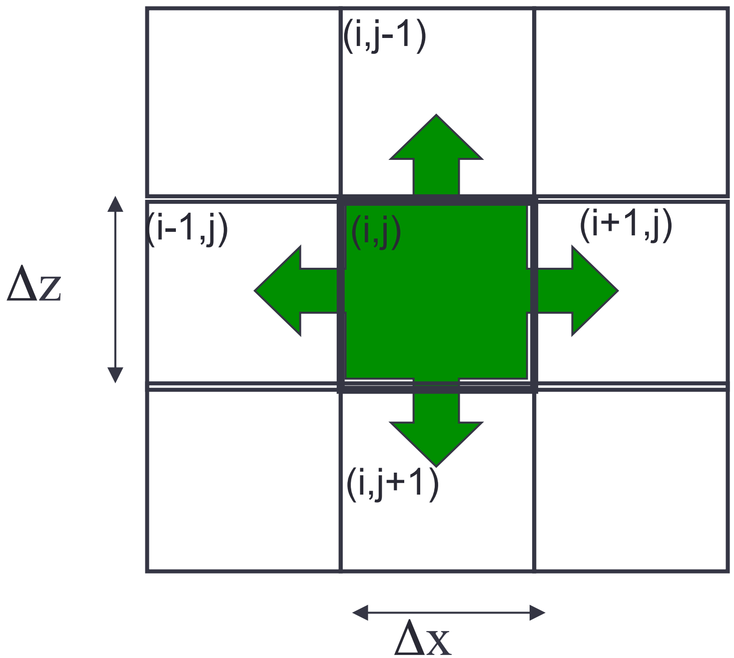
SUBITOP Understanding subduction zone topography through modelling of coupled shallow and deep processes

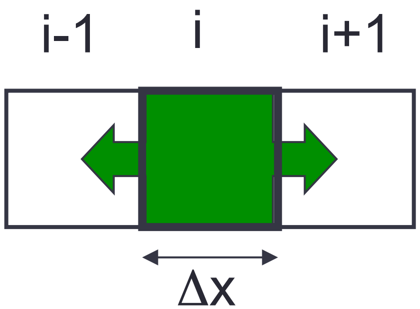
More dimensions

- Discretization
- Stable 2-D timestepping
- Advection-diffusion



Heat diffusion in 1-D: Euler forward

$$\rho C_P \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x} \right)$$



For constant k, C_p, k: $$\frac{\partial T}{\partial t} = \kappa \frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x} \right)$$

In Python:

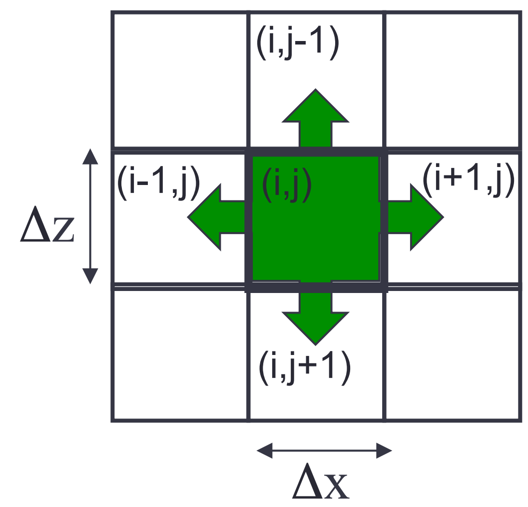
```
df = kappa*dt*(fin[2:]-2*fin[1:-1]+fin[0:-2])/dz**2
```

or

```
df=kappa*dt*np.diff(fin,n=2)/dz**2
```

Heat diffusion in 2-D: Euler forward

$$\rho C_P \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial z}\left(k \frac{\partial T}{\partial z}\right)$$



For constant k, C_p, k:

$$\frac{\partial T}{\partial t} = \kappa\left[\frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right)\right]$$

In Python:

Heat diffusion in 2-D: Euler forward

For constant k, C_p, k:

$$\frac{\partial T}{\partial t} = \kappa\left[\frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right)\right]$$

In Python:

```
df = kappa*dt*
  ( (fin[1:-1,2:]-2*fin[1:-1,1:-1]+fin[1:-1,0:-2])/dx**2
  +(fin[2:,1:-1]-2*fin[1:-1,1:-1]+fin[0:-2,1:-1])/dz**2 )
```

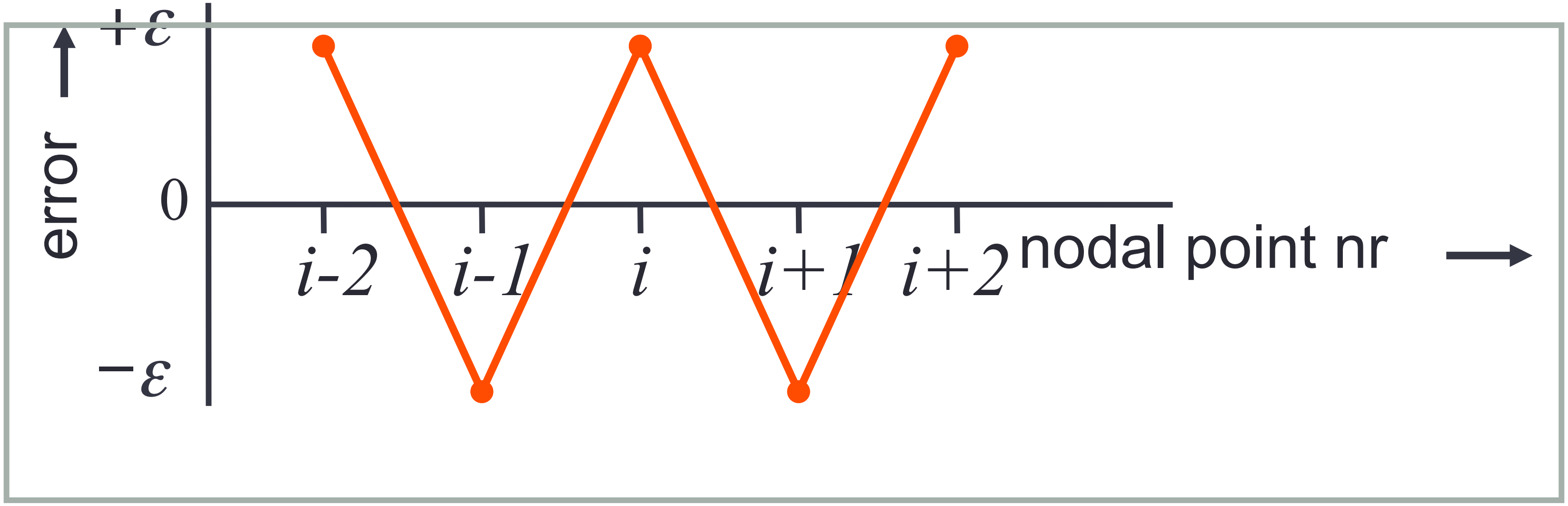
or

```
d2fdx2 = np.diff(fin,n=2,axis=1)/dx**2
d2fdz2 = np.diff(fin,n=2,axis=0)/dz**2
df = kappa*d2fdx2[1:-1,:]+kappa*d2fdz2[:,1:-1]
```

Stability criterion for 1-D heat diffusion

$$\varepsilon_i^{new} = r\varepsilon_{i+1}^{old} + (1-2r)\varepsilon_i^{old} + r\varepsilon_{i-1}^{old} \quad \text{with} \quad r = \frac{\kappa \Delta t}{\Delta x^2}$$

Worst case error scenario:



— $\varepsilon_i^{old} = -\varepsilon_{i-1}^{old} = -\varepsilon_{i+1}^{old}$ so that $\varepsilon_i^{new} = (1-4r)\varepsilon_i^{old}$

SUBITOP Understanding subduction zone topography through modelling of coupled shallow and deep processes

Stability criterion for 1-D heat diffusion

- $\varepsilon_i^{new} = (1-4r)\varepsilon_i^{old}$ with: $r = \dfrac{\kappa\Delta t}{\Delta x^2}$
- Avoiding amplification: $|1-4r| < 1$
- i.e.: $-1 < 1-4r$ or $r < \dfrac{1}{2}$ or
- So the 1-D forward Euler heat diffusion equation has following stability criterion:

$$\Delta t < \frac{\Delta x^2}{2\kappa}$$

Stability criterion for 2-D heat diffusion

For a uniform grid and κ = constant:

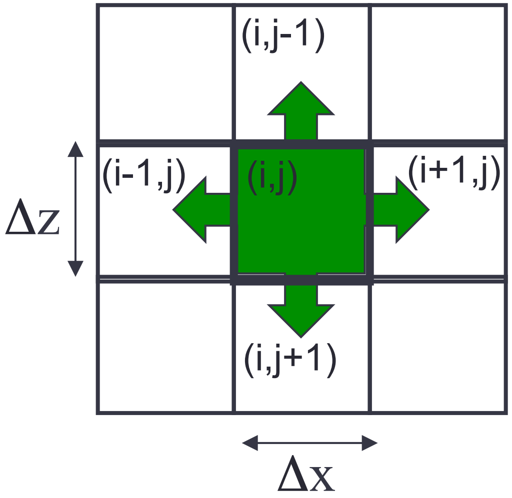
$$\frac{T_{i,j}^{new} - T_{i,j}^{old}}{\Delta t} = \kappa\left(\frac{T_{i+1,j}^{old} - 2T_{i,j}^{old} + T_{i-1,j}^{old}}{\Delta x^2} + \frac{T_{i,j+1}^{old} - 2T_{i,j}^{old} + T_{i,j-1}^{old}}{\Delta z^2}\right)$$

If $\Delta x = \Delta z = \Delta$:

$$T_{i,j}^{new} - T_{i,j}^{old} = r\left(T_{i+1,j}^{old} + T_{i,j+1}^{old} - 4T_{i,j}^{old} + T_{i-1,j}^{old} + T_{i,j-1}^{old}\right)$$

or:

$$T_{i,j}^{new} = rT_{i+1,j}^{old} + rT_{i,j+1}^{old} + (1-4r)T_{i,j}^{old} + rT_{i-1,j}^{old} + rT_{i,j-1}^{old}$$



Stability criterion for 2-D heat diffusion

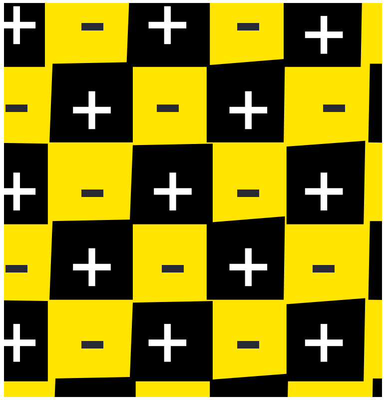
$$\varepsilon_{ij}^{new} = r\varepsilon_{i+1,j}^{old} + r\varepsilon_{i,j+1}^{old} + (1-4r)\varepsilon_{i,j}^{old} + r\varepsilon_{i-1,j}^{old} + r\varepsilon_{i,j-1}^{old}$$

with $r = \dfrac{\kappa \Delta t}{\Delta x^2}$

- Taking again the worst case scenario:

$$\varepsilon_{i,j}^{old} = -\varepsilon_{i-1,j}^{old} = -\varepsilon_{i+1,j}^{old} = -\varepsilon_{i,j-1}^{old} = -\varepsilon_{i,j+1}^{old}$$

- …



Stability criterion for 2-D heat diffusion

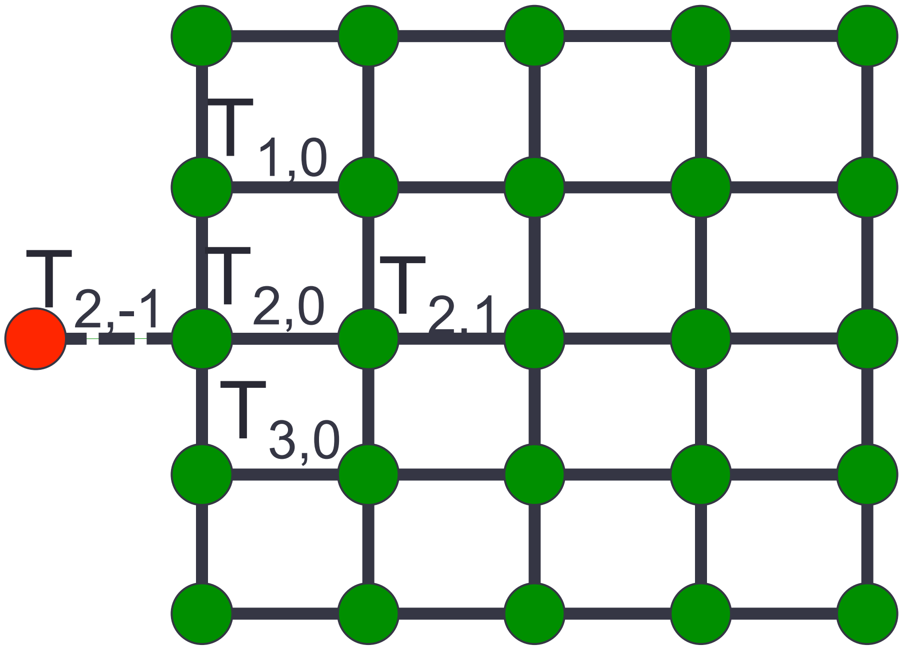
- $\varepsilon_i^{new} = (1-8r)\varepsilon_i^{old}$ with: $r = \dfrac{\kappa \Delta t}{\Delta x^2}$
- Avoiding amplification: $|1-8r| < 1$
- I.e.: $-1 < 1-8r$ or $r < \dfrac{1}{4}$ or $\dfrac{\kappa \Delta t}{\Delta x^2} < \dfrac{1}{4}$
- So the 1-D forward Euler heat diffusion equation has following stability criterion:

$$\Delta t < \frac{\Delta x^2}{4\kappa}$$

Natural boundary conditions in 2-D: Example for left boundary

$$\frac{\partial T}{\partial t} = \kappa\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2}\right)$$

$$\frac{dT}{dx} = c$$



$$T_{2,0}^{new} = T_{2,0}^{old} - \frac{\kappa \Delta t}{\Delta^2}\left(T_{2,-1} + T_{1,0} + T_{2,1} + T_{3,0} - 4T_{2,0}\right)$$

$$T_{2,-1} = T_{2,1} - 2c\Delta$$

Implementation issues

- Note the suggested index convention: T_{ij} → `T[j,i]`

  (so `T[`*row-nr*, *column-nr*`]`)

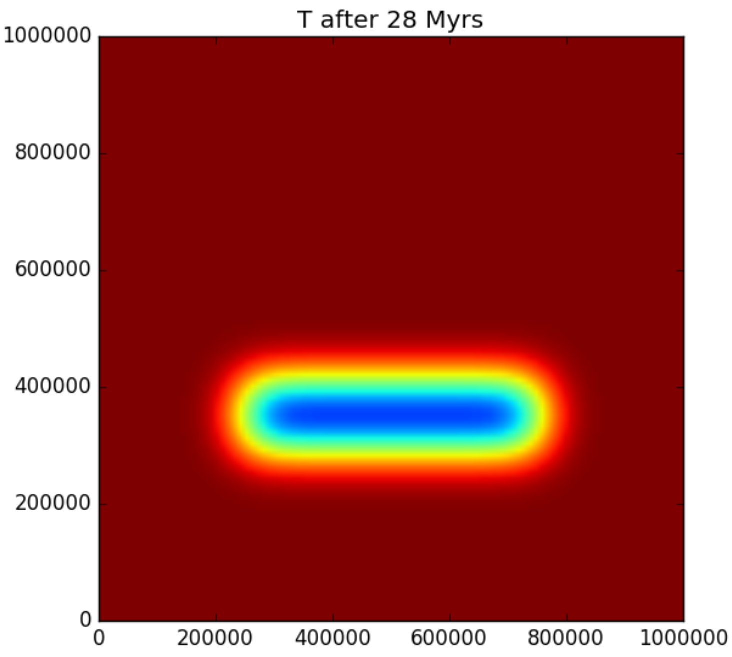
- So `np.diff(fin,n=1,axis=0)` calculates difference between subsequent rows (i.e. in z-direction), `np.diff(fin,n=1,axis=1)` calculates difference between subsequent columns (i.e. in x-direction)

- Defining a 2-D grid:

```
x       = np.linspace(0,w,nx)
z       = np.linspace(0,h,nz)
[xx,zz]= np.meshgrid(x,z)
```

Practical 3, Part 1:

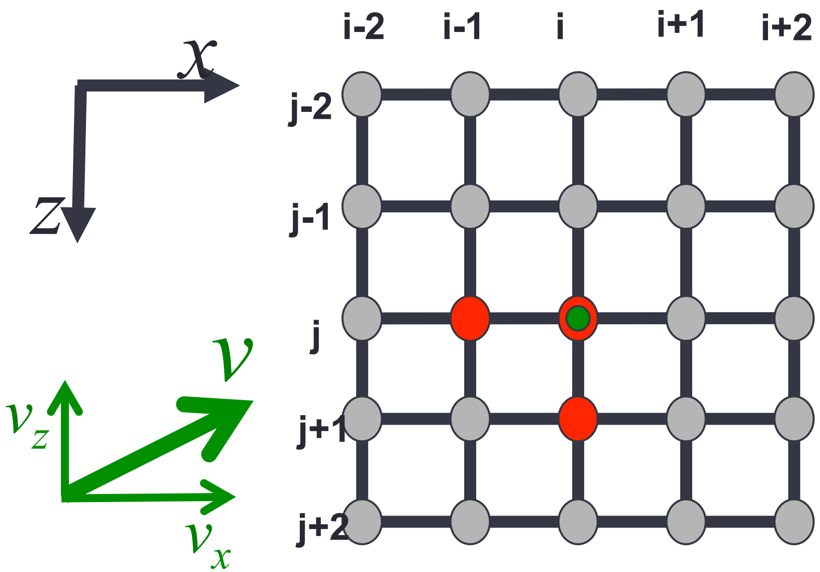
- ❑ Implement a 2-D heat diffusion solver
- ❑ Add natural boundary conditions



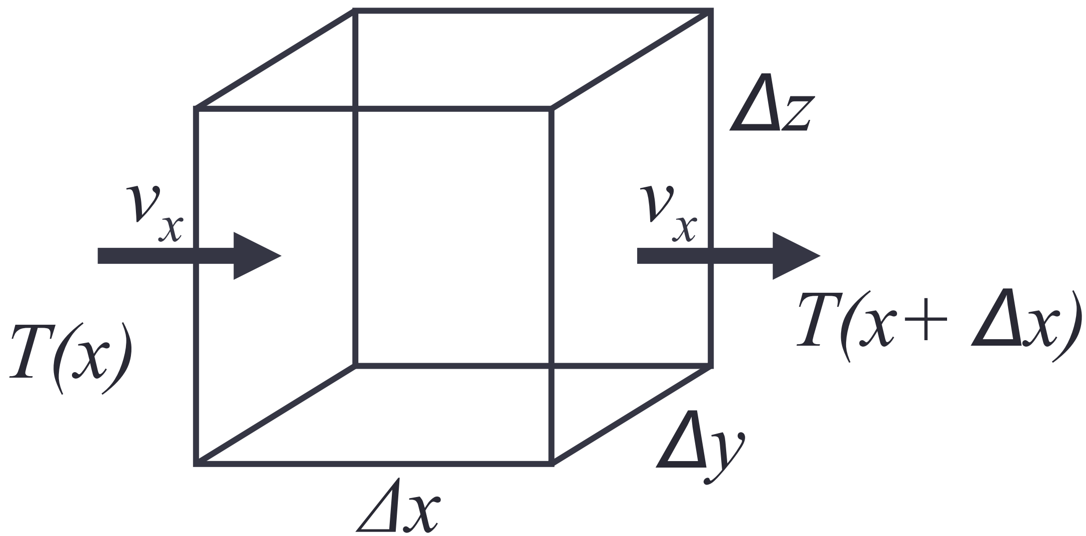
https://community.dur.ac.uk/jeroen.van-hunen/Subitop/session3.html

Advection-diffusion

- ❑ Advection schemes
- ❑ Advection-diffusion schemes
- ❑ Implementation



Heat advection



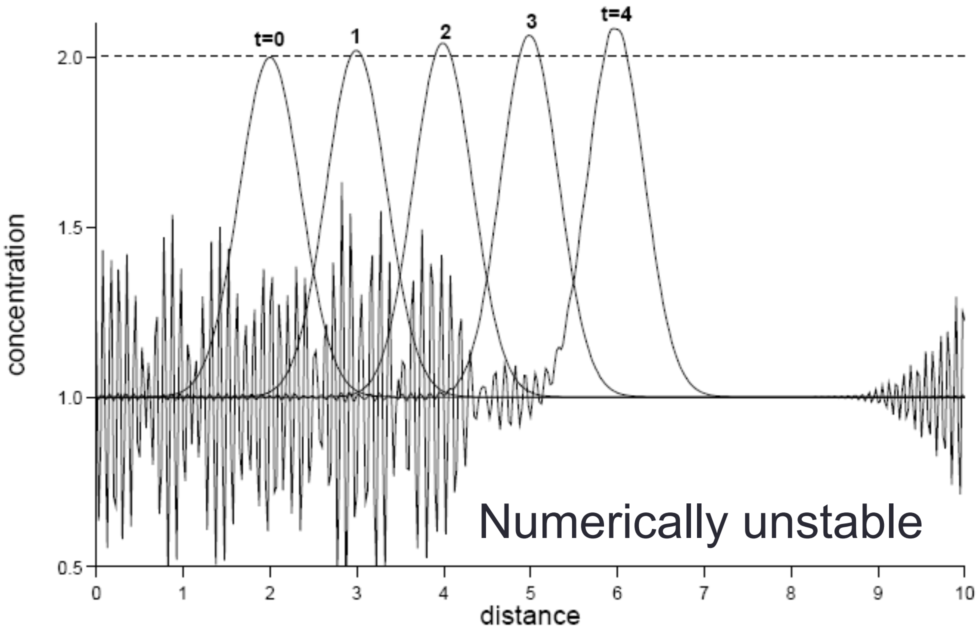
If inflowing material is hotter than outflowing (so $\frac{\partial T}{\partial x}$ <0)→ flow 'carries in' heat → T will rise:

$$\frac{\partial T}{\partial t} = -v_x \frac{\partial T}{\partial x}$$

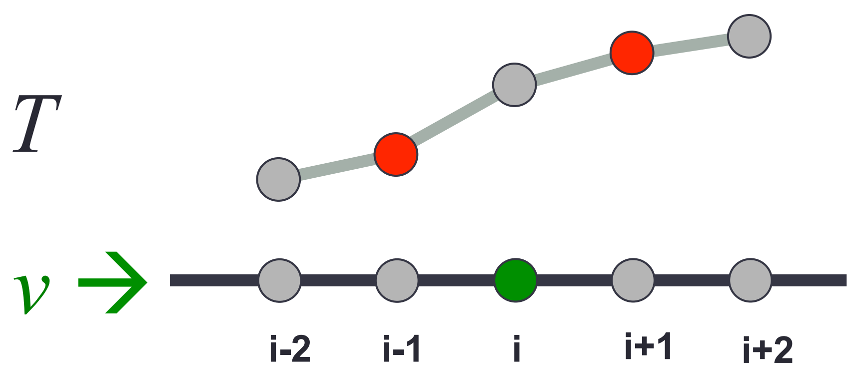
Advection: numerical methods in 1D

1. FTCS: Forward-Time-Central-Space

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



(Spiegelman, 2004)



Advection: numerical methods in 1D

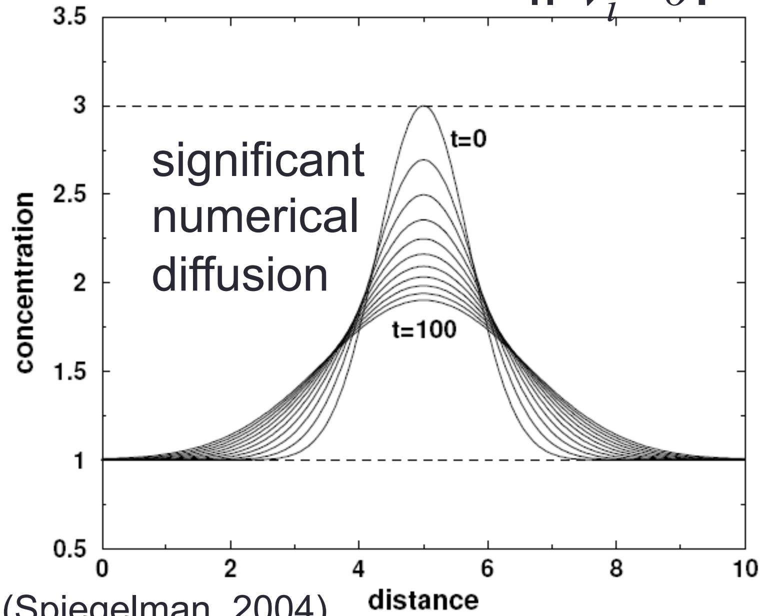
1. FTCS: Forward-Time-Central-Space

2. Upwinding: **if $v_i>0$:**

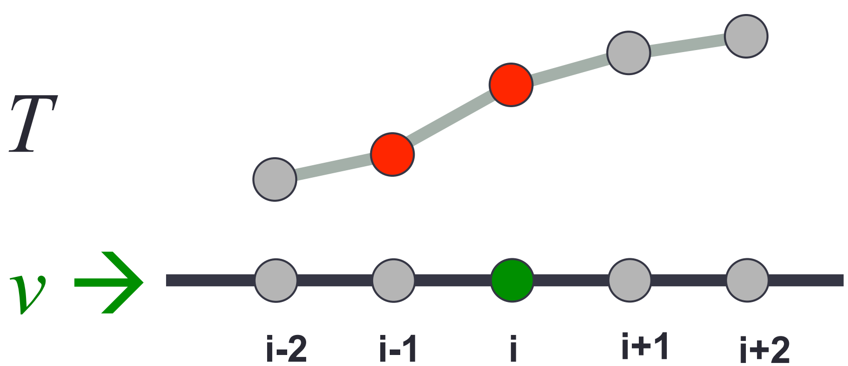
$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = -v_i \frac{T_i^n - T_{i-1}^n}{\Delta x}$$

if $v_i<0$:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = -v_i \frac{T_{i+1}^n - T_i^n}{\Delta x}$$



(Spiegelman, 2004)



Advection: numerical methods in 1D

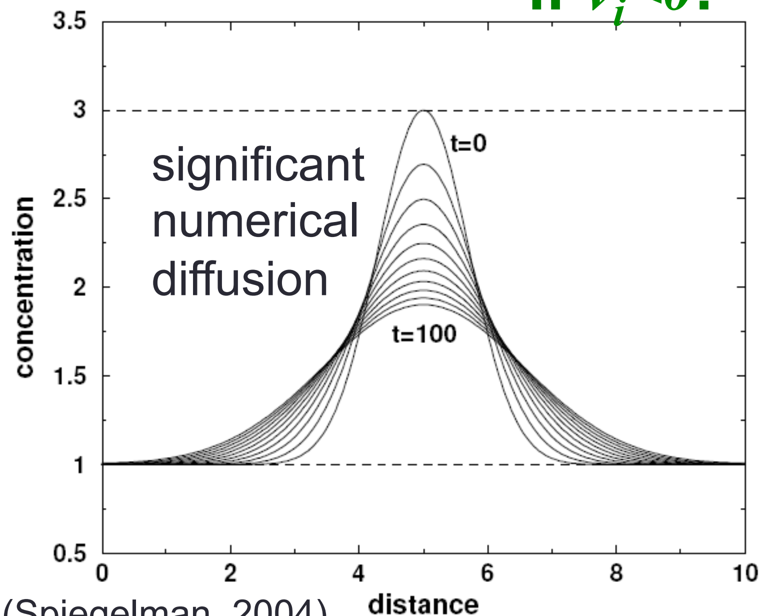
1. FTCS: Forward-Time-Central-Space

2. Upwinding: if $v_i>0$:

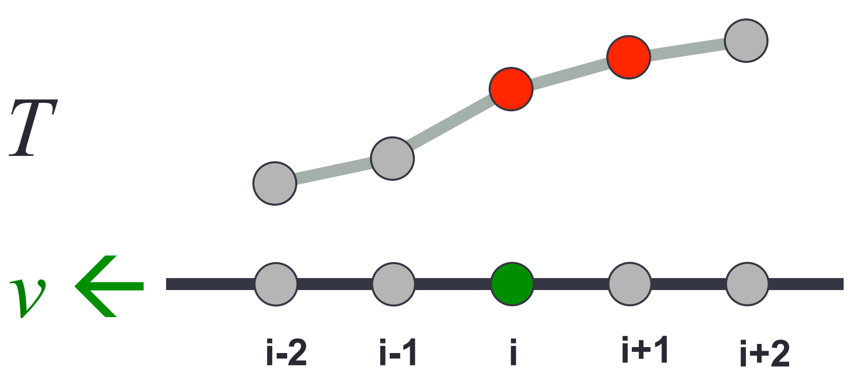
$$\frac{T_i^{n+1}-T_i^n}{\Delta t} = -v_i \frac{T_i^n - T_{i-1}^n}{\Delta x}$$

if $v_i<0$:

$$\frac{T_i^{n+1}-T_i^n}{\Delta t} = -v_i \frac{T_{i+1}^n - T_i^n}{\Delta x}$$



(Spiegelman, 2004)

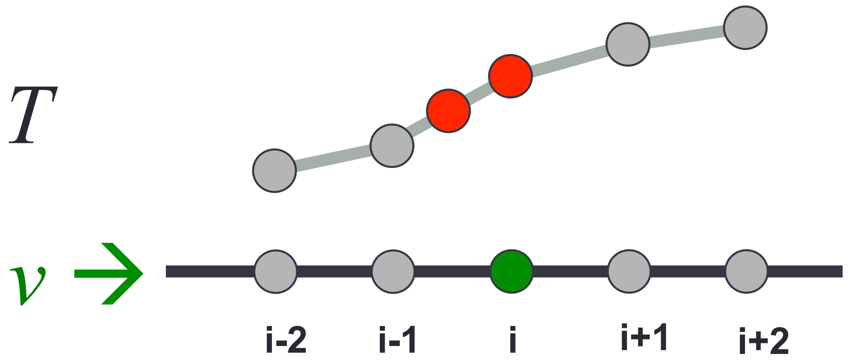


Advection: numerical methods in 1D

1. FTCS: Forward-Time-Central-Space

2. Upwinding

3. Semi-Lagrangian:
   - Use velocity field* to find location X where T_i advected from in Δt
   - Interpolate T from points T_{i-1} and T_i to X
   - Copy T into T_i



* Better velocity field can be found iteratively.

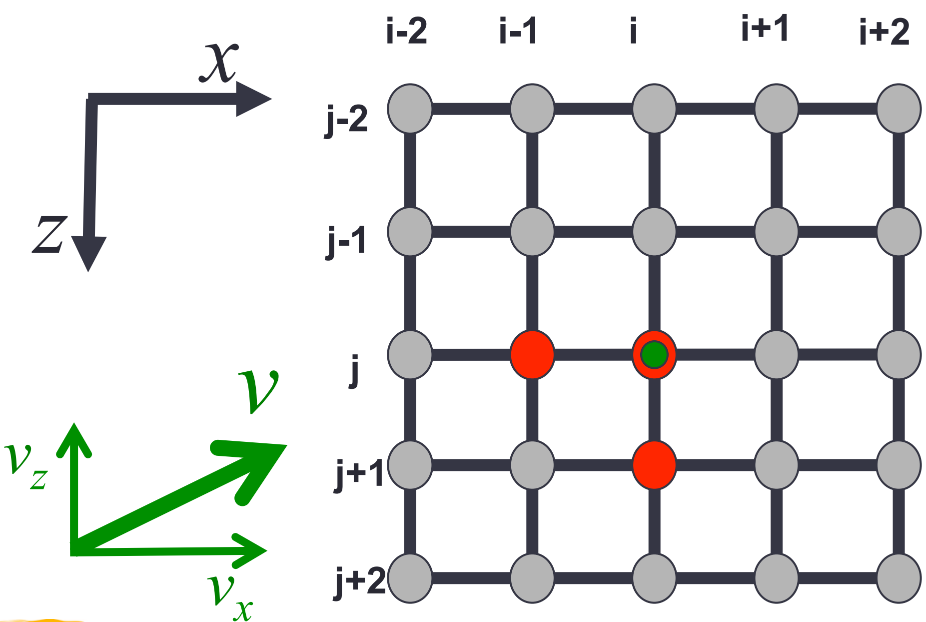
Advection: numerical methods in 1D

1. FTCS: Forward-Time-Central-Space
2. Upwinding
3. Semi-Lagrangian
4. (fully) Lagrangian: trivial
   - Lagrangian code (mesh moves with flow)
   - Particles

Advection: numerical methods in 2D

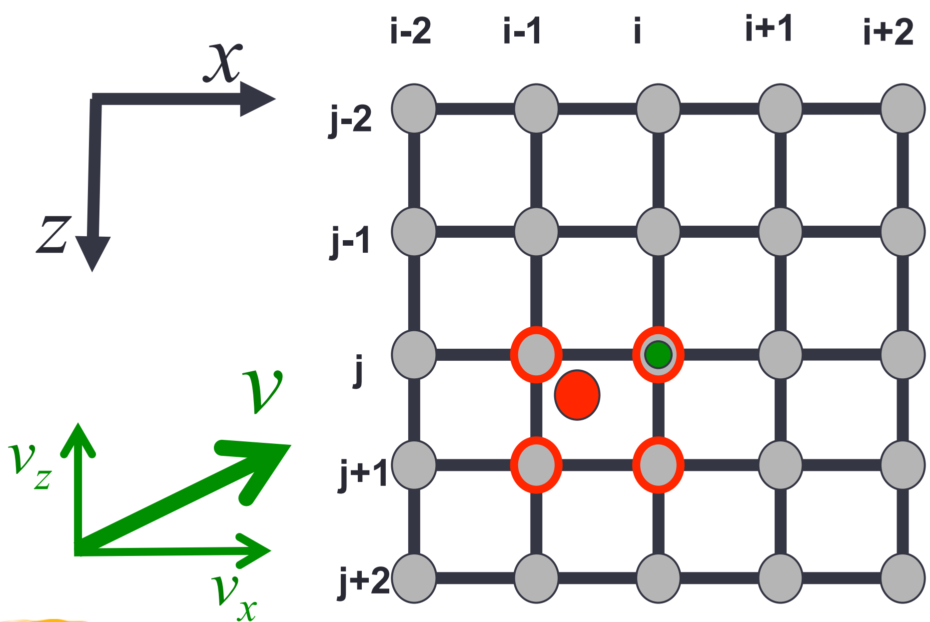
2. Upwinding: example:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = -v_{x,i} \frac{T_{i,j}{}^n - T_{i-1,j}{}^n}{\Delta x} - v_{z,i} \frac{T_{i,j+1}{}^n - T_{i,j}{}^n}{\Delta z}$$



Advection: numerical methods in 2D

2. Semi-Lagrangian: example:



Courant time step criterion for upwinding

$$\frac{\left(T_i^{n+1} + \varepsilon_i^{n+1}\right) - \left(T_i^n + \varepsilon_i^n\right)}{\Delta t} = -|v_i| \frac{\left(T_i^n + \varepsilon_i^n\right) - \left(T_{i-1}^n + \varepsilon_{i-1}^n\right)}{\Delta x}$$

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = -|v_i| \frac{T_i^n - T_{i-1}^n}{\Delta x}$$

—

$$\frac{\varepsilon_i^{n+1} - \varepsilon_i^n}{\Delta t} = -|v_i| \frac{\varepsilon_i^n - \varepsilon_{i-1}^n}{\Delta x}$$

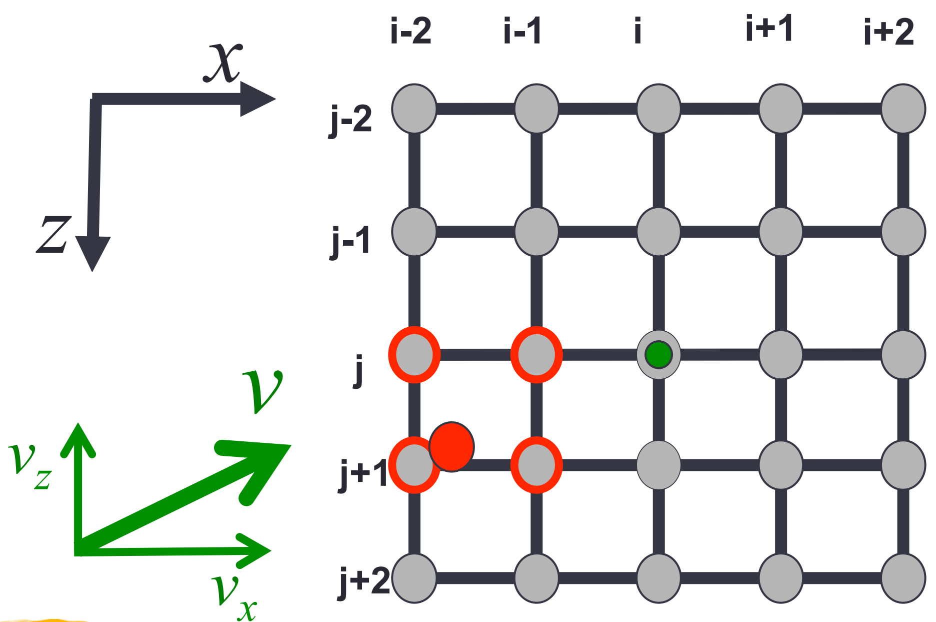
For $\varepsilon_i^{n+1} < \varepsilon_i^n$ and $\varepsilon_i^{n+1} = -\varepsilon_{i-1}^n$:

In 1-D: $-1 < 1 - \frac{2\Delta t|v|}{\Delta x} < 1$ or $\Delta t < \frac{\Delta x}{|v|}$

In 2-D: $\Delta t < \left(\frac{|v_x|}{\Delta x} + \frac{|v_z|}{\Delta z}\right)^{-1}$

No time step criterion for semi-Lagrangian method

2. Upwinding: time step needs to be smaller than advection time over 1 grid cell
3. Semi-Lagrangian: time step can be larger than advection time over 1 grid cell



Advection-diffusion equation

In 1-D:

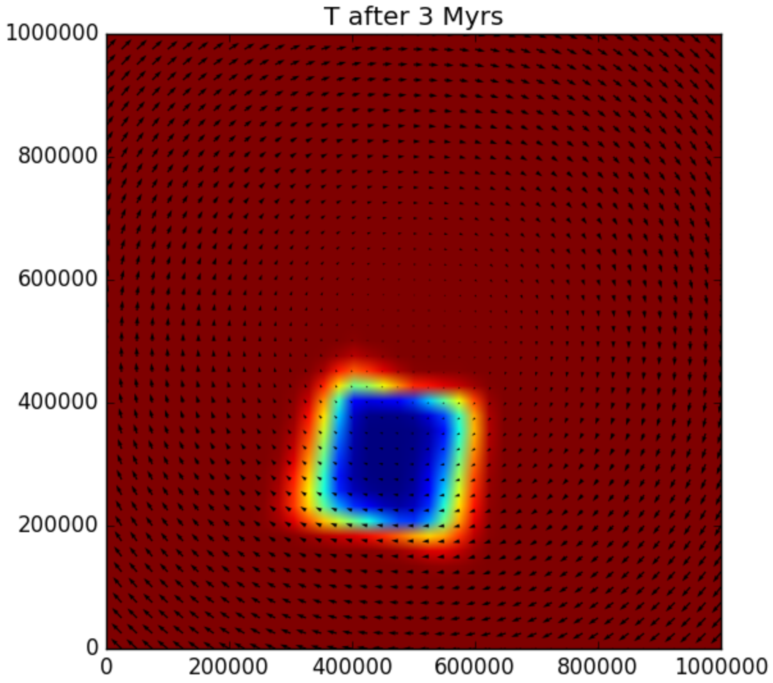
$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} - v_x \frac{\partial T}{\partial x}$$

In 2-D:

$$\rho C_p \frac{\partial T}{\partial t} = \kappa \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2} \right) - v_x \frac{\partial T}{\partial x} - v_z \frac{\partial T}{\partial z}$$

Practical 3, Part 2:

- Implement a 2-D heat advection-diffusion solver



https://community.dur.ac.uk/jeroen.van-hunen/Subitop/session3.html