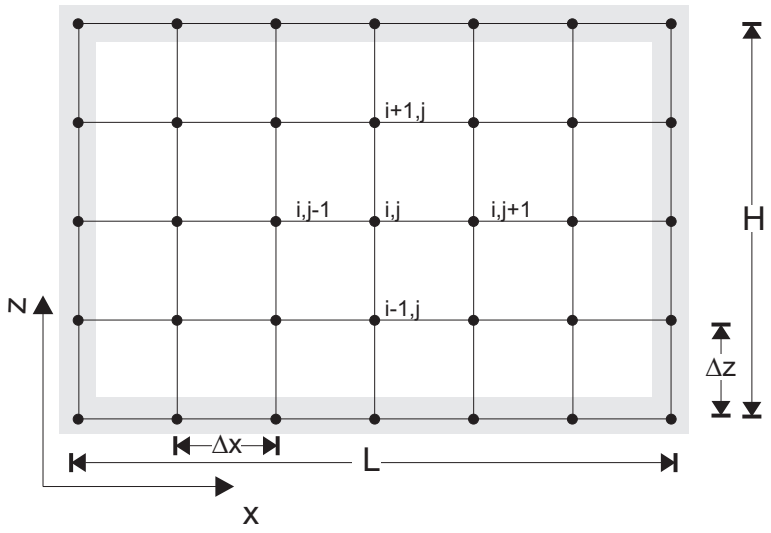

Figure 1: Finite difference discretization of the 2D heat problem.

1 Two-dimensional heat equation with FD

We now revisit the transient heat equation, this time with sources/sinks, as an example for two-dimensional FD problem. In 2D ($\{x, z\}$ space), we can write

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right) + Q \tag{1}$$

where, ρ is density, c_p heat capacity, $k_{x,z}$ the thermal conductivities in x and z direction, and Q radiogenic heat production.

If the thermal conductivity is isostropic ($k_x = k_z$) and constant, we can rewrite

$$\frac{\partial T}{\partial t} = \kappa\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2}\right) + \frac{Q}{\rho c_p}. \tag{2}$$

1.1 Explicit method

The simplest way to discretize eq. (2) on a domain, *e.g.* a box with width L and height H, is to employ an FTCS, explicit method like in 1-D

$$\frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} = \kappa\left(\frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{(\Delta x)^2} + \frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{(\Delta z)^2}\right) + \frac{Q_{i,j}^{n}}{\rho c_p}, \tag{3}$$

where Δx and Δz indicates the node spacing in both spatial directions, and there are now two indices for space, i and j for z_i and x_j, respectively (Figure 1). Rearranging gives

$$T_{i,j}^{n+1} = T_{i,j}^n + s_x \left(T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n \right) + s_z \left(T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n \right) + \frac{Q_{i,j}^n \Delta t}{\rho c_p}, \tag{4}$$

where

$$s_x = \frac{\kappa \Delta t}{(\Delta x)^2} \quad \text{and} \quad s_z = \frac{\kappa \Delta t}{(\Delta z)^2}. \tag{5}$$

Boundary conditions can be set the usual way. A constant (Dirichlet) temperature on the left-hand side of the domain (at $j = 1$), for example, is given by

$$T_{i,j=1} = T_{left} \quad \text{for all } i. \tag{6}$$

A constant flux (Neumann BC) on the same boundary at $\{i, j = 1\}$ is set through fictitious boundary points

$$\begin{aligned} \frac{\partial T}{\partial x} &= c_1 \\ \frac{T_{i,2} - T_{i,0}}{2\Delta x} &= c_1 \\ T_{i,0} &= T_{i,2} - 2\Delta x c_1 \end{aligned} \tag{7}$$

which can then be plugged into eq. (4) so that for $j = 1$, for example,

$$\begin{aligned} T_{i,1}^{n+1} = T_{i,1}^n \quad &+ \quad s_x \left(2T_{i,2}^n - 2(T_{i,1}^n + \Delta x c_1) \right) \\ &+ \quad s_z \left(T_{i+1,1}^n - 2T_{i,1}^n + T_{i-1,1}^n \right) + \frac{Q_{i,1}^n \Delta t}{\rho c_p} \end{aligned} \tag{8}$$

(compare eq. **??**).

The implementation of this approach is straightforward as T can be represented as a matrix with MATLAB , to be initialized, for example, for n_z and n_x rows and columns, respectively, as

$$\texttt{T} = \texttt{zeros(nz,nx)}; \tag{9}$$

and then accessed as `T(i,j)` for $T_{i,j}$. The major disadvantage of fully explicit schemes is, of course, that they are only stable if

$$\frac{2\kappa \Delta t}{\min((\Delta x)^2, (\Delta z)^2)} \leq 1. \tag{10}$$

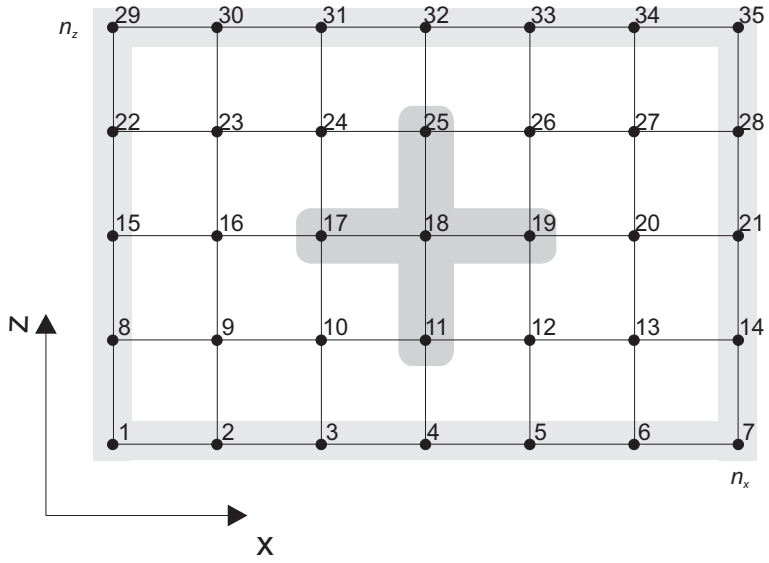

Figure 2: Numbering scheme for a 2D grid with $n_x = 7$ and $n_z = 5$.

1.2 Fully implicit method

If we employ a fully implicit, unconditionally stable discretization scheme as for the 1D exercise, eq. (2) can be written as

$$\frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} = \kappa \left(\frac{T_{i,j+1}^{n+1} - 2T_{i,j}^{n+1} + T_{i,j-1}^{n+1}}{(\Delta x)^2} + \frac{T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i-1,j}^{n+1}}{(\Delta z)^2} \right) + \frac{Q_{i,j}^{n}}{\rho c_p}. \tag{11}$$

Rearranging terms with $n+1$ on the left and terms with n on the right hand side gives

$$-s_z T_{i+1,j}^{n+1} - s_x T_{i,j+1}^{n+1} + (1 + 2s_z + 2s_x) T_{i,j}^{n+1} - s_z T_{i-1,j}^{n+1} - s_x T_{i,j-1}^{n+1} = T_{i,j}^{n} + \frac{Q_{i,j}^{n} \Delta t}{\rho c_p}. \tag{12}$$

As in the 1D case, we have to write these equations in a matrix $\underline{\underline{A}}$ and a vector $\underline{b}$ (and use MATLAB `x = A\b` to solve for T^{n+1}). From a practical point of view, this is a bit more complicated than in the 1D case, since we have to deal with "book-keeping" issues, *i.e.* the mapping of $T_{i,j}$ to the entries of a temperature vector `T(k)` (as opposed to the more intuitive matrix `T(i,j)` we could use for the explicit scheme).

If a 2D temperature field is to be solved for with an equivalent vector `T`, the nodes have to be numbered continuously, for example as in Figure 2. The derivative versus x-direction is then *e.g.*

$$\frac{\partial^2 T}{\partial x^2}|_{i=3,j=4} = \frac{1}{(\Delta x)^2} \left(T_{19} - 2T_{18} + T_{17} \right), \tag{13}$$

and the derivative versus z-direction is given by

$$\frac{\partial^2 T}{\partial z^2}|_{i=3,j=4} = \frac{1}{(\Delta z)^2}\left(T_{25} - 2T_{18} + T_{11}\right). \tag{14}$$

If n_x are the number of grid points in x-direction and n_z the number of points in z-direction, we can write eqs. (13) and (14) in a more general way as:

$$\frac{\partial^2 T}{\partial x^2}|_{i,j} = \frac{1}{(\Delta x)^2}\left(T_{(i-1)n_x+j+1} - 2T_{(i-1)n_x+j} + T_{(i-1)n_x+j-1}\right) \tag{15}$$

$$\frac{\partial^2 T}{\partial z^2}|_{i,j} = \frac{1}{(\Delta z)^2}\left(T_{i\cdot n_x+j} - 2T_{(i-1)n_x+j} + T_{(i-2)n_x+j}\right). \tag{16}$$

In matrix format this gives something like (*cf.* eq. 12)

$$\underline{\underline{A}} = \begin{pmatrix}
1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\
0 & 1 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & & & & & & & & & & & & \\
0 & 0 & & -s_z & \dots & -s_x & (1+2s_x+2s_z) & -s_x & \dots & -s_z & 0 & & 0 & 0 \\
0 & 0 & & 0 & -s_z & \dots & -s_x & (1+2s_x+2s_z) & -s_x & \dots & -s_z & & 0 & 0 \\
\vdots & \vdots & & & & & & & & & & & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1
\end{pmatrix}. \tag{17}$$

Note that we now have five diagonals filled with non-zero entries as opposed to three diagonals in the 1D case. The solution vector $\underline{x}$ is given by

$$\underline{x} = \begin{pmatrix}
T_1^{n+1} = T_{1,1} \\
T_2^{n+1} = T_{1,2} \\
\vdots \\
T_{(i-1)n_x+j}^{n+1} = T_{i,j} \\
T_{(i-1)n_x+j+1}^{n+1} = T_{i,j+1} \\
\vdots \\
T_{n_x n_z-1}^{n+1} = T_{n_z,n_x-1} \\
T_{n_x n_z}^{n+1} = T_{n_z,n_x}
\end{pmatrix}, \tag{18}$$

and the load (right hand side) vector is given by ($Q = 0$ for simplicity)

$$\underline{b} = \begin{pmatrix} T_{bottom} \\ T_{bottom} \\ \vdots \\ T^n_{(i-1)n_x+j} \\ T^n_{(i-1)n_x+j+1} \\ \vdots \\ T_{top} \\ T_{top} \end{pmatrix} . \tag{19}$$

1.3 Other methods

The fully implicit method discussed above works fine, but is only first order accurate in time (sec. **??**). A simple modification is to employ a Crank-Nicolson time step discretization which is second order accurate in time. In practice, this often does not make a big difference, but Crank-Nicolson is often preferred and does not cost much in terms of additional programming. You may consider using it for diffusion-type equations.

A different, and more serious, issue is the fact that the cost of solving `x = A\b` is a strong function of the size of `A`. This size depends on the number of grid points in x- (n_x) and z-direction (n_z). For a 2D problem with $n_x \times n_z$ internal points, $(n_x \times n_z)^2 \times (n_x \times n_z)^2$ equations have to be solved at every time step. This quickly fills the computer memory (especially if going to 3D cases).

For the special case of the temperature equation, different techniques have therefore been developed. One such technique, is the *alternating direction implicit* (ADI) method. It basically consists of solving the 2D equations half-explicit and half-implicit along 1D profiles (what you do is the following: (1) discretize the heat equation implicitly in the x-direction and explicit in the z-direction. (2) solve it for time $n + 1/2$, and (3) repeat the same but with an implicit discretization in the z-direction). Compared to the other methods, ADI is fast. However, ADI-methods only work if the governing equations have time-derivatives, and unfortunately this is often not the case in geodynamics. In the exercises, we therefore focus on the fully implicit formulation. If, however, you have to write a thermal solver at some point, you may strongly consider to use the ADI method (which is still very fast in 3D).

```
% Solves the 2D heat equation with an explicit finite difference scheme
clear
%Physical parameters
L       =   150e3;      %   Width of lithosphere    [m]
H       =   100e3;      %   Height of lithosphere   [m]
Tbot    =   1300;       %   Temperature of bottom lithosphere  [C]
Tsurf   =   0;          %   Temperature of country rock        [C]
Tplume  =   1500;       %   Temperature of plume               [C]
kappa   =   1e-6;       %   Thermal diffusivity of rock        [m2/s]
Wplume  =   25e3;       %   Width of plume                     [m]
day     =   3600*24;    %   # seconds per day
year    =   365.25*day; %   # seconds per year

% Numerical parameters
nx      =   101;         %   # gridpoints in x-direction
nz      =   51;         %   # gridpoints in z-direction
nt      =   500;          %   Number of timesteps to compute
dx      =   L/(nx-1);   %   Spacing of grid in x-direction
dz      =   H/(nz-1);   %   Spacing of grid in z-direction
[x2d,z2d] = meshgrid(-L/2:dx:L/2, -H:dz:0);  % create grid
% Compute stable timestep
dt         = min([dx,dz])^2/kappa/4;
% Setup initial linear temperature profile
T       =   abs(z2d./H)*Tbot;
% Imping plume beneath lithosphere
ind     =  find(abs(x2d(1,:)) <= Wplume/2);
T(1,ind) =  Tplume;
time    =   0;
for n=1:nt

    % Compute new temperature
    Tnew = zeros(nz,nx);
    sx  = kappa*dt/dx^2;
    sz  = kappa*dt/dz^2;
    for j=2:nx-1
        for i=2:nz-1
            Tnew(i,j) = ????;
        end
    end
    % Set boundary conditions
    Tnew(1,:)   =  T(1 ,: );
    Tnew(nz,:)  =  ?;
    for i=2:nz-1
        Tnew(i,1) = ?
        Tnew(i,nx) = ?
    end
    T           =   Tnew;
    time        =   time+dt;
    % Plot solution every 50 timesteps
    if (mod(n,50)==0)
        figure(1), clf
        pcolor(x2d/1e3,z2d/1e3,Tnew); shading interp, colorbar
        hold on
        contour(x2d/1e3,z2d/1e3,Tnew,[100:100:1500],'k');
        xlabel('x [km]')
        ylabel('z [km]')
        zlabel('Temperature [^oC]')
        title(['Temperature evolution after ',num2str(time/year/1e6),' Myrs'])
        drawnow
    end
end
```

Figure 3: MATLAB script `heat2D_explicit.m` to solve the 2D heat equation using the explicit approach.

1.4 Exercise: 2D heat equation with FD

You are to program the diffusion equation in 2D both with an explicit and an implicit discretization scheme, as discussed above. The problem to be considered is that of the thermal structure of a lithosphere of 100 km thickness, with an initial linear thermal gradient of 13 K/km. Suddenly a plume with $T = 1500^\circ$ C impinges at the bottom of the lithosphere. What happen with the thermal structure of the lithosphere? A related (structural geology) problem is that of the cooling of batholiths (like the ones in the Sierra Nevada).

```
% Solves the 2D heat equation with an implicit finite difference scheme
clear
%Physical parameters
L       =   150e3;      %   Width of lithosphere    [m]
H       =   100e3;      %   Height of lithosphere   [m]
Tbot    =   1300;       %   Temperature of bottom lithosphere  [C]
Tsurf   =   0;          %   Temperature of country rock        [C]
Tplume  =   1500;       %   Temperature of plume               [C]
kappa   =   1e-6;       %   Thermal diffusivity of rock        [m2/s]
Wplume  =   25e3;       %   Width of plume                     [m]
day     =   3600*24;    %   # seconds per day
year    =   365.25*day; %   # seconds per year
dt      =   100e6*year;   %   timestep
% Numerical parameters
nx      =   51;         %   # gridpoints in x-direction
nz      =   51;         %   # gridpoints in z-direction
nt      =   100;          %   Number of timesteps to compute
dx      =   L/(nx-1);   %   Spacing of grid in x-direction
dz      =   H/(nz-1);   %   Spacing of grid in z-direction
[x2d,z2d] = meshgrid(-L/2:dx:L/2, -H:dz:0);  % create grid
% Setup initial linear temperature profile
T       =   abs(z2d./H)*Tbot;
% Imping plume beneath lithosphere
ind     =  find(abs(x2d(1,:)) <= Wplume/2);
T(1,ind) =  Tplume;
% Setup numbering
num     =  1;
for i=1:nz
    for j=1:nx
        Number(i,j) = num;
        num = num+1;
    end
end
% Construct the A matrix
A       = sparse(nx*nz,nx*nz);
sx      = kappa*dt/dx^2;
sz      = kappa*dt/dz^2;
for i = 2:nz-1
    for j = 2:nx-1
        ii          = Number(i,j);
        A( ii, Number(i+1,j  ))  =   ??;
        A( ii, Number(i  ,j+1))  =   ??;
  ??
    end
end
% Set lower and upper BC
for j = 1:nx
??
end
% Set left and right BC
for i = 1:nz
??
end
time    =   0;
for n=1:nt
    % Compute rhs
    rhs   = zeros(nx*nz,1);
    for i = 1:nz
        for j = 1:nx
            ii      = Number(i,j);
          ??
        end
    end
    % Compute solution vector
    Tnew_vector =   A\rhs;
    % Create 2D matrix from vector
    Tnew        =   Tnew_vector(Number);
    T           =   Tnew;
    time        =   time+dt;
    % Plot solution every 50 timesteps
    if (mod(n,10)==0)
        figure(1), clf
        pcolor(x2d/1e3,z2d/1e3,Tnew); shading interp, colorbar
        hold on
        contour(x2d/1e3,z2d/1e3,Tnew,[100:100:1500],'k');
        xlabel('x [km]')
        ylabel('z [km]')
        zlabel('Temperature [^oC]')
        title(['Temperature evolution after ',num2str(time/year/1e6),' Myrs'])
        drawnow
    end
end
```

Figure 4: MATLAB script `heat2D_explicit.m` to solve the 2D heat equation using the implicit approach.

a) Fill in the question marks in the script `heat2Dexplicit.m` (Figure 3), by programming the explicit finite difference scheme. Employ zero flux boundary conditions $\frac{\partial T}{\partial x} = 0$ on the left and on the right-side of the domain (outside the top and bottom edges), and constant temperature conditions on the top and bottom. Ignore the effects of radioactive heat.

b) Finish the code `heat2Dimplicit.m` (Figure 4), by programming the implicit finite difference approximation of the 2D temperature equation.

c) A simple (time-dependent) analytical solution for the temperature equation exists for the case that the initial temperature distribution is

$$T(x,z,t=0) = T_{max} \exp\left[\frac{-(x^2+z^2)}{\sigma^2}\right] \tag{20}$$

where T_{max} is the maximum amplitude of the temperature perturbation at $(x,z) = (0,0)$ and σ its half-width. The solution is

$$T(x,z,t) = \frac{T_{max}}{\sqrt{1+4t\kappa/\sigma^2}} \exp\left[\frac{-(x^2+z^2)}{\sigma^2+4t\kappa}\right]. \tag{21}$$

(As for the 1D example, note that this uses a characteristic, time-dependent length-scale of $l_c \propto \sqrt{\kappa t}$, as expected for a diffusion problem (*cf.* sec. **??**), and see *Carslaw and Jaeger* (1959) for more analytical solutions.)

Program the analytical solution and compare it with the explicit and fully implicit numerical solutions with the same initial conditions at each time step. Comment on the accuracy of both methods for different values of dt.

d) Add the effects of radioactive heat to the explicit/implicit equations above. Use *Turcotte and Schubert* (2002) or Google to find typical values of Q, ρ, c_p for rocks.

e) *Bonus:* Write a code for the thermal equation with variable thermal conductivity k. Assume that the grid spacing Δx is constant. This type of code is not only relevant for thermal problems, but also for problems like hydro-geological problems (Darcy flow, *e.g.* how far did the chemical waste go into the aquifer?), fluid movements through the crust and through fault zones (which is related to the creation of ore deposits), magma migration through the mantle, geochemistry and mineral reactions at grain-boundary scale, and, aftershocks and fluids.

Bibliography

Carslaw, H. S., and J. C. Jaeger (1959), *Conduction of Heat in Solids*, 2nd ed., Oxford University Press, London, p. 243.

Turcotte, D. L., and G. Schubert (2002), *Geodynamics*, 2 ed., Cambridge University Press, Cambridge.