Moon's Crust Formation

Documentation for the numerical modeling

Description

This document describes how the numerical modelling of the lunar crust formation problem was carried out. It is a code made entirely with `Python`. The following sections present the numerical schemes used, the libraries needed for their use and the use of this code.

Numerical modelling

The numerical modelling is based on the solution of the 1D spherical advection-diffusion equation (see eq. 1).

$$\frac{\partial T}{\partial t} = \frac{\kappa}{r^2}\frac{\partial^2 T}{\partial t^2} - u\frac{\partial T}{\partial r} + hV \tag{1}$$

For this, the code is based on the use of finite volumes and a fully implicit scheme.

Diffusion

By integrating the diffusion part on a control volume :

$$\int_{CV} \frac{\partial T}{\partial t} dv = \kappa \int_{CV} \nabla(\nabla T) dv \; + \; \int_{CV} h dv \tag{2}$$

Divergence theorem :

$$\int_{CV} \frac{\partial T}{\partial t} dv = \kappa \int_{surface} (\nabla T)\mathbf{n} dS \; + \; \int_{CV} h dv \tag{3}$$

Yet $\nabla T = -q$:

$$\int_{CV} \frac{\partial T}{\partial t} dv = \kappa \int_{surface} (q_e - q_s)\mathbf{n} dS \; + \; \int_{CV} h dv \tag{4}$$

• implicit discretization :

$$\frac{\partial T}{\partial t} dv = \frac{T_i^{n+1} - T_i^n}{\Delta t} \times 4\pi r_i^2 \Delta r \tag{5}$$

$$q_e = -\kappa \frac{T_i^{n+1} - T_{i-1}^{n+1}}{\Delta r} \times 4\pi r_{i-1/2}^2 \tag{6}$$

$$q_s = -\kappa \frac{T_{i+1}^{n+1} - T_i^{n+1}}{\Delta r} \times 4\pi r_{i+1/2}^2 \tag{7}$$

$$h dv = h \times 4\pi r_i^2 \Delta r \tag{8}$$

So :

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} \times r_i^2 \Delta r = -\kappa \frac{T_i^{n+1} - T_{i-1}^{n+1}}{\Delta r} \times r_{i-1/2}^2 + \kappa \frac{T_{i+1}^{n+1} - T_i^{n+1}}{\Delta r} \times r_{i+1/2}^2 + h \times r_i^2 \Delta r \tag{9}$$

This scheme involves solving an inverse problem such as :

$$(\mathbf{I} - \Delta t \mathbf{D}) \mathbf{T^{n+1}} = \Delta t \mathbf{H} + \mathbf{T^n} \tag{10}$$

with $\mathbf{I}$ the identity matrix

The development of the matrix $\mathbf{A}$ gives, with a temperature boundary condition :

$$\mathbf{D} = \begin{pmatrix} CL_{01} & CL_{02} & & & \\ A_i & B_i & C_i & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & A_i & B_i & C_i \\ & & & CL_{I1} & CL_{I2} \end{pmatrix}$$

With $s_i = \dfrac{\kappa \Delta t}{r_i^2 \Delta r^2}$:

$$\begin{cases} A_i &= & s_i r_{i+1/2}^2 \\ B_i &= & -s_i r_{i+1/2}^2 - s_i r_{i1/2}^2 \\ C_i &= & s_i r_{i-1/2}^2 \end{cases}$$

and

$$\begin{cases} CL_{01} &= -2s_0 r_{-1/2}^2 - s_0 r_{1/2}^2 \\ CL_{02} &= s_0 r_{1/2}^2 \end{cases}, \quad \begin{cases} CL_{I1} &= s_I r_{I-1/2}^2 \\ CL_{I2} &= -2s_I r_{I+1/2}^2 - s_I r_{I-1/2}^2 \end{cases}$$

The boundary condition implies a rest such as :

$$\begin{pmatrix} R_0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ R_I \end{pmatrix}, \text{ with } \begin{cases} R_0 &= 2s_0 T_E r_{-1/2}^2 \\ R_I &= 2s_I T_S r_{I+1/2}^2 \end{cases}$$

The inverse problem becomes :

$$(\mathbf{I} - \Delta t \mathbf{D})\mathbf{T^{n+1}} = \Delta t \mathbf{H} + \mathbf{T^n} + \mathbf{R} \tag{11}$$

Advection

For the advective part, we use the Total Variation Diminishing (TVD) scheme

$$u\frac{\partial \tilde{T}}{\partial \tilde{r}} \sim \frac{F^+ - F^-}{\Delta r} \tag{12}$$

With :

$$F^+ = \frac{u}{2}[T_{i+2}^n(\frac{-\gamma(\theta_{i+1})}{2}) + T_{i+1}^n(1 + \frac{\gamma(\theta_{i+1})}{2} + \frac{\gamma(\theta_i)}{2}) + T_i^n(1 - \frac{\gamma(\theta_i)}{2})] - \frac{\mid u \mid}{2}[T_{i+2}^n(\frac{-\gamma(\theta_{i+1})}{2}) - T_{i+1}^n(1 + \frac{\gamma(\theta_{i+1})}{2} + \frac{-\gamma(\theta_i)}{2}) + T_i^n(\frac{\gamma(\theta_i)}{2} - 1)]$$

$$F^- = \frac{u}{2}[T_{i+1}^n(\frac{-\gamma(\theta_i)}{2}) + T_i^n(1 + \frac{\gamma(\theta_i)}{2} + \frac{\gamma(\theta_{i-1})}{2}) + T_{i-1}^n(1 - \frac{\gamma(\theta_{i-1})}{2})] - \frac{\mid u \mid}{2}[T_{i+1}^n(\frac{-\gamma(\theta_i)}{2}) - T_i^n(1 + \frac{\gamma(\theta_i)}{2} - \frac{\gamma(\theta_{i-1})}{2}) + T_{i-1}^n(\frac{\gamma(\theta_i)}{2} - 1)]$$

And :

$$\theta_i = \frac{T_i^n - T_{i-1}^n}{T_{i+1}^n - T_i^n}$$

$\gamma(\theta_i)$ corresponds to a flux limiter. Here, the choice is made to choose a flux limiter of the type Lax-Xendroff so $\gamma(\theta_i) = \gamma(\theta_{i+1}) = \gamma(\theta_{i-1}) = 1$

The distribution of radiogenic elements is governed by the advection equation. Thus the calculation of the $\mathbf{H}$ matrix is performed in the same way as for the temperature.

Putting the advective part as a matrix $\mathbf{A}$, the complete scheme is :

$$(\mathbf{I} - \Delta t\mathbf{D} + \Delta t\mathbf{A})\mathbf{T^{n+1}} = \mathbf{R} + \Delta t\mathbf{H} + \mathbf{T^n} \tag{13}$$

Requirement

List of all `Python` librairies requiried

- `numpy`
- `mathplotlib`
- `scipy.linalg`

Utilisation

The code is separated into two parts. The first part, `CrustFormation.py`, is a library containing all the necessary functions and the second part, `CrustFormation_fig.py`, is an executable file that returns different figures : evolution of the crust radius and solid cumulates, temperature and distribution of radiogenic elements in the crust and different fluxes.

The executable part allows to modify the different parameters such as : `k` the rmal conductivity, `rho` density, `Cp` heat capacity, `Lat` latent heat, `C_0` initial composition, `C_E` eutectic composition, `L0` initial condition for the size of the crust, `dR_bot0` initial condition for the speed of growth of the crust, `h0` initial radiogenic heat, `lambda` desintegration constant, `D_AN` partition coefficient of anorthite.

The executable file requires as input the number of iterations in space and a dictionary containing all the variable parameters