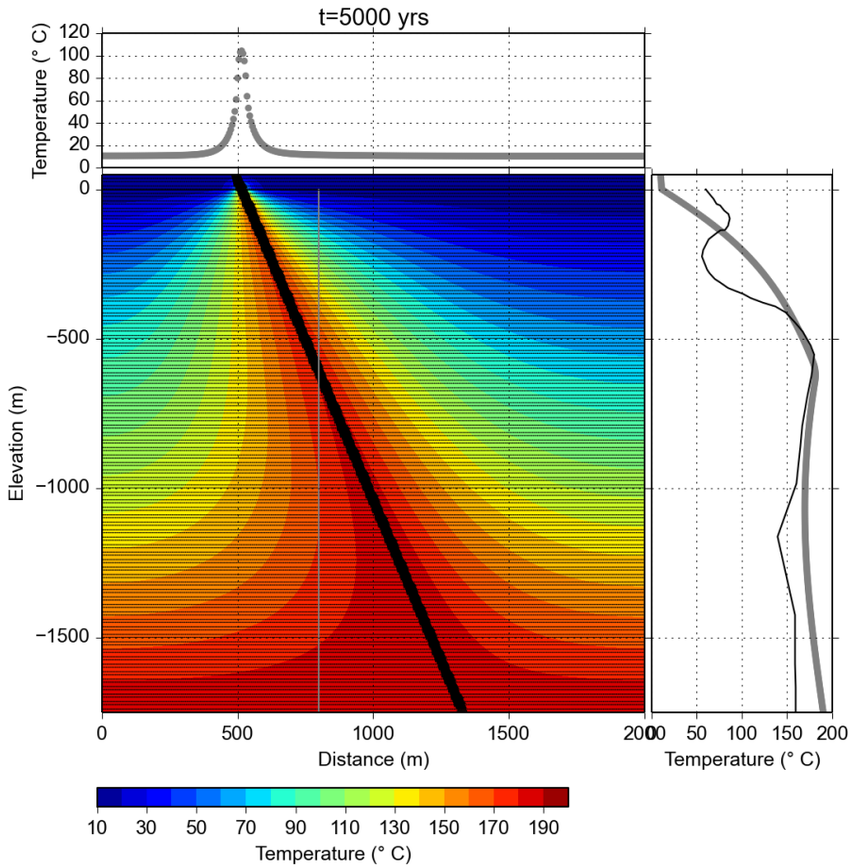
Exercise 3: Modelling transient heat flow using python and escript




```
1
2  dt= 1 * yr
3  Tc = 473*Celsius # Kelvin #the starting temperature of our country rock
4  rhoc = 2000*kg/m**3 #kg/m^{3} density
5  cpc = 920.*J/(kg*K) #j/kg.k specific heat
6  rhocp = rhoc*cpc #DENSITY * SPECIFIC HEAT
7  kappa = 1.9*W/m/K #watts/m.K thermal conductivity
8
9  ################################################ESTABLISHING PARAMETERS
10 #generate domain using rectangle
11 model = Rectangle(l0=mx,l1=my,n0=ndx, n1=ndy)
12
13 #create the PDE
14 mypde=LinearPDE(model) #assigns a domain to our PDE
15
16 #establish location of boundary between two materials
17 x=Function(model).getX()
18 bound = length(x-ic) - r #where the boundary will be located
19
20 #define our PDE coeffs
21 mypde.setValue(A=kappa*kronecker(model), D=rhocp/dt)
22
23 ##########################################################START ITERATION
24 while t<=tend:
25       i+=1 #counter
26       t+=h #current time
27       mypde.setValue(Y=qH + T*rhocp/dt)
28
29       # solve the PDE
30       T=mypde.getSolution()
```

M.Geo.239
11 Jan 2019
Elco Luijendijk, eluijen@gwdg.de

Summary of exercises

- Exercise 1 & 2: Steady-state groundwater & heat flow (=no change over time) in excel and Python

- Exercise 3: Transient groundwater flow -> how does fluid & heat flow in the crust change over time?

Solving the transient diffusion equation: explicit solution

- Step 1: calculate and initial steady-state value of u (in exercise 2)

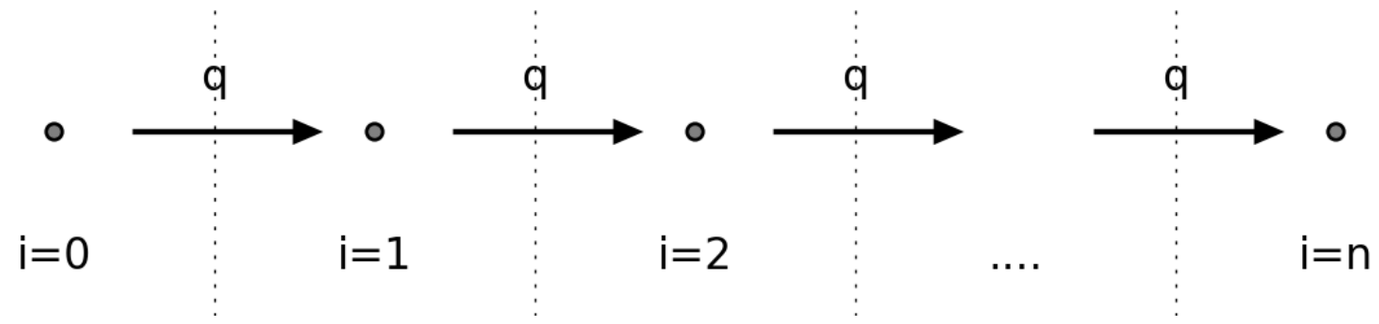
- Step 2: calculate flux between nodes:

$$q = K\frac{\partial u}{\partial x}$$

flux equation

$$q_{0-1} = K\frac{(u_0 - u_1)}{\Delta x}$$

discretized version for flux between node 0 and 1



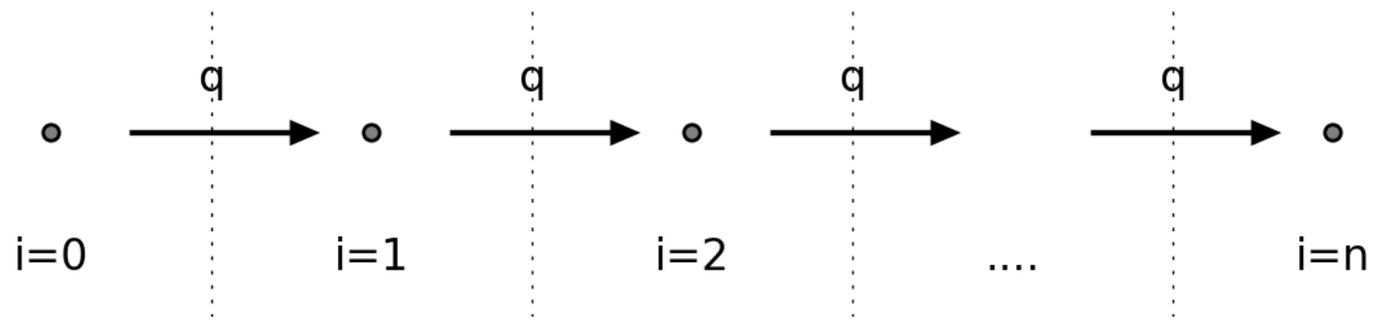
Solving the transient diffusion equation: explicit solution

- Step 3: use fluxes to update node values:

$$S\frac{\partial u}{\partial t} = \nabla K \nabla u + W$$ Full equation

$$S\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = -\frac{q_{i+1/2,j} - q_{i-1/2,j}}{2\Delta x} + W$$ Discretised equation

calculate the new node value of *u* at timestep j+1
as a function of flux *q* at current timestep j

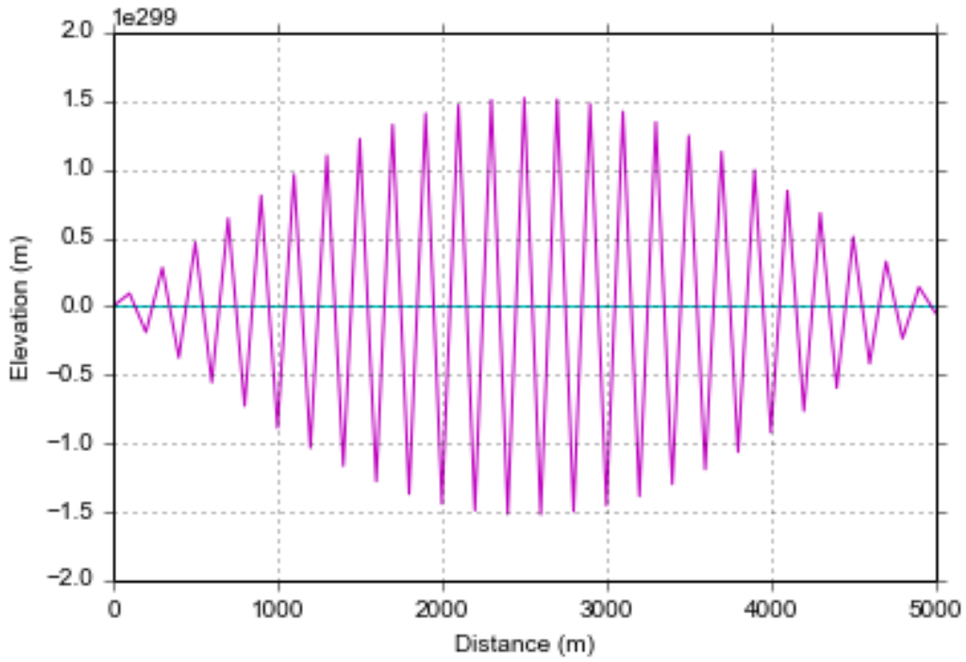


Solving the transient diffusion equation: explicit solution

- Explicit solutions: modelled temperature or hydraulic head is a function of the flux and temperature or head at the previous time step

- This method is relatively easy to implement, but has one important drawback: it is unstable at large time steps

- At large time steps hydraulic head or temperature gradients travel more than one node distance per time step. At the following time step the gradient is inverted at the node that was skipped and the solution starts to oscillate



Solving the transient diffusion equation: <u>implicit solution</u>

- Implicit solution: temperature or hydraulic head depends on flux & temperature or hydraulic head at the same time step

- This method is stable at any time step -> allows more efficient modelling of processes on long timescales

- However, also much more difficult to implement:

Heat flow equation $$\rho c \frac{\partial T}{\partial t} = \nabla q + Q \qquad q = K \nabla T$$

Heat flow equation, implicit discretisation: $$\rho c \frac{T_i^{t+1} - T_i^t}{\Delta T} = \frac{2K}{\Delta x^2} \left(T_{i+1}^{t+1} + T_{i-1}^{t+1} - 2T_i^{t+1} \right) + Q$$

Solving the transient diffusion equation: implicit solution

- Implicit solution results in one equation per node, with a single unknown
- Can be solved directly for low number of nodes: recall solving 2 equations with unknown x and y in your high school math classes
- Or more efficiently by casting these equations in the form of a matrix and using matrix solvers

discretized equation

$$\rho c \frac{T_i^{t+1} - T_i^t}{\Delta T} = \frac{2K}{\Delta x^2}\left(T_{i+1}^{t+1} + T_{i-1}^{t+1} - 2T_i^{t+1}\right) + Q$$

rewrite in matrix A
vector b to solve for unknown x

$$A\vec{x} = \vec{b}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -s & (1+2s) & -s & 0 & 0 & 0 \\ 0 & -s & (1+2s) & -s & 0 & 0 \\ 0 & 0 & -s & (1+2s) & -s & 0 \\ 0 & 0 & 0 & -s & (1+2s) & -s \\ 0 & 0 & 0 & 0 & -s & (1+s) \end{bmatrix} \quad \vec{b} = \begin{bmatrix} T_s \\ T_1^t + \frac{Q\Delta t}{\rho c} \\ T_2^t + \frac{Q\Delta t}{\rho c} \\ T_3^t + \frac{Q\Delta t}{\rho c} \\ T_4^t + \frac{Q\Delta t}{\rho c} \\ T_5^t + \frac{(Q+w/\Delta x)\Delta t}{\rho c} \end{bmatrix} \quad s = \frac{2K\Delta t}{\rho c \Delta x^2}$$

Solving the transient diffusion equation: implicit solution

- And this was a simple 1D example with constant thermal conductivity…
- However, no need to program your own solutions -> a lot of existing model codes can solve several forms of the diffusion equation for you: fluid flow, heat flow or solute transport model codes
- Choice of model codes: codes that solve a single or fixed set of equations:
  - Fluid flow: Modflow
  - Coupled fluid flow, heat flow and/or solute transport: Sutra, Seawat, Feflow

Modelling fluid & heat flow

- Or generic model codes that solve general partial differential equations of physical processes:
    - Comsol, FlexPDE, Abacus, Escript
- Advantages: more flexible, no need to stick to assumptions or simplifications in existing model codes. Can adjust the modeled processes to make them simpler and faster/easier to solve or more complex in case of more complex physics
- Examples:
    - recent questions on multi-phase (vapor, liquid) fluid flow in hydrothermal systems, oil reservoirs, CO2 storage
    - Coupled fluid, heat flow and deformation in induced seismicity, tracking
    - New research questions on the link between seismicity, heat and fluid flow
    - Plus also a simple advective heat transport model like the one you use in exercise 4: this is too simple for commercial codes

Modelling fluid and heat flow using escript

- We will use the generic model code Escript, developed at the university of Queensland
- Escript is a so-called finite element model thats can solve partial differential equations to simulate many different physical processes
- Advantages: Free, open source, flexible, relatively fast, can use multiple processors simultaneously (parallel processing)

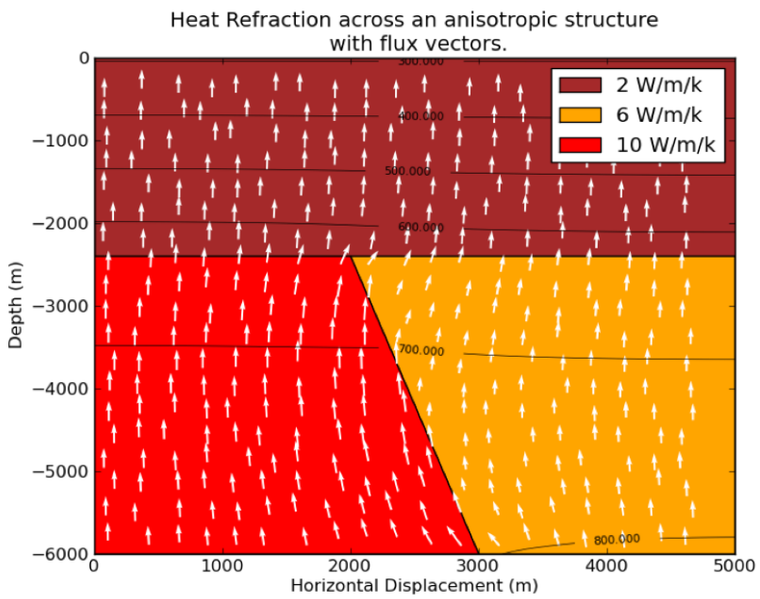


Figure 4.8: Example 6: Heat refraction model with three blocks and heat flux

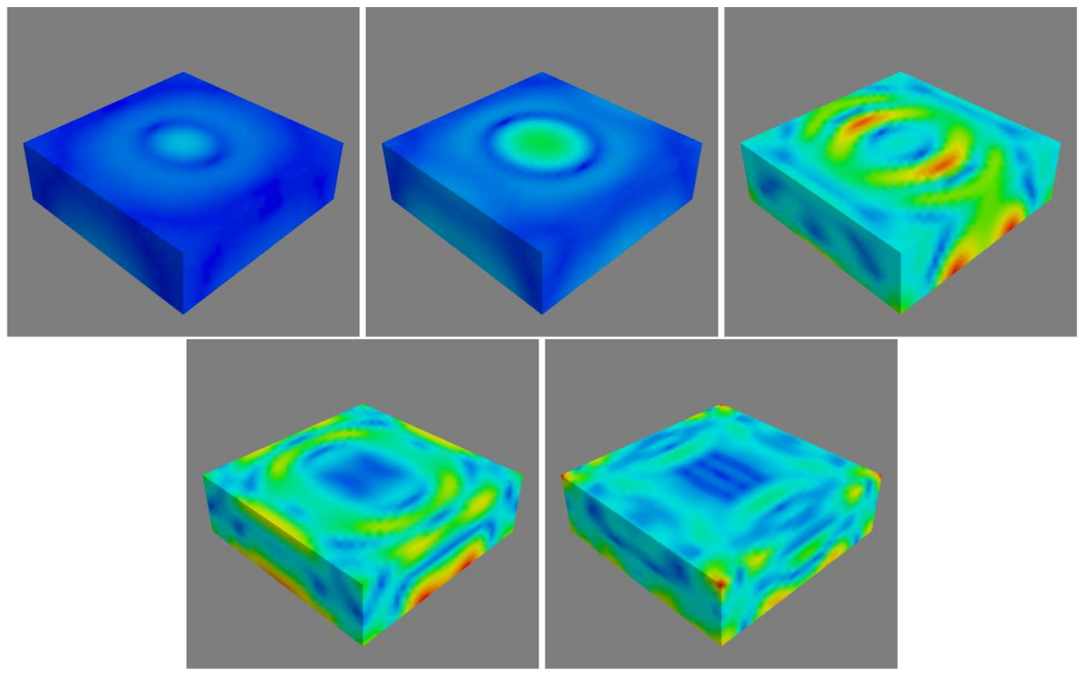


FIGURE 1.9: Selected time steps ($n = 11, 22, 32, 36$) of a wave propagation over a 10km $\times$ 10km $\times$ 3.125km block from a point source initially at (5km, 5km, 0) with time step size $h = 0.02083$. Color represents the displacement. Here the view is oriented onto the bottom face.

Modelling fluid and heat flow using Escript

- Under the hood Escript solves a fairly large partial differential equation
- The trick to use escript is to rewrite a partial differential equation of the process you are interested in in the same form as this equation
- And then use Python to tell escript what coefficients to use and solve the equation:

$$-\nabla(A\nabla u + Bu) + C\nabla u + Du = -\nabla X + Y$$

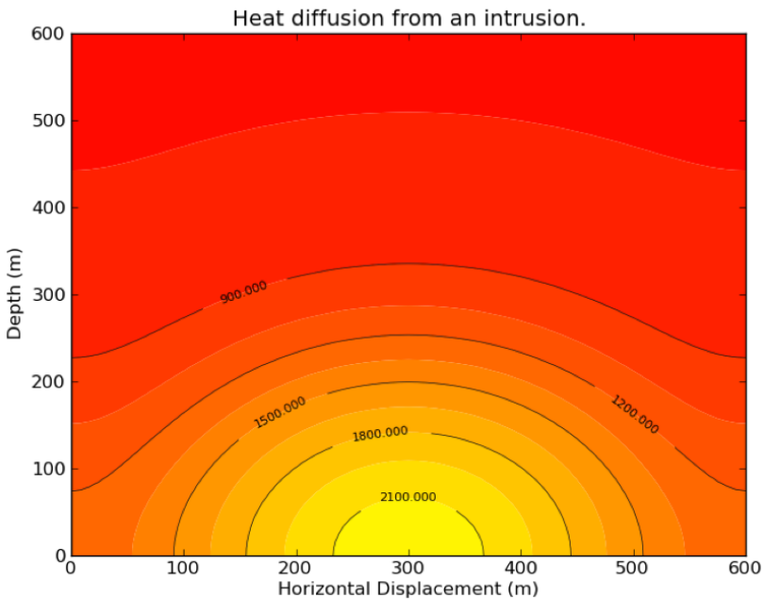
$$-\nabla dtK\nabla T^{t+1} - dt\rho_f c_f \vec{q}\nabla T^{t+1} + \rho_b c_b T^{t+1} = \rho_b c_b T^t$$

$$-\nabla \underbrace{dt\,K}_{A}\nabla T^{t+1} + (\underbrace{dt\rho_f c_f\,\vec{q}}_{C})\nabla u^{t+1} + \underbrace{\rho_b c_b}_{D} u^{t+1} = \underbrace{\rho_b c_b u^t}_{Y}$$

The discretised heat flow equation in escript format

Modelling fluid and heat flow using Escript

- Since escript does all of the heavy computational work in the background, your model code can stay relatively simple and compact:




```
1
2   dt= 1 * yr
3   Tc = 473*Celsius # Kelvin #the starting temperature of our country rock
4   rhoc = 2000*kg/m**3 #kg/m^{3} density
5   cpc = 920.*J/(kg*K) #j/kg.k specific heat
6   rhocp = rhoc*cpc #DENSITY * SPECIFIC HEAT
7   kappa = 1.9*W/m/K #watts/m.K thermal conductivity
8
9   ##################################################ESTABLISHING PARAMETERS
10  #generate domain using rectangle
11  model = Rectangle(l0=mx,l1=my,n0=ndx, n1=ndy)
12
13  #create the PDE
14  mypde=LinearPDE(model) #assigns a domain to our PDE
15
16  #establish location of boundary between two materials
17  x=Function(model).getX()
18  bound = length(x-ic) - r #where the boundary will be located
19
20  #define our PDE coeffs
21  mypde.setValue(A=kappa*kronecker(model), D=rhocp/dt)
22
23  ##########################################################START ITERATION
24  while t<=tend:
25        i+=1 #counter
26        t+=h #current time
27        mypde.setValue(Y=qH + T*rhocp/dt)
28
29        # solve the PDE
30        T=mypde.getSolution()
```

an example of a heat flow model in escript

What we will use escript for

- We will use Escript and python to model heat advection by fluid flow in a hydrothermal system
- This week & next three weeks: use escript to model a hydrothermal system
- Escript code, manual and examples available on https://launchpad.net/escript-finley
- However, no need to install this yourself, everything is preinstalled on the teaching laptops
- We will start with a simplified model of a hydrothermal system in the Basin and Range rift system, the Beowawe hydrothermal system
- However, the model is very simple and can be adapted to any hot spring that you'd like to model as long as there is data on the temperature and discharge of the spring, and a simple geological cross section
- Other potential candidates to model: the Aachen hot spring, several hot springs in the Alps, or an oceanic hydrothermal system at a seamount

Hydrothermal systems in fault zones

- Example of a non-magmatic hydrothermal system resulting from topography-driven flow: the Beowawe hydrothermal system

- Located in the basin and range rift, high density of active hydrothermal systems, probably because of active extensional tectonics, high deformation rates and high background heat flow (~110 mW m2) due to thinned crust and lithosphere

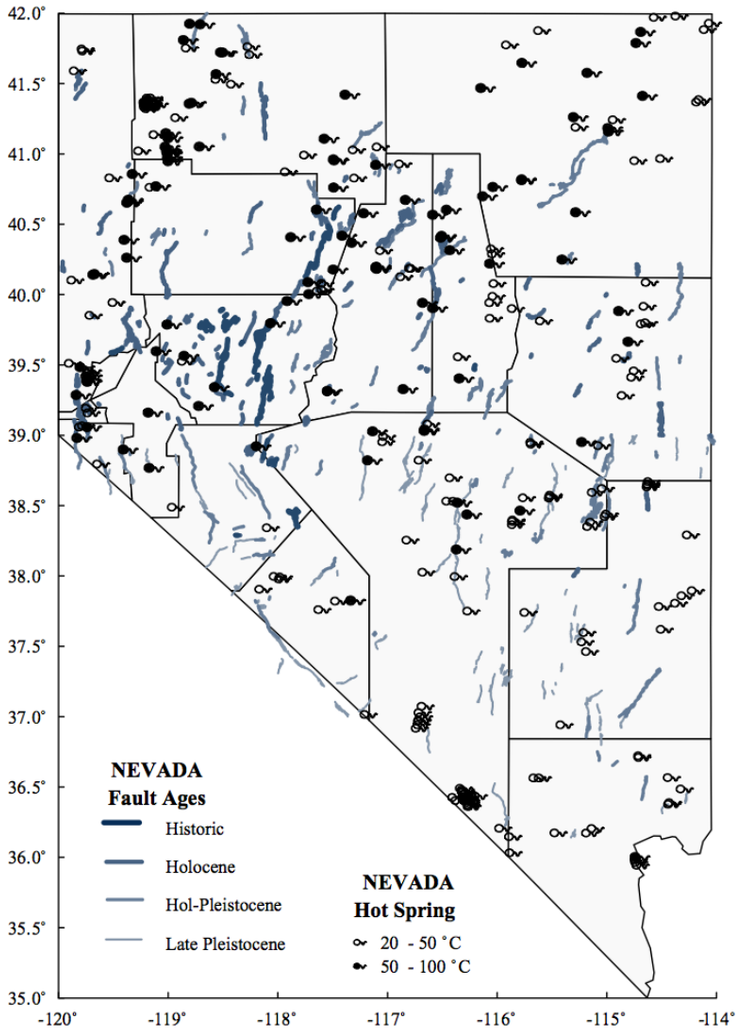


Fig. 9. High-temperature gradients, thermal springs, and faulting in Nevada.

Figure 4. Malpais fault zone and Beowawe Geysers terrace. View eastward; photograph by G.A. Thompson, taken October 18, 1951.

overview of hydrothermal sites
in the basin and range rift system, Nevada
McKenna & Blackwell (2004) Geothermics

Beowawe hydrothermal system

- Hydrothermal system evident from active geysers (up to 1960s)
- Largest geyser field outside of yellowstone

Figure 8. "Teakettle" Geyser (vent 6) erupting from its double vent, October 1, 1950. Stronger plume, 3 to 4½ m high, is inclined about 40° from vertical to the southwest. Weaker plume, largely of mist, is about 2 to 3½ m high and inclined about 50° from vertical to the northeast.

Figure 5. "Beowawe" Geyser erupting from vent 29 to a height of about 8 m on September 22, 1945.

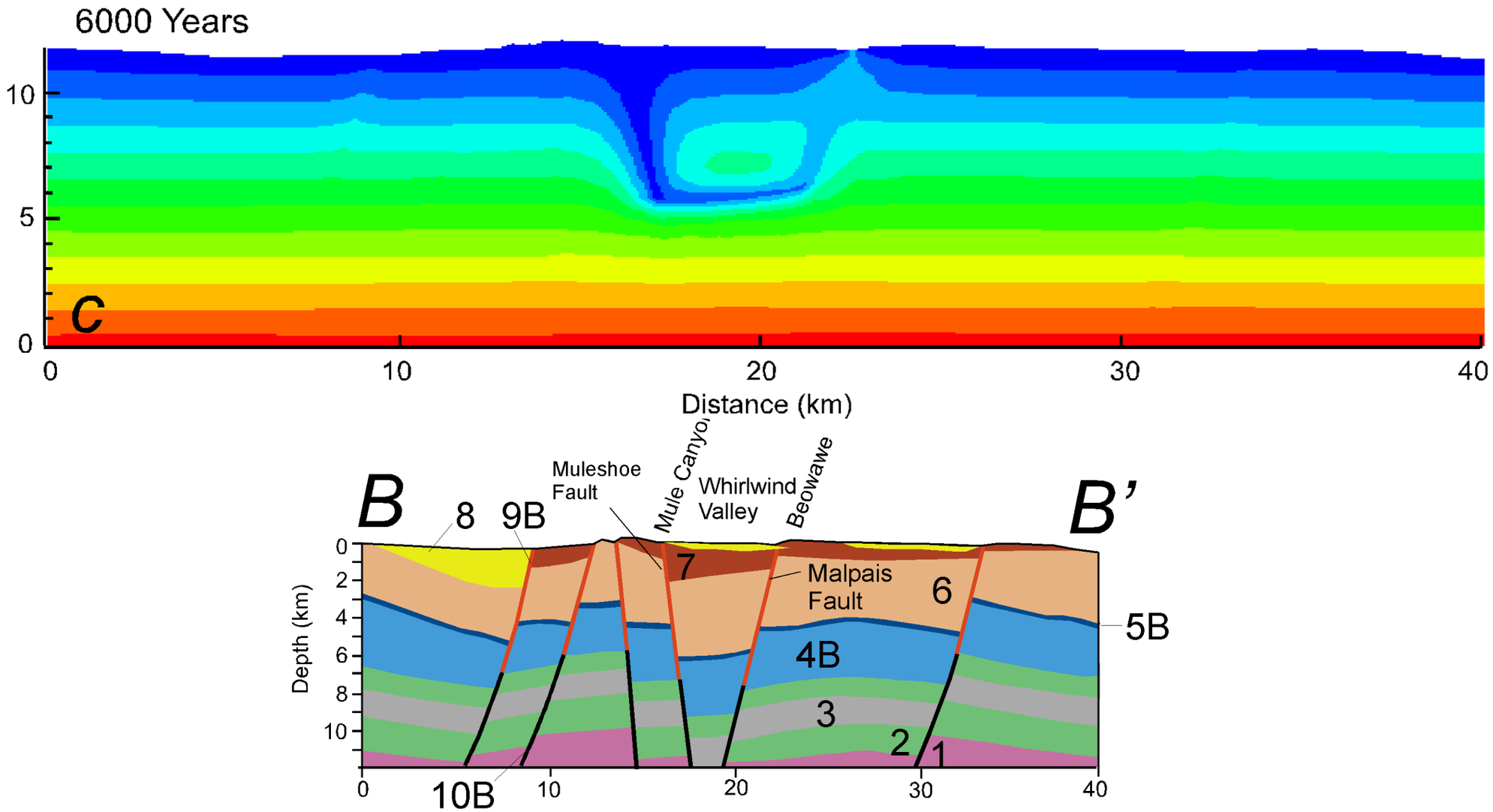
Beowawe hydrothermal system

- And massive hydrothermal sinter deposits consisting of amorphous silica, 60 m high, 1.5 km long

Beowawe hydrothermal system

- Modelling topography-driven flow: recharge at more elevated outcrop of Muleshoe fault to the NW, lateral flow through deep carbonates and upward flow along Malpais fault

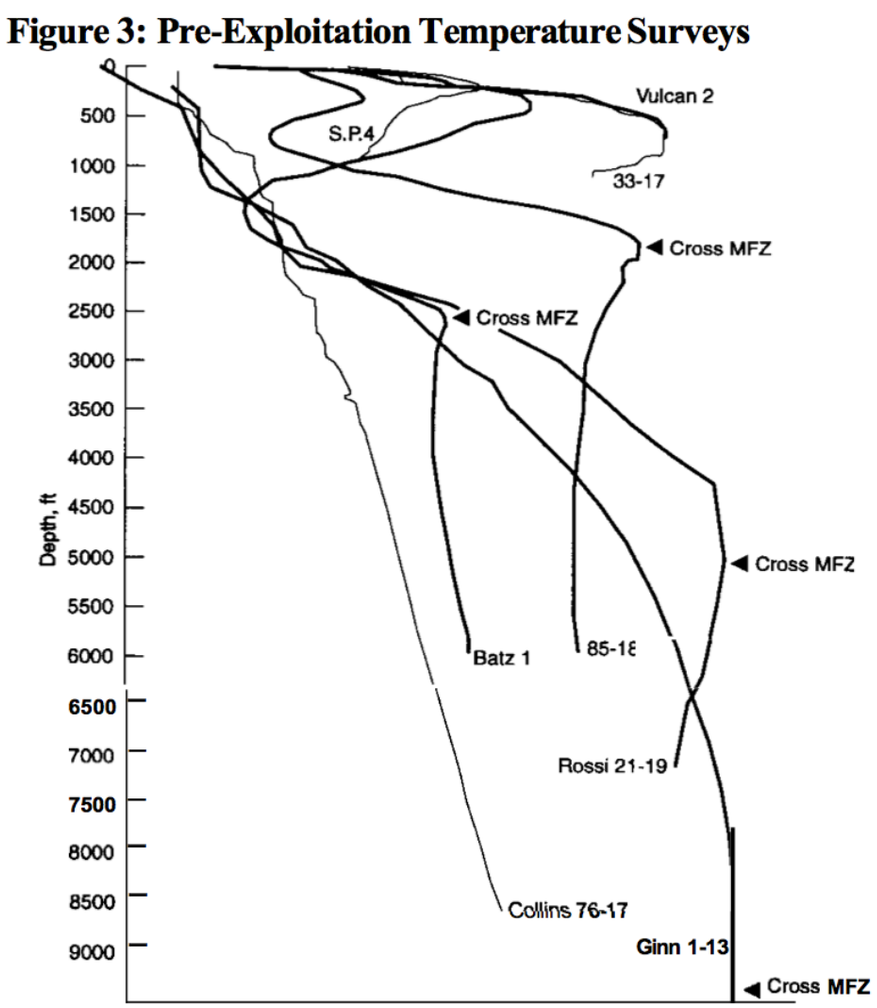
- First order characteristics fit: meteoric water discharges with temperatures of ~100 ˚C at the Mule Canyon fault



Beowawe hydrothermal system

- Numerical model also predicts changes in temperatures and chemistry over time, can we observe this in the field?

- Temperature data show overturned temperature-depth profiles:



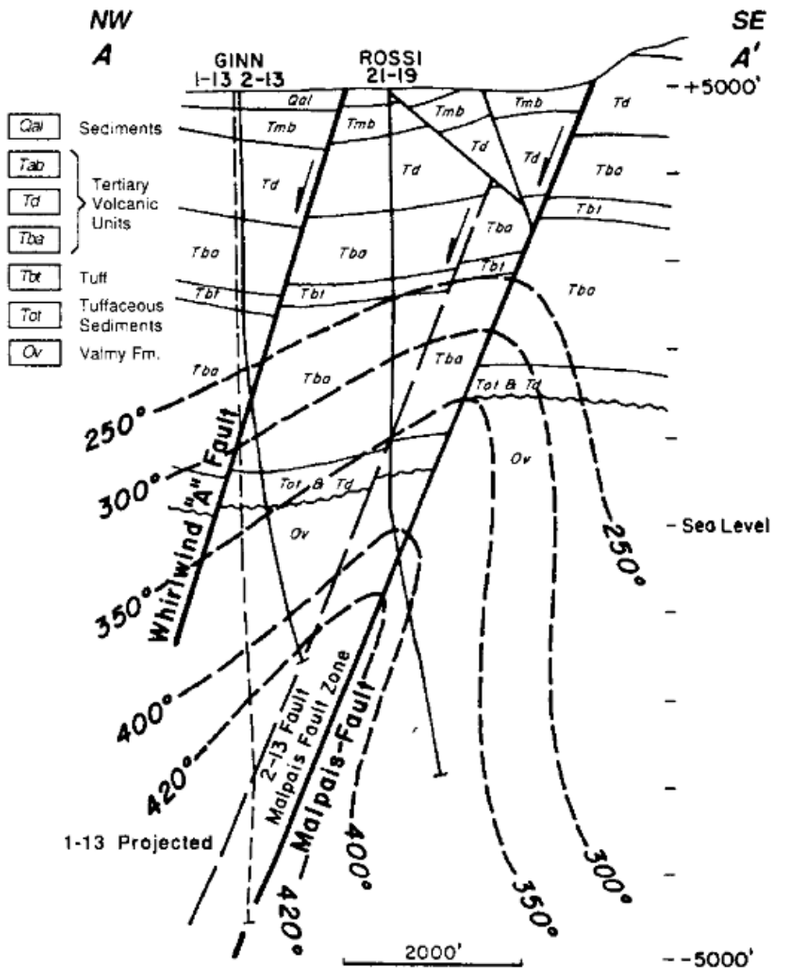
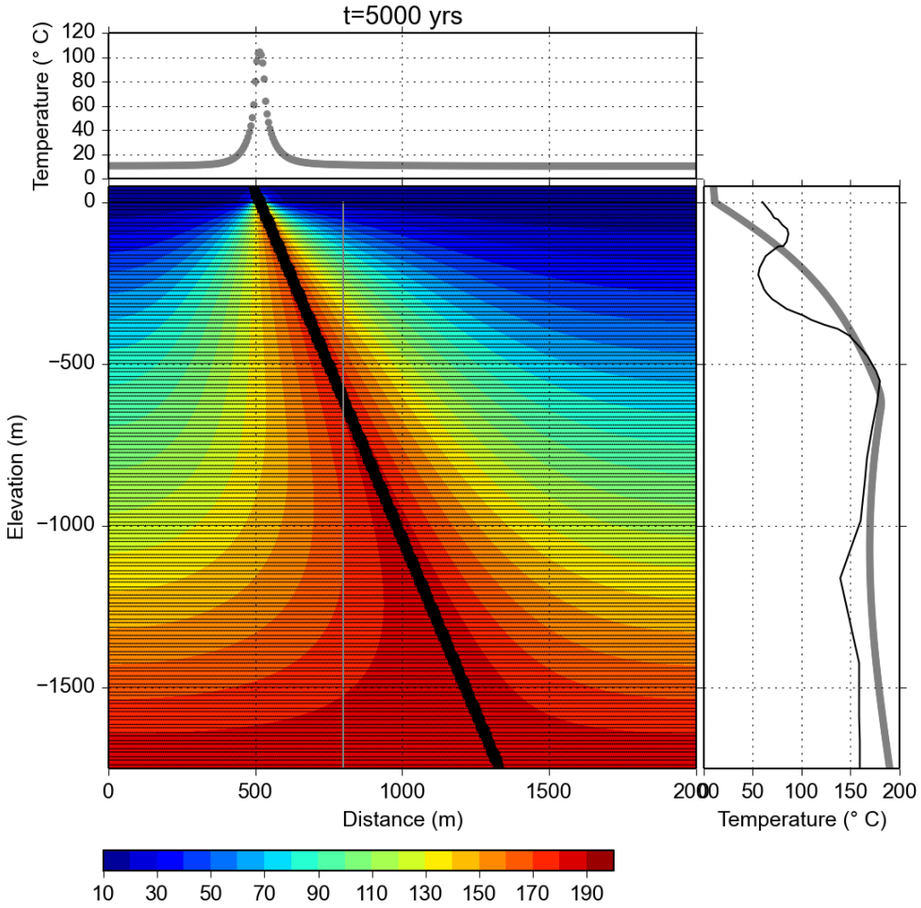
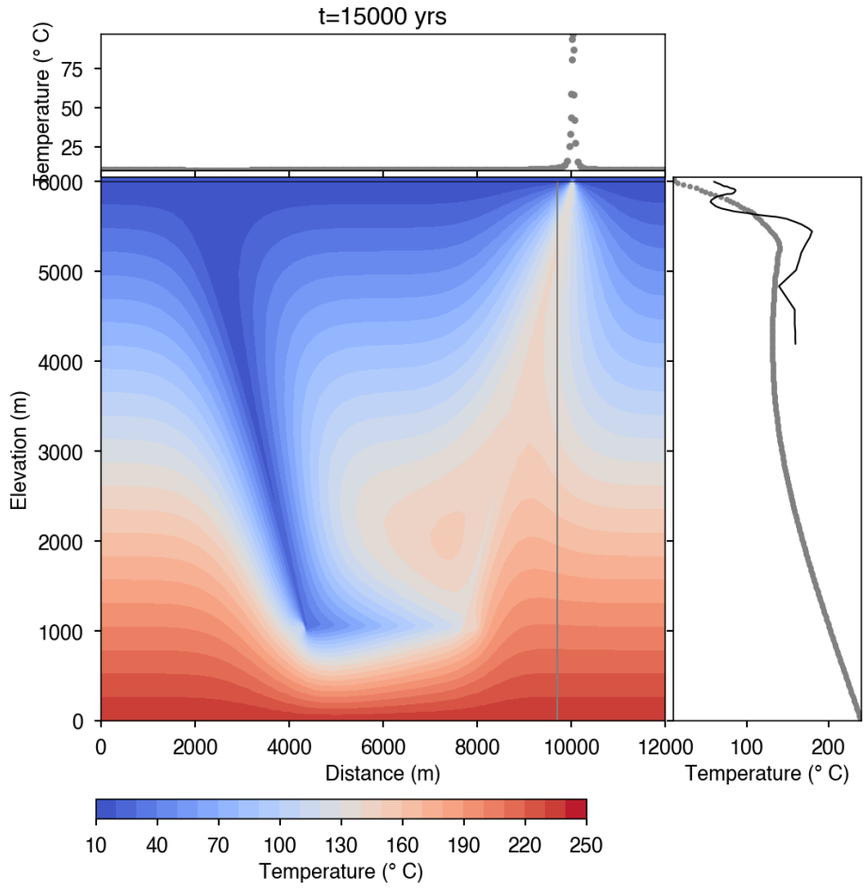


Figure 4. Geologic cross-section A-A'.

Hydrothermal model:

- What we will model:
- 2 options: simplified model of heat flow with user-specified fluid flux through faults and/or connected aquifers
- or a combined groundwater & heat flow model
- Calibration targets: Try to match the spring discharge, temperature, and measured temperatures in a borehole close to the hydrothermal system





Hydrothermal model

- Escript is pre-installed on the linux laptops
- Test escript by:
    - downloading the test script (example03a.py) from the escript cookbook
    - run this by opening the terminal, navigating to the folder where you saved the script and typing python example03a.py
        - The script should now run and make figures of the cooling of a granite intrusion over time
- To run the hydrothermal model:
    - Download the jupyter notebook and the python script for exercise 4 from stud.ip and save them to the same folder
    - Run the notebook
    - inspect the model output (figures, spreadsheet, VTK files with paraview)
    - install paraview by opening a terminal and typing: sudo apt install paraview
    - Change the parameters and rerun the model again