Pecube - version 3.0

Jean Braun

July 19, 2010

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 Ketcham's fission track algorithm, 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.

Contents

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.

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 and C 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 large datasets (topographic and age datasets) are stored in the *data* subdirectory.

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

All VTK files are stored in the *VTK* subdirectory. VTK files are used to display the results of the computation. VTK stands for *Visual TooolKit*. To visualize VTK files, you will need to install a VTK file viewer such as *Paraview* or *Mayavi*. Both of these work on any platform (Linux, MacOS, Windows) and are free.

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, go to the *source* directory:
> *cd source*
and make all utilities and PECUBE:
> *make all*

To run PECUBE, change the input files *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 character by using a fancy text editor 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, using:
> *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* programme 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.

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 yo have very large topographic datasets.

4 Input files

Two input files are now used in this new version of PECUBE. They are used by PECUBE to create an input file that has (approximately) the same format as the input file for former versions of PECUBE. This ensures some kind of backward

compatibility with the previous versions.

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	character*5
topofnme	Name of the input topographic file (DEM) containing a list of nx by ny (or -nx) height values in meters. 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* to *toponstep*. 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 exhumation rate and surface temperature computed by the LEM. They should be called *uplift0* to *upliftnstep* and *temp0* to *tempnstep*. 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.	character*(*)
nx ny	dimension of the input DEM; if nx and ny 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.	integers
dlon dlat	DEM longitude and latitude spacing in degrees (not used when $nx < 0$)	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 $nx < 0$)	integers
lon0 lat0	longitude and latitude of the bottom left corner of the DEM	float

nstep	the number of times steps in the geomorphic scenario	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 $\tau = 0$	float
time, ampli, offset, iout	per time step (i.e. you need to include nstep lines like this one) the time at which the step starts (in Ma in the past, i.e. geological time), the amplification factor by which the topo read in topofnme will be scaled, the offset factor by which it is moved up or down, and a flag to determine whether an output (temperature, velocities and detrital ages) is requested at this time (0 for no output, 1 for output) ($ampli$ and $offset$ are not used if $nx < 0$)	3 floats, 1 integer
iso, rhoc, rhom, E, nu, L, nxiso,nyiso	a flag to allow for flexural isostaty calculations (0 no isostasy, 1 isostasy), the crustal and mantle densities (in kg/m3), young modulus (in Pa), Poisson's ratio, elastic thickness (in km), and the resolutions in both directions of the FFT mesh used to calculate the isostatic flexural response (must be powers of 2). Note that iso must be set to 0 whenusing irregular grid ($nx < 0$)	1 integer, 5 floats, 1 integer
zl, nz, kappa, tbase, ttop, lapse, prod	crustal (model) thickness (in km), resolution in the z-direction, heat diffusivity (in km2/Myr), temperature at the base of the model (in °C), temperature at the top of the model (in °C), lapse rate (in °C/km), heat production (in °C/Myr)	1 float, 1 integer, 4 floats

agefnme	Name of the file containing observed ages (Nil if no data). On the first line the file should contain the number of observations nobs, then nobs lines containing: (1) the longitude (2) the latitude (3) elevation (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 Negative ages mean no data.	character*(*)

Tectonic input file

The second input file is called *fault_parameters.txt*. To understand the meaning of the various parameters, the user is referred to the following section (5). The tectonic input file 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).	integer
xlon1 xlat1 xlon2 xlat2	Two lon-lat pairs defining the trace of the fault at the surface. The fault is located to the right of that line.	4 floats
n	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	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 rigth, upper left, upper right)	2 or 4 floats/line
nstep	number of time intervals defining the movement on the fault	integer
tstart tend velo	nstep lines containing the starting and end times (in geological time) and a velocity (positive for normal faulting, negative for thrusting)	3 floats

5 The fault

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.

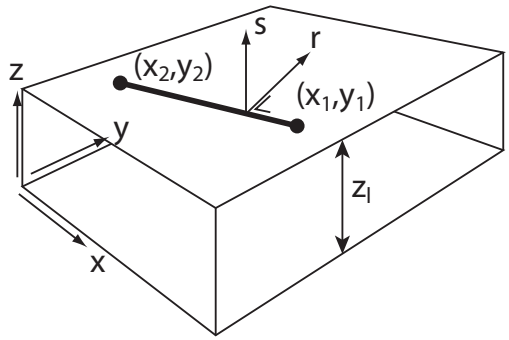

Figure 1: The local fault coordinate system and the global PECUBE system.

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).

6 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).

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.

7 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

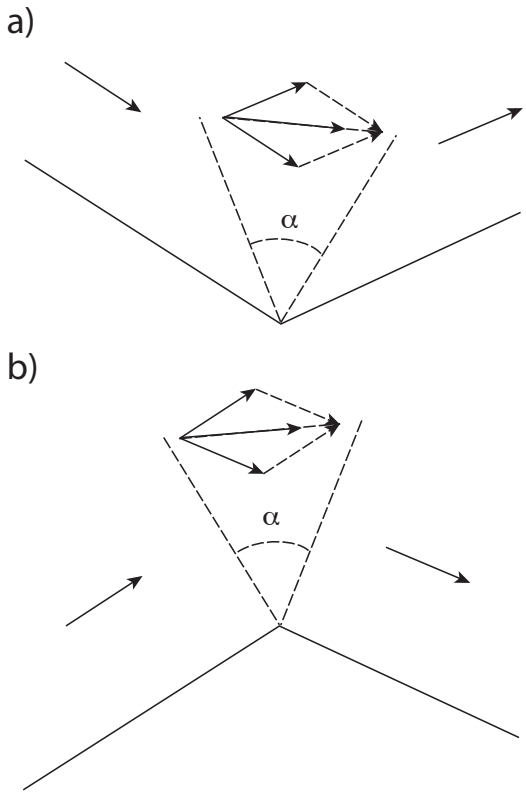

Figure 2: Imposed velocity field in the vicinity of a kink in the fault.

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.

8 Testing scenarios

The complexity of 3D fault movement is such that we have found it necessary to design a simple test program to visualize the kinematic and geomorphic scenarios that can now be defined for PECUBE. Once the input files describing these scenarios (see section below on input files) have been produced, one can run the program *Test* that generates a number of VTK files that can be viewed with a VTK visualizer such as *MayaVi*. They show the location of each fault (faultA001.vtk, faultA002.vtk, $\cdots$, faultAnnn.vtk for the first fault, successive faults being named B, C, $\cdots$) at the n times steps set in the scenario. Another set of VTK files (velo001.vtk, velo002.vtk, $\cdots$, velonnn.vtk) give the geometry of the velocity field at the various time steps.

9 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.

10 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*).

11 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.

12 Credit

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

References

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.