1. Basic theory

The *Mandyoc* code simulates thermochemical convection of the Earth's mantle. The following sections explain which equations are being solved by the code and the numerical approach that was used.

1.1. Basic equations

To simulate mantle thermochemical convection, we adopted the formulation for non-Newtonian fluids together with the Boussinesq approximation to solve the following equations of conservation of mass (Eq. 1.1), momentum (Eq. 1.2) and energy (Eq. 1.3) [1].

$$u_{i,i} = 0 \tag{1.1}$$

$$\sigma_{ij,j} + g_i \rho_0 (1 - \alpha(T - T_0)) = 0 \tag{1.2}$$

$$\frac{\partial T}{\partial t} + u_i T_{,i} = \kappa T_{,ii} + \frac{H}{c_p} - \frac{\alpha T g u_e}{c_p} \tag{1.3}$$

where u is the velocity in the i direction, g is the gravity acceleration, ρ_0 is the reference rock density at temperature T_0, α is the coefficient of volumetric expansion, T is the temperature, κ is the thermal diffusivity, H is the rate of radiogenic heat per unit of mass, c_p is the specific heat, δ_{ij} is the Kronecker delta, and σ_{ij} is the stress tensor:

$$\sigma_{ij} = -P\delta_{ij} + \eta(u_{i,j} + u_{j,i}) \tag{1.4}$$

where P is the dynamic pressure and η is the effective viscosity of the rock.

Note

For the adopted Einstein notation, repeated indexes in the same term represent summation and indexes after comma are partial derivative to a spatial coordinate.

1.2. Numerical approach

The equations of conservation of mass, momentum and energy are solved using the finite element method [1]. *Mandyoc* uses hexahedral elements for three dimensional grids and quadrilateral elements for two dimensional grids [2]. The Mass and momentum equations subsection presents the numerical methods used to solve the mass and momentum equations and the Energy equation subsection shows how an implicit formulation was used to solve the energy equation [3].

To simulate any scenario, the user **must** provide the parameter file `param.txt` and, if necessary, the ASCII files with the initial temperature field, velocity field and/or the initial interfaces of the model. To see how these files can be created/modified, see the section Parameter file and Input files. The flowchart in Fig. 1.1 summarizes the steps *Mandyoc* takes to solve the conservation equations and perform a simulation.

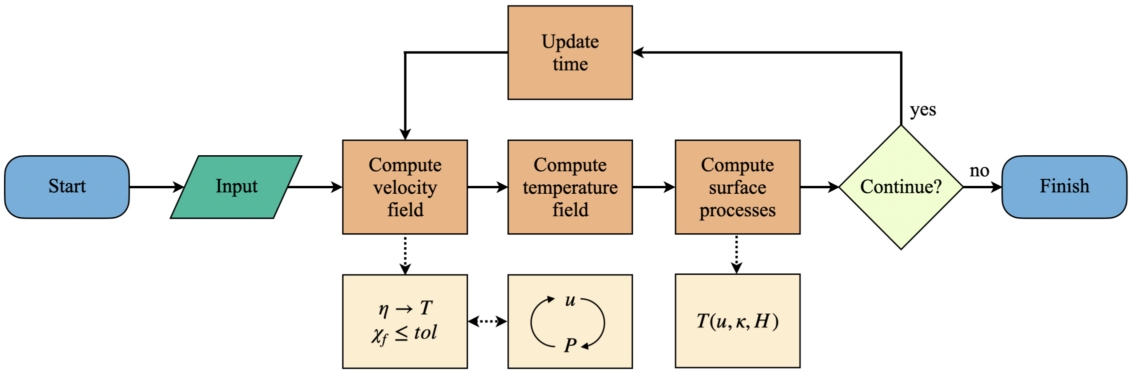

Fig. 1.1 *Flowchart showing the steps Mandyoc takes to solve the equations of conservation of mass, momentum and energy.*

Fig. 1.1 shows that once the code starts running and the input files are read (`param.txt` and the ASCII input files), *Mandyoc* uses the effective viscosity field η (Eq. 2.28) to calculate the velocity field u and checks if the convergence condition satisfies the tolerance tol as shown in Eq. 1.5 [4].

$$\chi_f = 1 - \frac{(\langle f^i - \langle f^i \rangle \rangle) \cdot (\langle f^{i+1} - \langle f^{i+1} \rangle \rangle)}{|\langle f^i - \langle f^i \rangle \rangle| \cdot |\langle f^{i+1} - \langle f^{i+1} \rangle \rangle|} \leq tol \tag{1.5}$$

where f is a vector with the components of the velocity u at all mesh nodes, i is the iteration number, the $\langle f \rangle$ represents the mean value of f, and tol is the tolerance parameter.

While the minimum tolerance is not reached, *Mandyoc* utilizes the Uzawa's method to iteratively calculate new u and P fields. The updated fields modify the viscosity field η, which in turn disturbs the velocity field again. These fields are updated until tolerance is reached. By default, the tolerance value for *Mandyoc* is 10^{-6}.

Additionally, the compositional factor C is evaluated for an advection as in the equation below. Its solution is calculated placing randomly a number of particles within each finite element of the mesh, which are displaced based on the adjacent node velocity values [5]. The individual value for each particle is obtained by linear interpolation of the node values.

$$\frac{\partial C}{\partial t} + u_i C_{,i} = 0 \tag{1.6}$$

Once the velocity field is solved, *Mandyoc* computes the temperature field as a function of the u, κ and H. In the next steps, the surface processes are computed and if the maximum time step or the maximum simulation time was not reached, the code updates the time and goes back to compute a new velocity field.