Pecube - version 3

Jean Braun

September 14, 2010

Contents

Abstract

This is the user guide for version 3 of PECUBE, a finite element code that solves the heat transport equation in three dimensions with the purpose of testing geomorphic and tectonic scenarios against thermochronological data. The code includes the effect of heat conduction and advection (resulting from tectonic transport, such as movement along faults or block uplift/subsidence), heat production by radioactive elements in the crust or resulting from shear heating, the effect of finite amplitude topography. This new version incorporates many changes, including the ability to predict ages below the surface (in a tunnel or in a well), to use surface geometries derived from a surface processes model, such as CASCADE, reading an external topography files constructed on an irregular (triangular) discretization, etc. We have also incorporated the Neighbourhood Algorithm of Malcolm Sambridge into PECUBE in its parallelized form to enable the direct inversion of thermochronological data. We have also enriched the library of thermochronological systems that can be predicted, including Richard Ketchams fission track algorithm, fission track length distributions for both apatite and zircon, as well as included a simple algorithm to predict age distribution at the surface of the model at any arbitrary time in the past, for direct comparison with detrital age distributions.

1 Purpose

Pecube was initially designed to predict cooling ages for low-temperature chronometers such as apatite fission track or Helium in apatite, mostly in response to landform evolution. It has been used by a relatively large community who has also contributed to its development through constant requests to see it improved and updated. It has consequently evolved from its original structure to be more user friendly but also more complex in the variety of scenarios that can be tested and thus in its setting up. This new version of the user guide attempts to help the new user to comprehend the code structure and basic functions; it also contains the information that will enable the more confirmed user to use all of its new or hidden functions.

When publishing results obtained with PECUBE, please make reference to Braun (2003).

This new version has benefited from the help of many, including Xavier Robert, Claire Perry, Christoph Glotzbach, Vivi Pedersen and Frédéric Herman,

2 Directory structure

To better organize the use of PECUBE, a directory structure has been adopted that separates the various components into subdirectories: input, output, VTK files, code, data, etc.

The *source* subdirectory contains the Fortran codes of the various subroutines, including those needed by the program *Test* and *Vtk*. These routines should be left unchanged unless you know what you are doing.

The executables are stored in the *bin* subdirectory. To run any of the programs, you must be in the main directory and call the executable by prefixing its name with *bin/*.

The *input* subdirectory contains the two input files described above.

All VTK files are stored in the *VTK* subdirectory.

All output files are stored in the *RUN00* subdirectory (or any other name given in the input file).

All large datasets (topographic and age datasets) are stored in the *data* subdiretory.

This documentation is in the *doc* subdirectory.

3 Compiling an running PECUBE

To compile PECUBE , go to the *source* directory:
>*cd source*
and make PECUBE:
>*make*
To compile all other utilities and PECUBE, go to the *source* directory:
>*cd source*
and
>*make all.*

PECUBE can also be compiled for use on a parallel computer, on which the

MPI library has been installed. You will find instructions in the Makefile to do this. Parallelisation will only improve PECUBE runs, not the pre- or post-processors. Note that this option is only meaningful when you run PECUBE in inversion mode. Note also that some compilers will not allow to have an 'include' command preceded by a 'd' as an indication of a commented line; if this is the case for your compiler, the routine *na.f* will not compile properly. You will need to edit the *na.f* file and remove the "d" in the first column of three lines that start with include 'mpif.h'.

To run Pecube, change the input files in *input/fault_parameters.txt* and *input/topo_parameters.txt* to define the setup of your run or experiment, by using your favorite text editor. Beware of not introducing spurious hidden characters by using a fancy word processor such as *WORD* or Pages. This would make the input files unreadable by PECUBE. Use a simple text file editor such as *WordPad* or *vi.* Then simply run PECUBE from the main directory, by typing:
>bin/Pecube.
Note that this command should not be used if using PECUBE in parallel (NA) mode. In this case, you should refer to your MPI user guide; submitting an MPI process usually looks like:
>mpirun -np 8 bin/Pecube
The output is sent to the *RUN00* directory (its exact name can be specified in *input/topo_parameters.txt*).

Before running PECUBE, you may wish to check that your input files correspond to what you are trying to do (shape of the faults, timing of movement on the faults, surface landform, etc...) To do that, run the Test program from the main directory:
>bin/Test
This will generate VTK files in the *VTK* directory that you can use to check the evolving geometry of the faults and the computed velocity field and the topography.

To produce VTK files from the outputs of a simulation, run the *Vtk* program from the main directory:
>bin/Vtk
This will generate VTK files in the *VTK* directory that you can use to display the temperature field, the ages and the exhumation rate.

You can automate these three actions (Test, Pecube, Vtk) by issuing a single command:
>*bin/run.sh*
Note that this command should not be used when running Pecube in NA mode.

To produce VTK files from the results of an inversion (using NA), run the *ShowNA* program from the main directory:
>*bin/ShowNA*
This will generate VTK files in the *VTK* directory that you can use to display the results of the inversion as a set of 3D plot files showing the location of the various runs in paramater space, using as many triplets of axes as required to show all possible combinations. The color of the circles representing the runs is proportional to the misfit value.

To create a transportable archive of PECUBE, run the *tarngo.sh* shell script from the main directory:
>*./tarngo.sh*
This will create a *Pecube.tar.gz* file that contains the content of the source, input and data directories; the other directory are also created in the tar file but they are empty (no solution is stored). This feature is useful if you want to report a bug or a difficulty in running PECUBE. The Pecube.tar.gz file should be relatively small to be sent by email, unless you have very large topographic datasets.

4 Input files

Two input files are used in this version of PECUBE. They are used by PECUBE to create an input file that has (approximately) the same format as the input file for the first version of PECUBE. This ensures some kind of backward compatibility with the previous versions. The input file for this version (3) is identical to that of version 2, except for a couple of lines added at the end of the *topo_parameters file.txt* and new options that can be reached by giving negative values to some of the integer parameters. Note that all PECUBE version 2 input files should still work with PECUBE version 3.

Both input files allow the introduction of comment lines (to help describing the input parameters) that start either with a dollar sign or an empty character. These lines are skipped by PECUBE.

Topographic input file

The first input file is called *topo_parameters.txt.* It must contain the following lines in the following order:

run	Name of the simulation or name of the directory in which the output of the simulation will be saved; note that the directory must exist prior to the simulation start (type character*5)
topofnme	Name of the input topographic file (DEM) containing a list of nx by ny height values in meters; the DEM starts at the bottom left corner and is given row by row. If the name of the input file ends with a / character, the topography should be stored in a subdirectory of the *data* directory in the form of a series of file named topo0, topo1 to topo(nstep). This feature is designed to enable the user to use landform geometries obtained from a Landscape Evolution Model (LEM) such as CASCADE. The user must also provide a series of uplift and temp files that contain instantaneous values of the uplift rate and surface temperature computed by the LEM. They should be called uplift0, uplift1 to uplift(nstep) and temp0, temp1 to temp(nstep), respectively. An option also exists to read in irregular topographies (i.e. stored on an irregular, non-rectangular grid). This option is described below. If topofnme is set to Nil the surface topography is assumed to be flat at all times.(type character*(*))
nx0,ny0	dimension of the input DEM; if nx0 and ny0 are negative, their meaning change: they become the number of nodes and triangles making up the irregular DEM. In that case, the user must also include a file named topofnme.geometry that contains ?nx pairs of coordinates (x and y) in degrees of longitude and latitude, followed by ?ny triplets of node numbers describing the triangular connectivity among the nodes. (type integers)
dlon,dlat	DEM longitude and latitude spacing in degrees (not used when $nx < 0$) (type float)
nskip	skipping factor used to decrease the resolution of the DEM; this parameter is very useful if you wish to test scenarios at low resolution; when constructing

	the 3D finite element mesh, both nx and ny are divided by nskip (not used when nx ¡ 0) (type integers)
lon0,lat0	longitude and latitude of the bottom left corner of the DEM (type floats)
nstep	the number of times steps in the geomorphic scenario (type integer)
tau	the erosional time scale (the successive topographies given at the various steps are interpolated using an exponential function of time with tau as a decay factor). Most of the time you will wish to turn this off and assume that the change in topography between two time steps is linear; to achieve this use $\tau = 0$ (type float)
t,a,o,io	1 of these per time step+1 (i.e. you need to include nstep+1 lines like this one), t is the time at which the step starts (in Ma in the past, i.e. geological time), a is the amplification factor by which the topo read in topofnme will be scaled, o is the offset factor by which it is moved up or down, and io is a flag to determine whether an output (temperature, velocities and detrital ages) is requested at this time (0 for no output, 1 for output) (a and o are not used if $nx < 0$)(type 3 floats and 1 integer)
f0,rc,rm,E,n,L,nx,ny	f0 is a flag to allow for flexural isostaty calculations (0 no isostasy, 1 isostasy), rc and rm are the crustal and mantle densities (in kg/m3) respectively, E is young modulus (in Pa), n is Poisson's ratio (dimensionless), L is elastic thickness (in km), and nx and ny the resolutions in both directions of the FFT mesh used to calculate the isostatic flexural response (must be powers of 2) (type 1 integer, 5 floats, 1 integer)
zl,nz,k,tb,tt,la,pr	zl is the crustal (model) thickness (in km), nz is the resolution in the z-direction, k is the heat diffusivity (in km2/Myr), tb is the temperature at the base of the model (in °C), tt is the temperature at the top of the model at mean sea level (in °C), la is the lapse rate (in °C/km), pr is the heat production (in °C/Myr) (type 1 float, 1 integer, 5 floats)
agefnme	Name of the file containing observed ages (Nil if no data). (type chatacter*(*)). The file should contain the number of observations nobs, then nobs lines containing

1. the longitude
2. the latitude
3. the elevation (note that if the age has been obtained from a well, you should input here the depth below the surface as a negative number)
4. apatite He age
5. error in apatite He age
6. apatite FT age
7. error in apatite FT age
8. zircon He age
9. error in zircon He age
10. zircon FT age
11. error in zircon FT age
12. Kspar Ar age
13. error in Kspar Ar age
14. biotite Ar age
15. error in biotite Ar age
16. muscovite Ar age
17. error in muscovite Ar age
18. hornblende Ar age
19. error in hornblende Ar age
20. normalized FT length distribution (apatite) as an array of length 17 (ftld(k) is the proportion of tracks with length between $k - 0.5$ and $k + 0.5$ microns; the sum of ftld(k) for $k = 1, 17$ must be equal to 1); if no FT length information use -1 for all 17 values

Negative ages mean no data.

tp,f1,f2,f3,fr tp is the time of the previous resetting event (for all systems); when a rock does not cross the closure temperature of a given system, its age is set to tp, unless tp is smaller than the time duration of the model run; f1 is a flag that determines which apatite FT routine is used (0 means Peter van der Beek's routine; 1 means Ketcham's routine); f2 is a flag that determines whether to use the differences of the ages (f2=0) or the differences in slopes of the age elevation relationship (f2=1) to construct the misfit function; f3 is a flag that determines whether the faults are advected by one another (f3=1) or not (f3=0); fr is friction coefficient used to multiply an arbitrary stress value of 100 Mpa to compute the heat produced by friction; the heat is calculated as the product of strain rate, stress, and friction coefficient, divided by the specific heat capacity (assumed to be 800 J/C/kg) (type 1 float, 3 integers, 1 float)

f1 to f9 a series of 9 flags to determine which of the ages should be computed; the flags correspond to f1: He in Apatite, f2: He in Zircon, f3: FT in Apatite, f4: FT in Zircon, f5: Ar in Kspar, f6: Ar in Biotite, f7: Ar in Muscovite (White Mica), f8: Ar in Hornblende, f9: Apatite FT length distribution. A value of zero means that the age is not computed (0 will appear in the output file), a value of one means that the age will be computed.

Tectonic input file

The second input file is called *fault_parameters.txt.* It must contain the following lines in the following order:

nfault number of faults (this is followed by a series of lines that must be entered for each of the nfault faults) (type integer)

x1,y1,x2,y2 Two lon-lat pairs defining the trace of the fault at the surface. The fault is located to the right of that line (not used if $n < 0$, as described below)(type floats)

n n is the number of points defining the fault; the order in which the points are listed will determine which of the two half spaces defined by the fault moves: it is the one to the right of the fault when going along the fault in the order given in the input file. Note that if n is negative, no fault is assumed and the tectonic velocity field is assumed to be purely vertical and vary as a bi-linear function of space, the value of which is pinned at the four corners of the mesh/DEM (type integer)

r$_i$,s$_i$ n lines containing the r-, s-coordinates of the points defining the fault; if nfault ¡ 0, only one line should be given, containing 4 numbers, i.e. the value of the velocity field at each of the four corners of the mesh (lower left, lower right, upper left, upper right) (type floats)

nstep number of time intervals defining the movement on the fault (type integer)

tstart,tend,velo nstep lines containing the starting and end times (in geological time; note that if you wish this time interval to start when the previous one finished, input a * (star/times symbol) instead of a time) and a velocity (positive for normal faulting, negative for thrusting) (in case where $n < 0$, a positive value means uplift, a negative value means subsidence) (type floats)

5 Faults

In this section we describe the way faults are defined by a set of linear segments and how the velocity field is determined. This is necessary as there is no uniquely defined solution to this kinematic problem.

Fault geometry

Let's us first consider the geometry of a single fault as incorporated in PECUBE. It is first defined by a local *fault coordinate system* (r, s) that is different from the *global three dimensional coordinate system* (x, y, z). As shown in Figure 1, this system is fully defined by the position of two points (x_1, y_1) and (x_2, y_2) in the horizontal plane and at the surface $z = z_l$, where z_l is the surface of the model, as defined in PECUBE. The r-axis of the fault coordinate system is located to the right of that line (when going along the line from point 1 to point 2), the s-axis is vertical and the origin is located anywhere along the line at $z = z_l$. To build a fault to the left of a given line, simply invert the order of the two points. That the origin is not strictly defined implies that the fault geometry is two-dimensional and its definition does not depend on its location along the line.

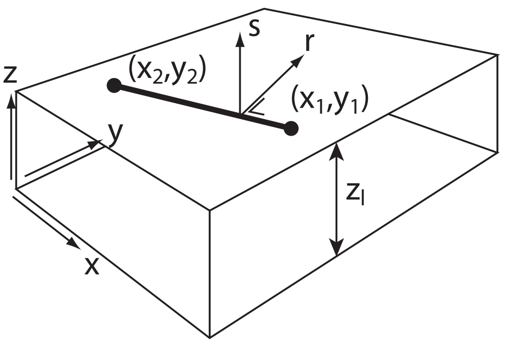

Figure 1: The local fault coordinate system and the global PECUBE system.

In the (r, s) plane, the fault is defined through a series of n connected points (r_i, s_i). The segments connecting the points define the fault trace in the (r, s) plane. The end segments are assumed to continue indefinitely.

The order in which the nodes are given is important: it determines which half-space (there is one on either side of the fault) moves with respect to the fault. When moving along the fault in the order in which the nodes are given, the half space that moves with respect to the fault is the one to the right of the fault. The other half-space is fixed. If one wishes to make it move too, one needs to define a second fault of the same geometry but with the nodes given in the reverse order.

The velocity of the half space along the fault v_0 is imposed along the fault surface and its sense is given by the sign of v_0: normal faults have a positive velocity, thrust fault a negative velocity.

Finally the fault is not defined (the velocity is nil) for regions outside of the infinite strip perpendicular to the two points (x_1, y_1) and (x_2, y_2).

Velocity field

The velocity field is calculated from the geometry of the various faults and their respective velocities. The algorithm is rather simple and easy to implement.

A two-dimensional velocity field is first calculated in the (r, s)-plane and later rotated and translated in the proper location in the (x, y, z)-space.

First, one considers each segment individually. In the region defined by the fault segment and its normals at each end of the fault segment, the velocity is set parallel to the fault with amplitude v_0. Two situations have to be considered next, when considering successive fault segments: they either form an acute (closed) or obtuse (open) angle (see Figure 2).

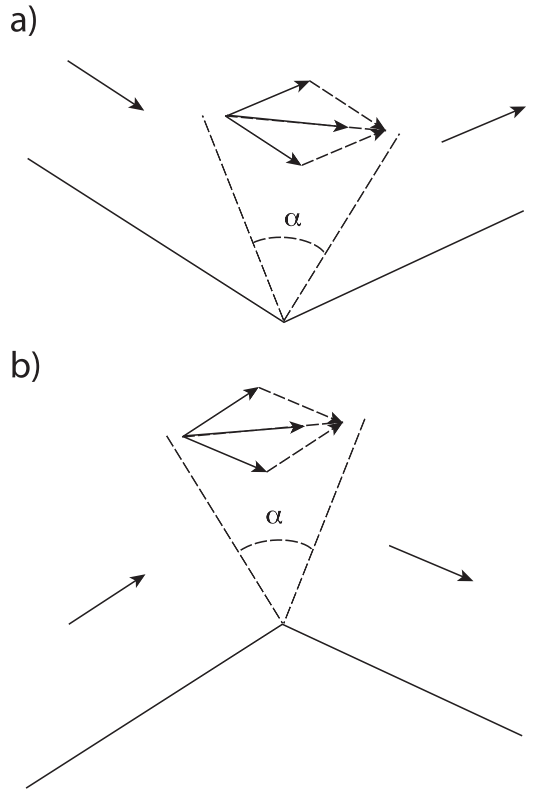

Figure 2: Imposed velocity field in the vicinity of a kink in the fault.

In the first case (acute angle), the direction of the velocity vector in the 'overlapping' region is set to the mean of the directions of the two segments (using the definition of the sum of two vectors to calculate the mean); its amplitude is:

$$v_0' = v_0 \frac{\cos\alpha}{\cos\frac{\alpha}{2}} \tag{1}$$

where α is the angle made by the the two normals to the segments. In the second case, the direction is also the mean of the directions of the segments, but the amplitude is given by:

$$v_0' = v_0 \frac{1}{\cos\frac{\alpha}{2}} \tag{2}$$

The amplitudes are obtained by imposing continuity of the normal component of the velocity across the boundaries defining the various regions, to ensure mass conservation.

More than one fault

When considering the compounded movement of several faults, one must take into account the advection of the position (and potentially the geometry) of one fault with respect to the other. In this new version of PECUBE, the faults are advected by considering which fault is on the 'moving side' of all the others and applying the corresponding displacement (product of the velocity by the time step) to each of the points defining the fault. Note that this works well when the fault being displaced lies within a uniform velocity field and straight segments remain straight; but it breaks down when the fault is displaced by a non uniform velocity field (such as in the vicinity of a kink). This is thus not recommended or it is requested that to maintain some kind of accuracy, the fault being displaced ought to be discretized by a large number of segments.

6 Detrital ages

PECUBE's main purpose is to compute synthetic thermochronological ages from given/imposed tectonic and geomorphic scenarios. To compute ages, one needs to estimate the temperature history of points that end up at the surface of the model at the end of a simulation. This requires a two-pass algorithm. Firstly, one places points at the surface at $t =$now and, using the set tectonic velocities (and the component from the isostatic rebound if necessary), one works out the original position of those points by playing the tectonic scenario backwards. Secondly, during the temperature (forward) calculations, the points are gradually advected towards their final position recording their temperature on the way. At the end of the simulation the ages are calculated from the $T - t$-paths.

To predict detrital ages, one must perform this calculation for points that end up at the surface of the model at set times in the past. In this new version of PECUBE, the number of particles/points being tracked is equal to the number of points defining the surface ($nsurf$) times the number of time steps ($nstep$). During the first pass of the calculations, these points are located at the surface at the set times (when one needs a detrital age distribution) and advected backwards. During the forward model calculations, the points are advected towards the surface; at the set times, they are 'frozen' in their final position and their computed $T - t$-paths are used to calculate the detrital ages.

Note that the predicted detrital ages must be weighted by the predicted/imposed exhumation rate at the time, before they can be compared to observed detrital distributions.

7 Chronometers

In this new version of PECUBE, eight thermochronometers are simulated: U-Th/He in apatite and zircon, Fission Track in apatite and zircon and K-Ar in Kspar, biotite, white mica (muscovite) and hornblende. The diffusion parameters used are those given in Braun et al. (2006); a choice of annealing parameters are available for the fission track simulations (see the corresponding routines *Mad_Trax.f* and *Mad_Trax_Zircon.f*). In this version we are providing the choice of two routines for FT ages in apatites based on van der Beek (1995) or Ketcham et al. (2000). We have also included the possibility of predicting FT length distributions for both apatite and zircon and comparing them to observed distributions, that must be included in the observation file.

8 Output

The ages are given as Text files (*Ages001.txt*, *Ages002.txt*, $\cdots$, *Agesnnn.txt*) corresponding to each time steps requested. The ages, the geometry of the surface and the exhumation rate are also stored in a binary file (*Ages.out*) that can be used to produce VTK files (*Ages001.vtk*, *Ages002.vtk*, $\cdots$, *Agesnnn.vtk*) by using the *Vtk* program. *Vtk* also produces VTK files (*Pecube001.vtk*, *Pecube002.vtk*, $\cdots$, *Pecubennn.vtk*) of the temperature field and the velocities from another binary output file called *Pecube.out*.

9 Inversion or NA mode

In this new version of PECUBE, we have included the basic set of NA (Neighbourhood Algorithm of Sambridge (1999). This algorithm will try to find the optimum values of given parameters of the model that will minimize a misfit function defined by the difference between observed ages and predictions. Note that data (ages) is necessary for this option to work. Two misfit functions can be used (see above).

To enable this feature, you simply need to replace the value of the parameter you wish to invert for (in the *fault_parameters.txt* or *topo_parameters.txt* files) by a range of values separated by a colon (:). NA will search within that range.

You also need to create or modify the file called *na.in* in the *NA* subdirectory to include:

i1	:	Algorithm type (NA or Uniform MC: 1=MC,0=NA)
i2	:	Maximum number of iterations
i3	:	Sample size for first iteration
i4	:	Sample size for all other iterations
i5	:	Number of cells to re-sample
i6,i7	:	Use Quasi random number generator ? (y/n);random seed
i8	:	Type of initial sample (0=random;1=read in a NAD file)
i9	:	Output information level (0=silent,1=summary info,2=1+models)
c1	:	Turn timing mode on ? (y/n)
c2	:	Turn debug mode on ? (y/n)

10 Examples

Example 1: King Canyon

In this example we will use the DEM around Kings Canyon, California, as well as the He in apatite dataset published in House et al. (1998). We will run two distinct scenarios, one in which the present-day topography is assumed to be old (i.e. it formed by linear decay of a larger amplitude topograpyh that was created 90 Myr ago) and one in which it is assumed to be young (created in the last 5 Myr). In both scenarios, we will rapidly exhume rocks (at 1 km/Myr) during an orogenic event that spans the 110 to 90 Ma time span, and then very slowly (at 0.03 km/Myr) during the remaining 90 Myr. Note the use of the '*' symbol to specify the start of the second exhumation period as equal to the end of the previous period (90 Myr).

First go in the input directory and copy the *topo_parameters.txt.KingCanyon* file to *topo_parameters.txt*:

>cp topo_parameters.txt.KingCanyon topo_parameters.txt

as well as the fault_parameters.txt.KingCanyon to fault_parameters.txt:

>cp fault_parameters.txt.KingCanyon fault_parameters.txt

Go back to the main *Pecube* directory and run PECUBE:

>bin/run.sh

This should take a few minutes on a reasonably new computer. Once the run is finished, you can use a VTK viewer (PARAVIEW, for example) to display and inspect the various VTK files created by PECUBE.

In Figures 3 and 4, we show the results of the first and second model runs, respectively, in terms of the He in apatite ages contoured on the topography.

Now you should change the *topo_parameters.txt* file and replace the value of the basal temperature by a range of values (400:600, for example) and the *fault_parameters.txt* file to change the timing of the end of the tectonic episode from 100:70 and the exhumation velocity in the slow exhuming episode from 0.02:0.04. Alternatively you can copy the *fault_parameters.txt.Inversion* to *fault_parameters.txt* and the *topo_parameters.txt.Inversion* to *topo_parameters.txt.*

We advise you to compile PECUBE using MPI-enabled compilers and run this example by using the following command

>mpirun -np 8 bin/Pecube

replace 8 by the sample size per iteration defined in *na.in* to optimize the run.

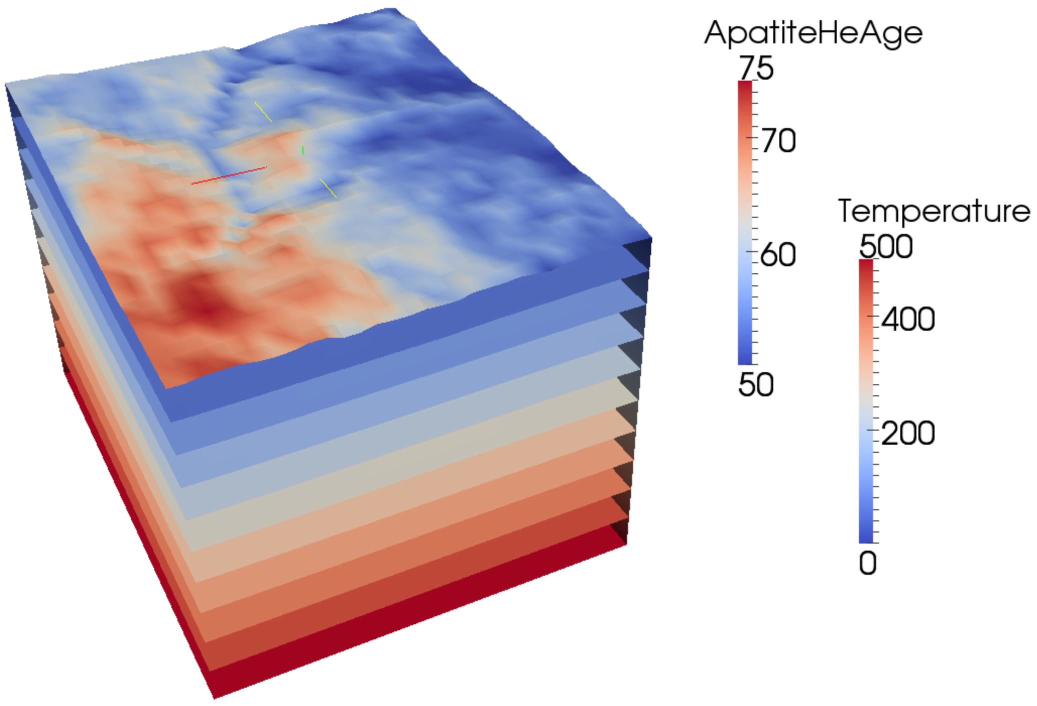

Figure 3: Results of a PECUBE experiment in which the relief of Kings Canyon is assumed to be old.

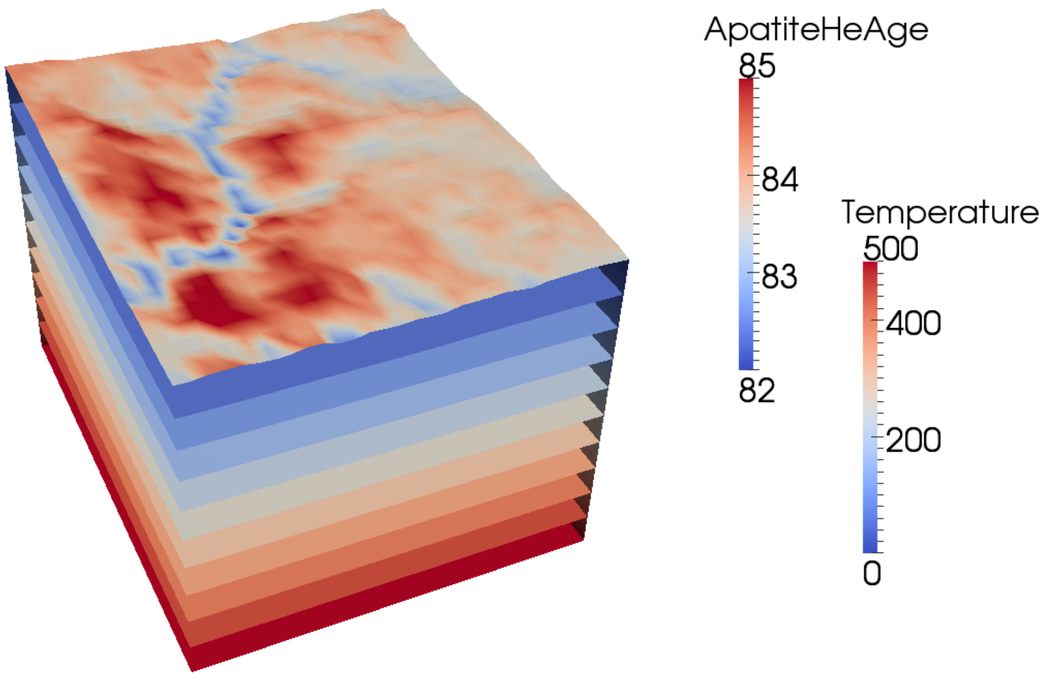

Figure 4: Results of two separate PECUBE experiments in which the relief of Kings Canyon is assumed to be young.

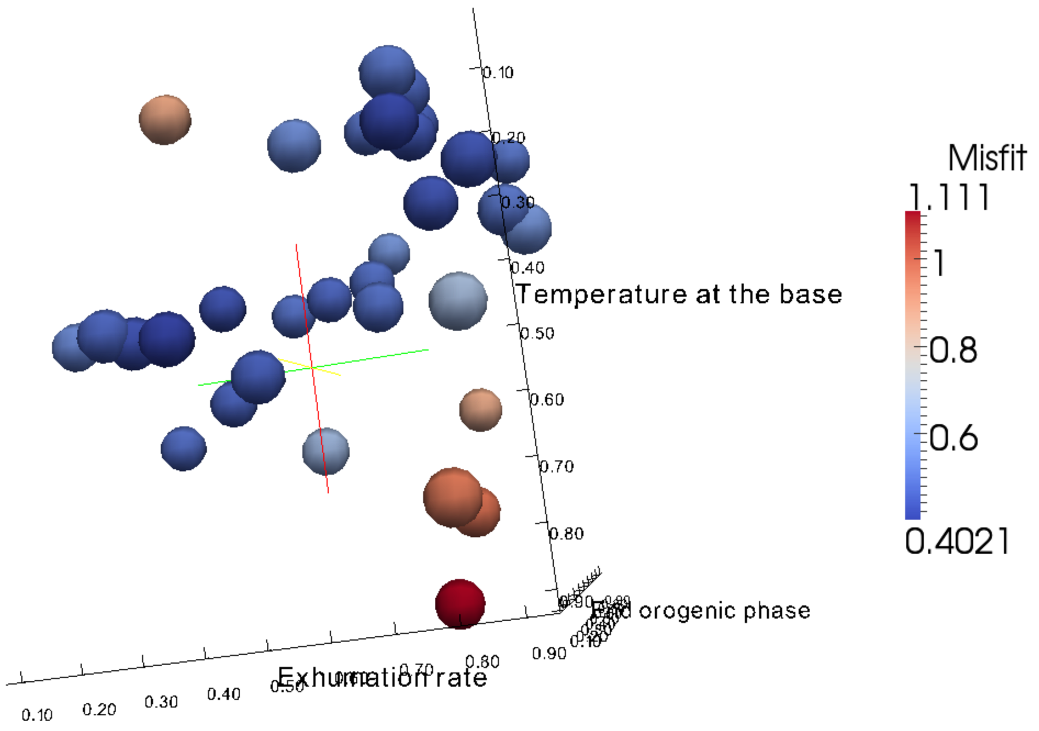

Figure 5: Results of an inversion run using the KingsCanyon dataset to compute the misfit.

Example 2: Whataroa Valley

The second example illustrates the use of a fault to define the tectonic velocity field. The two input files are *topo_parameters.txt.Whataroa* and *fault_parameters.txt.Whataroa*. They use a topographic dataset from around the Whataroa Valley in the central section of the South Island, New Zealand, where the rapid relative motion between the Pacific and Australian plates is accommodated by the Alpine Fault.

Note how the fault surface trace is defined by pairs of two pairs of longitude-latitude coordinates corresponding to the extremities of the line defining the width of the fault segment. Note that the order in which these points are given defines where the fault dips (always to the right of the line), which means that, in this case, the southwestern corner is given first. The fault geometry is then defined by a series of distance-depth pairs in a direction perpendicular to the fault trace. Note first that the depth are given as negative numbers, and second, that the order in which the points defining the fault segments are given determines which of the two blocks (the footwall or the hanging wall) moves. By convention, it is the block to the right of the fault (when moving along the fault in the order given in the input file) that moves. In this case we want the hanging wall to move; consequently, we give the points from the deepest to the shallowest. Here we define the fault as made of two segments defined by three distance-depth points: a first horizontal segment at a depth of 25 km and a second segment dipping at approximately 60° to the east.

The velocity on the fault is finally given as a negative number to indicate that it is a thrust fault. The velocity field corresponding to this geometry is computed internally by PECUBE as explained above. It consists of three regions: to the right (east), along the flat/horizontal segment of the fault, the velocity is horizontal, to the left (west), above the dipping segment of the fault, the velocity is parallel to the fault, i.e. dipping at 60°, and in an intermediate triangular region, the velocity is the mean of the velocities in the other two regions. This velocity field is shown in Figure 6 along a plane perpendicular to the fault.

We imposed that the present day convergence started about 6 Ma and that the topography started growing at about the same time to reach its present-day value at about 4 Ma. Note, however, that, in the current version of PECUBE, no strike-slip motion can be imposed on faults. This is because the strike-slip component

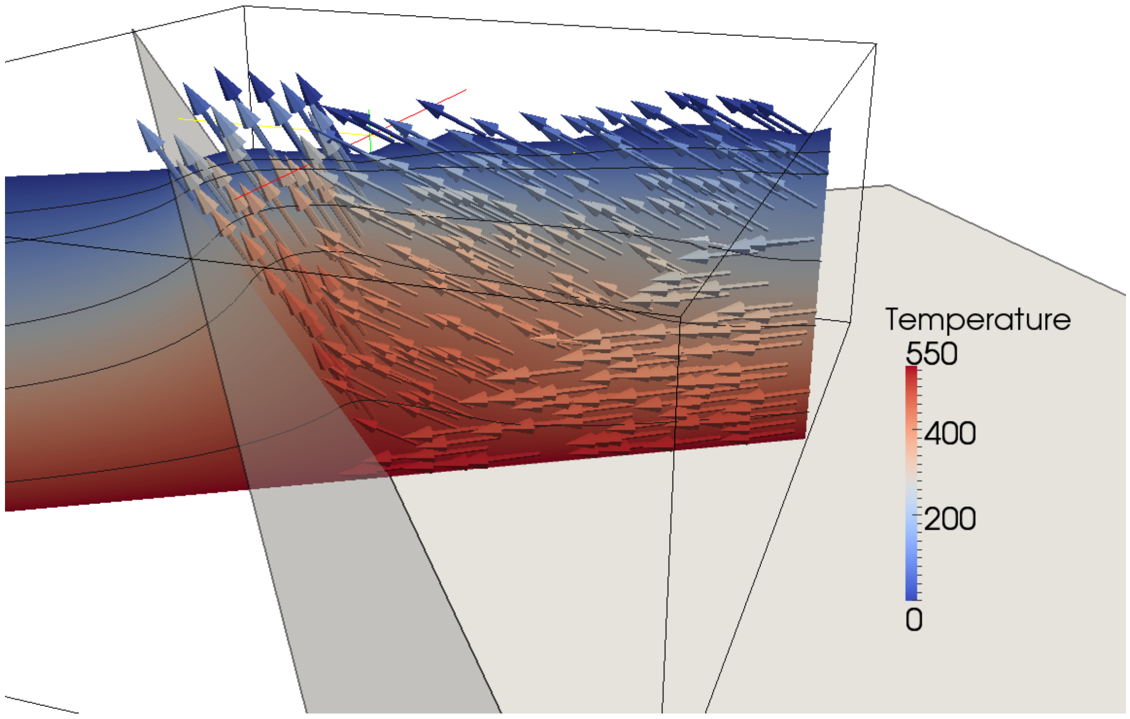

Figure 6: Geometry of the fault used in the second example and corresponding velocity field displayed along a plane perpendicular to the fault trace.

of deformation does not influence much the distribution of cooling ages along a linear structure.

Finally, it is worth noting that we imposed a timing of 100 Ma for the previous tectonic event. We have also limited the number of age/systems being computed to fission track in apatite only (see the flags at the bottom of the input *topo_parameters.txt* file).

Results of the computations are shown in Figure 7. Note how the isotherms are compressed towards the surface in the vicinity of the fault (on the upthrow side) as a result of heat advection by the imposed velocity field (shown in Figure 6). Consequently the FT ages are very young near the fault (i.e. close to 0 Ma, as observed by Tippett and Kamp (1993)) and along most of the western side of the orogen. Along the eastern flank, ages get older and reach pre-Alpine values of approximately 100 Ma, as imposed in the input file.

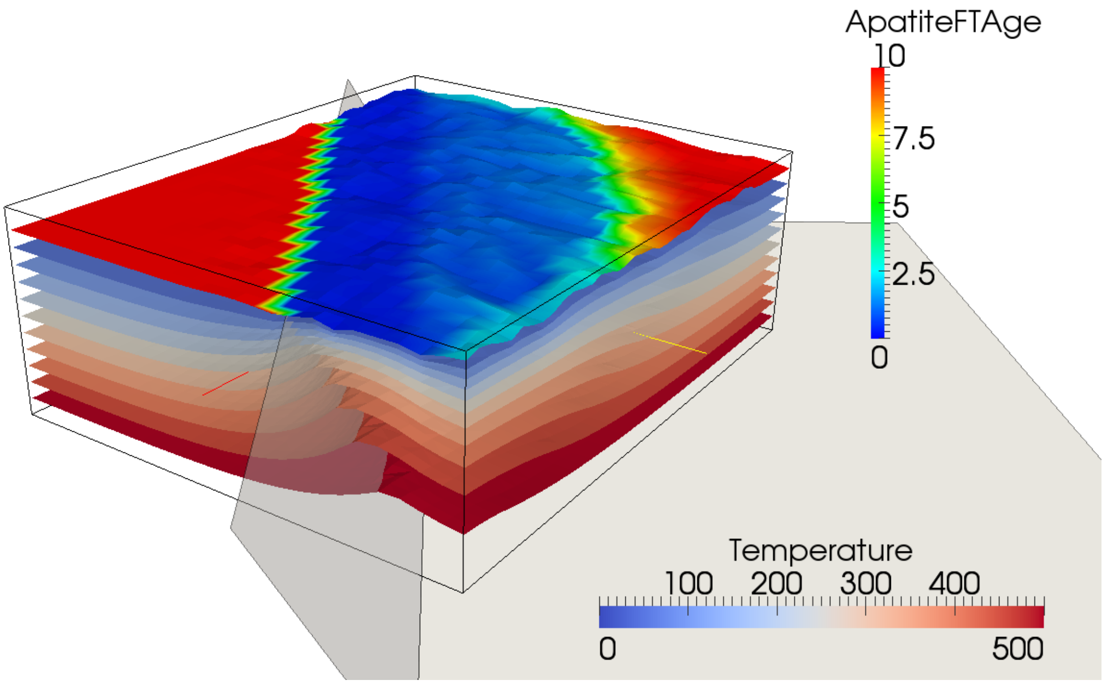

Figure 7: Computed Apatite fission track ages, temperature field and imposed fault geometry.

We suggest to the user to modify the input file to include frictional heating along the fault by setting the coefficient of friction at 0.1, for example, to represent a relatively weak fault. The new solution is shown in Figure 8. The heat produced by friction along the fault causes a further upwarping of the isotherms on either sides of the fault.

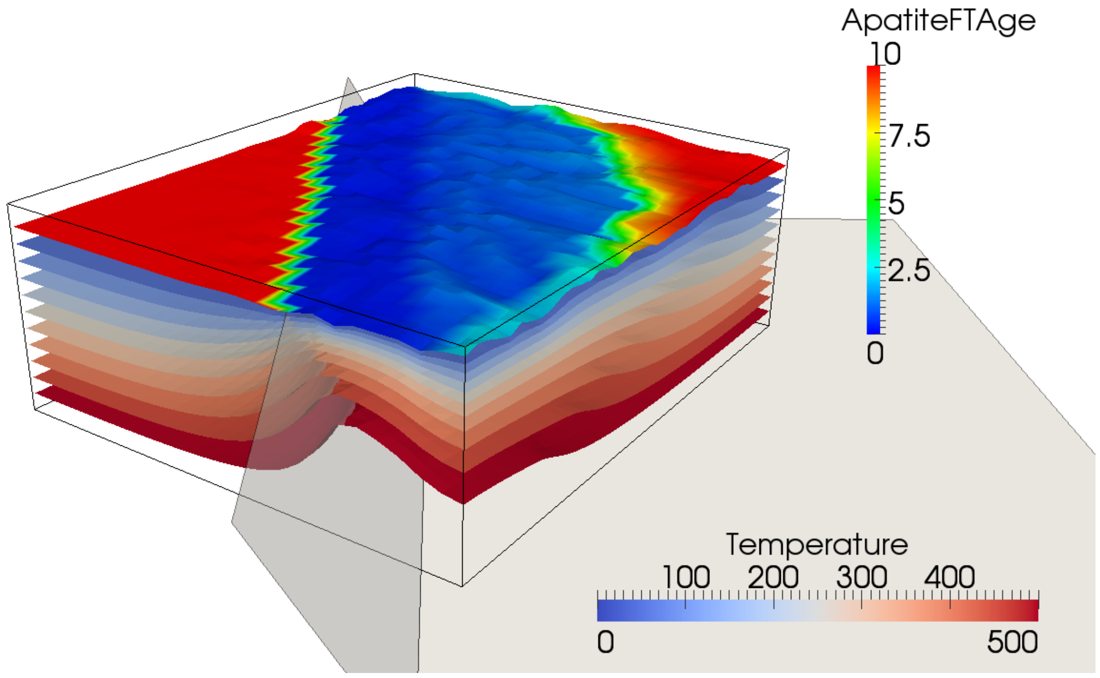

Figure 8: Computed Apatite fission track ages, temperature field and imposed fault geometry. Same as Figure 7 but including frictional heat.

Example 3: Synthetic landscapes

In this third example, we use synthetic landscapes produced by the CASCADE software as input topographic files for PECUBE. This requires a series of topographic, uplift and surface temperature files generated from the output of a CASCADE run. To create these files, we have written a simple FORTRAN code named *cascade2Pecube.f90* that can be found in the source subdirectory. If using another surface processes model (SPM) to produce the synthetic landscapes, the user will need to modify this Fortran code to accept the output created by the other SPM.

Output files (connectivity, geometry, uplift rate, topography) from a cascade run need to be stored in the *cascade_output* subdirectory before running *bin/cascade2Pecube* from the main Pecube directory. You will be prompted for a directory file name that will need to create in the data directory before running cascade2Pecube. This will create as many topo, temp and uplift files as there are time steps in the CASCADE output files. You will then need to modify the *topo_parameters.txt* file to indicate the number of time steps and their associated times. Parameters such as the skip factor or the spacing in degrees will not be used. Set the bottom left corner of the mesh to longitude-latitude (0,0). Use the *topo_parameters.txt.Cascade* as a template.

The *fault_parameters.txt* files should not be used to define an additional uplift/velocity field. Make sure that the number of faults is set to zero, right at the top of the file.

We will use this example to compute histograms of age distribution at the various intermediate time steps 0, 1, 2, 3, 4 and 5 Ma (corresponding to *Ages010.txt*, *Ages008.txt*, *Ages006.txt*, *Ages004.txt*, *Ages002.txt* and *Ages000.txt*, respectively) for the He in apatite thermochronometer. These are shown in Figure 9. They show a relatively constant lag time (Bernet et al., 2004) for the past 3-4 Myr.

References

Bernet, M., Brandon, M., Garver, J., and Molitor, B. (2004). Fundamentals of detrital zircon fission-track analysis for provenance and exhumation studies with examples from the European Alps. In Bernet, M. and Spiegel, C., editors, *Detrital thermochronology - Provenance analysis, exhumation and landscape evolution of mountain belts*, volume 378, pages 25–36, Boulder, Colorado. Geological Society of America Special Paper.

Braun, J. (2003). Pecube: A new finite element code to solve the heat transport equation in three dimensions in the Earth's crust including the effects of a time-varying, finite amplitude surface topography. *Comput. Geosci.*, 29:787–794.

Braun, J., van der Beek, P., and Batt, G. (2006). *Quantitative Thermochronology*. Cambridge University Press.

House, M., Wernicke, B., and Farley, K. (1998). Dating topography of the Sierra Nevada, California, using apatite (U-Th)/He ages. *Nature*, 396:66–69.

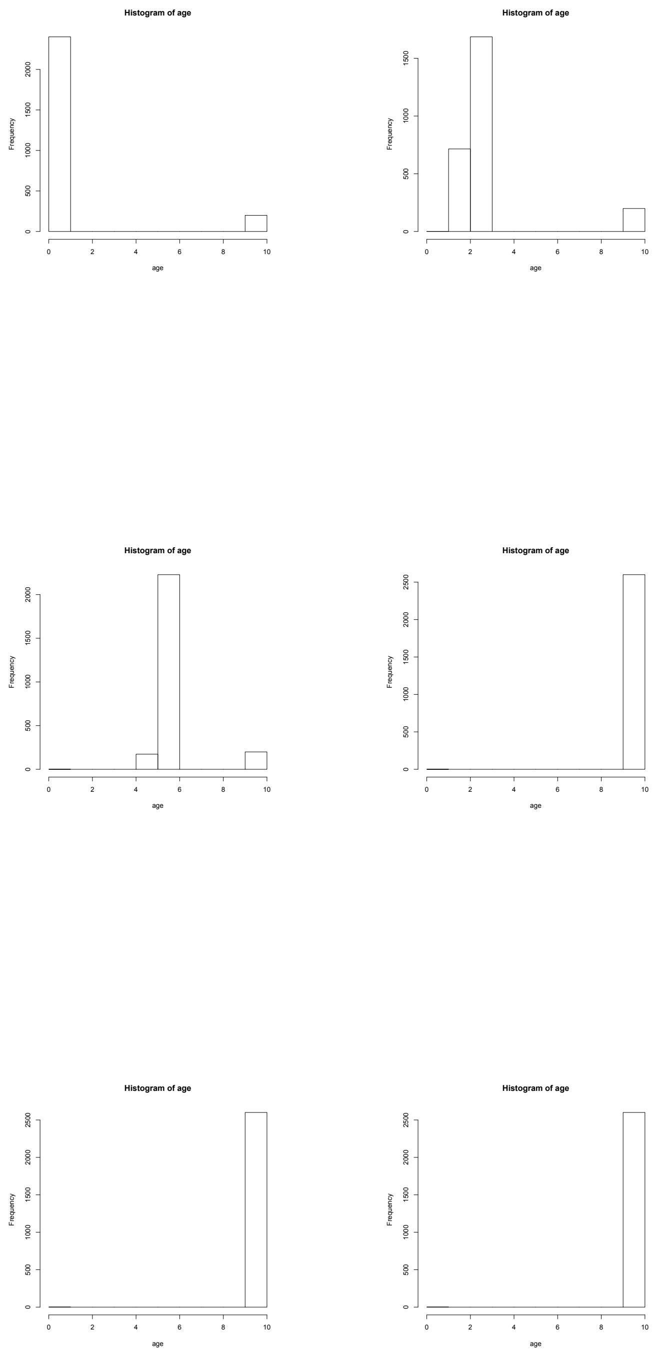

Figure 9: Computed He apatite age distributitions obtained with PECUBE at t=0, 1, 2, 3, 4 and 5 Ma.

Ketcham, R., Donelick, R., and Donelick, M. (2000). AFTSolve: A program for multi-kinetic modeling of apatite fission-track data. *Geological Materials Research*, 2(1).

Sambridge, M. (1999). Geophysical Inversion with a Neighbourhood Algorithm -I. Searching a parameter space. *Geophys. J. Int.*, 138:479–494.

Tippett, J. and Kamp, P. (1993). Fission track analysis of the late Cenozoic vertical kinematics of continental Pacific Crust, South Island, New Zealand. *J. Geophys. Res.*, 98:16,119–16,148.

van der Beek, P. (1995). *Tectonic evolution of continental rifts: Inferences from numerical modelling and fission track thermochronology.* PhD thesis, Vrije Universiteit, Amsterdam, the Netherlands.