TUTORIAL 03:
NUMERICAL ASPECT I: FINITE DIFFERENCES

In this tutorial, we will model the temperature evolution in a case study that reduces to solve the one-dimensional diffusion equation, also often referred to as a heat equation.

1: The 1D diffusion equation

→ From CROCO 3D temperature equation:

$$\frac{\partial T}{\partial t} + \boldsymbol{u}\nabla T = \nabla_h(K_{Th}\nabla_h T) + \frac{\partial}{\partial z}\left(K_{Tv}\frac{\partial T}{\partial z}\right)$$

➥ We simplify the processes at work by studying a simple case study, where:

- there is no surface forcing (adiabatic).
- there is no current in the swimming pool, i.e. we can cross-out the non-linear advection terms ($\boldsymbol{u}\nabla T$).
- there is no variation of temperature with depth (barotropic case), i.e. we can cross-out the vertical turbulent diffusion term.
- the horizontal diffusion coefficient (K_{Th}) is constant.

→ From the 3D temperature, we need to solve the 1D diffusion equation:

$$\frac{\partial T}{\partial t} = K_{Tv}\frac{\partial^2 T}{\partial x^2} \qquad x \in [O, L], \qquad t \in [0, T] \quad (1)$$

➥ With only a first-order derivative in time, only one initial condition is needed: $T(x, 0) = I(x)$.

➥ Contrastingly the second-order derivative in space leads to a demand for two boundary conditions.

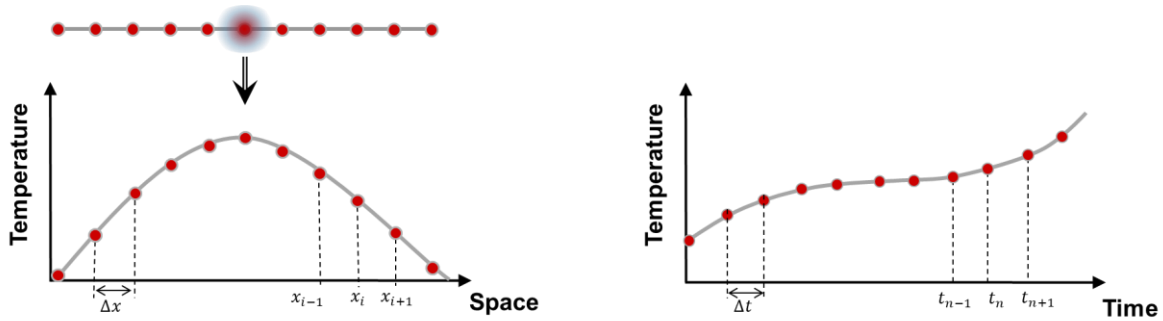
➥ The parameter K_{Tv} must be given and is referred to as the diffusion coefficient.

📖 Partial Diffusion Equations (PDE) like this one have a wide range of applications throughout physical, biological, and financial sciences. One of the most common applications is propagation of heat, where $T(x, t)$ represents the temperature of some substance at point x and time t.

2: Spatial and temporal discretisation

→ The first step in the discretization procedure is to replace the domain $[0, L] \times [0, T]$ by a set of mesh points. Here we apply equally spaced mesh points

$$x_i = i\Delta x, i = 1, \ldots, N_x \quad \text{and} \quad t_n = n\Delta t, n = 1, \ldots, N_t$$



→ The next step is to replace the derivatives by finite difference approximations. The computationally simplest method arises from using a forward difference in time and a central difference in space:

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\Delta t} \qquad \frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

→ Using this ***forward Euler scheme***, we turn the 1D diffusion equation into a discrete equation, where everything at time step n is known, such that T_i^{n+1} is the only unknown. The formulation is explicit, such that:

$$T_i^{n+1} = T_i^n + K_{Tv}\frac{\Delta t}{\Delta x^2}(T_{i+1}^n - 2T_i^n + T_{i-1}^n) \quad (2)$$

3: Computational algorithm

→ The computational algorithm consists of the following steps:

❶ Initialise the vector state variable ($T_i^1, i = 1, \dots, N_x$),

❷ Plot T^1 at every point in the domain,

for $n = 1, \dots, N_t$

❸ Apply equation (2) for all the internal points, $i = 2, \dots, N_x - 1$,

❹ Set the boundary values for $i = 1$ and $i = N_x$ (T_1^{n+1} and $T_{N_x}^{n+1}$),

❺ Plot T^2 at every point in the domain,

➢ Rince and repeat (steps ❸ ❹ ❺).

4: Your turn

→ To solve the 1D diffusion equation, we will use **MATLAB** on the Lengau cluster.

→ Log onto the Lengau cluster by executing the following command from a terminal/konsole:

```
ssh -X login@lengau.chpc.ac.za
```

Replace **login** with your corresponding account number.

→ Reserve one interactive processor to use Matlab (Step 4 from **#TUTORIAL01**):

```
[login@login2 ~]$ qsubi1
[login@cnode0220 ~]$
```

→ Go into your lustre directory and create a directory for this hands-on session: (**lustre/diff**):

```
[login@cnode0220 ~]$ cd lustre
[login@cnode0220 lustre]$ mkdir diff
[login@cnode0220 lustre]$ cd diff
[login@cnode0220 diff]$
```

→ Copy a template of the computational algorithm

```
[login@cnode0220 diff]$ cp /mnt/lustre/users/sillig/
CROCO_TRAINING_Week1/3_Some_files/My_diffusion_1D.m .
[login@cnode0220 diff]$
```

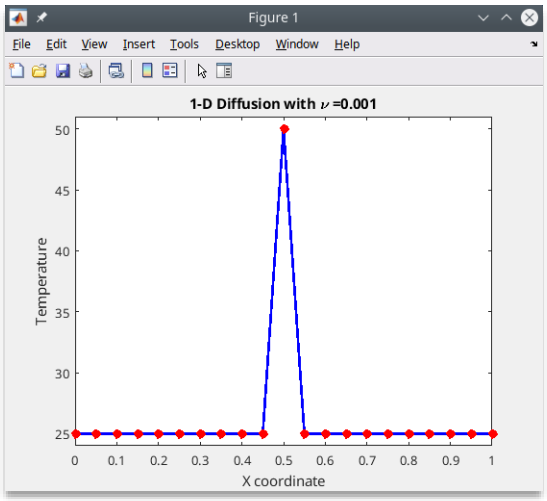
→ Start Matlab with the command **matlab -nodesktop** (or the alias **mat**):

```
[login@cnode0220 diff]$ matlab -nodesktop &
[login@cnode0220 diff]$
```

→ Edit **My_diffusion_1D.m** using the Matlab command edit:

```
>> edit My_diffusion_1D.m
>>
```

→ The Matlab script is listed in the following page. You have to complete this script at lines 29, 53 and 56 (see ➤➤➤ below). Here are the different parts of the algorithm:



➥ Line 14: K is the constant horizontal diffusion coefficient.

➥ Lines 16-18: the length of the swimming pool is descetized into 21 equaly-spaced points.

➥ Lines 20-21: We begin at $n = 1$, with $\Delta t = 0.1s$.

➥ Line 25: The swimming temperature pool is initialized at 25°C, everywhere. This corresponds to step ❶ of the computationsal algorithm (see **#3**).

➤➤➤ Complete line 29, to initialise (❶) the 11th spatial point at **50°C**.

➥ Lines 32-38: This is a plot (❷) of the temperature in the swimming pool.

➤➤➤ At line 53, start by applying equation (2) for the 11th spatial point (❸), such that:
`temperature_new(11)= temperature(11)+...`

↳ You can define a coefficient `alpha`, such that $\alpha = K\Delta t/\Delta x^2$

➤➤➤ At line 53, apply equation (2) for all the internal points, $i = 2, \dots, N_x - 1$ (❸) using a **for loop** (`for i=2,nx-1`)

➤➤➤ At line 56, apply the boundary conditions at $i = 1$ and $i = N_x$ (❹). Either the temperature at these points remains at 25°C, or you can copy the temperature of the closest internal point.

➤➤➤ When it is working, increase the number of time steps nt (at line 20)

➤➤➤ Decrease and the increase the time step dt (line 21). What do you observe?

5: Exiting

→ When you are done, exit Matlab and logout from the cluster:

```
[login@cnode0220 Run_Clim]$ exit
logout
qsub: job 4416950.sched01 completed
[login@login2 ~]$ exit
```

6: Matlab template script

```
% Simulating the 1-D Diffusion equation by the Finite Difference Method
% Numerical scheme used is a first order upwind in time and a second order
...central difference in space (both Implicit and Explicit)

% The equation is :
% dT/dt = D x laplacian(T)
% F is the Diffusion coefficient

%%
clear all;

%%
% Parameters
K=0.001;          % Diffusion coefficient/viscosity

nx=21;              % Number of steps in space(x)
dx=1/(nx-1);        % Width of space step
x=0:dx:1;           % Range of x (0,1) and specifying the grid points

nt=1;              % Number of time steps
dt=0.1;             % Width of each time step

%%
% Initial Conditions
temperature=zeros(nx,1)+25.; %--> 25°C everywhere

%%
% Warm or cool a little bit at point 11
% ---> Complete HERE

%Plotting the initial temperature profile
figure(1)
plot(x,temperature,'-bo','LineWidth',2,'MarkerFaceColor','r','MarkerEdgeColor','r','MarkerSize',6);
axis([0 1 24 51]);
title({['1-D Diffusion with \nu =',num2str(vis)]});
xlabel('X coordinate');
ylabel('Temperature');
drawnow;

%%
% Go ahead : loop over time
temperature_new=temperature;
for it=1:nt
    %Plotting the temperature profile
    plot(x,temperature,'-bo','LineWidth',2,'MarkerFaceColor','r','MarkerEdgeColor','r','MarkerSize',6);
    axis([0 1 24 51]);
    title({['1-D Diffusion with \nu =',num2str(vis)];['time(\itt) = ',num2str(dt*it)]});
    xlabel('X coordinate');
    ylabel('Temperature');
    drawnow;

    % Solve the problem :
    % ---> Complete HERE (temperature(i)=...)

    % Boundaries conditions
    % ---> Complete HERE


    % Update temperature vector
    temperature=temperature_new;

    pause(0.1)
end
```