FV scheme for conservation law: $u_t + \mathcal{F}_x = \mathcal{S}$ (in 1D)

$x \in \Omega = (a,b)$, $t > 0$

$u(x,t)$: conserved quantity per unit volume at x, at time t

$\mathcal{F}(x,t)$: flux of u: amount crossing a unit area per unit time

$\mathcal{S}(x,t)$: source of u: amount generated (or lost) per unit volume per unit time

... will derive this later ...

Discetize space $[a,b]$ into M control volumes V_i, $i = 1:M$

» time $[0, t_{end}]$ into N_{max} timesteps t_n, $n = 0:N_{max}$

integrating the PDE over each V_i and over $[t_n, t_{n+1}]$ ... details later ...

explicit FV scheme: $$U_i^{n+1} = U_i^n + \frac{\Delta t}{\Delta x_i}\left[F_{i-\frac{1}{2}}^n - F_{i+\frac{1}{2}}^n \right] + \Delta t\, S_i^n$$

$i = 1:M$, $n = 0, 1, 2, \ldots, N_{max}$

where

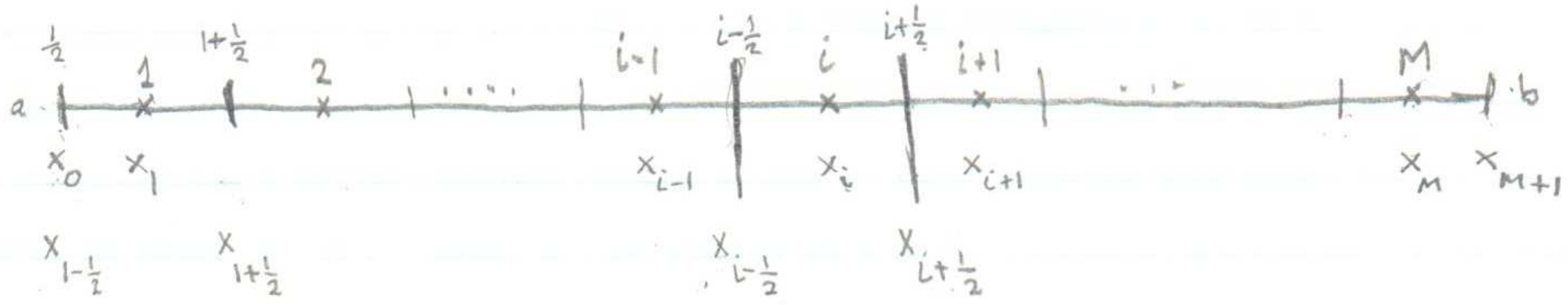
U_i^n = mean value of u over V_i at time t_n $= \frac{1}{V_i}\int_{V_i} u(x, t_n)\,dx \approx u(x_i, t_n)$

$F_{i-\frac{1}{2}}^n$ = mean flux during $[t_n, t_{n+1}]$ across face $x_{i-\frac{1}{2}}$ $= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} F(x_{i-\frac{1}{2}}, t)\,dt$

S_i^n = mean source over V_i and timestep $= \frac{1}{\Delta t}\frac{1}{V_i}\int_{t_n}^{t_{n+1}}\int_{V_i} S(x,t)\,dx\,dt$

$\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ = volume of $V_i = \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$

$\Delta t = t_{n+1} - t_n$ = timestep

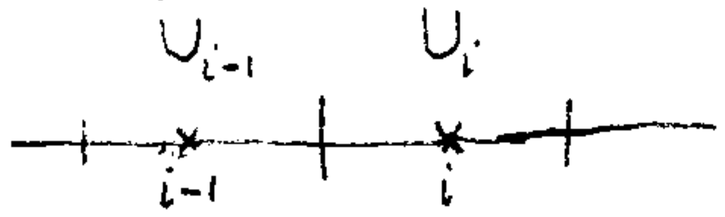


arrays: $X(0:M+1)$, $U(0:M+1)$, $F(1:M+1)$ no time-index

Explicit scheme – CFL condition for diffusion

Diffusion: $\mathcal{F} = -D u_x$ (Fick's Law, Fourier Law)

Approximation: $\mathcal{F} \approx F_{i-\frac{1}{2}} = -D_{i-\frac{1}{2}} \dfrac{U_i - U_{i-1}}{x_i - x_{i-1}}$



$= -D \dfrac{U_i - U_{i-1}}{\Delta x}$ for uniform mesh, const. D
$i = 2:M$ (internal fluxes)

This is a major approximation, for expediency, simplicity:

we interpret <u>mean</u> values U_i <u>as nodal</u> values (at x_i)

Explicit FV scheme for $u_t + \mathcal{F}_x = 0$:

$$U_i^{n+1} = U_i^n + \frac{\Delta t}{\Delta x}\left[F_{i-\frac{1}{2}}^n - F_{i+\frac{1}{2}}^n \right] \qquad \begin{matrix} i = 1:M \\ n = 1: N_{max} \end{matrix}$$

This is the simplest, most physical, and most accurate 1st order in time
2nd order in space
(discretization error $= O(\Delta t + \Delta x^2)$)

The penalty for such simplicity is that we must restrict Δt for stability.

Stability condition: plug in $F_{i-\frac{1}{2}}$ to get updating formula for U_i:

$$U_i^{n+1} = U_i^n + \frac{D\Delta t}{\Delta x^2}\left[U_{i-1}^n - 2U_i^n + U_{i+1}^n \right]$$

centered finite difference $\approx u_{xx}$, 2nd order in Δx

$$\Rightarrow U_i^{n+1} = \left[1 - 2\frac{D\Delta t}{\Delta x^2}\right] U_i^n + \frac{D\Delta t}{\Delta x^2}\left[U_{i-1}^n + U_{i+1}^n \right]$$

Set $\mu = \dfrac{D\Delta t}{\Delta x^2}$ = CFL number, dimensionless (Courant-Friedrichs-Lewy ~1927?)

$$\Rightarrow U_i^{n+1} = (1 - 2\mu) U_i^n + \mu U_{i-1}^n + \mu U_{i+1}^n$$

Choose Δx, Δt so that, say, $\mu = 5$: $U_i^{n+1} = -9U_i + 5U_{i-1} + 5U_{i+1}$

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
$n=0$	0	1	2	1	0
$n=1$	0	$-9+5\cdot2 = +1$	$-9\cdot2+5\cdot2 = -8$	$-9+10 = +1$	0
$n=2$	0	$-9-40 = -49$	$+72+10 = +82$	$= -49$	0
$n=3$	0	+851	−1220	+851	0

Clearly nonsense! why?

CFL condition

Trouble arose because $1-2\mu < 0$ can produce negative values from positive ones, so can produce smaller (and larger) values than bdy or initial values.

This violates the Maximum Principle, a fundamental property of heat eqn:

> The smallest and largest values of $u(x,t)$ must occur on the boundary or earlier (initially)

To prevent it, should respect the

Positive Coefficient Rule: in the update formula $U_i^{n+1} = \dots$ all coefficients must be ~~positive~~ non-negative.

So, we must demand $1-2\mu \geq 0$ i.e $\mu \leq \frac{1}{2}$, this is the famous

Courant-Friedrichs-Lewy (CFL) condition: $0 < \mu = \frac{D\Delta t}{\Delta x^2} \leq \frac{1}{2}$

so must restrict the timestep: $\boxed{\Delta t \leq \frac{\Delta x^2}{2D} =: \Delta t_{expl}}$

For safety against roundoff: $\Delta t \lneq \Delta t_{expl}$

or, better, choose $\Delta t = \text{factor} \cdot \Delta t_{expl}$ and play with factor

0.99 or 0.95 or 0.90 ...

or 1 or 1.01 or 1.1 ...

Code skeleton

```
!1. Inputs to be read from a "dat" file :
      MM, tend, dtout, factor, D, a, b
```

MM = # of nodes per unit length
dtout = how often to print output

```
2. Set grid
     Dx = 1.d0/MM, M = (b - a) * MM
     CALL MESH(M, a, b, Dx, x)                    returns x array

3. Set timestep (for stability of explicit scheme)
     DtEXPL = Dx * Dx / (2 * D)
     Dt = factor * DtEXPL
     Nmax = int(tend / Dt) + 1
      CALL INIT(...) ; CALL OUTPUT(...)           profile at t = 0
4. Initialize
     nsteps = 0
     time = 0.d0
     tout = MAX(dtout, Dt)

5. Execute timestepping
     for nsteps = 1: Nmax
          CALL FLUX(                              returns F array
          CALL PDE(                               returns U array
          time = (time + Dt =) nsteps * Dt
          if( time >= tout)
               CALL OUTPUT(M, x, U, time, nsteps)
               tout = tout + dtout
          endif
     end for

6. Finish
     print: 'DONE, at time = ', time, ' after nsteps = ', nsteps
     END
```