle.

Functions and parameters for users wishing to use the Experiment Interface to modify parameters In the initialization function (Experiment::Init(), case 4) the following functions contain parameters that can be changed:

- WeakenAll(2.0,1,1.0) Change spring constants of all springs (multiply by 2)
- SetPhase(0.0,0.0,1.0,1.0) Set a linear distribution on breaking strength of single bonds.
- SetFracturePlot(20,1) Make a plot after 20 bonds are fractured

In the run function (Experiment::Run(), case 4) the following functions contain parameters that can be changed:

- ShrinkBox(0.001,1,1) Shrink all particles in the box by the amount 0.001 times area.

Experiment 20 - Visco-elastic deformation and fractures

This example illustrates the development of structures in visco-elastic materials. The background of the software is explained in Chap. 4.6 and is based on a Lattice-Spring model (Chap. 2.8).

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. As input file you should use a rather low resolution file unless you have a very fast computer (**a** & **b** work fine, **c** is ok, **d** becomes problematic and **e** takes a long time even on a fast computer), **c** gives good results in reasonable time (several hours). Two sets of possible sub-experiments, which simply vary the density of unodes are provided for the case of a single brittle grain in a viscous matrix, and for a brittle layer in a viscous matrix:

a	brittle grain 2850 particles 🚶
b	brittle grain 11500 particles 🚶
c	brittle grain 46000 particles 🚶
d	brittle layer 2850 particles 🚶
e	brittle layer 11500 particles 🚶
f	brittle layer 46000 particles 🚶

Once the file is loaded, select **Run** from the **Run menu** of the *Elle* window to watch the system evolve. Visco-elastic deformation takes longer to calculate than purely elastic deformation!

Interface: These files will each load their appropriate preferences file automatically, and only the unodes will be visible. You can now choose to view the Young's moduli of different grains, the fractures and the differential and mean stress. In order to view fractures, choose values between -1.0 and 0.0, unodes with fractured bonds will be blue and unfractured unodes red. Differential and mean stress should be scaled during a run, normal values are: differential stress 0.0 to 0.01 and mean stress - 0.01 to 0.01 (negative is compression).

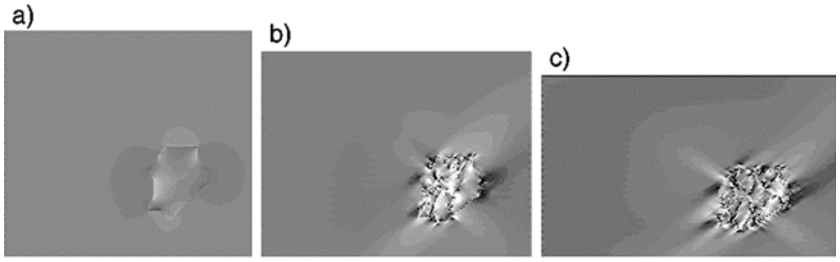


Fig. B25. Viscoelastic deformation of a hard grain in a more viscous matrix. Plot shows the mean stress (values -0.04 to 0.01) where light colour is high compressive stress (negative in this case) and dark colour low compressive stress. The hard grain is deforming by internal fractures that are opening whereas the viscous matrix flows around it without fracturing

Examples: The first example of visco-elastic deformation and fracturing is illustrated in Fig. B25 where a hard grain is embedded in a weaker matrix. The input file is res100.elle.

During the simulation the hard grain fractures whereas the matrix flows around the hard grain without fracturing. Plotted is the mean stress where light colour is large compressive stress and dark colour low compressive stress. Fractures in the hard grain open and may be filled in nature with vein material.

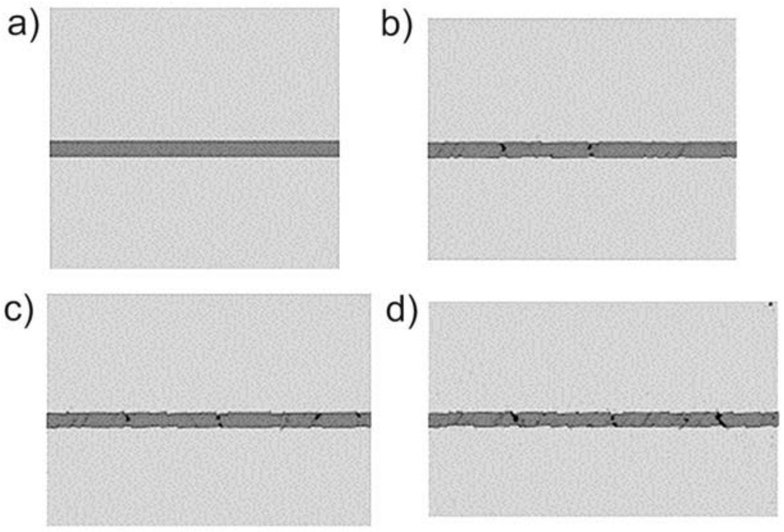


Fig. B26. Viscoelastic deformation leading to the formation of rotating and folding boudins. Steps are **(a)** 20, **(b)** 80, **(c)** 120 and **(d)** 180

Figure B26 shows the progressive deformation of the matrix and the layer. The more competent layer fractures and folds during extension. Small flanking folds develop and the boudins are slightly rotating and necking. In the end some of the larger fractures are opening between boudins.

Functions and parameters for users wishing to use the Experiment Interface to modify parameters The parameters for the brittle grain example are:

- SetWallBoundaries(1,1.0),
- SetPhase(0.0,0.0,0.2,1.2),
- WeakenAll(0.8,1.1,1.0),
- WeakenGrain(7,10,10,1),
- DeformLatticePureShear(0.001,1),
- ViscousRelax(1,1e11).

The parameters for the brittle layer example are:

- SetWallBoundaries(1,1.0),
- SetPhase(0.0,0.0,1.0,0.6),
- WeakenAll(0.1,1.1,1.0),
- WeakenHorizontalParticleLayer(0.47,0.52,5.0,5.0,1),
- DeformLatticePureShear(0.001,1),
- ViscousRelax(1,1e11).

In the initialization function (Experiment::Init(), case 5) the following functions contain parameters that can be changed:

- SetWallBoundaries(1,1.0) Set external boundary walls that repel particles. 1 means leave the particle walls and 1.0 is the repulsion constant of the walls.
- SetPhase(0.0,0.0,0.2,1.2) Set a linear distribution on breaking strength of bonds. Mean multiplied by 0.2 and 1.2 is the size of the distribution.
- WeakenAll(0.8,1.1,1.0) Change elastic constants of all particles (multiply by 0.8)
- WeakenGrain(7,10,10,1) Change elastic constant, viscosity and breaking strength of grain number 7. First number behind the grain number is the elastic constant (times 10), second number the viscosity (times 10) and last number the breaking strength (no change, times 1).
- WeakenHorizontalParticleLayer(0.47,0.52,20.0, 20.0,1.0) Define a horizontal layer between $y = 0.47$ and y is 0.52 with an elastic constant that is 20 times higher than the matrix, a viscosity that is 20 times higher and a remaining breaking strength (times 1.0).

- SetFracturePlot(50,1) Make a plot after 50 fractures developed.

In the run function (Experiment::Run(), case 5) the following functions contain parameters that can be changed:

- DeformLatticePureShear(0.001,1) Deform the lattice by pure shear deformation with a vertical strain component of 0.001 and plot a picture (1 is plot).
- ViscousRelax(1,1e11) Use the viscous relaxation routine, 1 is plot a picture and 1e11 the viscous time for one deformation step (in seconds). With the above strain this leads to a strain rate of 10^{-12}.

Experiment 21 - Strain localisation and rigid object kinematics

This section gives details for running one of the deformation experiments involving porphyroblasts with deformable mica caps described in section 4.7.2.

Execution: In order to run one of these experiments, start up the *Experiment Launcher* and select the appropriate experiment from the **Experiments menu**. Possible sub-experiments, which simply vary the relative viscosity of the mica caps are:

a	viscosity ratio 1 ➡
b	viscosity ratio 5 ➡
c	viscosity ratio 25 ☕

As these are multi-process experiments, they will start automatically and you can either use *showelle* to view the microstructural evolution (see Appendix C) or *Sybil* (see appendix F) to follow the stress/strain evolution of the sample.

Interface: The easiest way to monitor the progress is to view the newly generated *Elle* files with *showelle*. Alternatively, the velocity, mechanical quantities and file geometry can all be viewed in *Sybil*. To view only the grain geometry in Fig. 4.7.3, use showelle to open any *Elle* file generated by this experiment.

To reproduce the velocity, vorticity or deviatoric shear stress plots in Fig. 4.7.3, use *Sybil* to open any solution file generated by this experiment, then select **Contour** and the appropriate quantity from the **Graphics menu**, and click the box to use the maximum and minimum range of data values. To overlay the grain geometry on your plot, choose **Elle** from the **XYPlot menu**, and select the *Elle* file that corresponds to the solution to have plotted. To view the Lagrangian mesh for the experiments, go to **XYPlot**, the **LGMesh** and choose **elements**.

Examples: The physical model is shown in Fig. 4.7.2. A central, hard inclusion is intended to simulate a garnet porphyroblast. This grain is surrounded above and below by "caps" that are intended to simulate material that is relatively weak in shear, such as micas with or without thin fluid layers. The surrounding polygonal framework is intended to simulate a background matrix with a homogenous, isotropic viscosity structure. The matrix and central inclusion are assigned constant viscosities, whereas the relative viscosity of the caps was varied from one experiment to another (Fig. B27).

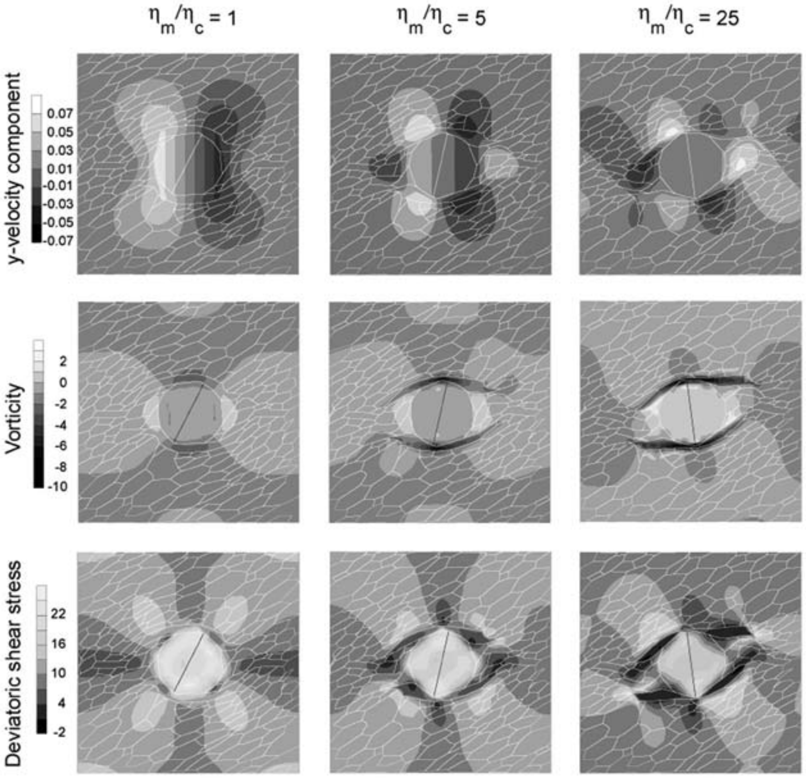


Fig. B27. Contour plots of the *y*-velocity component, vorticity and deviatoric shear stress for the three experiments with different viscosity caps. Total shear strain in each is 1.0. The deformed microstructure from Fig. 4.7.2 has been overlain on the contour plots

Processes: This experiment uses only *Basil*- a Finite Element formulation for linear or non-linear viscous flow.

Functions and parameters for users wishing to modify parameters in the Shelle Script: exp21.elle is the initial elle file that consists of a simple foam texture with a semi-circular grain in the centre. The default viscosities of all matrix grains is set to 10, the viscosity of the larger central grain is set to 1000, and the viscosity of the lubricating caps is set to 10, 2 and 0.4 in experiments (a), (b) and (c) respectively.
exp21.shelle/bat is a batch file generated using Shelle24.html to set the following parameters which control the Elle_experiment:

1. Experiment Name **exp21_out**, the base for naming saved output files
2. *Elle* Input File exp21.elle, the initial microstructure
3. Last Time **30**, the number of iterations for the experiment loop
4. Save Interval **1**, output files from all processes will be saved every iteration with the following processes [and their process id] turned on:
5. **Basil** [1 2 3 13] The file exp21.in controls the *Basil* code, and in this series of experiments you can change the stress exponent by altering the value SE on line 10. The experiments in section 4.7.2 were run with SE=1, or a linear viscous rheology.
6. The saved output file has the process list LIST= "1 2 3 13". All processes in the list are called once per time step.

Appendix C The *Elle* graphical user interface

Jens Becker

C.1 Introduction

A graphical user interface has been developed to simplify the usage of the *Elle* platform. In the following, a short introduction to the usage of this graphical user interface (GUI) is given. However, users vaguely familiar with computers in general will have no problems to use it since it follows standard, menu-driven applications. Upon program start-up (either a process or a utility), the main application window displays a menu bar at the top and a status bar at the bottom of the main application window. In the main bar, the name of the process currently used is displayed.

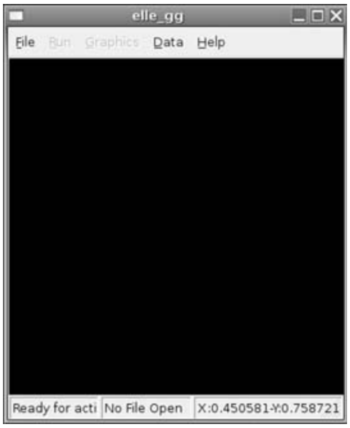


Fig. C1. Main window at program start-up (no file is loaded). The **Run** and **Graphics menus** are disabled as long as no input file is loaded

Start-up procedure: When an *Elle* utility or process starts up or a new *Elle* file is loaded from the GUI, you may pass additional information. If you are in a command line system such as Linux or Msys, you can use the userdata to transfer extra control parameters (see Appendix D). If however a zip file with the same name as the *Elle* file or <u>defaults.zip</u> is found in the same directory as the *Elle* file, this file is loaded, the userdata information is taken from <u>userdata.txt</u> within that zip file. This zip file also includes three other files that control: the graphical appearance of the microstructure (dsettings.txt); the colour look up table (cmap.txt) and the run options (runopts.txt), which are also loaded at the same time. Whenever an *Elle* file is saved using the **Save menu** item from the GUI, a zip file containing all four of these files is also created.

C.2 The status bar

The status bar is divided into three fields (Fig. C1). The first field has two functions: during an experiment it displays the stage numbers or indicates whether there was an error. While there is no experiment running, it displays various status messages. The second field displays the name of the file currently loaded (or the message "No File Open" if no file is loaded). The third field displays the coordinates of the mouse pointer (in *Elle* units, which range from 0 to 1, the origin being in the lower left corner).

C.3 The menu-bar

The menu bar has 5 entries, **File**, **Run**, **Graphics**, **Data** and **Help**. Upon program start-up and during an experiment, the **Run** and **Graphics** menus are disabled to prevent the user from making potentially harmful changes of the simulation while it is running. In the **File** menu, you can choose to load and save *Elle* files as well as quit the program. Of particular importance is the entry "Show Log". This opens another window where logging output of the *Elle* simulations package is displayed, where error messages will be stored.

C.3.1 The File Menu

1. **Open:** Open an *Elle* file
2. **Save:** Save an *Elle* file and a preferences zip file containing all the current graphical and runtime options
3. **Show Log:** Open a window showing he debugging information
4. **Exit:** Quit the *Elle* process

C.3.2 The Run Menu

1. **Run:** Run an *Elle* process or update the graphics if a new file has been loaded in *showelle***.** The process runs the specified number of time steps each time **Run** is selected, using the just modified microstructure as the starting point.
2. **Rerun:** Restarts an *Elle* process using the initial microstructure defined in the *Elle* file.
3. **Run Options:** Opens the Run Options Dialogue

C.3.3 The Graphics Menu

1. **Preferences:** Opens the Graphics Preferences Dialogue
2. **Zoom:** Opens the Zooming Window
3. **Always Overlay:** Displays each new image on top of the previous ones, without clearing screen first
4. **Save:** Opens a submenu where you can choose between saving the currently displayed *Elle* file as a bitmap, saving all *Elle* files in a certain directory as bitmaps, or saving a stereoplot of the three principal axes of each grain (if defined)
5. **Redraw:** Simply redraws the current *Elle* file

C.3.4 The Data Menu

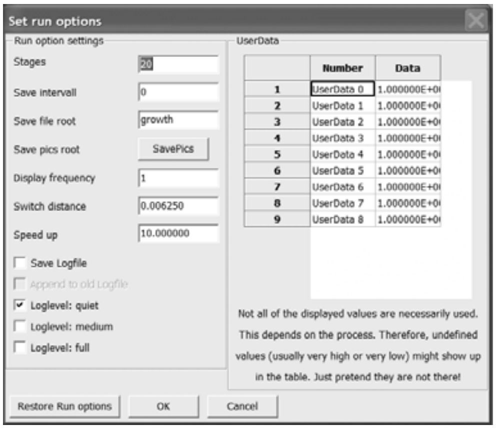
1. **Flynns:** Open the flynn Data Window
2. **Bnodes:** Open the bnodes Data Window
3. **Unodes:** Open the unodes Data Window

C.3.5 The Help Menu

1. **About:** Acknowledgements
2. **Help:** Not yet implemented

C.4 The Run Options dialog

The Run Options Dialog is used to control important settings of the simulations.



Set run options

Run option settings

Stages	20
Save intervall	0
Save file root	growth
Save pics root	SavePics
Display frequency	1
Switch distance	0.006250
Speed up	10.000000

☐ Save Logfile
☐ Append to old Logfile
☑ Loglevel: quiet
☐ Loglevel: medium
☐ Loglevel: full

UserData

	Number	Data
1	UserData 0	1.000000E+0
2	UserData 1	1.000000E+0
3	UserData 2	1.000000E+0
4	UserData 3	1.000000E+0
5	UserData 4	1.000000E+0
6	UserData 5	1.000000E+0
7	UserData 6	1.000000E+0
8	UserData 7	1.000000E+0
9	UserData 8	1.000000E+0

Not all of the displayed values are necessarily used. This depends on the process. Therefore, undefined values (usually very high or very low) might show up in the table. Just pretend they are not there!

Fig. C2. The Run Options dialog. This dialog is used to control important settings of the simulation

Stages: Enter the number of time steps (stages) the simulation should run.
Save interval: Enter the frequency of saving out *Elle* files. If set to 10, a file will be saved every 10th time step.
Save file root: The filename of the saved files. Each file will begin with the filename (e.g. growth, see Fig. C2) followed by the stage number and the ending elle. A complete filename of a single process run therefore looks like this: growth1200.elle.
Save pics root: Next to the *Elle* file (which is a regular ASCII-text file), pictures in various formats can also be saved during the simulation. Enter the root filename of the pictures here.
Display frequency: Only update the *Elle* window every x-th stage.
Switch distance: Adjust the spacing between nodes.
Speed up: Setting this option to anything else than 1 will speed up (or slow down in case of values <1) the simulation. If the speed up is set to 1 and the node would move a distance of 0.25, a speed up of 2 would cause the node to move a distance of 0.5, a speedup of 10 would cause a movement of 2.5 etc. Use with caution since large node movements frequently cause topology problems.
Save log file: During the simulation, the logging messages can be saved out to a new file or appended to an old one. Several log-levels are available: *quiet* will only show the most important messages *medium* will show some debugging information *full* will show a lot of information. This is only useful for debugging.
User Data: Any input data read from the command line or via an associated zip file.

C.5 The Preferences Window

The preferences window can be opened from the **Graphics menu**. It will open as a book with tabs for flynns, bnodes, unodes etc. It is possible to load and save the settings made in the preferences dialog by using the load and save buttons. Apply applies all the settings to the current microstructure but does not close the preferences window. Ok applies all the changes and closes the preferences window. Cancel does not apply the current changes (nor does it reset changes already made to default values) and closes the window.

C.5.1 Flynn Preferences

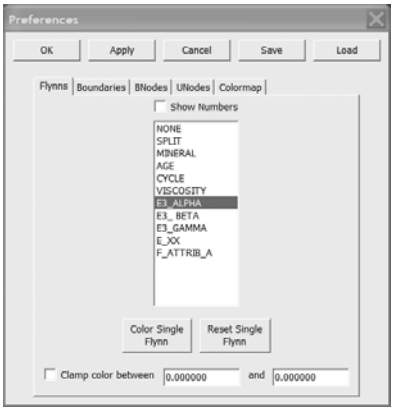
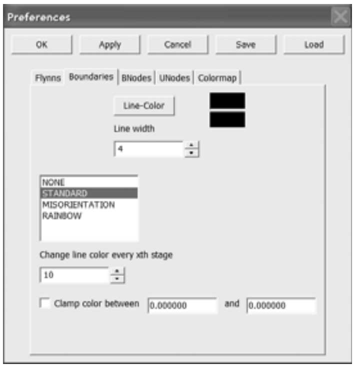


Fig. C3. Preferences page of flynn settings

Show Numbers: Shows the flynns numbers.
List of attributes: The flynn attributes are shown here. Chose one of them to colour the flynns accordingly.
Color single flynn: Give the number of a flynn and it's colour that should be marked. The colour will be displayed during runs so it is possible to (visually) mark a flynn during a simulation.
Reset single flynn: Reset the colour of a flynn again to its normal state (according to its attributes). Asks for the flynn number to be reset.
Clamp color between: Limits the range of the colour look up table to the lower and upper values defined to the right of this check box.

C.5.2 Boundaries Preferences



Line width
4

NONE
STANDARD
MISORIENTATION
RAINBOW

Change line color every xth stage
10

☐ Clamp color between 0.000000 and 0.000000

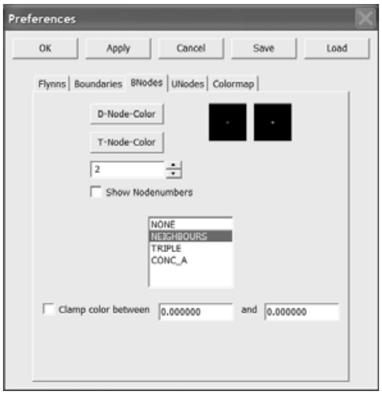
Fig. C4. Preferences page of boundary settings

Line-color: The line colour and its size can be changed. The current colour and size are shown in the box next to the button.
Line width: The line width can be adjusted using the text field.
List of attributes: It can be chosen in which way the boundaries are displayed. If the rainbow option is chosen, the colour of the boundaries will change according to the current colour map. The frequency of colour changes can be adjusted (see below).
Change line color every xth stage: This adjust how often the line colour changes if the rainbow option is chosen. A value of 10 for example indicates that the line colour will change every 10 time steps. This option only has an effect if the rainbow option is chosen, and only makes sense if "Always Overlay" is turned on in the graphics menu.
Clamp color between: Limits the range of the colour look up table to the lower and upper values defined to the right of this check box.

C.5.3 Bnode Preferences

D-Node and T-Node color: The colour of double and triple nodes can be changed independently, the size of nodes is the same for both. The respective colours of double and triple nodes and their sizes are shown in the boxes.
Show Node numbers: If checked this will display all the node numbers in the main window. This is only useful during zooming as there are (usually) far too many nodes so the node numbers cannot be read.

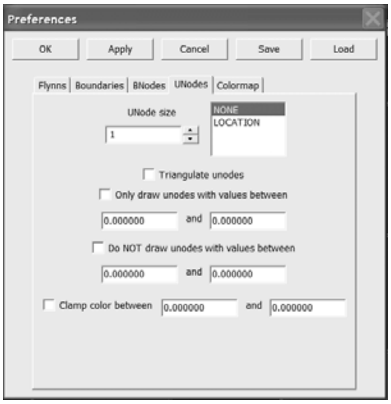
List of attributes: As with flynns and boundaries, the attributes chosen here will be used to colour the nodes.
Clamp color between: Limits the range of the colour look up table to the lower and upper values defined to the right of this check box.



2
Show Nodenumbers
NONE
NEIGHBOURS
TRIPLE
CONC_A
Clamp color between 0.000000 and 0.000000

Fig. C5. Preferences page of bnodes settings

C.5.4 Unode Preferences



UNode size 1
NONE
LOCATION
Triangulate unodes
Only draw unodes with values between 0.000000 and 0.000000
Do NOT draw unodes with values between 0.000000 and 0.000000
Clamp color between 0.000000 and 0.000000

Fig. C6. Preferences page of unodes settings

Unode size: Adjust the size of the unodes here. This will simply draw every unode with the given size, there is no triangulation or recalculation of unode values!
List of attributes: As with the other pages, the attributes can be chosen here that is used to colour the unodes.

Triangulate Unodes: This will display the unodes as a continuous image, rather than showing each unode separately.
Only draw unodes with values between: This will only colour values within the given range. If you select an attribute other than "none" or "location", the min and max values of this attribute will be shown in the two text fields.
Do NOT draw unodes with values between: The two fields (draw unodes and NOT draw unodes) can be combined. The following is important to know to get exact results and not to misinterpret the display: unodes that have the values that were entered in the boxes WILL be displayed (unodes with values equal to max or min).
Clamp color between: Limits the range of the colour look up table to the lower and upper values defined to the right of this check box.

C.5.5 The colour map

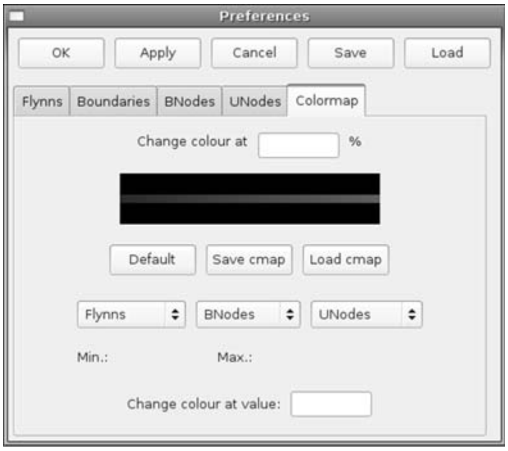


Fig. C7. Preferences page of colour settings

In the colour map window, the colours used to display attributes can be adjusted. The upper text box expects a percentage. An input of 0 will change the colour for the lowest value, an input of 100 will change the highest value. It is not necessary to know the min and max values of the attribute.
Default: The colours can be reset to the original by pressing the default button.
Save/Load cmap: The colour map can be saved or loaded for later reuse. The file saved will be a regular text file with 256 colours. See below for an example file of default.cmap:

```
Colormap for Elle-files... Please do not edit
          0 0 255 1
          1 0 254 0
          [...]
          [...]
          [...]
          254 0 1 0
          255 0 0 1
```

The first number indicates the number of the colour, the next are RGB values of the colour.
Flynns/BNodes/Unodes: If the user wants to change a specific colour value, he has to know the min and max values of the attributes. These might (or better most certainly will) have changed during a simulation. These three drop-down lists can be used to choose an attribute, the min and max values of this attribute will be calculated and displayed. The lower text box can be used to enter a specific value at which the colour will be changed.

C.6 The zooming window

Once the zooming window is opened, you have a few options to choose from: zooming in and out of the structure, moving the structure, stop zooming entirely, get info on bnodes and unodes, number all bnodes.

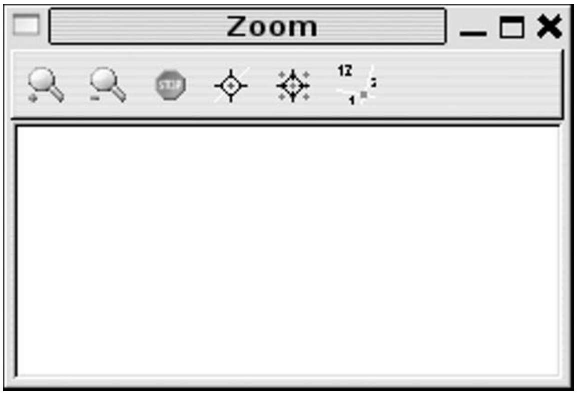


Fig. C8. The zooming window. For explanation of the buttons see text

Zoom into the view. Zooming will always use upper left corner as the the origin.

Zoom out of the view.

Stop zooming. The screen is redrawn to its full size.

Get info on bnodes. With this tool, bnodes can be chosen (by clicking on them in the canvas. The pointer to do so changes to black so you might have a hard time seeing it when no flynn-colours are displayed!) and information about the values of the current bnode are displayed. If you click to far away from a node, the window displays the message No Hit!. If there are no bnode-attributes defined, the window will say so EVEN IF YOU HIT A BNODE!

Get info on unodes. Same as above, only for unodes.

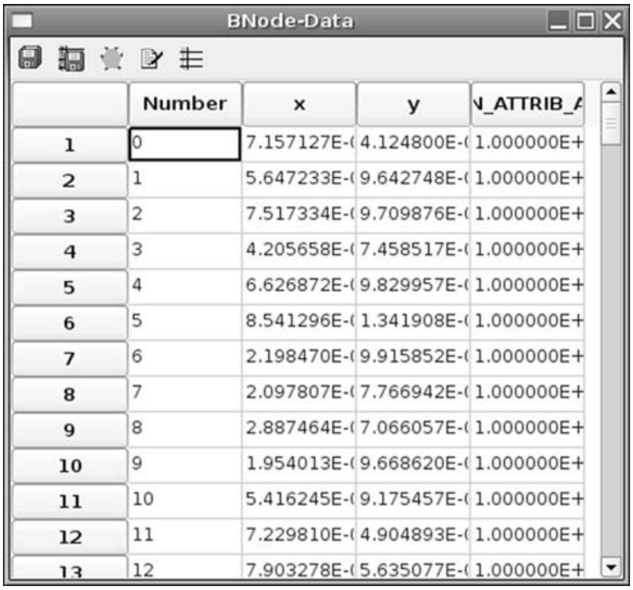
Labels all the flynns and bnodes. Since there might be a lot of bnodes, this only makes sense if the zoom is large.

Another feature of the zoom is that you can zoom in on any area you are interested in and start your simulation. During the simulation, the canvas will only show the area you have zoomed in. It is possible to change the viewing area of the zoom and/or to stop zooming entirely during the simulation.

C.7 The data-windows

The table windows will open if the respective menu under Data is chosen. They will display either flynn, bnode or unode attributes, if there are any defined. The values can be changed by simply clicking on the cell: a small box will pop-up and ask for the new value.

It is not possible to change the numbers of flynns, bnodes or unodes or the location of bnodes and unodes. It is also not possible to insert or delete flynns or bnodes or unodes from the file. If you click on the row-labels, the respective flynn, bnode or unode will be highlighted. Clicking on a row of a previously selected item will deselect it.


BNode-Data

	Number	x	y	N_ATTRIB_A
1	0	7.157127E-(	4.124800E-(	1.000000E+
2	1	5.647233E-(	9.642748E-(	1.000000E+
3	2	7.517334E-(	9.709876E-(	1.000000E+
4	3	4.205658E-(	7.458517E-(	1.000000E+
5	4	6.626872E-(	9.829957E-(	1.000000E+
6	5	8.541296E-(	1.341908E-(	1.000000E+
7	6	2.198470E-(	9.915852E-(	1.000000E+
8	7	2.097807E-(	7.766942E-(	1.000000E+
9	8	2.887464E-(	7.066057E-(	1.000000E+
10	9	1.954013E-(	9.668620E-(	1.000000E+
11	10	5.416245E-(	9.175457E-(	1.000000E+
12	11	7.229810E-(	4.904893E-(	1.000000E+
13	12	7.903278E-(	5.635077E-(	1.000000E+

Fig. C9. Example of a data window (here bnode-data are displayed). If no attributes for nodes or flynns have been set, the window will not show anything. This is a *feature*, not a bug!

Save all data as text file. The floppy disk icon lets you save all the data.

Save selected data as text file. This icon lets you only save the selected data.

Change multiple values. If more than one point should be changed, it is possible to mark the respective area by clicking on the icon. You will then be able to mark an area in the canvas by drawing a polygon around the area you want to change. Note that only flynns, bnodes or unodes that are completely within the polygon will be highlighted and hence changed. The area that is marked can not have more than 1000 vertices. The last vertex is set by double-clicking on the desired position (left-double-click).

If multiple entries are selected (using the polygon tool described above) you can simultaneously change all of their values with this tool. If you have lots of flynns or nodes highlighted you may not see some of them. In that case, either scroll down until you find one, click on the desired cell and change the value for all of them or press this icon: a message box will appear asking you for the new value.

Deselect Rows. You can deselect the rows by clicking on this icon.

Appendix D Different ways to run an *Elle* experiment

Mark W. Jessell

At a minimum, an *Elle* experiment consists of a single input file (an *Elle* file) that describes the starting microstructure, and a single *Elle* processes, that describes the effect of a single process on that microstructure. Single-process experiment may be run in graphics mode, with continual update of the microstructure visualisation. Multi-process experiments must always be run via a shell script (called a batch script on the Windows platform), and the results of the experiment may be followed by opening output files using the *showelle* microstructure display tool.

D.1 Via the *Experiment Launcher*

The *Experiment Launcher* is a simple program that allows you to launch any of the example experiments in this Appendix. Once started up, the Launcher offers a single menu and two panels: the **File menu**, the **Examples panel** and the **Utilities panel**. The **Experiment panel** and its submenus list all the experiments in this Appendix. To run a particular example, select it from the **Experiment menu** and you will be provided with a brief overview of the experiment. If you wish to start the experiment, simply click on the **Go! button**.

Single-Process Experiments If a single-process was started, an *Elle* window will open up showing the starting microstructure. To enact the experiment select **Run→Run** from within the *Elle* Window. See Appendix C for details on graphical user interface for single processes.

Multi-Process Experiments If a multi-process was started, the experimented will be started automatically in the background, and to view any of the output *Elle* files, select *showelle* from the **Utilities menu** of the Launcher, and load in the desired file. See Appendix C for details on graphical user interface for *showelle*. All output files will be created in the same directory as the experiment, and have a naming convention described in Appendix E.

Starting Utilities The Experiment Launcher may also be used to launch utilities such as *showelle*, without any input file pre-loaded.

D.2 Direct launching of experiments

Single-Process Experiments To run a single-process experiment, double click on the icon for the particular process (e.g. *elle_gg*), go to the **File menu** and **Open** an *Elle* file (e.g. exp3.elle) If there is a zip file with the same suffix as the *Elle* file in the same directory (e.g. exp3.zip) it will be loaded at this time, and it will define the experimental conditions and the graphical display properties of the experiment. To run the experiment, simply select **Run** from the **Run menu**. The graphical display will then be updated each time step, or less frequently, depending on the display options. A complete description of the *Elle* graphical interface, which is common to all processes, may be found in Appendix C.

Multi-Process Experiments To run a multi-process experiment, double click on the batch file (e.g. exp4.bat for windows) and a window will open up giving some runtime debugging information. Any output files will be saved in the same directory as the batch file, and the *Elle* files may be viewed by starting *showelle*, loading the *Elle* file. If there is a zip file containing the graphics preferences called defaults.zip, this will be loaded automatically. A web page to create new shell scripts/batch files is described in Appendix E. If *Basil* was used in the experiment, and its post-processor *Sybil* was installed, detailed information on the stress and strain state within the microstructure may also be displayed (see Appendix F).

D.3 Developer mode (Windows and Linux)

In addition to the GUI based modes described above, Windows and Linux Developers may also run *Elle* experiments from the command line. In Windows, you need to start up *MSys*, which will provide a shell, and in Linux, you need to open a new shell window.

Single-Process Experiments To run a single-process experiment, one command line call is needed with options specifying information such as the number of steps to be simulated, the input and output file and the simulation mode.

Synopsis: `process_name [-cefhinsuv]`
Options:

`f`	the next argument will set the frequency for saving the *Elle* data during a run; i.e. 10 will mean that the data is saved after every 10th stage
`h`	help, list the valid options
`i`	the next argument will be the filename of an *Elle* input data file
`n`	no display (run without the graphical display)
`s`	the number of stages to run (default 20)
`u`	may be followed by up to 9 numbers which allow the user to vary parameters defined for a particular process e.g. *elle_gbdiff* interprets the first value as the diffusion rate.
`v`	the boundary velocity (only used by *elle_manuel*)

These are the only options, which can currently be changed at runtime when executing in batch mode.

An example of calling an *Elle* process:

```
elle_pblast –i start.elle –s 25 –f 25 –n
```

will run *elle_pblast* in batch mode (no display), reading the input file start.elle, processing 25 stages then saving the result. Only one file is saved as it is set to save every 25th stage. The options set by the command line will override the runtime and userdada setting read from a zip file (see App. D.2).

Multi-Process Experiments To run a multi-process experiment, change to the directory which contains the batch file and execute the batch file directly: ./exp4.shelle.

Any output files will be saved in the same directory as the batch file, and the *Elle* files may be viewed by starting ***showelle***, loading the *Elle* file. If there is a zip file containing the graphics preferences, this can be manually loaded by selecting **Preferences** from the **Graphics** menu, then clicking on the **Load button**. A web page to create new shell scripts/batch files is described in Appendix E. If ***Basil*** was used in the experiment, detailed information on the stress and strain state within the microstructure may also be displayed using the *Sybil* post-processor (see Appendix F).

Appendix E How to create multiple-process experiments

Lynn Evans

E.1 Shelle scripts/batch scripts

Running multiple processes requires the use of a control script, which we call a Shelle Script, this is in fact a shell script for under Linux and MSys, and a Batch Script under Windows, which defines the behaviour of the experimental run (Fig. E1):

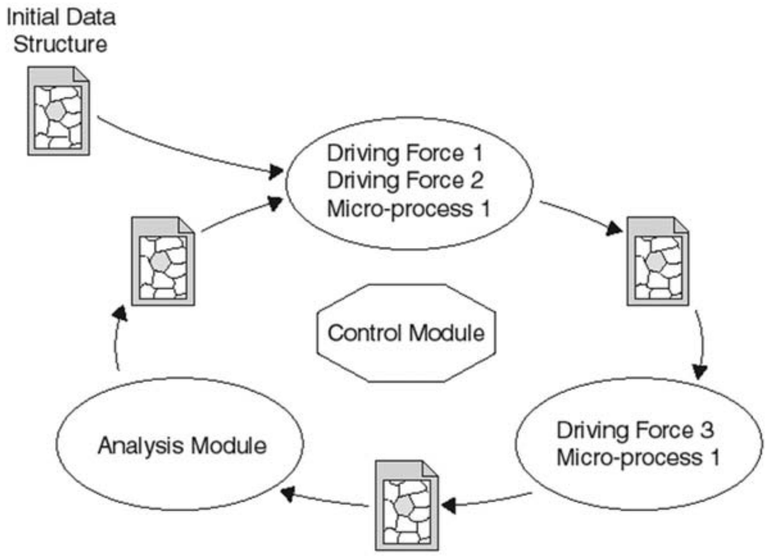


Fig. E1. *Elle* control module controls the way in which processes are executed and data is stored

We have simplified the task of creating these control scripts by developing a GUI that allows the user to specify the steps and parameters required for their simulation then generate and run the control program. Once created, it is possible for the user to write or edit the resulting text file that is a shell script similar to elle/examples/init/control.template. All the controlling variables, e.g. STAGES, LASTTIME, are set in the first section above the line that looks something like:

```
LIST="1 2 3 13 4 16 15"
```

LIST specifies the processes in the order in which they will be executed. This line may be edited if different processes are desired. If DEBUG is set to echo, the program will write the command line to the screen rather than running it. This is useful for checking that the program calling order and parameters are correct.
To run a simulation in the background type:

```
NOHUP CONTROLFILE >& DUMPFILE &
```

where controlfile is the name of the file created by the shelle script generator and dumpfile is the output redirected from the screen and stderr. The dumpfile can be deleted if the simulation runs successfully.

E.2 Shelle: An *Elle* Shell Script Generator (v1.24)

In order to run more than one process, it is necessary to create a shell script that will call each process in turn, taking into account all the file names and formats needed to ensure a smooth transfer of information.

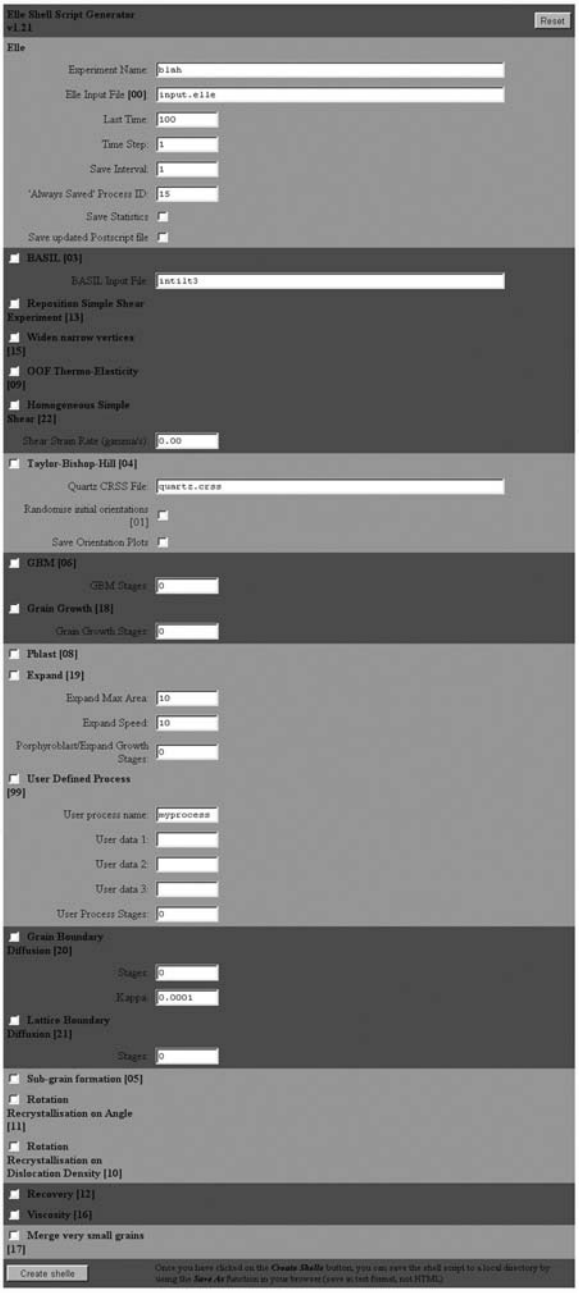
The *Elle* implementation page contains a link to a web page that lets you fill in a form that then automatically generates the appropriate shell script. It currently allows you to create a shell script for any combination of the processes listed in this manual.

E.2.1 How to use the Shelle Script Generator

Select the **Shelle Script Generator** from the **Utilities** section of the Elle Experiment Launcher and click on **Go**. Alternatively, start up a JavaScript compatible web browser (not as scary as it sounds, any Firefox browser or any Internet Explorer version from 3.0 will do) and select the Shell script generator:

- http://www.microstructure.info/elle/shelle.html
- or elle/utilitities/shelle/shellev24.html.

You will be presented with a form like Fig. E2.


Elle Shell Script Generator v1.21

Reset

Elle

Experiment Name:	blah
Elle Input File **[00]**	input.elle
Last Time:	100
Time Step:	1
Save Interval:	1
'Always Saved' Process ID:	15
Save Statistics	☐
Save updated Postscript file	☐

☐ **BASIL [03]**

BASIL Input File:	intilt3

☐ **Reposition Simple Shear Experiment [13]**

☐ **Widen narrow vertices [15]**

☐ **OOF Thermo-Elasticity [09]**

☐ **Homogeneous Simple Shear [22]**

Shear Strain Rate (gamma/s):	0.00

☐ **Taylor-Bishop-Hill [04]**

Quartz CRSS File:	quartz.crss
Randomise initial orientations [01]	☐
Save Orientation Plots	☐

☐ **GBM [06]**

GBM Stages:	0

☐ **Grain Growth [18]**

Grain Growth Stages:	0

☐ **Pblast [08]**

☐ **Expand [19]**

Expand Max Area:	10
Expand Speed:	10
Porphyroblast/Expand Growth Stages:	0

☐ **User Defined Process [99]**

User process name:	myprocess
User data 1:	
User data 2:	
User data 3:	
User Process Stages:	0

☐ **Grain Boundary Diffusion [20]**

Stages:	0
Kappa:	0.0001

☐ **Lattice Boundary Diffusion [21]**

Stages:	0

☐ **Sub-grain formation [05]**

☐ **Rotation Recrystallisation on Angle [11]**

☐ **Rotation Recrystallisation on Dislocation Density [10]**

☐ **Recovery [12]**

☐ **Viscosity [16]**

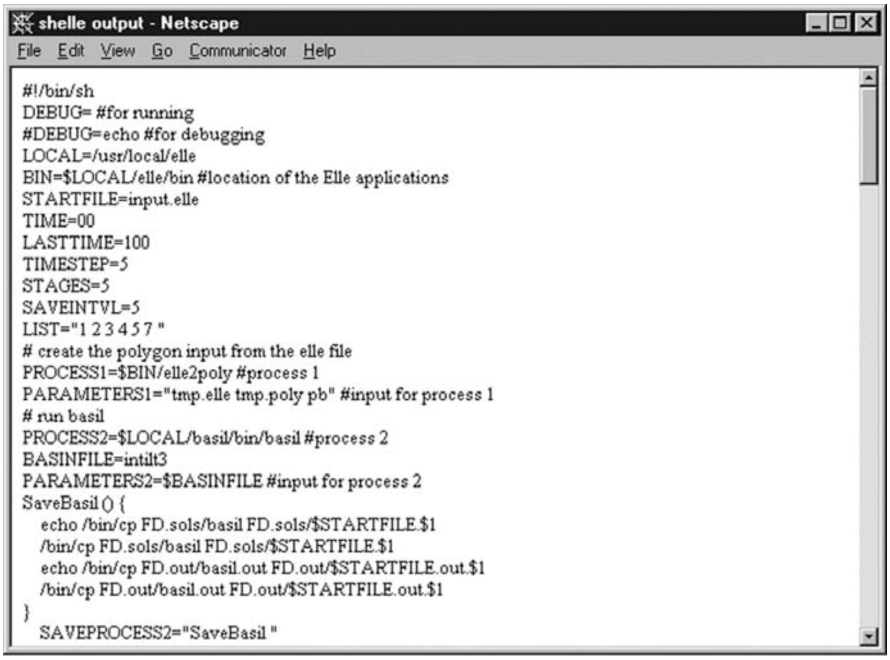
☐ **Merge very small grains [17]**

Create shelle

Once you have clicked on the ***Create Shells*** button, you can save the shell script to a local directory by using the ***Save As*** function in your browser (save in text format, not HTML.)

Fig. E2. The shelle script generator

To activate a process, click on the check box to the left of its name, and fill in the appropriate information in the subsequent fields. Then click on **Create Shelle** for Linux, and Windows Developers using *MSys*, or **Create Batch** for Windows Users and a new window will appear that contains the generated shell script.



```
#!/bin/sh
DEBUG= #for running
#DEBUG=echo #for debugging
LOCAL=/usr/local/elle
BIN=$LOCAL/elle/bin #location of the Elle applications
STARTFILE=input.elle
TIME=00
LASTTIME=100
TIMESTEP=5
STAGES=5
SAVEINTVL=5
LIST="1 2 3 4 5 7 "
# create the polygon input from the elle file
PROCESS1=$BIN/elle2poly #process 1
PARAMETERS1="tmp.elle tmp.poly pb" #input for process 1
# run basil
PROCESS2=$LOCAL/basil/bin/basil #process 2
BASINFILE=intilt3
PARAMETERS2=$BASINFILE #input for process 2
SaveBasil () {
   echo /bin/cp FD.sols/basil FD.sols/$STARTFILE.$1
   /bin/cp FD.sols/basil FD.sols/$STARTFILE.$1
   echo /bin/cp FD.out/basil.out FD.out/$STARTFILE.out.$1
   /bin/cp FD.out/basil.out FD.out/$STARTFILE.out.$1
}
   SAVEPROCESS2="SaveBasil "
```

Fig. E3. Output of the shelle script generator (Shell Script)

If you like what you see, you can save this shell script by using the **File→Save As** menu from the browser window, remembering to change the save as type to Plain Text **(txt)**. If you don't like what you see, go back to the input form window, click on reset and try again.

In order to actually use this script under Linux or using Windows Developers using Msys you will have to change its privileges to executable: if the shell script is called myshell.txt, type in

```
chmod +x myshelle.txt
```

in the same directory as the file (or use your file manager to access the Permissions section of the file properties and set Exec).

E.2.2 Notes on the Shelle Script

a) *File name convention:* Output elle files resulting from using this shelle script will be of the form **Experiment_Name.xxx.yy.elle** where xxx is the time step and yy is the process code (shown above in square brackets), as follows:

00 = input file
01 = following initial randomisation of crystal axis orientations
03 = following a BASIL step
04 = following a TBH step
05 = following a Subgrain Split step
06 = following a GBM step
08 = following a Porphyroblast growth step
09 = following an OOF step
10 = following a Nucleation/Rotation Recrystallisation step
11 = following a Misorientation/ Rotation Recrystallisation step
12 = following a Recovery calculation step
13 = following a Repositioning step
15 = following an Angle check step
16 = following a Viscosity calculation step
17 = following a Merge calculation step
18 = following a Grain Growth step
19 = following an Expand step
20 = following a Grain Boundary Diffusion step
21 = following a Lattice Diffusion step
22 = following a Homogeneous Simple Shear step
99 = following a User Defined Process step

b) *Randomise Orientations:* When checked, this option randomises the lattice orientations of any quartz grains, which have not already been defined, one time only, at the start of the experiment.

c) *Save Orientation Plots:* When checked, at the save interval frequency, c-, a- and r- axis pole figures of quartz grains will be saved out as post-script files with the naming convention *ellefilename_ax.ps* .

d) *Reposition Simple Shear Experiment:* This normalises the elle space to a square after each BASIL deformation step.

e) *Widen narrow vertices:* Angle check to removes narrow vertices from flynns to make triangulation better.

f) *Merge calculation:* Removes very small grains.

g) *Statistics:* Simply outputs information to file.

h) *Save updated Postscript file:* If this option is turned on, a postscript file of the current *Elle* microstructure called *tmp.ps* is created (using *showelle.in* defaults), if you use *ghostview* with the "watch file" option enabled, the window will update with the new file each save step.

E.2.3 Debugging Shell Scripts

It is advisable to test the shell script without actually running the processes first. To do this use your favourite text editor to remove just the first # from the third line, which reads #DEBUG=echo #for debugging, so it reads DEBUG=echo #for debugging. If you run the script now it will go through the motions of the script without actually doing anything. If the results look good, cross your fingers, put the # back and run the script.

Appendix F *Sybil* - The *Basil* post-processor

Lynn Evans, Terence Barr and Greg Houseman

F.1 Introduction

Sybil is the graphical post-processor for the *Basil* Finite Element code, and is a stand-alone program that reads in *Basil* solution files (which will be found in a directory called FD.sols within the directory containing the *Elle* experiment. It can also overlay grain boundary maps from *Elle* files.

Windows To start *Sybil* from Windows, first start up the *Experiment Launcher* and then select *Sybil* from the Utilities Menu.
Linux To start up *Sybil*, type **`./sybil`** in the same directory as your experiment. You will then be presented with a blank window, with a series of menus at the top

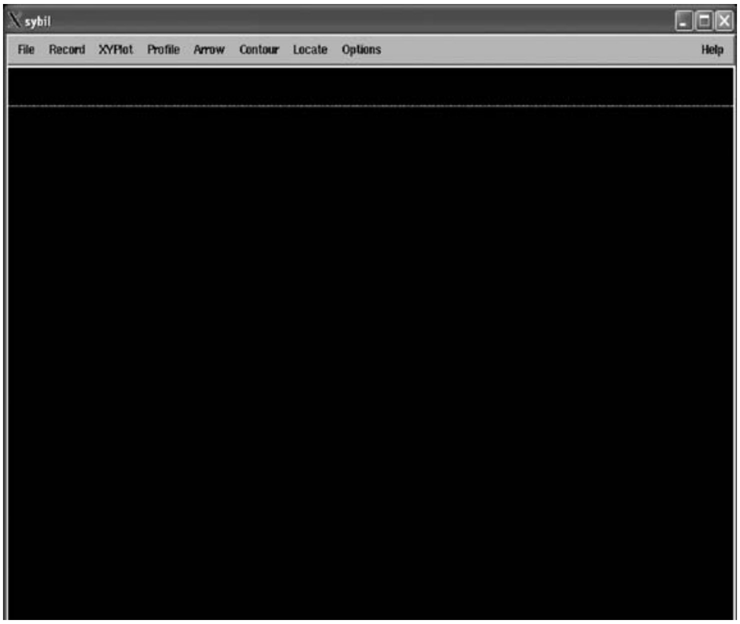


Fig. F1. *Sybil* window at start-up

You can use the various Menu Options, that are outlined below, to display the stress and strain state of the microstructure at different stages of the experiment.

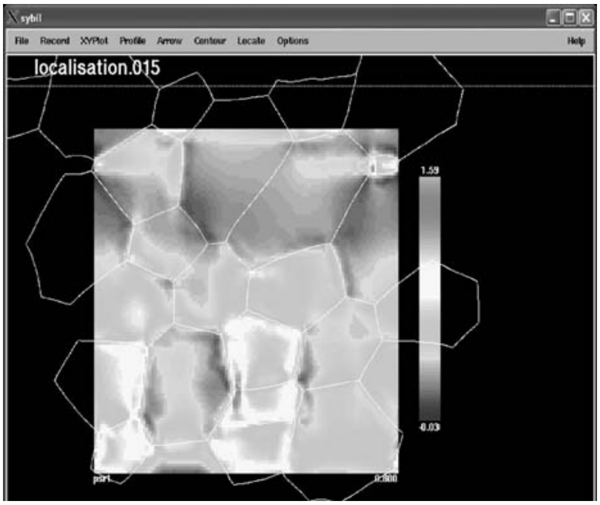

sybil
File Record XYPlot Profile Arrow Contour Locate Options Help
localisation.015
1.59
-0.03

Fig. F2. *Sybil* window showing strain state in a microstructure

F.2 Menu Options

F.2.1 File

1. **Open**: opens a file, either **Solution**: to open a solution file, or **Log**: to open and execute a previously saved log file
2. **Save**: writes a file, either **Log**: to write a log file that recreates the current drawing, or **Log+Postscript**: to write the log file, and then cause *sybilps* to execute and write a postscript file of the current drawing.
3. **Exit**: terminates execution of *Sybil*, after checking.

F.2.2 Record

1. **Current**: selects the solution record for subsequent drawing, **Next**: selects the next record in the current file, **Previous**: selects the previous record in the current file, **Last**: selects the last record in the current file, **Specify**: type input number of solution record (1 is the first record and the value of max in the info line at the bottom of the window is the last record)
2. **Reference**: selects the solution record to define an initial geometry relative to which subsequent deformations are measured, **First**: uses the first record of the current file, **Specify**: uses the record number you identify, as above.

F.2.3 XYPlot

1. **Mesh**: creates an X-Y plot of the Finite Element mesh, either **elements**: outline of all elements in the mesh **element+num**: outline as above plus element numbers of some elements **boundary**: external mesh boundary only **viscosity**: shows regions with anomalous viscosity
2. **Deform**: draws deformation indicators relative to a previously saved reference record (see **Record→Reference** above), **ellp**: draws finite deformation ellipses for a subset of triangles **tria**: draws the subset of triangles used for the ellipse calculations **trel**: draws ellipses and triangles, as specified by two preceding options.
3. **Bounding box**: draws a rectangular box around the current plotting area, excluding margins.
4. **LGMesh**: draws the Lagrangian mesh that may be used by *Basil* to track deformation, **elements**: draws the entire Lagrangian mesh **boundary**: draws the boundary only
5. **StrainMark**: draws any strain markers defined by *Basil* to track internal deformation
6. **Elle**: Draws the grain boundary network defined in an *Elle* File

F.2.4 Profile

1. **1_D**: constructs a 1-D profile of selected parameters between 2 points on the *x-y* plane. After selecting a quantity to profile from the lists below, input of the *x-y* coordinates of the two profile endpoints is requested. Min and Max values on the profile are then shown and, the user must input values for the Min and Max on the vertical axis of the profile plot. Default values are the min and max values of the function on the first call, and scale is preserved for subsequent profile plots. **Velocity**: select from the following parameters: **Ux**: *x*-component of velocity, **Uy**: *y* component of velocity, **Ur**: radial component of velocity (relative to local origin), **Uth**: tangential component of velocity. **Strain**: select from the following quantities related to strain-rate: **edxx**: the *xx*-component of strain-rate, **edyy**: the yy-component of strain-rate, **edzz**: the *zz*-component of strain-rate, **edxy**: the *xy*-component of strain-rate, **psr1**: maximum principal strain-rate in *x-y* plane, **psr2**: minimum principal strain-rate in *x-y* plane, **msst**: maximum shear strain-rate in the *x-y* plane, **cang**: orientation of principal compressional axis,, **tang**: orientation of principal extensional axis, **sang**: orientation of direction of maximum shear, **dblc**: parameter that indicates type of faulting, **vort**: vertical component of vorticity, **ed2i**: 2nd invariant of the strain-rate

tensor, **vota**: ratio of vorticity to shear strain-rate. **Stress**: select from the following quantities related to stress, **taud**: direction of maximum deviatoric stress, **taum**: maximum deviatoric shear stress, **taxx**: the *xx* component of deviatoric stress, **tayy**: the *yy*-component of deviatoric stress, **tazz**: the *zz*-component of deviatoric stress, **taxy**: the *xy*-component of deviatoric stress, **tau1**: maximum principal deviatoric stress, **tau2**: minimum principal deviatoric stress, **sixx**: the *xx*-component of total stress, **siyy**: the *yy*-component of total stress, **sizz**: the *zz*-component of total stress, **sig1**: maximum principal total stress, **sig2**: minimum principal total stress, **thdi**: thermal dissipation function (stress*strain-rate), **pres**: the pressure field, **brit**: parameter indicating type of faulting, **bri2**: parameter indicating type of faulting, **visc**: effective viscosity parameter, **Thickness**: layer thickness in the *z*-direction, **Density**: density parameter.

2. **2_D**: constructs a horizontal or vertical profile of selected quantities integrated in the orthogonal direction. The same set of parameters can be profiled, as listed above for 1_D. After choosing the quantity to be profiled, choose whether an *x*-direction, or a *y*-direction profile is required, and input the upper and lower limits of the orthogonal integra- tion variable. E.g. To show how the integral of *Ux.dy* varies as a function of *x*, choose the *x*-direction profile and the two limits are the extreme values of *y* for the integration. Avoid integrating outside the solution region if meaningful results are required. Input of vertical axis limits is then requested, as for 1_D profile.
3. **Mark**: is used to draw a line in the current cell showing the path of the 1-D profile in the *x*-*y* plane. The profile path can be plotted over a previously drawn contour plot of the variable that is to be plotted, in order to show where the profile has come from, and to check correlation with features in the 2D plane. After entering the coordinates of the 2 endpoints, use preview to check position of the profile that can then be amended as necessary. The Mark command can be used also for drawing any additional lines required on a *Sybil* drawing, regardless of whether a profile plot is required.

F.2.5 Arrow

1. **Velocity**: constructs an arrow plot, consisting of equally spaced arrow heads, whose direction is parallel to the velocity field and whose length is proportional to the magnitude of the velocity. The spacing of arrows can be set under **Options→Plot→mp and np** (refer 'Options' below).

2. **Strain:** constructs an arrow plot, consisting of equally spaced crosses, whose orientation indicates the directions of the principal axes of strain-rate. For direction-only plots all crosses are the same size, for magnitude plots, the length of each arm is proportional to the magnitude of the principal axis. Options are: **pstm**: Principal strain-rate magnitudes, **pstd**: Principal strain-rate directions only, **mssr**: planes and magnitudes of maximum shear strain-rate, **msft**: direction of strike-slip faulting planes,
3. **Stress**: constructs an arrow plot as for Strain (see above), but with magnitudes proportional to the relevant stress quantity. Options are: **taum**: principal deviatoric stress magnitudes, **taud**: principal deviatoric stress, directions only, **sigd**: principal stress (total), directions only, **sigm**: principal stress (total) magnitudes.

F.2.6 Contour

The Contour command is used to display 2-D quantities using a graphical display in which areas are coloured according to the value of the physical parameter being displayed, or equally spaced contours of this physical parameter are drawn, or both. The same set of physical parameters listed above under **Profile→1_D** is available for contouring. After choosing the parameter to be contoured, the user is asked to confirm or enter a set of parameters that affect the display. Shown at the top left are the actual minimum and maximum values of the quantity to be contoured. The two values at the top right, entered by the user, determine the mapping of colour to number. With the default colour scale, blue is mapped to the minimum, and magenta to the maximum. Contours, if present, are chosen in equal increments of ‘Step’ above and below ‘Level’. A maximum number of contours is also available. The number labelled scale is only used to multiply the labels that appear on the colour scale. The settings are preserved for successive calls to ‘Contour’ so that the default setting usually is to keep the same scales as for the preceding plot. Various default parameters affecting the display may be set using the **Options** command (see below).

F.2.7 Locate

The Locate command is used to specify which of the current drawing cells is active, as follows:

1. **Next**: Move to the next cell, either the cell to the right, or if already at the right limit of the page, the left most cell of the next row.
2. **Prev**: Move to the previous cell.

3. **Specify**: Increment or decrement the row or column number of the current cell address using the buttons provided in order to move directly to a particular cell.

F.2.8 Options

1. **Label**: is used to add labels to the plot as follows:
2. **Edit**: enter a string to be drawn on the plot. Then click with the left mouse button at the point on the drawing where the lower left corner of the string is to be located. The same string can be added to the plot as many times as required and the string can be edited at any time in the labelling step.
3. **Font**: choose one of the 2 fonts (Helvetica, Symbol) and 5 or 6 font sizes (8, 10, 12, 14, 18, 24) that are provided as options.
4. **Colour**: choose one of the 8 standard colours to be used in subsequent drawing and labelling operations.
5. **Line**: choose solid, dashed, or dotted line for subsequent drawing operations.
6. **Rescale**: On making the first plot, the scale is set automatically so that the entire solution region will appear within the plotting cell, including a small margin on each side. Subsequent plots are drawn at the same scale, with the origin shifted to the new drawing cell. The scale will be automatically reset for the next plot, only if this option is first selected. Judicious choice of cell numbers and margin widths may be necessary to obtain a particular scale.
7. **Delete**: deletes previous drawings from the screen. Note that the first option is also available at any time by locating the cursor within the current cell and pressing the middle button of the mouse.
8. **Current cell**: only deletes contents of one cell
9. **All Cells**: deletes contents of all cells
10. **Plot**: requests input of certain options and parameters that affect the visual display, as follows: **nx3**: determines the number of interpolation points to be used in contouring and other operations. A number in the order of 100 is recommended. Greater numbers will improve the resolution of the plot, but will require more time to plot. **mp, np**: in constructing arrow plots (see above) an arrow is situated at every (mp)th interpolation point in the *x*-direction and every (np)th interpolation point in the *y*-direction. The size of the maximum arrow is equal to the distance between arrows. Numbers of 4 or 8 are recommended. Note that if nx3 is doubled, mp and np should be doubled to preserve the same arrow spacing. **profile_pts**: the number of sample points to take along a

given profile, **stipple**: if > 0, stippling is added to contour plots in specified regions, with reference to the Level parameter (see Contour above): stipple = 1 → stipple if below Level, stipple = 2 → stipple if above Level, stipple = 3 → stipple if abs value above Level, stipple = 4 → stipple if abs value below level. **solution_rot**: This parameter should be a number between 1 and 3, to rotate the solution by quanta of 90 degrees anticlockwise. The rotation is done immediately that the solution is input, and any reference to x and y- coordinates then applies to the solution as displayed in its new reference frame. **dble**: Causes most area plots to be displayed twice, with a horizontal offset equal to the width of the solution region. This option is only used for solutions with periodicity in the x-direction. **label**: causes all automatic labelling to be switched on or off. This option does not affect any labels added manually, as above. **flip**: causes the x-axis to be reversed on the display. **Contour Options**: either or both of the following may be selected. **Lines**: contour lines are added to all contour plots. **Shading**: shading by colour is added to all contour plots. **Bar**: the colour bar for contour plots may be vertical, to the right of the plot, or horizontal, underneath the plot, or may be omitted. **Tics**: ticks on profile plots may be internal to the plot, external, centred, or omitted.

11. **Verbose**: outputs numbers (lots of them) to standard out, from profiles, contour plots, arrow plots and strain-marker plots, in order that other external processing may be applied to these data. This option should only be turned on temporarily to extract a particular data set and is not yet implemented uniformly.
12. **Mark Cell**: shows the current plotting cell by a line drawn around its perimeter. This boundary line does not appear on the postscript plot.

Appendix G How to use *Elle_Latte*

Till Sachau and Daniel Koehn

G.1 Introduction

Elle_Latte (simply called *Latte* in this appendix) is a Lattice-Spring code that is included in *Elle*. Latte is written as a separate process that uses *Elle* functions and the *Elle* data set.

Latte is build up by three main classes, the particle class, the lattice class (lattice.cc, lattice.h, particle.cc and particle.h) and the experiment class (experiment.cc, experiment.h). A third basic class that inherits the lattice class is used as a base class for phase transformations (base_phase.cc, base_phase.h). This base phase class is inherited by the classes phase_lattice for fluid-solid reactions (phase_lattice.cc, phase_lattice.h) and min_trans_lattice for solid-solid phase transformations (min_trans_lattice.cc, min_trans.h). In addition a third class is used for heat diffusion (heat_lattice.cc, heat_lattice.h) and a class that includes the grain growth process (graingrowth.cc and graingrowth.h).

The User does not have to be concerned with these classes directly as we have developed a graphical interface where the user can choose one of the processes that is included in the experiment class and can change parameters without having to recompile.

Functions are basically separated into initialization functions that give different flynns, single particles or defined layers different properties, and run functions that define the proposed deformation or internal changes and dump statistics to plot for example stress-strain curves. The user can run an experiment in two possible ways: a) use the interface to select the desired process and its parameters or b) change parameters in the experiment.cc class, recompile and select the process when calling *Latte* using the **-u command**.

G.2 The experiment interface for Windows and Linux users

G.2.1 Introduction

As an attempt to simplify the handling of *Latte* and to avoid users to get into the *Latte* source code and recompile it every time to adapt it to their needs, a simple graphical user interface (GUI) has been developed. This GUI is not linked to the *Latte*-executable itself, but calls it as an external process. Thus it is designed as a graphical editor for a preference-file that *Latte* can read and execute. The examples given in the book can be started either from this interface or, alternatively, directly from the command-line. Functions that can be controlled from the interface are described by a simple tool-tip, while the mouse-pointer moves over a particular button.

As an alternative to run *Latte* directly from this window, the preference file in the generated zip-archive can be accessed from *Latte* by starting it directly from the command-line with

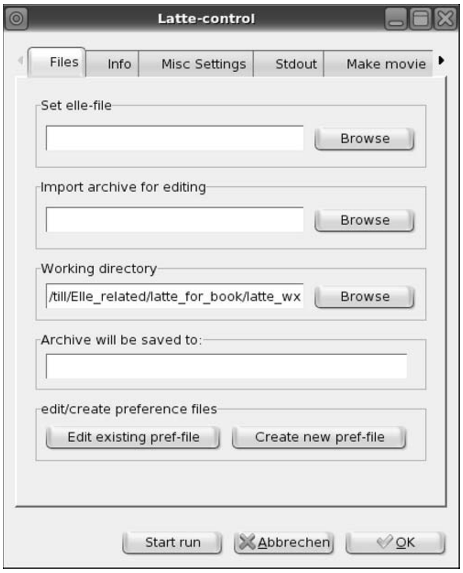
```
./latte_elle -u 0 -i inputfilename
```

The inputfilename is identical to the name of the zip-archive, apart from the file-ending, which will be '.zip' instead of '.elle'.

G.2.2 Installation notes

The interface is linked with wxWidgets - even though *Elle*/*Latte* can be installed using *lesstif* instead – therefore the Wx-package has to be installed on the system in order to use the interface.

On start-up the program needs to read an xml-file, called lattestart.xrc, containing mainly graphical information about the GUI. This file will be looked up in the directory, where the interface was started, and, if not found there, in the directories given by the 'PATH' variable (on *nix systems). Thus, if the executable will be copied to another location, either the xrc-file has to be copied to this location too, or the bin directory of the *Elle* package has to be included in the mentioned variable.



Working directory: /till/Elle_related/latte_for_book/latte_wx

Fig. G1. Interface-window on program start-up. Different functionalities can be accessed via tabs. In the open 'file'-tab, the necessary settings for a single simulation can be done

G.2.3 The main-window

The main window uses tabs, which give access to different basic functionalities of the interface. The following description is thus subdivided according to those tabs. The **Start**, **Cancel** and **OK** buttons are related to the *Latte*-executable itself and can be accessed from every tab.

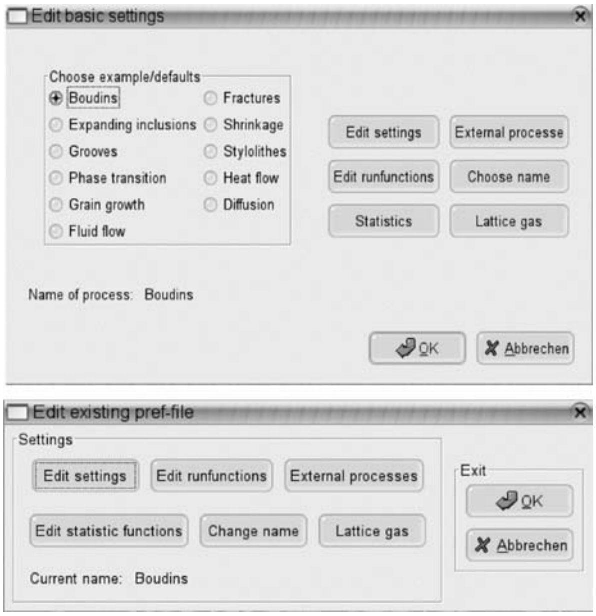
G.2.4 The 'Files' tab

Latte can be almost entirely controlled with the **Files** tab alone. Users can navigate to a working directory, where the input-file is located, and also edit and/or import preference-files from already existing zip-archives. The name of the created archive will be the same as the name of the input-file, except for the file ending.

To prepare the execution of *Latte*, two things have to be done: first an input-file has to be chosen by clicking the **Choose start-file** button, and browsing to an *Elle* file. Clicking **Create new pref-file** gives the opportunity to choose one of several predefined samples, which will set the preference file to useful default values, or, after importing a pref-file from a zip-archive, this can be edited with **Edit existing pref-file. Start run** will execute *Latte*.

The **Save-interval** spin box (in the **Misc settings** tab) allows to control how many pictures will be saved to disc. Output of the running simulation is displayed in the **Stdout** tab.

User-defined notes for a preference file will be displayed in the **Notes** tab, and can be edited there. They will be saved in the archive on finishing the interface.



Choose example/defaults: [✓ Boudins]

Name of process: Boudins

Current name: Boudins

Fig. G2. The 'New File'-window will set up a basic preference-file after choosing an example. The buttons will open other windows, which allow the refinement of the settings

G.2.5 The New file and the Edit file window

After having chosen to create a new preference-file, the dialogue on top of Fig. A16 will appear. It displays several predefined examples, which have been discussed in the book, and will give acceptable default-values. These settings, however, can be tweaked using the buttons on the right-hand side, which give control over the important functions and their parameters.

From this step on, the functionality of the **New file** and the **Edit file** window is identical. The edited variables will be saved, if the **OK** button is clicked in one of these windows.

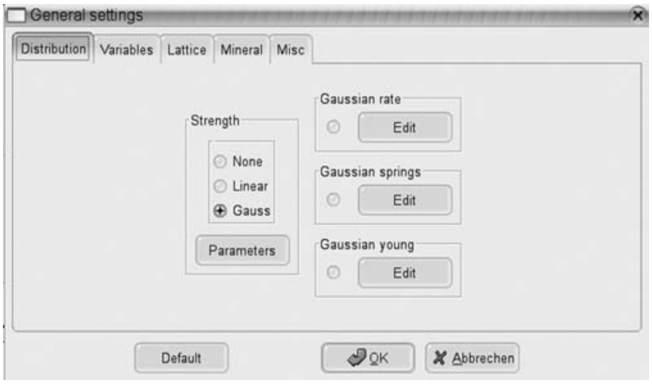


Fig. G3. General, non-run-function-settings for a *Latte*-run

The **Edit settings** button will pop up a window, which allows the adjustment of basic properties of the lattice or the discrete elements. These settings range from basic distributions on several variables to settings concerning the number of the produced output-files.

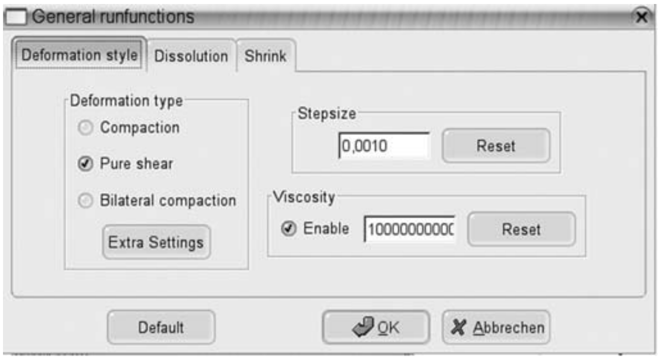


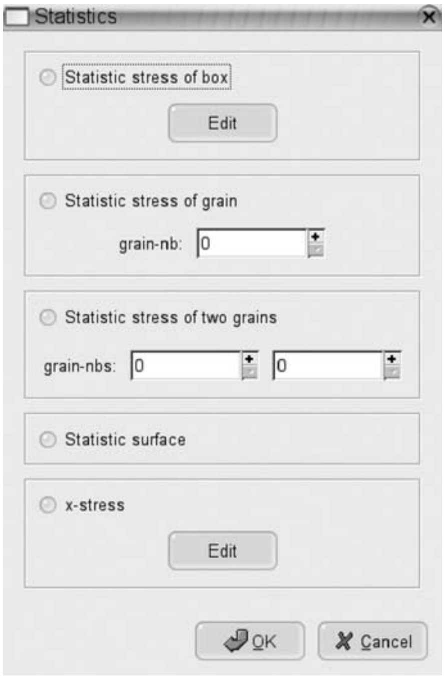
Fig. G4. General run functions and the style of deformation

Similarly the run-functions dialogue covers functions, which are called during every single loop of the program and control the deformation behaviour. It consists of tabs for the deformation style, the dissolution and for shrinkage of either all particles of the system or single grains.

The **deformation style** tab gives control over the movement of the system-walls. **Compaction:** stands for unilateral compression, pressing the upper wall of the system down without moving the remaining walls. **Pure shear:** Pushes the top wall down and extends the systems on the right-wall-side. **Bilateral compaction:** Compresses the system by moving the top and bottom wall. Used in stylolite-simulations.

In addition it is possible to combine two deformation-types at a time or using different deformation-types at different time-steps using the **Extra settings** button. The user may also chose to have a visco-elastic lattice via a checkbox, and the viscosity can be entered in a text-field.

On the tab **Dissolution**, stylolite dissolution and dissolution strain can be applied. Please have a look for the corresponding function-descriptions and chapters. The same advice can be given for the **Shrink** tab, which allows the shrinkage of grains and/or the total system.


Statistics

Statistic stress of box

Edit

Statistic stress of grain

grain-nb: 0

Statistic stress of two grains

grain-nbs: 0 0

Statistic surface

x-stress

Edit

OK Cancel

Fig. G5. The statistic options

G.2.6 The Statistics window

From the statistics window it is possible to enable the output of a file containing statistics. Note that these functions will be called every time the runtime-functions have been finished. See the list of functions in the appendix for details of the given functions.

G.2.7 Remaining tabs

The ‘Info’ tab: This includes a simple text control, where a description of the preference-file may be entered and saved. If importing a preference-file from a different zip-archive, this will be read and displayed.

The ‘Misc settings’ tab: The most important function of this tab, is the possibility to tell the interface where to look for the *Latte*-executable. If no distinct path to an executable is given, the current working directory will be the default, if the program finds it there. If there is still none, the program will have a look into the users ‘PATH’-variable (on unix/linux), and, finally, it will try to find it two levels up in the file-hierarchy in the ‘bin’ directory. In addition, one can define in this tab after how many time steps an elle-file will be saved to the hard drive.

The ‘Stdout’ tab: Displays the standard-output of latte, which would be displayed on the command-line if started from there. (This doesn’t display error-messages, if the program hangs! If you’re developing and testing your own routines, don’t start *Latte* from here, but use a shell). Unfortunately, the displayed output is not always synchronized with what really happens in latte (this seems to be an wxWidgets issue for which, however, the author didn’t find a workaround). It is possible to save this output to a user-defined text-file.

The ‘Make movie’ tab: This is an attempt to provide a more or less convenient way of shooting movies from the *Elle* files produced by a run of latte. The usage is straightforward: choose the first and the last file of the sequence you want to shoot a film of, select a name for it and hit the start-button.

The interface-program relies on some external executables. Firstly it has to find *showelleps* on your system, which is in the bin/binx/’binwx directory within the *Elle* location. It is most advisable - not only for this particular purpose - to integrate this directory into your ‘PATH’-variable anyway. Alternatively, the program will be looked up in the current working directory itself, and finally – if there is still no success – it tries to find the bin-directory relative to the working-directory (two levels up in the file hierarchy). To convert the generated postscript-files into an animated gif-file, the ‘imagemagick’-package is needed – this should be found without problems, if installed on the system.

The ‘Convert movie’ tab: If you’re not interested in the animated gif, which was created by **Make movie**, you can convert it into avi or mpeg, which will consume much less space on your hard drive and thus load much faster - and back to gif, if needed.

As usual an external program is needed, in this case *mencoder*, which is in turn part of the mplayer-package. If installed on your system, there shouldn't be any difficulties to use it from within the interface.

G.3 The command line interface for Windows and Linux developers

A switch command is used in the initialization and the run routine in the experiment class to define processes. When the experiment is called the -u command followed by an Integer calls the right process:

```
./elle_tte -i file.elle -u 1
```

for example, calls a simple fracture experiment.
The following processes are included in Latte as examples:

-u 1	fracturing of a granular aggregate,
-u 2	fracture boudinage of horizontal layers,
-u 3	fractures around expanding grains,
-u 4	shrinkage cracks,
-u 5	visco-elastic deformation and fracturing,
-u 6	dissolution grooves,
-u 7	stylolite roughening,
-u 8	combine grain growth and fracturing,
-u 9	solid-solid phase changes under stress,
-u 10	heat diffusion,
-u 11	grain growth,
-u 12	Lattice-Gas example diffusion,
-u 13	Lattice-Gas example 2d fluid flow.

Parameters in functions and basic functions for processes can be changed within the experiment class. The file experiment.cc starts with a number of header calls followed by a constructor (Experiment::Experiment()). After the constructor the function Experiment::Init() is used to initialize the different processes. Within this function the *Elle* file is called, desired lattices are build and initial geometrical configurations are applied to the data set (statistical distributions, layering etc.). The initialization function is called if the **Run** button in the *Elle* interface is chosen at the beginning and also if the **ReRun** button is chosen. Please note that a rebuilding of some lattices with the **ReRun** command may not be possible, especially if the microstructure is already deformed.

The run functions are then executed in another switch call in the function Experiment::Run().

In this function the user can similar to the init function change parameters in existing functions or add new functions for different processes. Basically all functions that are included in the lattice, phase_base, phase_lattice, min_trans_lattice and graingrowth classes can be called within the experiment class.

Appendix H Miscellaneous processes and utilities

Mark Jessell

H.1 Introduction

In addition to the processes already documented in this chapter, we have also developed a number of other processes that are briefly described here for the sake of completeness, and because they may act as template for those wishing to develop their own codes. We also describe the functionality of the utility codes, which are not meant to represent specific geological processes, but instead perform useful conversion functions.

H.2 *elle_gbdiff*

This code performs a Finite Difference grain boundary diffusion calculation for a uniform-width grain boundary network and constant diffusivity. The code is currently set to diffuse the bnode property CONC_A, but could be easily modified.

H.3 *elle_diff*

This code performs a Finite Difference lattice diffusion calculation, which ignores the grain boundary network, and assumes constant diffusivity. The code is currently set to diffuse the flynn property CONC_A, but could be easily modified. This code performs a similar function to the lattice diffusion part of the *elle_exchange* code, except that it is not limited to intragranular diffusion.

H.4 *elle_met*

This is a prototype code derived from the *elle_melt* code that attempts to simulate polyphase solid-state metamorphic reactions. It uses a standalone array of thermodynamic data derived, for example from the *Perplex* code (http://www.perplex.ethz.ch), to simulate reactions of the type A+B=C+D.

H.5 *elle_recovery*

This code performs a simple time-based recovery of the flynn DISLOCDEN attribute (dislocation density) that allows the gradual decay of this property.

H.6 *elle_viscosity*

This code assigns each flynn a viscosity as a function of its properties (e.g. grain size and/or dislocation density). This code is typically rewritten for each new type of experiment.

H.7 *elle_expand* & *elle_pblast*

These two codes force any grains with their flynn attribute EXPAND set to 1 to slowly grow, consuming their matrix.

H.8 *elle_manuel*

This code applies a homogeneous simple shear to the entire microstructure.

H.9 *reposition*

This utility code transforms the deformed *Elle* microstructure from a Finite Element or *elle_manuel* calculation so that all nodes lie within the original bounding box. This is possible because *Elle* models have cyclic boundaries.

H.10 *tidy*

This utility code transforms the *Elle* microstructure in a number of different ways, including changing the bnode spacing, adding or removing unodes, randomising Euler angles of all grains etc. To modify an Elle file, you can use the tidy utility with the following user data settings, accessible from the Run Options Window:

- **User Data 1** Change default node spacing. Input value wil be the new switch distance. Setting the value to 0 keeps the current switch distance.
- **User Data 2** Set to anything but 0 to randomise the Euler angles.
- **User Data 3** If the value set here is equal to F_ATTRIB_A of a Flynn, this Flynn will get the mineral attribute "MICA". All other Flynns will be assigned the mineral attribute "QUARTZ". Set to zero to not use this function.
- **User Data 4** Setting the input to anything other than zero reorients the C-axis (Euler angle) of "MICA"-Flynns to become parallel to the long axis of the Flynn.
- **User Data 5** Add or delete unodes from file. Any valy smaller than zero will delete all unodes, while values bigger than zero determine the number of unodes.
- **User Data 6** Set the unode pattern: 0 = hexagonal array; 1 = square array; 2 = random; 3=semi_random.
- **User Data 7** If "User Data 6" was set to 3 (semi random), it defines the number of unodes per cell, otherwise set to 0.

H.11 *elle2poly*, *basil2elle*, *elle2oof*, *goof2elle*

These utility codes transforms the *Elle* microstructure into a format that the Finite Element codes *Basil* (*elle2poly*) and *OOF* (*elle2oof*) can understand, and similarly transforms the outputs from *Basil* (*basil2elle*) and *OOF* (*goof2elle*) back into the *Elle* format.

H.12 *ppm2elle*

This utility code takes a binary format *ppm* format raster image and transforms it into an *Elle* format file. For this to work the microstructure in the *ppm* file must consist of grains with completely uniform colours, with no lines demarking grain boundaries, and island grains completely within other grains are not allowed.

H.13 *ebsd2elle*, *elle2ebsd*

This utility code *ebsd2elle* takes the Euler angle information that describes the crystallographic orientations in an HKL format EBSD image and maps it onto the unodes of an *Elle* file. The grain boundary network information needs to be transferred as a separate step using *ppm2elle. elle2ebsd* maps the unode Euler angle information onto a regular grid and writes a file in HKL-format.

H.14 *plotaxes*

The utility code creates an equal angle upper hemisphere postscript plot of the c- a- and r- axes of the quartz grains in an *Elle* file.

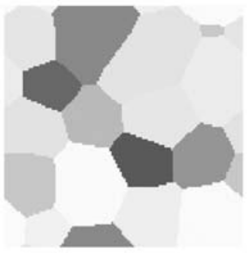
Appendix I How to Create an *Elle* file

Mark W. Jessell

We can create an arbitrary geometry *Elle* input file from any rectangular binary raster image. Any rectangular bounding box will be mapped to a square, so generally a square image is a good place to start, with dimensions of around 100x100 pixels. Each patch of pixels with a unique colour will be considered to be a single grain, so in order to minimise the number of individual grains, you should limit the number of different colours that appear in the image. Do not draw the grain boundaries, only the grains. In this example we use the Open Source drawing package *gimp* (http://www.gimp.org) to create the image, but any raster or vector package that can output a raster image (ppm-format) could also be used. We assume that there is a pre-existing raster that you want to modify, but you could equally start from a blank screen.

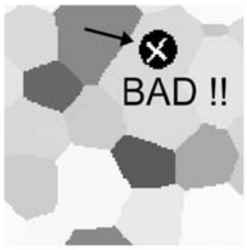
The example we use was actually created by saving out an existing *Elle* microstructure as a graphics file, with no grain boundaries displayed (this is important as otherwise the grain boundaries would be considered to be a different grain).

1. Start up *gimp* by typing in **gimp** (Linux) or click on the appropriate icon (Windows).
2. Select **Open** from the **File** menu and load the image foam.ppm (found in the ...elle/extras/ppm_files directory).

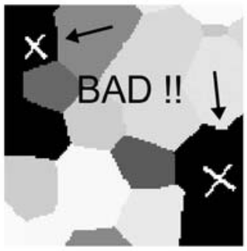


3. Use the **Pencil** tool ✎ to modify your microstructure, using a different colour for each neighbouring grain. Grains can wrap around horizontal and vertical boundaries, but:

Don't make island grains (i.e. grains completely surrounded other grains)



Don't make any single grain larger than about 30% of the length of the box (be careful with the wrapping resulting in grains being bigger than they appear).



4. Save the image by right-clicking in the image window and select **Save as**, and call you file any_name_you_like.ppm.
5. Quit from *gimp*.
6. Open up the *Experiment Launcher* and select and run *ppm2elle.*
7. Load up the ppm-format file you just created, which will be converted as it is loaded and then save the file as a new *Elle*-format file.
8. Next, go back to the *Experiment Launcher* and select and run **tidy**. Load the new Elle file, and go to **Run→Run Options** and change "Stages" to 1 and the **UserData field 1** to 0.01, and change all the other **Userdata** values to 0.0. Now select **Run→Run**. This has changed the bnode spacing to 0.01 *Elle* units.
9. Save out the new *Elle* file.
10. When the normal Elle window appears, open the *Elle* file you created in the previous steps and then go to **Run→Run Options** and change "Stages" to 100. Finally select **Run→Run** and you will see the new microstructure evolve through 100 time steps.

You will now have an *Elle* file that describes the geometry and topology of your microstructure, but will not have any specific physical or chemical attributes set as yet. See Appendix J to see how to add attributes with a text editor. Before you use the new data file for a simulation, it is often advisable to run ten or more steps of grain growth (step 10). This way you can see that the microstructure has no hidden problems and unwanted small (one or more pixel) grains are removed.

Appendix J The *Elle* file format

Mark W. Jessell

The *Elle* File is a text file that describes the general experimental conditions, followed by a description of the geometric, physical and chemical characteristics of the specimen, i.e. its microstructure. The file can be modified with any simple text editor, or even a word processor, as long as it is saved out again as a text file. If you have created a new microstructure using *ppm2elle*, you may add lists of new flynn, bnode or unode attributes at the end of the file (the order of the parameter blocks is not important). Often you will simply want to set default values, perhaps defining a few elements as having non-default values.

J.1 Example *Elle* file

```
# Created by test_unodes: elle version 2.3.2  Fri Jun  8 13:38:25
2001
# LINES THAT START WITH A # ARE COMMENTS

# THE OPTIONS BLOCK DEFINES GLOBAL PARAMETERS

OPTIONS
SwitchDistance 2.50000000e-02
MaxNodeSeparation 5.50000000e-02
MinNodeSeparation 2.50000000e-02
SpeedUp 1.0
CellBoundingBox 0.00000000e+00 0.00000000e+00
                1.00000000e+00 0.00000000e+00
                1.00000000e+00 1.00000000e+00
                0.00000000e+00 1.00000000e+00
SimpleShearOffset 0.00000000e+00
CumulativeSimpleShear 0.00000000e+00
Timestep 3.15000000e+07
UnitLength 1.00000000e-02
Temperature 2.50000000e+01
Pressure 1.00000000e+00
MassIncrement 0.00000000e+00

#THE NEXT BLOCK DEFINES THE POLYGONAL STRUCTURES: THE FIRST NUMBER IS
THE ID <E.G. 3>, THE SECOND THE NUMBER OF NODES THAT DEFINE THE
POLYGON <E.G. 23>, ANS FOLLOWED BY AN ANTICLOCKWISE LIST OF NODE IDS
E.G. <133 272 140 ETC.>

FLYNNS
3 23 133 272 140 146 112 174 87 15 147 150 211 129 17 39 62 94 119 20
13 8 72 135 7
7 36 21 76 31 29 73 107 279 302 166 280 167 30 224 156 209 27 131 25
16 77 132 2 0 141 4 121 1 10 86 122 155 227 22 3 37 105
```

```
11 24 20 119 94 62 39 17 33 165 34 48 49 162 51 50 88 36 38 56 78 29
31 76 21 142
#etc.

#THE NEXT BLOCKS DEFINE FLYNN PAAMETERS SO TAHT FOR EXAMPLE, AN
EXPAND PARAMETER IS DEFINED FOR THESE FLYNNS, THE DEFAULT VALUE IS 1
AND THE FLYNN WITH ID 17 IS SET TO 1

EXPAND
Default 1
17 0
DISLOCDEN
Default 0.00000000e+00
MINERAL
Default QUARTZ
EULER_3
Default 0.00000000e+00 0.00000000e+00 0.00000000e+00

#THE LOCATION BLOCK DEFINES THE LOCATION OF BOUNDARY NODES. THE FIRST
NUMBER IS THE BNODE ID, THE SECOND IS X, THE THIRD IS Y

LOCATION
0 0.7157127300 0.0004124800
1 0.5647232500 0.9642747600
2 0.7517333600 0.9709875600
3 0.4205657800 0.7458517000
4 0.6626872400 0.9829956900
272 0.1584355500 0.7560001600
279 0.5814173200 0.6308649800
280 0.6592591400 0.6547707300
284 0.5687853700 0.4511538400
289 0.1280806400 0.3809446100
293 0.2484257500 0.1165237400
302 0.6079290500 0.6402233800
#etc.

#NEXT COMES BNODE PROPERTIES
CONC_A
Default 0.00000000e+00
29 4.0

#THE NEXT BLOCK DEFINES THE POSITION OF UNCONNECTED NODES. THE FIRST
NUMBER IS THE UNODE ID, THE SECOND IS X, THE THIRD IS Y

UNODES
0 0.000000 0.008660
1 0.020000 0.008660
2 0.040000 0.008660
3 0.060000 0.008660
4 0.080000 0.008660
5 0.100000 0.008660
6 0.120000 0.008660
#etc.

#NEXT COMES UNODE PROPERTIES
U_CONC_A
Default 1
0 1.2
```

J.2 *Elle* file blocks

The file format consists of a series of 7 sets of data blocks:

The **OPTIONS Block** - This defines the general experimental conditions:

```
SwitchDistance The distance between triple nodes before they are
      switched
MaxNodeSeparation The maximum distance between double nodes before a
      new node is inserted
MinNodeSeparation The minimum distance between double nodes before
      they are merged
SpeedUp Arbitrary rate multiplyer for some processes
CellBoundingBox Bottom Left, Bottom Right, Top Right and Top Left
      coordinates of bounding Box
SimpleShearOffset Used to determine the Bounding box shape for simple
      shear deformation experiments
CumulativeSimpleShear Used to determine the Bounding box shape for
      simple shear deformation experiments
Timestep Single timestep in seconds
UnitLength Real World width of unit bounding box
Temperature Temperature in Kelvin
Pressure Pressure
MassIncrement Used to add mass to grian boundaries for validation
      tests
```

The **FLYNNS Block** - The next block defines the polygonal grain structures: the first number is the id <e.g. 3>, the second the number of nodes that define the polygon <e.g. 23>, followed by an anticlockwise list of node ids e.g. <133 272 140 etc.> All flynns will be defined here. The bnode ids refer to the ids found in the Bnode Block.

```
FLYNNS
3 23 133 272 140 146 112 174 87 15 147 150 211 129 17 39 62 94 119 20
13 8 72 135 7
7 36 21 76 31 29 73 107 279 302 166 280 167 30 224 156 209 27 131 25
16 77 132 2 0 141 4 121 1 10 86 122 155 227 22 3 37 105
11 24 20 119 94 62 39 17 33 165 34 48 49 162 51 50 88 36 38 56 78 29
31 76 21 142
17 25 50 123 110 204 164 158 64 53 67 115 128 85 68 24 91 180 40 23
185 182 284 96 38 36 88
#etc.
#etc.
#etc.
```

The **FLYNN Parameters Block(s)** - The next set of blocks define flynn parameters so that, for example, an EXPAND parameter is defined for these flynns, the default value is 1 and the flynn with id 17 is set to 0. You can only define parameters that the *Elle* system recognises, however three dummy parameters are provided.

Each defined parameter consists of its name, followed by a Default value definition, followed by the list of flynns that do not have the default value and their values. All FYLNN parameters are optional, and all definitions apart from the default value are optional, if a parameter is defined as being present. In the example below all flynns would have a value for EXPAND of 1, expect flynn number 17, which has a value of 0.

```
EXPAND
Default 1
17 0
DISLOCDEN
Default 0.00000000e+00
3 21.9
17 23.32
MINERAL
Default QUARTZ
EULER_3
Default 0.00000000e+00 0.00000000e+00 0.00000000e+00
```

The current list of valid flynn parameters is:

F_ATTRIB_I, F_ATTRIB_J, F_ATTRIB_K (dummy integer attributes to be assigned by the user)
MINERAL (Currently, valid keywords are QUARTZ, FELDSPAR, MICA, GARNET, CALCITE, MINERAL_A, MINERAL_B, MINERAL_C)
EXPAND (toggles grain expansion for elle_pblast and elle_expand)
COLOUR (this attribute will become redundant, a number in the range 8-63)
ENERGY (Arbitrary volume energy term)
VISCOSITY (Viscosity term)
S_EXPONENT (Stress exponent for power-law viscosity)
DISLOCDEN (dislocation density)
F_ATTRIB_A, F_ATTRIB_B, F_ATTRIB_C (dummy floating point attributes to be assigned by the user)
CAXIS (polar coordinates of c-axis orientation)
EULER_3 (Euler space coordinates of lattice orientations)
FLYNN_STRAIN (current strain state as calculated by *Basil* [followed by at least one of E_XX, E_XY, E_YY, E_YX, E_ZZ, F_INCR_S, F_BULK_S])

The **BNODES Block** - The location block defines the location of boundary nodes. The first number it the bnode id, the second is *x*, the third is *y* coordinate.

```
LOCATION
0 0.7157127300 0.0004124800
1 0.5647232500 0.9642747600
2 0.7517333600 0.9709875600
3 0.4205657800 0.7458517000
4 0.6626872400 0.9829956900
5 0.0854129600 0.0134190800
255 0.1195778800 0.1301631600
272 0.1584355500 0.7560001600
279 0.5814173200 0.6308649800
280 0.6592591400 0.6547707300
284 0.5687853700 0.4511538400
289 0.1280806400 0.3809446100
293 0.2484257500 0.1165237400
302 0.6079290500 0.6402233800
#etc.
#etc.
#etc.
```

The **BNODES Parameters Block(s)** - The same logic applies as for flynn parameters.

```
CONC_A
Default 0.00000000e+00
29 4.0
```

The current list of valid bnode parameters is:

N_ATTRIB_A, N_ATTRIB_B, N_ATTRIB_C (user assigned node attributes)
VELOCITY VEL_X VEL_Y (the velocities in the *x* and *y* directions)
STRESS The local stress state as calculated using *Basil*. [TAU_XX TAU_YY TAU_ZZ TAU_XY TAU_1 PRESSURE] (at least one)
CONC_A (a dummy concentration attribute)
STRAIN (the local strain state as calculated using *Basil* [INCR_S BULK_S]; at least one)

The **UNODES Block** - The next block defines the position of unconnected nodes. The first number it the unode id, the second is *x*, the third is *y* coordinate. This block and the unodes parameter blocks are optional.

```
UNODES
0 0.000000 0.008660
1 0.020000 0.008660
2 0.040000 0.008660
3 0.060000 0.008660
4 0.080000 0.008660
5 0.100000 0.008660
6 0.120000 0.008660
7 0.140000 0.008660
#etc.
#etc.
#etc.
```

The **UNODES Parameters Block(s)** - The same logic applies as for flynn and bnode parameters.

```
U_CONC_A
Default 1
0 1.2
```

The current list of valid unode parameters is:

U_CONC_A (Attribute used to store chemical concentration)
U_ATTRIB_A, U_ATTRIB_B, U_ATTRIB_C (user assigned unode attributes)
U_STRAIN START_S_X START_S_Y PREV_S_X PREV_S_Y CURR_S_X CURR_S_Y (the *x*,*y* coordinates for the starting position, the position before the last increment of strain and the current position)
U_CAXIS (Polar coordinates of c-axis orientation)
U_EULER_3 (Euler space coordinates of lattice orientations)
U_ENERGY (Arbitrary volume energy term)
U_VISCOSITY (Viscosity term)
U_DISLOCDEN (dislocation density)

Index

Page numbers referring to the entries discussed in detail are given in **BOLD.**

Printing: Krips bv, Meppel, The Netherlands
Binding: Stürtz, Würzburg, Germany

LNES

LECTURE NOTES IN EARTH SCIENCES

106

Microdynamics Simulation

Paul D. Bons
Daniel Koehn
Mark W. Jessell
Editors

COMPACT disc CD-ROM

ISBN 3-540-25522-2

Springer

Bons · Koehn · Jessell (Eds.)

Microdynamics Simulation

This volume deals with the simulation of metamorphic and tectonic microstrucutres in rocks with a special emphasis on the modelling package „Elle". The first part provides a review of the problems and opportunities in the modelling of microstructures, followed by an introduction to various numerical modelling techniques. In the second part examples of the modelling of different processes which alter rock microstructure are presented, beginning with individual processes and progressing to modelling multiple coupled processes. The accompanying CD not only enables readers to carry out these simulations themselves and but also helps them to design their own simulations.

with CD-ROM

System Requirements:

ISSN 0930-0317

ISBN 978-3-540-25522-2

9 783540 255222

springer.com