Program ftee version 1.13

last update October 2023

1. ABOUT

Calculates alpha-loss correction for U-Th-Sm/He dating, using a Monte-Carlo model for several symmetrical geometries for apatite, zircon, titanite/sphene, monazite and rutile. The geometries are: ellipsoid (includes sphere); cylinder and flattened cylinder (pinacoidal only); and tetragonal prism (pinacoidal or with optional pyramidal terminations; includes cube).

IMPORTANT. This code has been checked against analytical solutions for spherical, cylindrical, ellipsoidal, and prismatic tetragonal geometries (solutions from Ketcham et al., 2011). Earlier versions were tested for tetragons with tips against concrete examples from Hourigan et al. I have not exhaustively tested routines that calculate parent abundances, Ft-equivalent spheres, especially for the case of combining different decay schemes and coping with multi-grain aliquots. What checking I did do showed that results were good and experience with the code shows that its results make sense, but **caveat emptor.**

---

3. AUTHOR: Dr. Peter K. Zeitler
Department of Earth and Environmental Sciences
1 W. Packer Avenue
Lehigh University
Bethlehem, Pennsylvania 18015-3188 U.S.A.
phone: (610) 758-3671 fax: (610) 758-3677
peter.zeitler@lehigh.edu

---

2. COMPILATION and USAGE

Compilation

ftee is just a simple plain-vanilla C program. However, it is programmed to take advantage of multiple cores using openMP. If you run into issues trying to compile with the `-fopenmp` flag, the code should compile as a

regular program without it though you may first have to excise a few statements related to multicore timing.

Compile **ftee** as follows. This example assumes you are using the gcc compiler; the code is simple C so other compilers should work fine, though it's best to use one with OpenMP support.

```
gcc ftee113.c -o ftee113 -O3 -fopenmp
```

To run the code, enter:

```
[./]ftee113  filename
```

where *filename* is the name of an existing tab-delimited input file of the correct format (see below), located in your working directory. Note that if you place the executable in a directory that's in your PATH, you can execute the code using just the program name.

Input Conventions

Input takes the form of tab-delimited text file containing sample information. A template file is available. See also additional details below, in Section 4.

1. Dimensions are in microns

2. For each grain, **ftee** requires Length, Width1, Width2. You can optionally append the axis-parallel length of one or two tips (Tip1 and Tip2). In the case of tetragons with one or two tips, Length is the TOTAL grain length, tip-to-tip or end-to-end. The widths are the grain's widths across the two perpendicular minor axes. Thus the length of the prism segment is Length - Tip1 -Tip2.

3. The Ft-equivalent sphere value reported by **ftee** is a radius.

3. U, Th, and Sm amounts must be in ng in order for the ppm and eU calculations to work. **ftee** will still give you good alpha-loss corrections if your parents are not entered as ng, provided the ratios between the parents are correct.

Outputs

A tab-delimited text file. The file should be viewable in any text or spreadsheet editor. Also, a summary of the sample info is reported to the console.

In the output file, the first block reports results for the sample as a whole: Ft for the sample, ppm values for parents, Ft-equivalent radius, and eU, followed by input data.

The second block reports results for individual-grain data as best it can – for bulk multi-grain analyses, concentration and eU results can't be determined for component grains.

Notes:

- The eU calculation follows Cooperdock et al. (2019) in being U + 0.238Th + 0.0012Sm
- The reported grain volumes and mass are calculated from the input grain dimensions, chosen geometry, and in the case of mass, the assigned mineral density (reported in the table).

3. APPROACH

The code involves brute-force monte-carlo simulation: random x, y, and z coordinates are determined from within the problem bounds, and this point is checked to see if it is within the simulated grain or not. If it is, between 1 and 100 random alpha-ejections are modeled for each of four nuclides, originating from the random point. The coordinates of each of these new points is checked to see if it is on or within the grain; if yes, it is counted. The code is quite fast, so sub-micron sampling densities are easy to obtain. The bounds of the problem space extend from tip1 to tip2 of the grain and across the two grain widths.

The current code uses mean ejection lengths for each decay chain. It does not, as discussed in Ketcham et al. (2011), account for the significant range in stopping distances *within* the various chains. This is

not a significant issue given other uncertainties in grain measurement and in U-He dating in general.

Each geometry has a custom routine based on tricks related to peculiarities of the geometry. The ellipsoid (and sphere) use the formula for an ellipsoid in x-y-z space. The (optionally flattened) cylinder uses the formula for an ellipse in the x-y plane because in the case of the cylinder there is no dependence on z. The tetragonal prism (and cube) uses simply the grain dimensions for the prism faces but for the pyramids (if there are tips) we use a projection in which the xy bounds of the rectangle that wraps the pyramidal faces at each z-level is determined from simple geometry and linear scaling of proportions (easier shown than explained!). Other geometries could be built in a similar fashion (e.g., hexagonal prism with or without pyramidal terminations). More complex geometries are, in principle, simple to add, although in practice they might be nasty and tedious to define in code.

At the moment, there is no provision for U and Th zonation. In principle, one could imagine entering a profile and then extrapolating to 3D using symmetry, but I am not sure that such artificial symmetry will produce better results for most samples and in any event it's still rare to have quantitative information about zoning.

The user has the option of blending the total number of monte-carlo positions in the crystal with the number of random-direction alpha ejections that are modeled from each position. Experience shows that increasing the number of sampled positions is far more important than modeling lots of ejections, and in fact, if the number of positions drops too much (say below 100,000) accuracy decreases because there are positions in the grain that are not getting explored. Generally, I suggest running a total of 1e7 to 1e8 decays using a minimum of 1e6 monte-carlo positions. It's easy to experiment and see how the tradeoff influences the results.

It depends a bit on geometry and size, but 1e7 total decays produces results that are usually precise to about 0.05% or better.

This code uses the high-quality Mersenne Twister random-number generator from Nishimura and Matsumoto. For a monte-carlo routine

like this that consumes enormous numbers of calls to random(), it is important to have random numbers that are as random as possible and have an extremely large repeat sequence.

4. INPUT FILE FORMAT

1 header line containing:

```
#_analyses #_monte-carlo_iterations #_of_ejections_per_site
```

followed by

1 or more lines each containing the following (see explanation for permitted values):

```
sample_name    number_analyses_per_sample    U    Th    Sm
mineral    calibration    geometry    Length    Width1    Width2
[Tip1]    [Tip2]
```

Explanation of input parameters:

sample_name – no spaces;
number_analyses_per_sample – number of separate grains making up the sample (related grains must be contiguous in the file!)
U – sample U content in nanograms
Th – sample Th content in nanograms
Sm – sample Sm content in nanograms (enter 0.0 as placeholder if not measured)
mineral – code for mineral type, options are:
apatite zircon sphene-or-titanite monazite rutile
calibration – stopping-distance calibration: oldfarley farley or ketcham (the newer values used in Hefty from Ketcham et al. (2011); See NOTE #1 below)
geometry – grain geometry, options are:
ellipsoid cylinder tetragonal (See NOTE #2, below)
Length – tip-to-tip grain length in microns -- should be the largest dimension
Width1 – second largest dimension, microns
Width2 – third largest dimension, microns (enter same value as Width1 if crystal is symmetrical or only one value is available)

Tip1 – tip-to-prism axis-parallel distance (set this to 0.0 if not measured)
Tip2 – second tip-to-prism axis-parallel distance (set this to 0.0 if not measured, set this to Tip1 if crystal is symmetric)

NOTE #1 – if you are modeling apatite the oldfarley calibration is available), which uses the original values from the first alpha-stopping paper. The Farley calibration uses values reported in Farley and Ehlers for a range of minerals. The Ketcham values are those used in Hefty as reported in Ketcham's *On-Track* note and also Ketcham et al. (2011). If new stopping values are published or you want to add a new calibration, you will need to go into the appropriate block of code at the start of this program and hard-code the changes at that spot before recompiling.

NOTE #2 - the ellipsoid geometry can handle the sphere. The tetragonal geometry can handle a cube and can handle both pinacoidal and pyramidal terminations. The cylinder currently handles pinacoidal geometry only. Obvious extensions would be hexagonal prism, and pyramidal terminations for the cylinder and hexagon. Good luck dealing with sphene.

NOTE #3 – the maximum number of analyses is 500. It's hard to imagine why, from a single data file, you would want to run anything close to that many analyses. Note that an "analysis" in this context means a grain: the number of samples will be equal to or smaller than then number of analyses. In modern work most of your inputs will likely be single-grain samples.

When you obtained this code you should have also received one or more examples of working input files. Note that you must be sure to create plain-text input files with UNIX line breaks and no rich-text formatting. Various GUI editors on Macs and PCs do not always handle this well.

Output files are in tab-delimited format and have the suffix ".txt" appended to them – it should be trivial and fast to open these in Excel using "open with…".