GEOS 3104

Practical Exercise

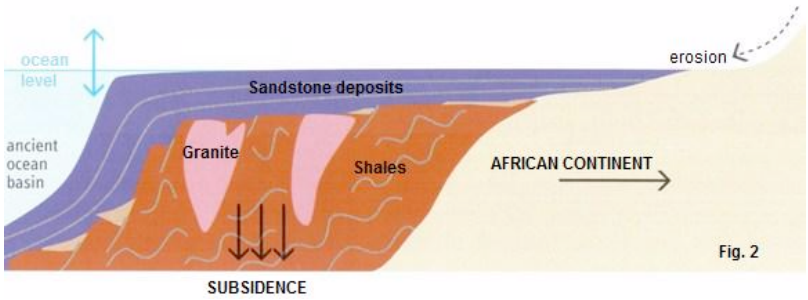
Thursday 20 August 2009, 2 pm. The reports are due in one week.

Diffusion Equation with Matlab

We start a series of three practical exercises on **Numerical modeling** with the most classical and easiest of the equations; the diffusion equation. We will investigate two solving techniques, both based on the *finite difference* methodology, using two different formulations, one in 1D and another in 2D.

Let's assume heat diffusion of a non-deformed medium with constant thermal conductivity (k) and density (ρ) and heat capacity (Cp) and, as a consequence, constant thermal diffusivity $\kappa = k/(\rho Cp)$. This example is a good approximation for the oceanic lithosphere as it drifts over the Earth's mantle, prior to being subducted. The oceanic lithosphere is almost rigid and temperature changes occur mostly via conduction. We can therefore suppose to solve its thermal evolution as a function of its age.

In particular we will investigate numerically the time that a magma inclusion in the crust needs to reach a thermal equilibrium with the surrounding rocks. An example of such a system is a granitic intrusion which does not reach the surface during its cooling.



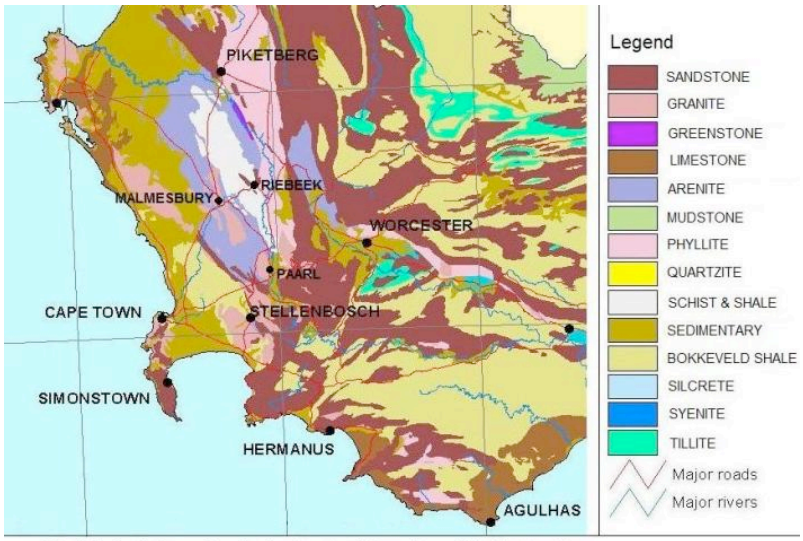


Fig 4. Geological map showing dominant rock formations of the Western Cape

Figure 1: Magmatic Intrusion

Technical tools that we use

Assuming you want to calculate the development over time of a solid body with one or more initial thermal anomalies or with a non-equalized thermal jump at its extremities (i.e. a rock suddenly immersed in a magma chamber or a magma inclusion in a rock matrix). A solution to this problem can be reached with the *Finite Difference* technique which we apply today.

In this context the continuous functions we are going to use are discretized as we have seen in the first lectures, i.e. the time and space are divided in regular intervals (t_1, t_2,…, t_n) and (x_1,x_2,…,x_n) (see Figure 2), and a function, temperature in our case, is defined for each t_i and x_i, e.g. $T(t_i, x_i)$. Because most of the equations of interest in geodynamics are differential equation, i.e. equations that involve derivatives of a function, we need a technique to express derivatives as a discretized function. This technique, or methodology, is called *Finite Difference Method*.

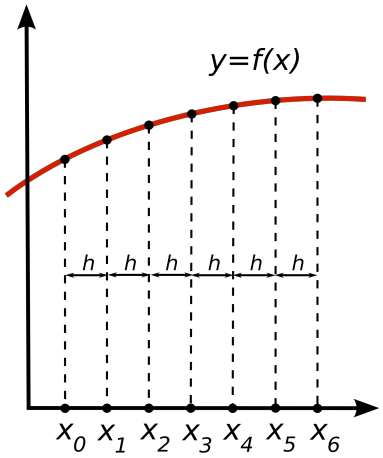


Figure 2. Discretization of a function y(x) for regular intervals h of x.

In finite differences space and time derivatives are calculated using the discretization of the definition of derivation. Given a function f(x), discretized in a domain defined by the array (x_1,x_2,…,x_n), where the values of the function are $f_i=f(x_i)$, the local derivative is expressed as:

$$\frac{df(x)}{dx} \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \qquad (1)$$

We will use this formula to define the time derivative of the temperature T(x, t) in 1D of our initial very hot anomaly:

$$\frac{dT(x,t)}{dt} \approx \frac{T(x,t+\Delta t) - T(x,t)}{\Delta t} \qquad (2)$$

In the second part of this practical we will need the above discretization in the time domain and the discretization of the laplacian (second derivative) in the space domain. In finite difference in one dimension the second derivative is normally expressed as:

$$\frac{d^2 f(x)}{d^2 x} \approx 4\frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{\left(x_{i+1} - x_{i-1}\right)^2} \qquad (3)$$

We will make use of these two relationships to look at the diffusion in time using an initially square thermal anomaly in 2D.

Diffusion of a thermal anomaly in 1D

As an initial example we first use a "toy model". A simulation in 1D where only the discretization in time is used. This means we solve the expression (2) with an empirical right hand side, that represents the fraction of heat that goes from each cell to every neighbouring ones, in 1D one on the left and one on the right:

$$\frac{dT(x,t)}{dt} \approx \frac{T(x,t+\Delta t)-T(x,t)}{\Delta t} = Diff[T(x,t)] \qquad (4)$$

In order to explicitly solve our system, we first have to create the mesh in space and time. We define the number of space cells and then the time steps.

```
ncells = 100;         % Number of Cells in the Array
NTimes = 200;         % Number of Time steps to perform
```

You can now define the fraction of heat diffused laterally. For example:

```
FracLeft  = 1/3;    % Fraction of heat diffusing to the left
FracSame  = 1/3;    % Fraction of heat which remains in same cell
FracRight = 1/3;    % Fraction of heat diffusing to the right
```

Generate a vector that contains the starting conditions and will contain the time evolution of the temperature profile:

```
Temps = zeros(ncells,1);  % Initialize the Array that contains the
temperature values to zero
TMax = 100;                  % Maximum Temperature
Temps(ncells/2) = TMax;    % Initially set the Temperature at the
midpoint
```

Now we need to iterate using cycle "for" in order to update the model in function of time:

```
for i=1:NTimes  % Perform a total of NTimes time steps
  NewTemps = Temps*FracSame;  % New temperatures initially a
fraction of original
```

And a second cycle "for" in order to diffuse the heat:

```
  for j=2:ncells-1  % For each cell except leftmost or rightmost
        NewTemps(j-1) = NewTemps(j-1)+Temps(j)*FracLeft;  %
Diffuse Left
        NewTemps(j+1) = NewTemps(j+1)+Temps(j)*FracRight; %
Diffuse Right
  end % for j=1...
  Temps = NewTemps;     % The updated temperatures become the
current temperatures
```

Finally we plot the solution at each time step:

```
  plot(Temps)             % Plot the values
  axis([1 ncells 0 TMax]) % Set the minima and maxima on axes
  xlabel('Position')      % Label the axes and the title
  ylabel('Temperature')
  title(['After Time Step: ' num2str(i)])
```

```
  drawnow                    % Make Matlab display the graph
                             % (Normally it only displays at the end
of a program)
end  % for i=1...
```

Save the script in a file and try to run it.

What happens if you change the lateral diffusion fractions? For example:

```
   FracLeft  = 1/20;    % Fraction diffusing to the left
   FracSame  = 9/10;    % Fraction which remains in same cell
   FracRight = 1/20;    % Fraction diffusing to the right
```

A diffusion can also be asymmetric. This produces a lateral drift of the inclusion. Only on one side:

```
   FracLeft  = 0;       % Fraction diffusing to the left
   FracSame  = 9/10;    % Fraction which remains in same cell
   FracRight = 1/10;    % Fraction diffusing to the right
```

The sum of the FracLeft+FracSame+FracRight must be always one. Do you know why? If not, try to change them in a way that the sum is more or less then one and see what happens.

Finite Differences: Diffusion equation in 2D

The FracLeft, FracSame and FracRight properties of the prior section have been empirically set and give us an intuitive physical understanding of the diffusion process. We want to model now the same system as before, but employing a more exact treatment and in 2D, in order to capture better its dynamics and consider also lateral heterogeneities. We visualize it as a square of high temperature that is surrounded by cold material. By applying the finite difference method, we can model how the heat diffuses from the hot part into the surrounding cooler material.

For this we first need to solve the Poisson equation:

$$\frac{df(x,y,t)}{dt} = k\nabla^2 f(x,y,t), \text{ in detail: } \frac{df(x,y,t)}{dt} = k\left(\frac{d^2 f(x,y,t)}{d^2 x} + \frac{d^2 f(x,y,t)}{d^2 y}\right)$$

in its *Finite Difference* Formulation to discretize it for our purposes:

$$\frac{T(x_i, y_i, t+\Delta t) - T(x_i, y_i, t)}{\Delta t} =$$

$$4k\left(\frac{f(x_{i+1}, y_i, t) - 2f(x_i, y_i, t) + f(x_{i-1}, y_i, t)}{(x_{i+1} - x_{i-1})^2} + \frac{f(x_i, y_{i+1}, t) - 2f(x_i, y_i, t) + f(x_i, y_{i-1}, t)}{(y_{i+1} - y_{i-1})^2}\right)$$

Let's take a squared 2D grid, with constant thermal diffusivity, no advection (velocity), no heat sources and constant temperature at the boundary. Our two dimensional discretization looks like Figure 3:

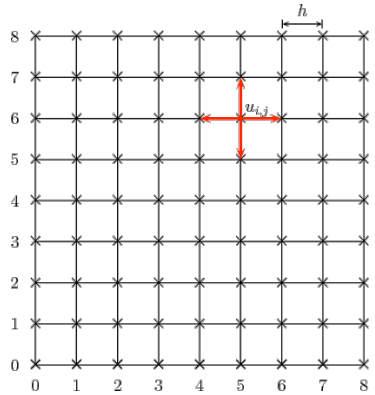


Figure 3. Discretization of the Finite Differences Formulation in 2D

A similar grid as we see in Figure 2 needs to be defined:

```
% Defining grid in the horizontal direction
xmin=0.0; xmax=3000.0; xnum=99;
xstp=(xmax-xmin)/(xnum-1);
ymin=0.0; ymax=3000.0; ynum=99;
ystp=(ymax-ymin)/(ynum-1);
```

We can now define the material properties that control the diffusion process:

```
k=3.0; % Thermal conductivity, W/m/K
ro=3000.0; %Density, kg/m^3
cp=1000.0; % Isobaric heat capacity, J/kg
kappa=k/ro/cp; % Thermal diffusivity, m^2/s
```

The explicit algorithm that we employ in this section is very simple to apply (in practice at each

time step the increment of diffused temperature is added to the old field) but it has the important limitation that it is intrinsically unstable beyond a threshold "time increment". The timestep that we employ is close to such a limit. When you finish writing the script, try to run the simulation with a time step 2 to 5 times larger to see what "unstable behaviour" looks like:

```
timestep=0.75*((xstp^2+ystp^2)/2.0)/kappa/3.0; % Crucial
parameter: Explicit timestep

% Defining Initial Temperature structure
tmin=500;
tmax=1000;
for xn = 1:xnum
    for yn = 1:ynum
        T0(xn,yn)=tmin; % Background Temperature, K
        if ((xn-1)/(xnum-1)>0.25 & (xn-1)/(xnum-1)<0.75 & (yn-
1)/(ynum-1)>0.25  & (yn-1)/(ynum-1)<0.75)
            T0(xn,yn)=tmax; % Rectangular Hot body in the middle
        end
    end
end
```

The increment of temperature is calculated employing the *Finite Difference* expression (2) for the new temperature field T1, from T0. Presently, kappaX and kappaY are equal but they can be given asymmetric values in order to represent asymmetric diffusion in the X and Y directions. When the model run, try to impose a kappaY ten times smaller to see what happens.

```
% Solving temperature equation by explicit method
for cycle = 0:25
    for xn = 1:xnum
        for yn = 1:ynum
            % Boundary nodes:
            if (xn==1 | xn==xnum | yn==1 | yn==ynum)
                T1(xn,yn)=tmin; % T=const
            else
                kappaX=kappa;
                kappaY=kappa;
                T1(xn,yn)=T0(xn,yn)+timestep*kappaX*( T0(xn-1,yn)-
2.0.*T0(xn,yn)+T0(xn+1,yn) )/(xstp*xstp); %x contribution
                T1(xn,yn)=T1(xn,yn)+timestep*kappaY*( T0(xn,yn-1)-
2.0.*T0(xn,yn)+T0(xn,yn+1) )/(ystp*ystp); %y contribution
            end
        end
    end
    T0=T1;% Reloading solutions to T0

     % Creating vectors for x and y axes
     x = xmin:xstp:xmax;
     y = ymin:ystp:ymax;
     x=x/1000.;% renormalizes m in km, only for plotting
     y=y/1000.;% renormalizes m in km, only for plotting

     % Visualizing results
     % Ploting RO as colormap
     figure(1);
```

```
        surf(x,y,T0);
        view (0,90);
        caxis([tmin tmax]);
        colorbar;
        title(['Temperature, K For time step ',num2str(cycle)])
        xlabel('X, km')
        ylabel('Y, km')
end
```

Exercises

1) The timestep used above is almost the largest one that produces stable reliable solution. Try to run the code with a greater timestep and to see how instability develops.

2) Try different kappaX and kappaY. For example kappaY ten times smaller then kappaX, and see what happens.

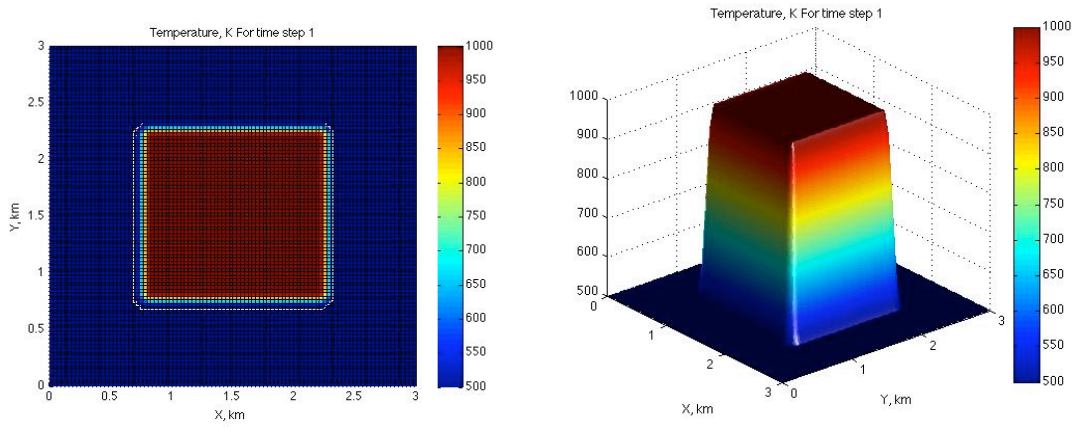
3) You might want to try to see how the system evolves with a smaller inclusion. Go in the initialization loop and change the thermal anomaly size from half of the box to a smaller value. For example if you want to have an inclusion that has a size $1/10^{th}$ of the box, try:

```
        if ((xn-1)/(xnum-1)>0.45 & (xn-1)/(xnum-1)<0.55 & (yn-
1)/(ynum-1)>0.45  & (yn-1)/(ynum-1)<0.55)
                T0(xn,yn)=tmax; % square hot body in the middle
        end
```

A smaller inclusion equalizes faster then a bigger one. How fast? You can follow better the peak temperature of the inclusion using a 3D projection in which the z axis is proportional to the temperature. In order to do so, just replace the view(…,…) line with the following:

```
light; lighting phong; view (50,30); shading interp;
```

The difference between the two views should look similar to the following figures:



4) Finally we want to use this code for a practical purpose. This program explicitly calculates the thermal diffusion of an initial square body of 1000K immersed in a matrix at 500K. For example it might represent a hot magma inclusion immobilized in the middle of the upper crust, so it can be used for estimating the time necessary to thermally equalize the system.

We can consider the inclusion equalized when the thermal anomaly has become only 10% of the

initial one, i.e. when the initial temperature is reduced from 1000K down to 550K that corresponds to a differential temperature that is reduced from 500K down to 50K. Use the small inclusion that you have already tested (10% of the box) because it takes less time to reach equalization. Such an inclusion corresponds to a size of about 300m (see the box definition).

In order to achieve the necessary equalization you cannot increase the timestep because this would trigger instabilities in the code. Therefore you must reise the number of iterations (see "for cycle= 0:25") and substitute the maximum iteration with a greater number, e.g. 100 or 200 or more until your thermal anomaly has reached the equilibration, i.e. its maximum value has gone below 550K. Use the three dimensional visualization to detect precisely the exact temperature peak.

Now, using this last simulation you can estimate the time necessary for equalizing a body of 300m of diameter that has initially 1000K, when immersed in a crust at 500K. Use the number of timesteps and the length of one timestep. Convert this time from seconds to years.

Not due. Just if you have extratime: *you can try an inclusion of 600m and see if the time necessary for the equalization is 2 or 4 times or more of the time necessary for the 300m inclusion. Do you guess why?*

Report structure

Write a short discussion of your results and answer the questions above. You can save Matlab images as jpg files and place them into a MS Word doc. Make sure all figure are labelled properly (axis labels, title). Submit a MS Word .doc or .pdf file by email to kmat3920@usyd.edu.au.

Your report should have the following structure and be no longer than 4 pages including figures:

Title
The title must be concise and informative. It should state what the report is about, not simply the name of the assignment.

Introduction
A short paragraph describing which problems were investigated.

Methodology
A short paragraph describing the methodology.

Results
Results of the exercise.

Conclusions