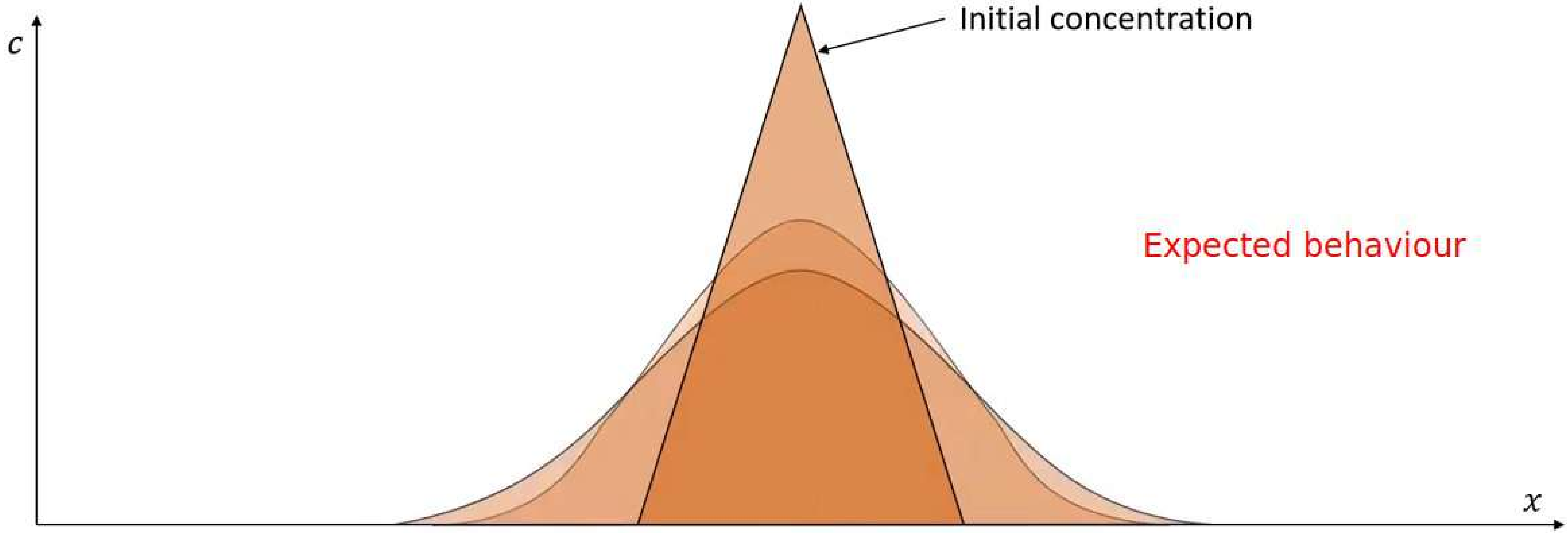
The 1D Diffusion Equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

with: c = concentration of dissolved substance (mg/m^3)
D = diffusion coefficient (m^2/s)



Finite Difference Schemes for Advection and Diffusion

The 1D Diffusion Equation

$$\frac{\partial c}{\partial t} = D\,\frac{\partial^2 c}{\partial x^2}$$

with: c = concentration of dissolved substance (mg/m³)
D = diffusion coefficient (m²/s)

$$\frac{\partial c}{\partial t} = \frac{c_{i,j+1} - c_{i,j}}{\Delta t} + O(\Delta t)$$

$$\frac{\partial^2 c}{\partial x^2} = \frac{c_{i+1,j} - 2c_{i,j} + c_{i-1,j}}{\Delta x^2} + O(\Delta x^2)$$

$$\frac{c_{i,j+1} - c_{i,j}}{\Delta t} = D\left(\frac{c_{i+1,j} - 2c_{i,j} + c_{i-1,j}}{\Delta x^2}\right)$$

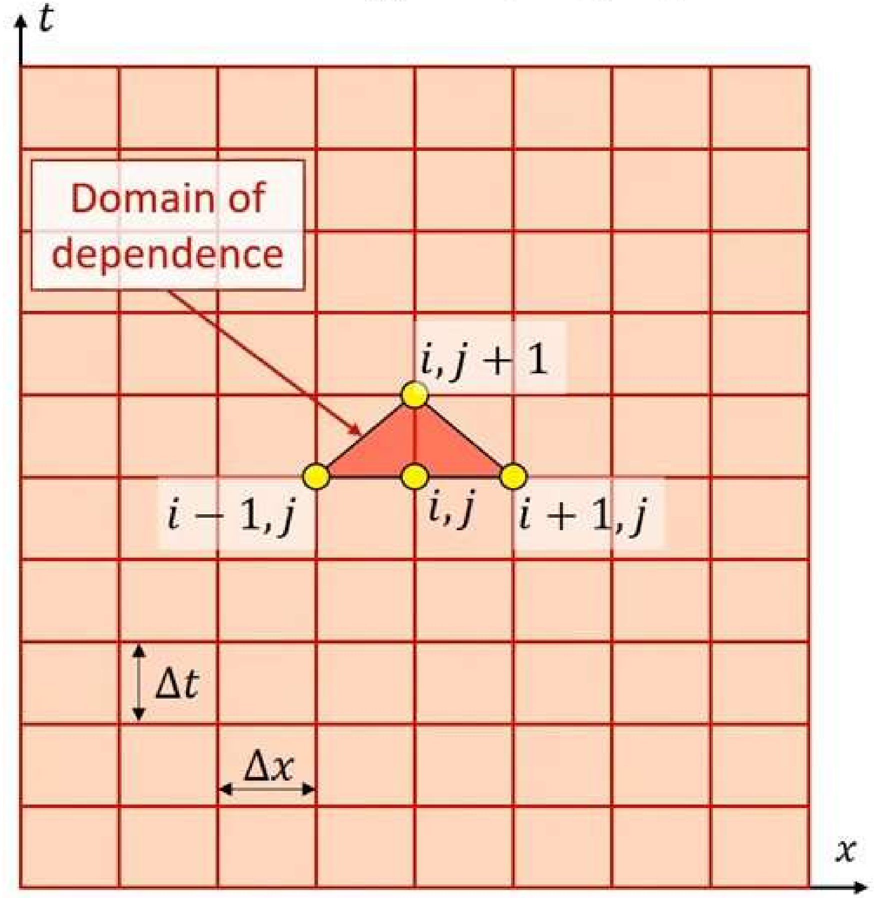
$$c_{i,j+1} = c_{i,j} + r\big(c_{i+1,j} - 2c_{i,j} + c_{i-1,j}\big) \qquad r = D\,\frac{\Delta t}{\Delta x^2}$$

Fourier number

$$\boxed{c_{i,j+1} = rc_{i+1,j} + (1-2r)c_{i,j} + rc_{i-1,j}}$$

Explicit scheme --> obs. First Order scheme (in time)

Notation: $c_{i,j} = c(i\Delta x, j\Delta t)$



The 1D Diffusion Equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

with: c = concentration of dissolved substance (mg/m³)
D = diffusion coefficient (m²/s)

$$\frac{\partial c}{\partial t} = \frac{c_{i,j+1} - c_{i,j}}{\Delta t} + O(\Delta t)$$

$$\frac{\partial^2 c}{\partial x^2} = \frac{c_{i+1,j} - 2c_{i,j} + c_{i-1,j}}{\Delta x^2} + O(\Delta x^2) \qquad \frac{\partial^2 c}{\partial x^2} = \frac{c_{i+1,j+1} - 2c_{i,j+1} + c_{i-1,j+1}}{\Delta x^2} + O(\Delta x^2)$$

$$\frac{c_{i,j+1} - c_{i,j}}{\Delta t} = \tfrac{1}{2}D\left(\frac{c_{i+1,j} - 2c_{i,j} + c_{i-1,j}}{\Delta x^2}\right) + \tfrac{1}{2}D\left(\frac{c_{i+1,j+1} - 2c_{i,j+1} + c_{i-1,j+1}}{\Delta x^2}\right)$$

$$c_{i,j+1} = c_{i,j} + \tfrac{1}{2}r\left(c_{i+1,j} - 2c_{i,j} + c_{i-1,j} + c_{i+1,j+1} - 2c_{i,j+1} + c_{i-1,j+1}\right)$$

$$r = D\frac{\Delta t}{\Delta x^2}$$

Fourier number

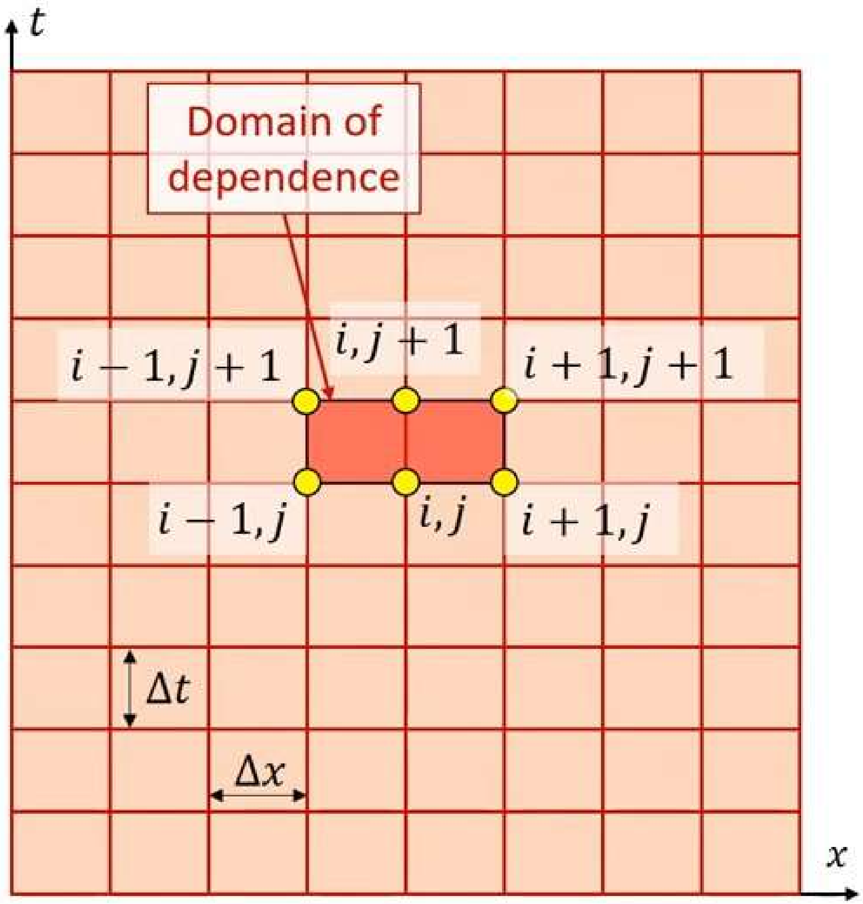
First order Crank-Nicolson scheme --> (implicit scheme)

$$\boxed{-\tfrac{1}{2}rc_{i-1,j+1} + (1+r)c_{i,j+1} - \tfrac{1}{2}rc_{i+1,j+1} = \tfrac{1}{2}rc_{i-1,j} + (1-r)c_{i,j} + \tfrac{1}{2}rc_{i+1,j}}$$

The 1D Diffusion Equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

with: c = concentration of dissolved substance (mg/m³)
D = diffusion coefficient (m²/s)



First order Crank-Nicolson scheme

$$-\tfrac{1}{2}rc_{i-1,j+1} + (1+r)c_{i,j+1} - \tfrac{1}{2}rc_{i+1,j+1} = \tfrac{1}{2}rc_{i-1,j} + (1-r)c_{i,j} + \tfrac{1}{2}rc_{i+1,j}$$

$$r = D\frac{\Delta t}{\Delta x^2}$$

Crank-Nicolson (implicit) scheme

$$\boxed{-\tfrac{1}{2}rc_{i-1,j+1} + (1+r)c_{i,j+1} - \tfrac{1}{2}rc_{i+1,j+1} = \tfrac{1}{2}rc_{i-1,j} + (1-r)c_{i,j} + \tfrac{1}{2}rc_{i+1,j}} \qquad r = D\frac{\Delta t}{\Delta x^2}$$

Tridiagonal system of equations:

$$\begin{bmatrix} (1+r) & -\frac{1}{2}r & 0 & 0 & \cdots & 0 \\ -\frac{1}{2}r & (1+r) & -\frac{1}{2}r & 0 & \cdots & 0 \\ 0 & -\frac{1}{2}r & (1+r) & -\frac{1}{2}r & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \\ & & 0 & -\frac{1}{2}r & (1+r) & -\frac{1}{2}r \\ 0 & \cdots & 0 & 0 & -\frac{1}{2}r & (1+r) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_N \end{bmatrix}_{j+1} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{N-1} \\ \beta_N \end{bmatrix}_j$$

where $\beta_{i,j} = \frac{1}{2}rc_{i-1,j} + (1-r)c_{i,j} + \frac{1}{2}rc_{i+1,j}$

Boundary conditions

The diffusion equation: $\dfrac{\partial c}{\partial t} = D\dfrac{\partial^2 c}{\partial x^2}$

1. c is known: $c(0,t) = f(t), \quad c(L,t) = g(t), \quad t \geq 0.$ <u>Dirichlet condition</u>

2. No material is leaving the domain: $\dfrac{\partial c}{\partial x}(0,t) = 0, \quad \dfrac{\partial c}{\partial x}(L,t) = 0, \quad t \geq 0.$ <u>Neumann condition</u>

   Closed BC

3. $\dfrac{\partial c}{\partial x}(0,t) = F(c,t), \quad \dfrac{\partial c}{\partial x}(L,t) = G(c,t), \quad t \geq 0.$

Boundary conditions

Neumann condition $\dfrac{\partial c}{\partial x}(0,t) = 0, \quad \dfrac{\partial c}{\partial x}(L,t) = 0$ where L is the length of the solution domain

Forward difference at $x = 0$

$$\frac{\partial c}{\partial x}(0, j\Delta t) = \frac{c_{1,j} - c_{0,j}}{\Delta x}$$

$$\boxed{c_{0,j} = c_{1,j}}$$

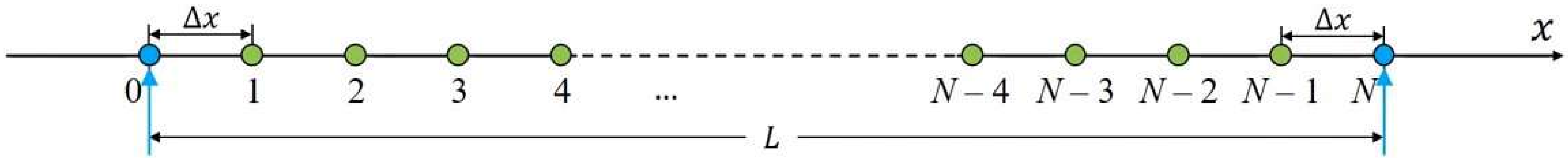
Accurate to $O(\Delta x)$

First order

Backward difference at $x = N\Delta x = L$

$$\frac{\partial c}{\partial x}(L, j\Delta t) = \frac{c_{N,j} - c_{N-1,j}}{\Delta x}$$

$$\boxed{c_{N,j} = c_{N-1,j}}$$




```python
def oneStepExplicitFirstOrderDiffusion(yIN, Fr):
  yOUT = np.zeros_like(yIN)
  # internal points
  for i in range(1,len(x)-1):
    yOUT[i] = Fr*yIN[i-1] + (1.-2*Fr)*yIN[i] + Fr*yIN[i+1]
  # closed BC (Neumann)
  yOUT[0] = yOUT[1]
  yOUT[-1] = yOUT[-2]
  return yOUT
```

--> Second Order accurate disc. in space

--> First order accurate BC discretization

Finite Difference Schemes for Advection and Diffusion

Boundary conditions

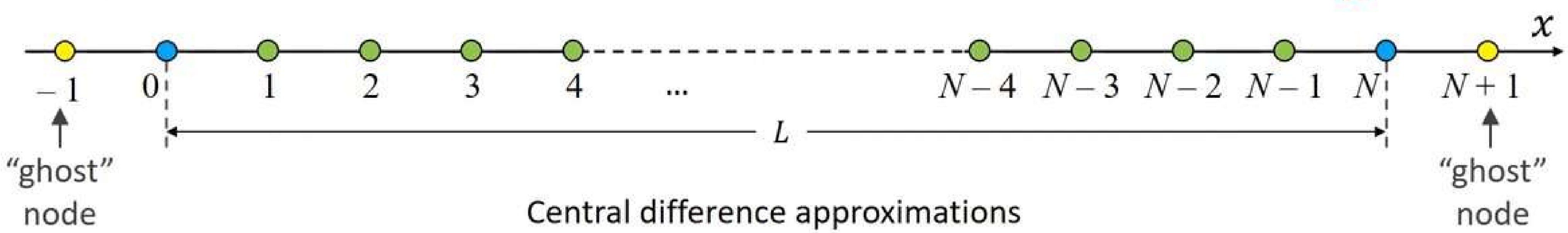
Neumann condition $\dfrac{\partial c}{\partial x}(0,t) = 0, \quad \dfrac{\partial c}{\partial x}(L,t) = 0$ where L is the length of the solution domain

```
def oneStepExplicitFirstOrderDiffusion(yIN, Fr):
  yOUT = np.zeros_like(yIN)
  # internal points
  for i in range(1,len(x)-1):
    yOUT[i] = Fr*yIN[i-1] + (1.-2*Fr)*yIN[i] + Fr*yIN[i+1]
  # closed BC (Neumann)
  yOUT[0] = (1.-2*Fr)*yIN[0] + 2.*Fr*yIN[1]
  yOUT[-1] = 2.*Fr*yIN[-2] + (1.-2*Fr)*yIN[-1]
  return yOUT
```

--> Second Order disc. for internal points

--> Second Order disc. for BCs

Consistent disc.



Central difference approximations

$$\frac{\partial c}{\partial x}(0, j\Delta t) = \frac{c_{1,j} - c_{-1,j}}{2\Delta x}$$

$$\frac{\partial c}{\partial x}(N\Delta x, j\Delta t) = \frac{c_{N+1,j} - c_{N-1,j}}{2\Delta x}$$

Accurate to $O(\Delta x^2)$

$\boxed{c_{-1,j} = c_{1,j}}$

Second order

$\boxed{c_{N+1,j} = c_{N-1,j}}$

Boundary conditions

Neumann condition $\dfrac{\partial c}{\partial x}(0,t) = 0, \quad \dfrac{\partial c}{\partial x}(L,t) = 0$ where L is the length of the solution domain

$$\boxed{c_{i,j+1} = rc_{i+1,j} + (1-2r)c_{i,j} + rc_{i-1,j}} \qquad r = D\frac{\Delta t}{\Delta x^2}$$

Explicit scheme for the diffusion equation

Left hand boundary: $\boxed{c_{-1,j} = c_{1,j}}$ [Direct implementation – requires external node]

$$c_{0,j+1} = rc_{1,j} + (1-2r)c_{0,j} + rc_{-1,j}$$

$$\boxed{c_{0,j+1} = (1-2r)c_{0,j} + 2rc_{1,j}}$$

[Indirect/Implicit implementation]

--> `yOUT[0] = (1.-2*Fr)*yIN[0] + 2.*Fr*yIN[1]`

Right hand boundary: $\boxed{c_{N+1,j} = c_{N-1,j}}$ [Direct implementation]

$$c_{N,j+1} = rc_{N+1,j} + (1-2r)c_{N,j} + rc_{N-1,j}$$

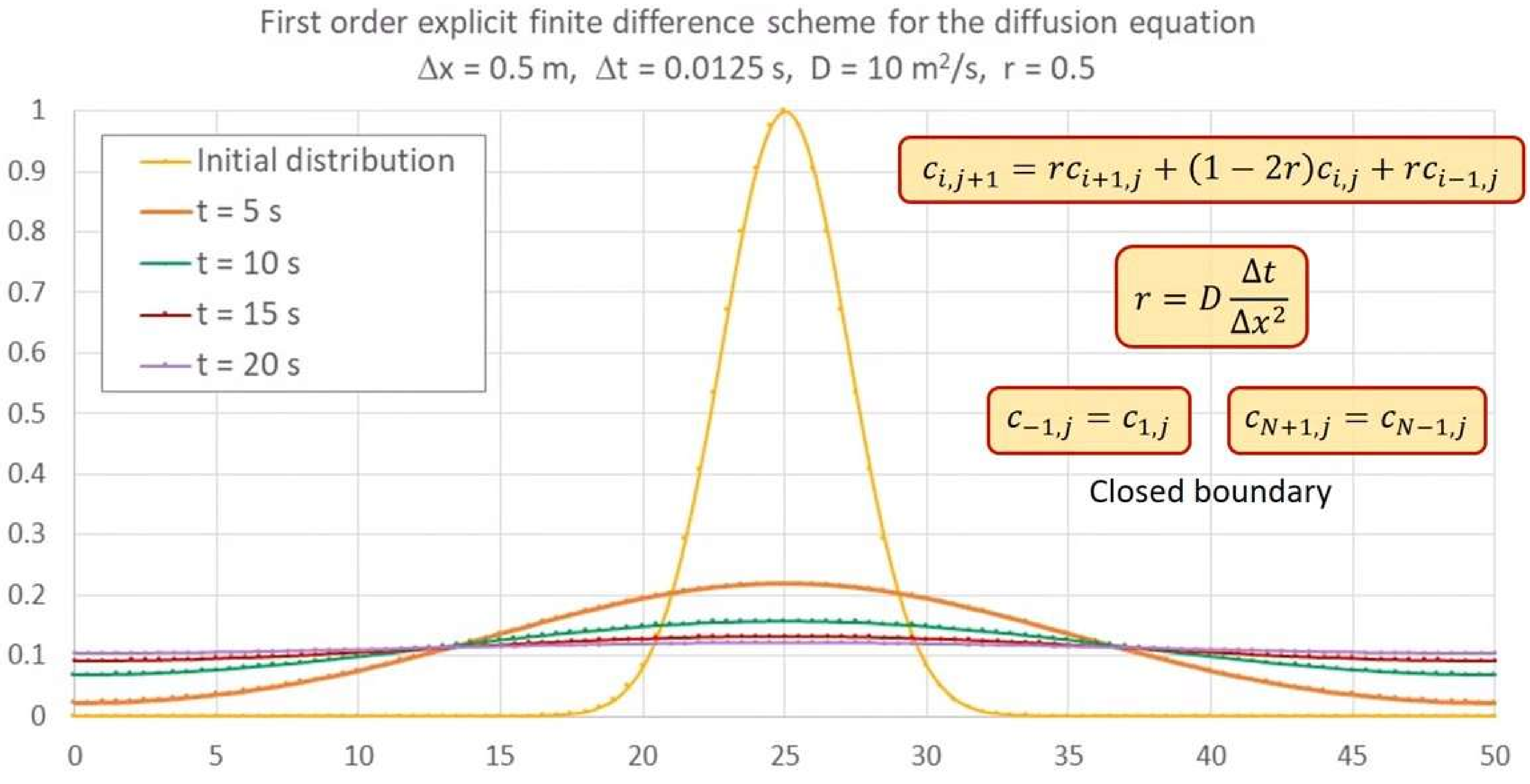
$$\boxed{c_{N,j+1} = 2rc_{N-1,j} + (1-2r)c_{N,j}}$$

[Indirect/Implicit implementation]

--> `yOUT[-1] = 2.*Fr*yIN[-2] + (1.-2*Fr)*yIN[-1]`

Preferable since the **order of the discretization error is preserved** throughout the domain (second order).

Explicit scheme for the diffusion equation



Finite Difference Schemes for Advection and Diffusion

Caspar Hewett

1,05 k abonnés

S'abonner

165

Partager

Télécharger

Extrait

...

```python
import numpy as np

# space discretization
xL = 0.
xR = 50.
dx = 0.5
delX = xR - xL
nx = int(delX/dx)
x = np.linspace(xL, xR, nx+1)
```

1

```python
# time discretization
Fr = 0.5 # Fourier Number: Fr = D*dt/dx**2
D = 10.
dt = Fr*dx**2/D
delT = 5.
stepsNbr = int(delT/dt)
```

2

```python
# unkown's initialization
def yInit_(x, yMin, yMax, xc, width, isRectangular = True):
  if isRectangular:
    # rectangular distribuition
    return np.array([yMax if xc-width < xi and xi < xc+width else yMin for xi in x])
  else:
    # Gaussian distribuition
    return yMin + yMax * np.exp(-(x - xc) ** 2 / (2 * width ** 2))
yMin = 0.
yMax = 1.
yInit = yInit_(x=x,
               yMin=yMin,
               yMax=yMax,
               xc=25.,
               width=2.,
               isRectangular=False)
y = np.copy(yInit)
```

3

```python
def oneStepExplicitFirstOrderDiffusion(yIN, Fr):
  yOUT = np.zeros_like(yIN)
  # internal points
  for i in range(1,len(x)-1):
    yOUT[i] = Fr*yIN[i-1] + (1.-2*Fr)*yIN[i] + Fr*yIN[i+1]
  # closed BC (Neumann)
  yOUT[0] = (1.-2*Fr)*yIN[0] + 2.*Fr*yIN[1]
  yOUT[-1] = 2.*Fr*yIN[-2] + (1.-2*Fr)*yIN[-1]
  return yOUT
```

4

```python
for i in range(stepsNbr):
  y = oneStepExplicitFirstOrderDiffusion(y, Fr)
```

6

```python
print('dx = ', dx, '\n'
      'Fr = ', Fr, '\n'
      'D = ', D, '\n'
      'dt = ', dt, '\n'
      'delT = ', delT, '\n'
      'stepsNbr = ', stepsNbr)
```

5

Output

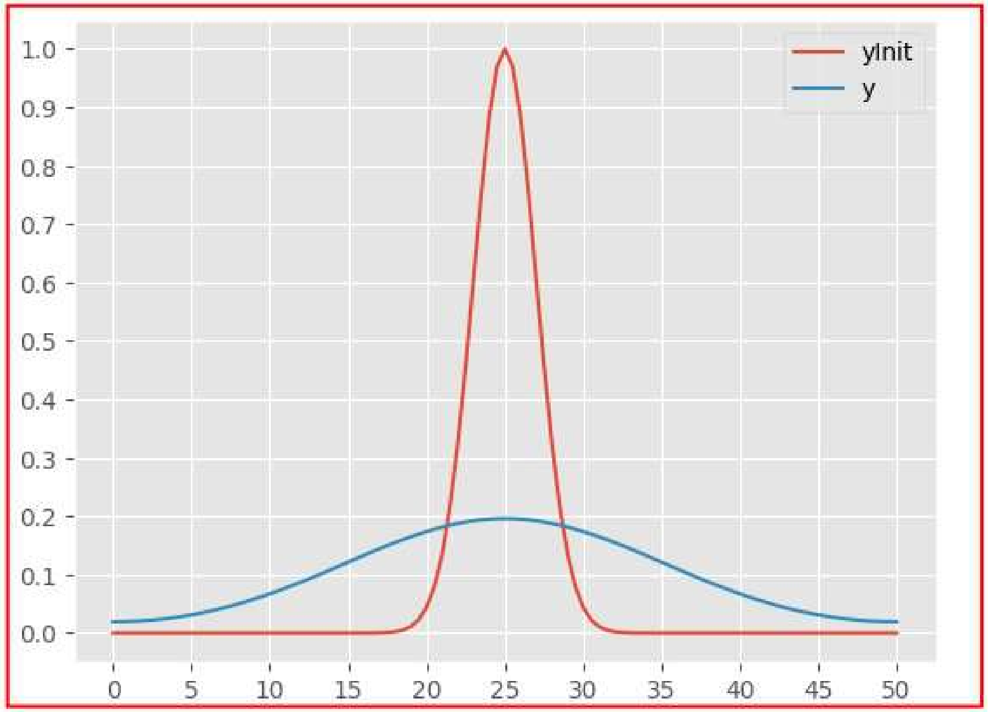
```
dx =  0.5
Fr =  0.5
D =  10.0
dt =  0.0125
delT =  5.0
stepsNbr =  400
```

1

```python
import matplotlib.pyplot as plt
plt.style.use("ggplot")
plt.xticks(np.arange(xL, xR+.1, 5))
plt.yticks(np.arange(yMin, yMax+.1, .1))
plt.plot(x, yInit, label='yInit')
plt.plot(x, y, label='y')
plt.legend()
plt.show()
```

7

Output



2

1D Advection Equation

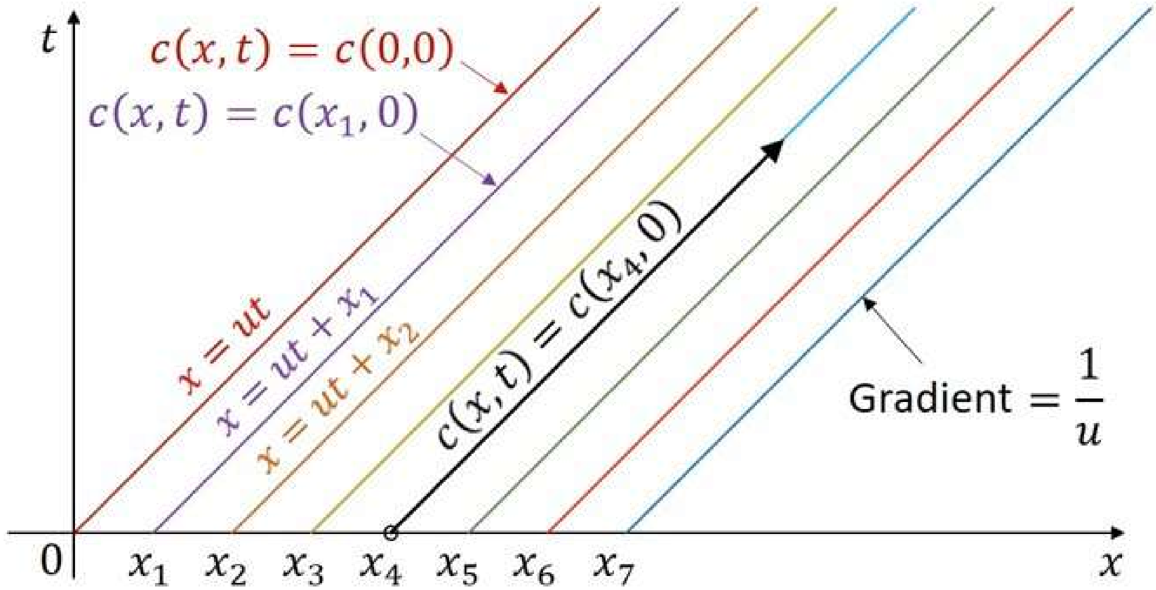
$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0$$

with: $c = c(x, t)$ = concentration (mg/m³)
u = flow velocity in the x direction (m/s)

Let us assume $u > 0$ and is constant

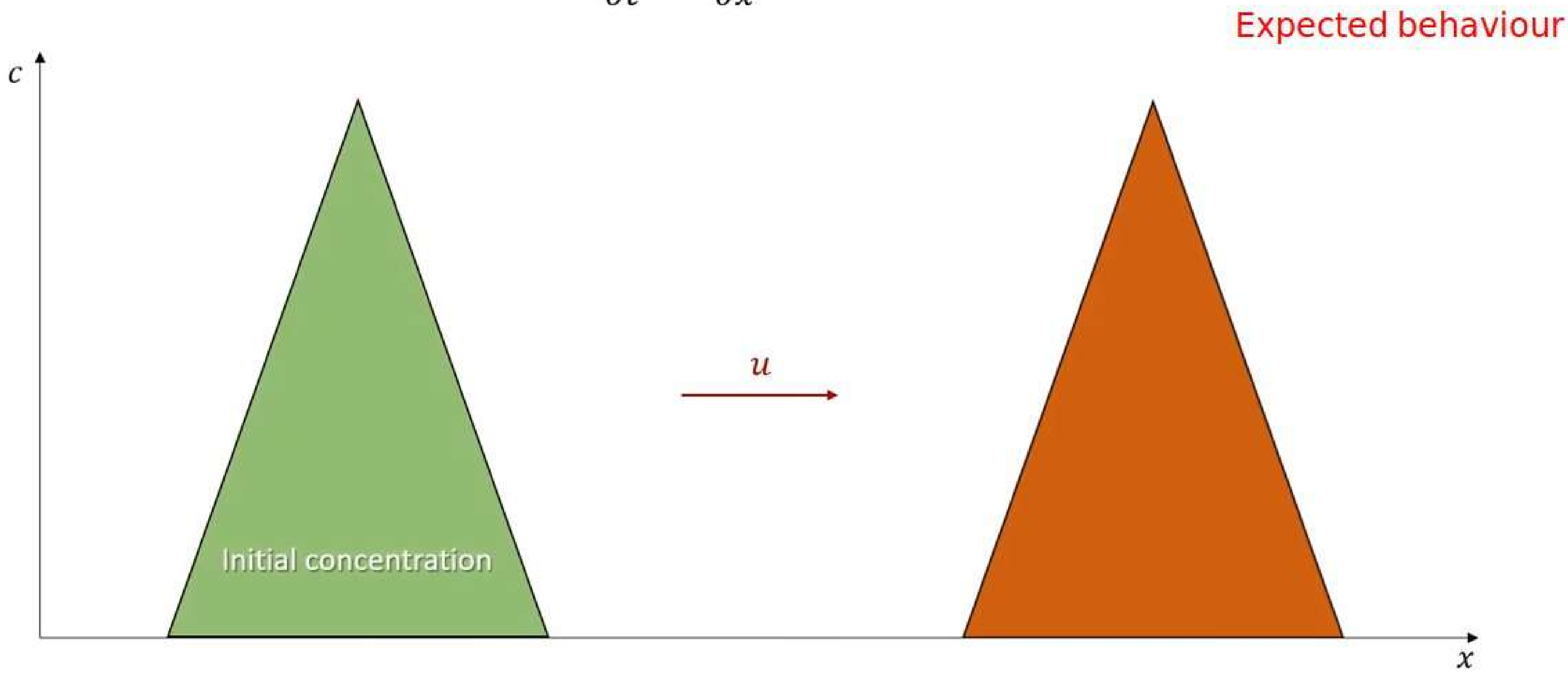
Characteristics: $\dfrac{dx}{dt} = u$

$x = ut$ + constant



1D Advection Equation

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0 \ , u > 0$$



Explicit Upwind Finite Difference Solution to the Advection Equation

1D Advection Equation

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0 \ , \boxed{u > 0}$$

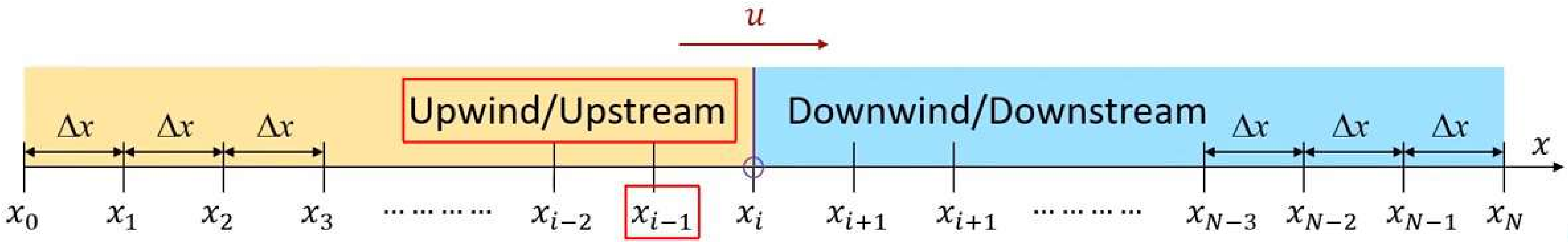
$$\frac{\partial c}{\partial t} \approx \frac{c(x, t+\Delta t) - c(x,t)}{\Delta t}$$

Forward time difference approximation ($O(\Delta t)$)

$$\frac{\partial c}{\partial x} \approx \frac{c(x,t) - \boxed{c(x-\Delta x, t)}}{\Delta x}$$

Backward spatial difference approximation ($O(\Delta x)$) <u>Upwind scheme</u>

Obs. u>0 --> Backward = Upwind or Upstream



Explicit Upwind Finite Difference Solution to the Advection Equation

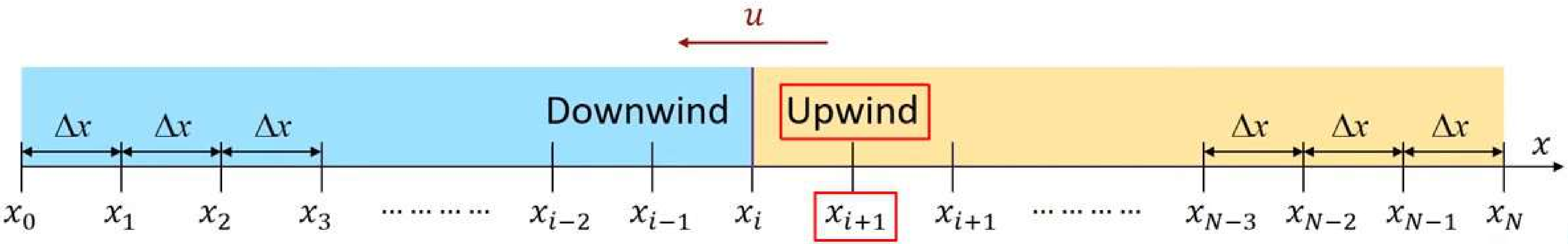
1D Advection Equation

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0 \ , \boxed{u < 0}$$

$$\frac{\partial c}{\partial t} \approx \frac{c(x, t+\Delta t) - c(x,t)}{\Delta t}$$ Forward time difference approximation

$$\frac{\partial c}{\partial x} \approx \frac{\boxed{c(x+\Delta x, t)} - c(x,t)}{\Delta x}$$ Forward spatial difference approximation <u>Upwind scheme</u>

Obs. u<0 --> Forward = Upwind or Upstream



Explicit Upwind Finite Difference Solution to the Advection Equation

1D Advection Equation

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0 \; , u > 0$$

Notation: $c_{i,j} = c(i\Delta x, j\Delta t)$

$$\frac{\partial c}{\partial t} \approx \frac{c(x, t + \Delta t) - c(x, t)}{\Delta t} \qquad \frac{\partial c}{\partial t} \approx \frac{c_{i,j+1} - c_{i,j}}{\Delta t}$$

$$\frac{\partial c}{\partial x} \approx \frac{c(x, t) - c(x - \Delta x, t)}{\Delta x} \qquad \frac{\partial c}{\partial x} \approx \frac{c_{i,j} - c_{i-1,j}}{\Delta x}$$

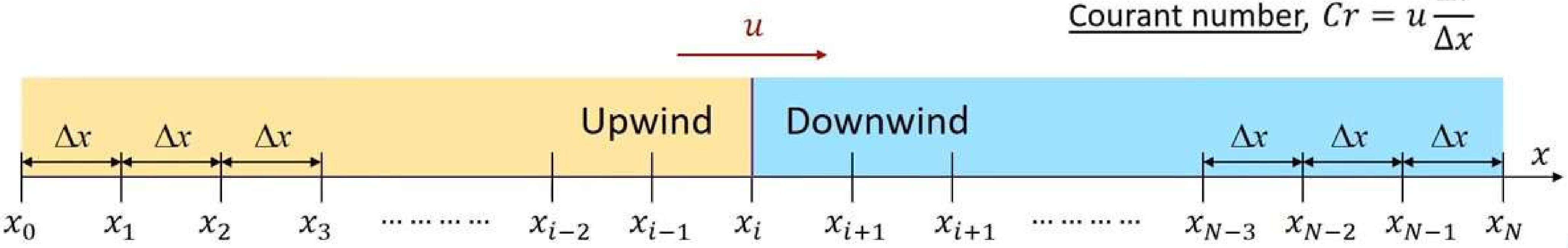
$$\frac{c_{i,j+1} - c_{i,j}}{\Delta t} + u\frac{c_{i,j} - c_{i-1,j}}{\Delta x} = 0$$

$$c_{i,j+1} = c_{i,j} - u\frac{\Delta t}{\Delta x}(c_{i,j} - c_{i-1,j})$$

$$\boxed{c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}}$$

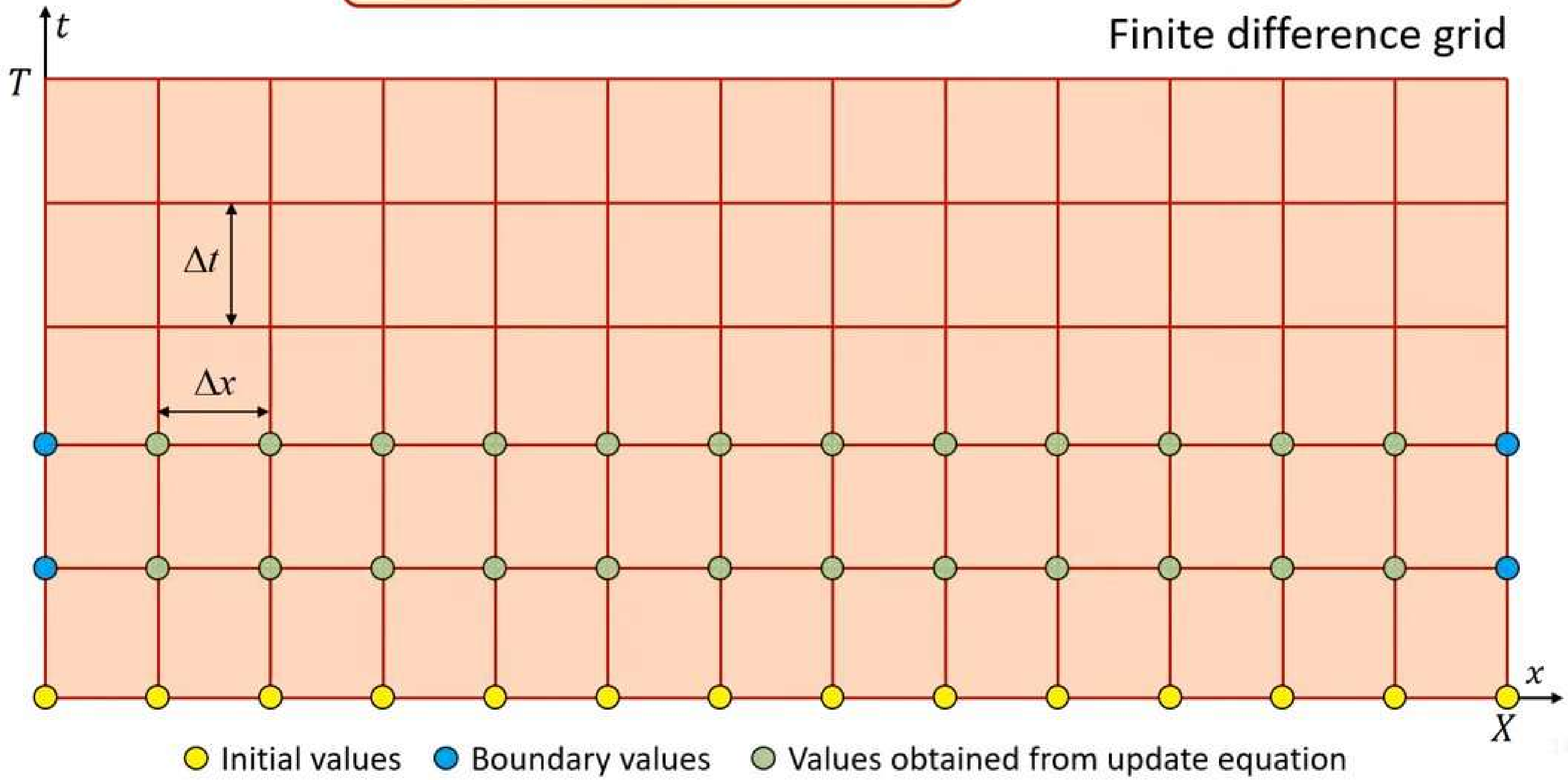
First order explicit upwind finite difference solution to the advection equation

Courant number, $Cr = u\frac{\Delta t}{\Delta x}$



Explicit Finite Difference schemes

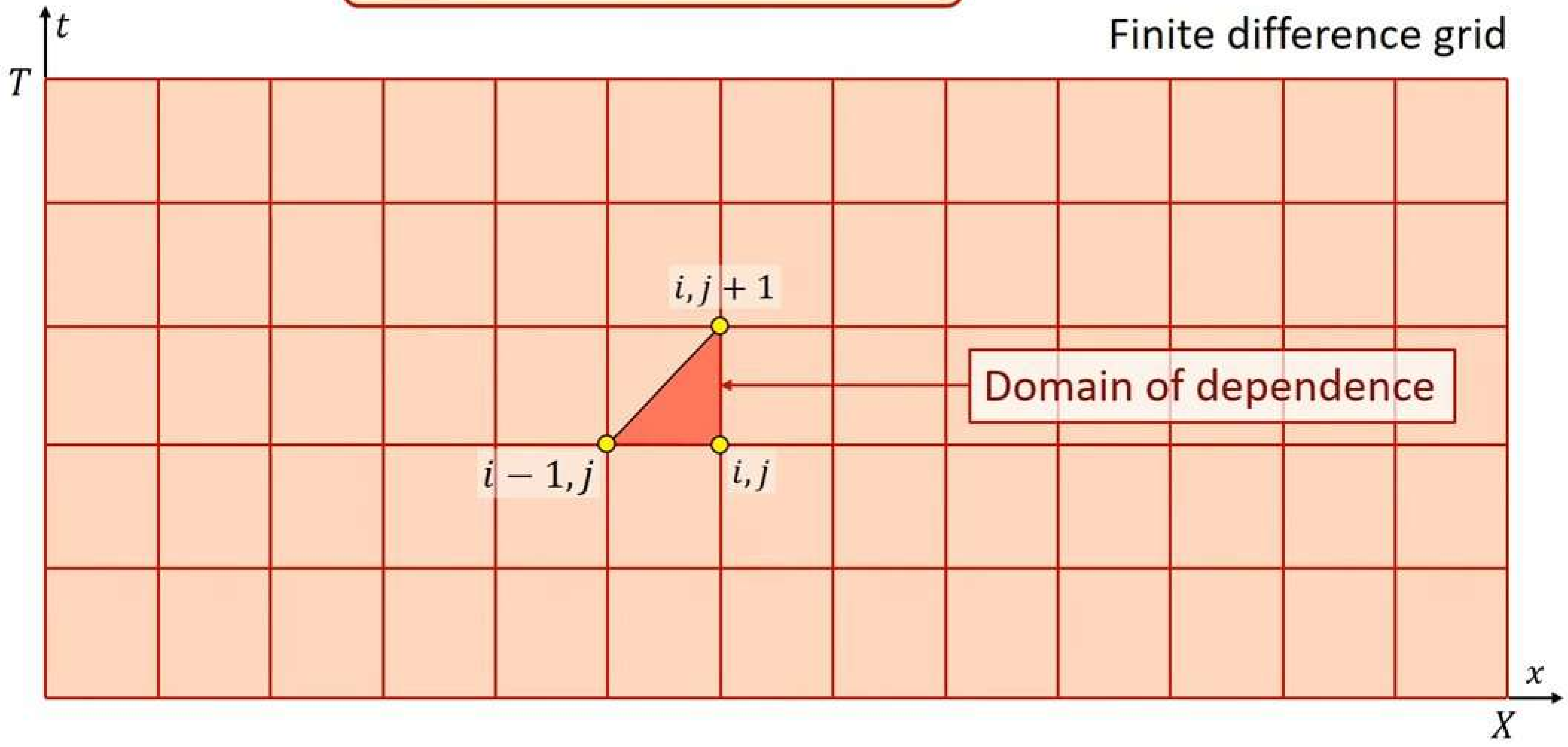
$$c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}$$



Explicit Upwind Finite Difference Solution to the Advection Equation

Explicit Finite Difference schemes

$$c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}$$

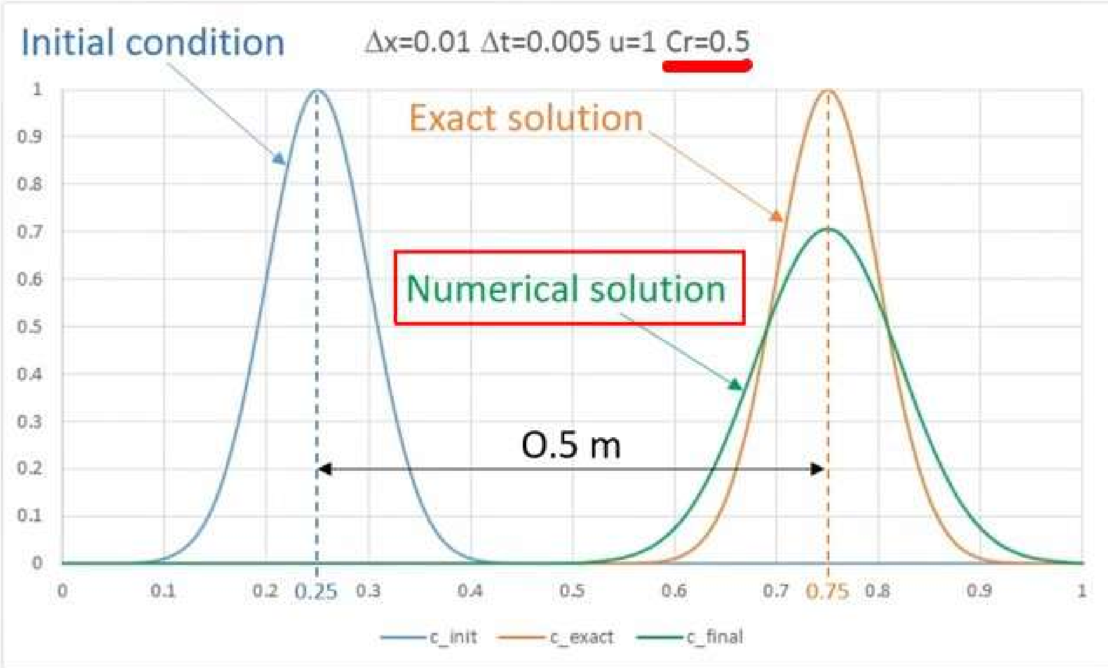


Explicit Finite Difference schemes

First order explicit upwind finite difference solution to the advection equation:

$$\boxed{c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}} \qquad Cr = u\frac{\Delta t}{\Delta x}$$

$Cr = 0.5$



After 0.5 s

Explicit Upwind Finite Difference Solution to the Advection Equation

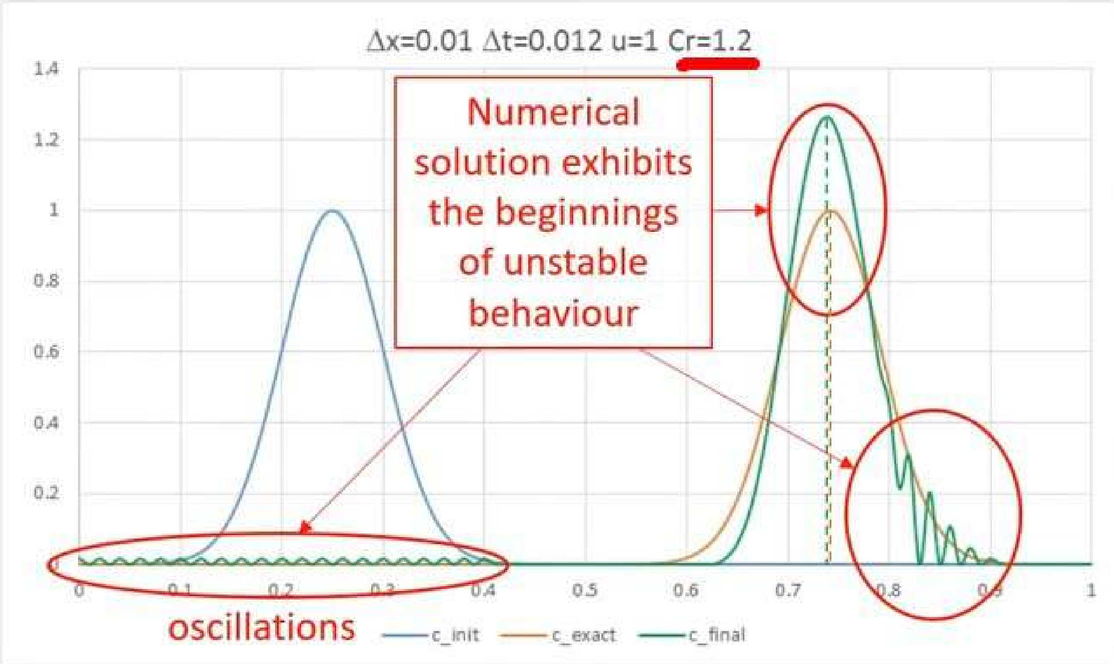
Explicit Finite Difference schemes

First order explicit upwind finite difference solution to the advection equation:

$$\boxed{c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}}$$

$$Cr = u\frac{\Delta t}{\Delta x}$$

$Cr = 1.2$



After 0.5 s

<u>Unstable</u>

Solution eventually 'blows up'

Explicit Upwind Finite Difference Solution to the Advection Equation

Caspar Hewett

1,05 k abonnés

S'abonner

130

Partager

Télécharger

Extrait

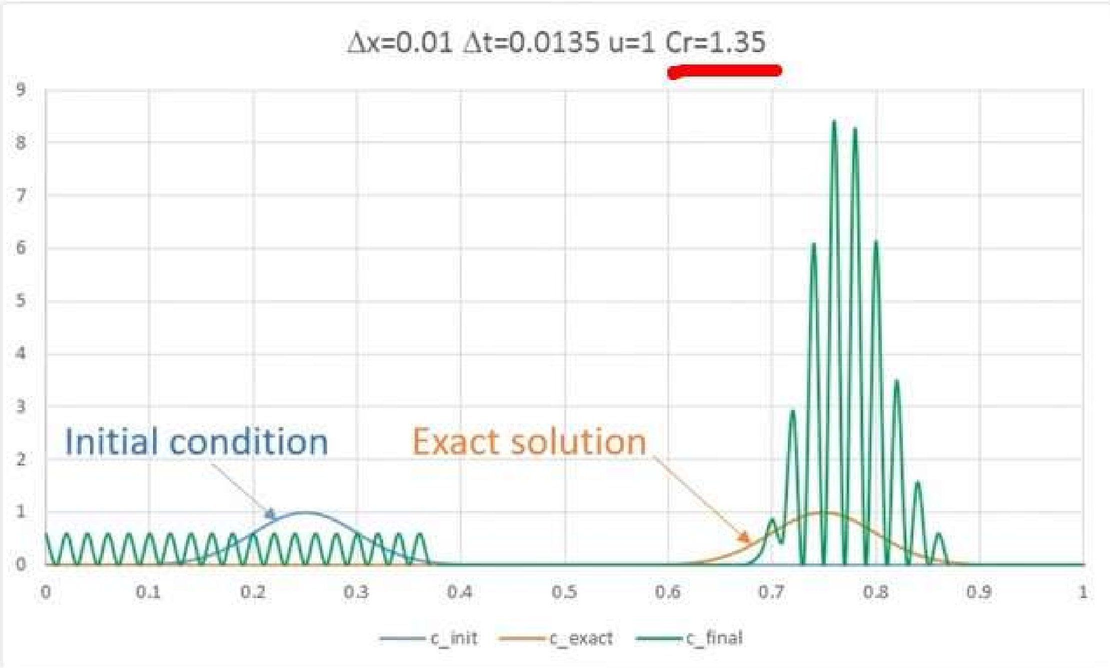
Enregistrer

Explicit Finite Difference schemes

First order explicit upwind finite difference solution to the advection equation:

$$\boxed{c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}} \qquad Cr = u\frac{\Delta t}{\Delta x}$$

$Cr = 1.35$



After 0.5 s

<u>Unstable</u>

Explicit Finite Difference schemes

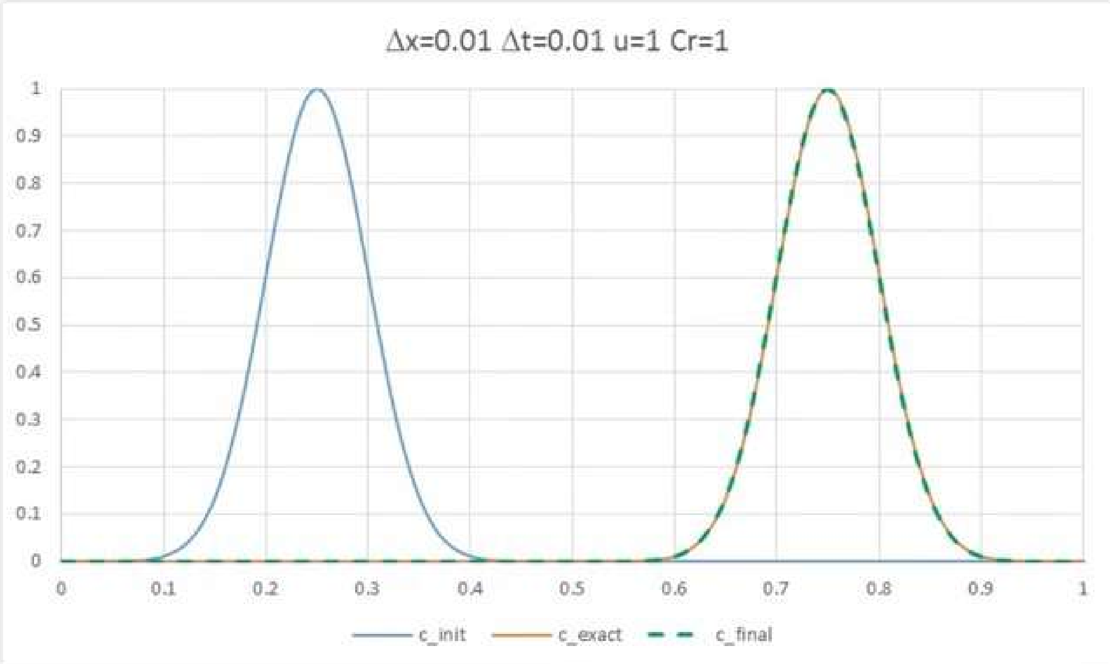
First order explicit upwind finite difference solution to the advection equation:

$$\boxed{c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}} \qquad Cr = u\frac{\Delta t}{\Delta x}$$

$$Cr = 1$$

$$\Rightarrow \boxed{c_{i,j+1} = c_{i-1,j}}$$

Reproduces the exact solution



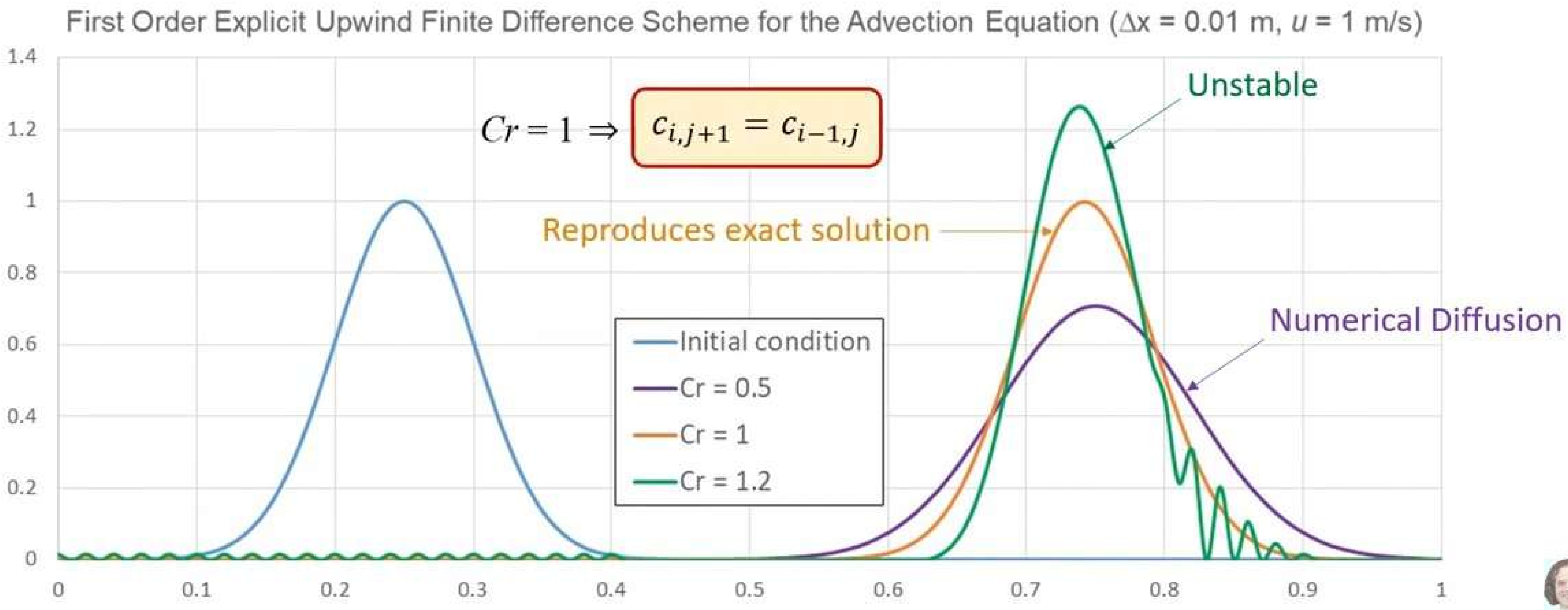
After 0.5 s

Explicit Upwind Finite Difference Solution to the Advection Equation

Explicit Finite Difference schemes

Summary

$$\boxed{c_{i,j+1} = (1 - Cr)c_{i,j} + (Cr)c_{i-1,j}} \qquad Cr = u\frac{\Delta t}{\Delta x}$$



Consistency

1st order upwind explicit scheme for the advection equation:

Finite difference equation

$$\boxed{c_{i,j+1} = (1-\rho)c_{i,j} + \rho c_{i-1,j}}$$

$$\boxed{\rho = u\frac{\Delta t}{\Delta x}}$$

Taylor series:

$$c_{i,j+1} = c_{ij} + \Delta t\frac{\partial c}{\partial t} + \frac{\Delta t}{2!}\frac{\partial^2 c}{\partial t^2} + \frac{\Delta t^2}{3!}\frac{\partial^3 c}{\partial t^3} + \cdots \quad \text{and} \quad c_{i-1,j} = c_{i,j} - \Delta x\frac{\partial c}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 c}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 c}{\partial x^3} + \cdots$$

Substituting into the finite difference equation

$$c_{ij} + \Delta t\frac{\partial c}{\partial t} + \frac{\Delta t^2}{2!}\frac{\partial^2 c}{\partial t^2} + \frac{\Delta t^3}{3!}\frac{\partial^3 c}{\partial t^3} + \cdots = (1-\rho)c_{i,j} + \rho\left(c_{i,j} - \Delta x\frac{\partial c}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 c}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 c}{\partial x^3} + \cdots\right)$$

$$\Delta t\frac{\partial c}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 c}{\partial t^2} + \frac{\Delta t^3}{6}\frac{\partial^3 c}{\partial t^3} + \cdots = -\rho\Delta x\frac{\partial c}{\partial x} + \rho\frac{\Delta x^2}{2}\frac{\partial^2 c}{\partial x^2} - \rho\frac{\Delta x^3}{6}\frac{\partial^3 c}{\partial x^3} + \cdots$$

$$\Delta t\frac{\partial c}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 c}{\partial t^2} + \frac{\Delta t^3}{6}\frac{\partial^3 c}{\partial t^3} + \cdots = -u\Delta t\frac{\partial c}{\partial x} + u\Delta t\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2} - u\Delta t\frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3} + \cdots$$

$$\boxed{\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0}$$

Advection equation

$$\frac{\partial c}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2} + \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3} + \cdots = -u\frac{\partial c}{\partial x} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2} - u\frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3} + \cdots$$

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = \underbrace{\underbrace{-\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2}}_{\text{Local truncation error}} - \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3} - u\frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3} + \cdots}_{\text{Truncation error}}$$

Consistency

1st order upwind explicit scheme for the advection equation

$$c_{i,j+1} = (1-\rho)c_{i,j} + \rho c_{i-1,j} \Rightarrow \quad \frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = -\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2} - \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3} - u\frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3} + \cdots$$

$$\text{Truncation Error (T.E.)} = -\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2} - \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3} - u\frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3} + \cdots$$

$\boxed{\lim_{\Delta x, \Delta t \to 0} (\text{T.E.}) = 0 \quad \Rightarrow \text{The scheme is consistent with the original PDE.}}$

Truncation error analysis

Consistency and Numerical Diffusion

Consistency

1st order upwind explicit scheme for the advection equation

$$\boxed{\rho = u\frac{\Delta t}{\Delta x}}$$

$$c_{i,j+1} = (1-\rho)c_{i,j} + \rho c_{i-1,j} \Rightarrow \frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = -\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2} - \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3} - u\frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3} + \cdots$$

$$\text{Local Truncation Error (L. T. E.)} = -\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2}$$

$$\frac{\partial c}{\partial t} = -u\frac{\partial c}{\partial x} \Rightarrow \frac{\partial^2 c}{\partial t^2} = \frac{\partial}{\partial t}\left(-u\frac{\partial c}{\partial x}\right) = -u\frac{\partial}{\partial t}\left(\frac{\partial c}{\partial x}\right) = -u\frac{\partial}{\partial x}\left(\frac{\partial c}{\partial t}\right) = -u\frac{\partial}{\partial x}\left(-u\frac{\partial c}{\partial x}\right)$$

$$\Rightarrow \frac{\partial^2 c}{\partial t^2} = u^2\frac{\partial^2 c}{\partial x^2} \Rightarrow \text{L. T. E.} = -\frac{\Delta t}{2}u^2\frac{\partial^2 c}{\partial x^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2}$$

$$\boxed{\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0}$$

Advection equation

$$\boxed{\rho\Delta x = u\Delta t} \Rightarrow \text{L. T. E.} = -u\frac{\rho\Delta x}{2}\frac{\partial^2 c}{\partial x^2} + u\frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2}$$

$$\boxed{\text{L. T. E.} = \tfrac{1}{2}u\Delta x(1-\rho)\frac{\partial^2 c}{\partial x^2}}$$

Source of numerical diffusion

Numerical diffusion (false dispersion)

Courant number

$$\rho = u\frac{\Delta t}{\Delta x}$$

1st order upwind explicit scheme for the advection equation

$$\boxed{c_{i,j+1} = (1-\rho)c_{i,j} + \rho c_{i-1,j}}$$

$$\boxed{\mathrm{L.T.E.} = \tfrac{1}{2}u\Delta x(1-\rho)\frac{\partial^2 c}{\partial x^2}}$$

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = \tfrac{1}{2}u\Delta x(1-\rho)\frac{\partial^2 c}{\partial x^2} + \text{Higher Order Terms}$$

- If $\rho < 1$ the local truncation error is positive and has a *diffusive* effect. Numerical diffusion
- If $\rho > 1$ the local truncation error is negative and has an *amplifying* effect and is a cause of instability.
- When $\rho = 1$ the local truncation error is zero. No numerical diffusion.
- When ρ is close to 1 the local truncation error is small and thus so is the numerical diffusion (for $\rho < 1$).

$$\boxed{\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0}$$

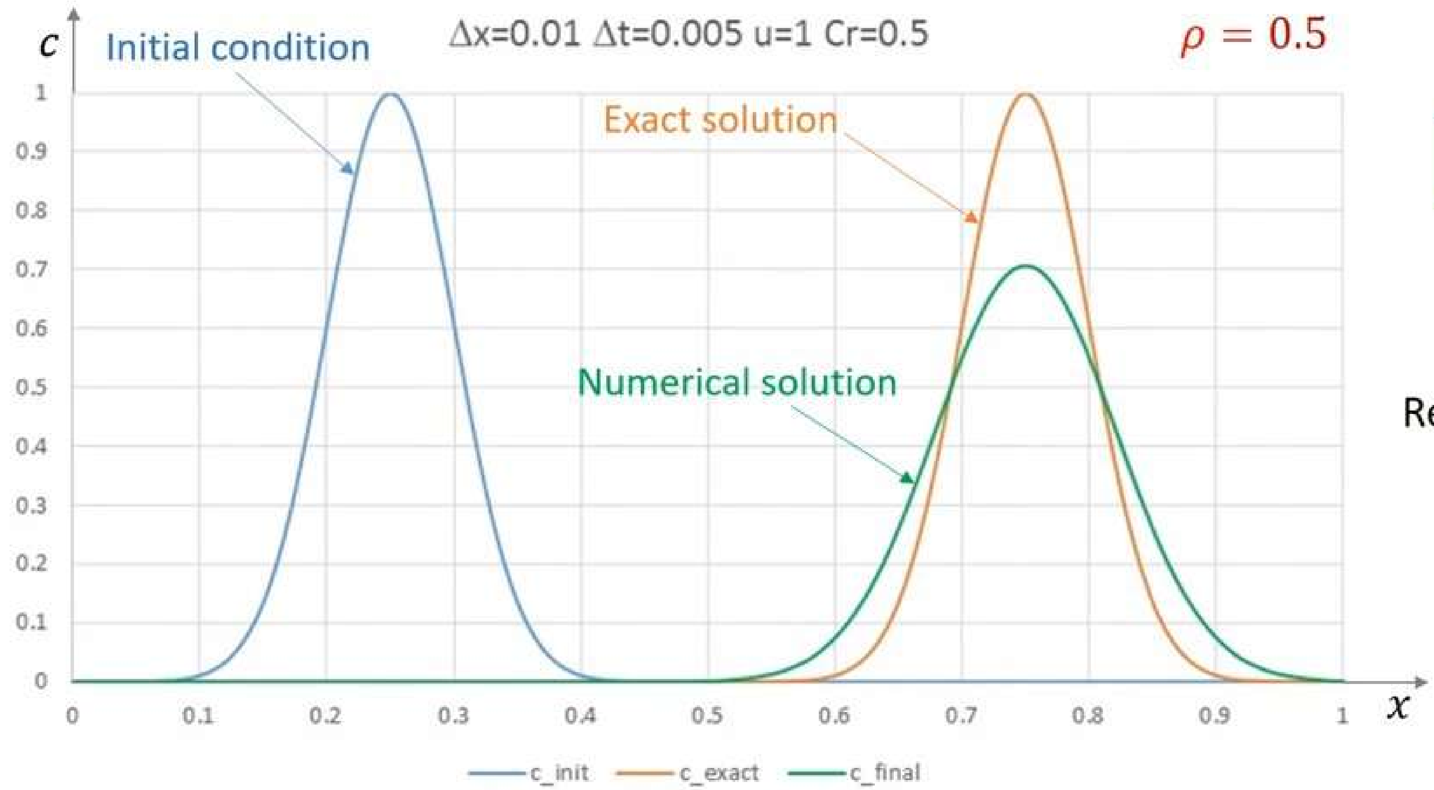
Advection equation

Consistency and Numerical Diffusion

Numerical diffusion

1st order upwind explicit scheme for the advection equation

$$\boxed{c_{i,j+1} = (1-\rho)c_{i,j} + \rho c_{i-1,j}}$$



$$\mathrm{T.E.} = \underbrace{\tfrac{1}{2}u\Delta x(1-\rho)\frac{\partial^2 c}{\partial x^2}}_{\text{NUMERICAL DIFFUSION TERM}} + \underbrace{\mathrm{H.O.T}}_{\text{HIGHER ORDER TERMS}}$$

Positive & Diffusive

Courant number, $\rho < 1$

Results obtained for $t = 0.5$ seconds

Consistency and Numerical Diffusion

Consistency of a numerical scheme is its property that when the discretization step tends to zero its finite difference approximation tends towards the original differential equation.

To *prove* a scheme is consistent with the differential equation it represents we perform truncation error analysis.

Numerical diffusion is a form of inaccuracy of a numerical scheme that manifests itself through an unrealistic attenuation of the propagated disturbance (wave height or concentration). The effects of numerical diffusion are similar to physical diffusion and therefore it is often very hard to distinguish between the two.

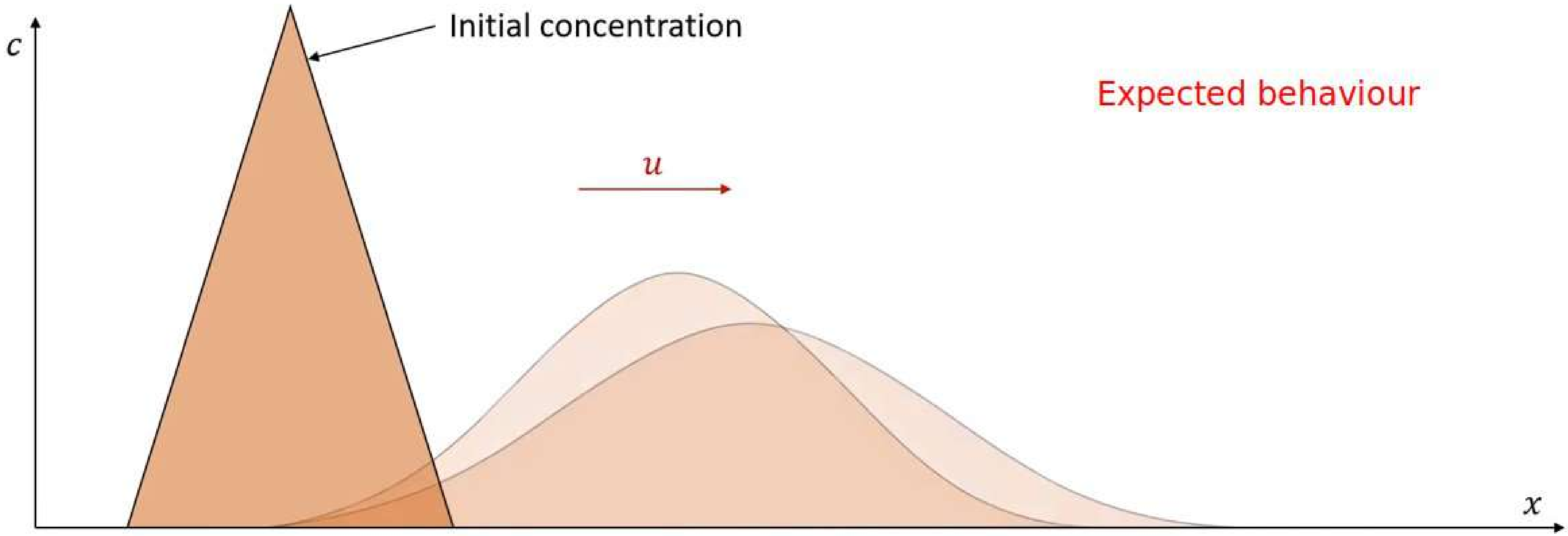
Truncation error can cause amplification of the numerical solution, leading to numerical instability.

The first order upwind explicit scheme for the advection equation is consistent with the advection equation.

Advection and Diffusion

Consider the transport and diffusion of a dissolved substance in one spatial dimension $$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = D\frac{\partial^2 c}{\partial x^2}$$

with: c = concentration of dissolved substance (mg/m^3)
u = celerity of flow (m/s)
D = diffusion coefficient (m^2/s)



Advection and Diffusion

Consider the transport and diffusion of a dissolved substance in one spatial dimension $\dfrac{\partial c}{\partial t} + u\dfrac{\partial c}{\partial x} = D\dfrac{\partial^2 c}{\partial x^2}$

with: $c =$ concentration of dissolved substance (mg/m^3)
$u =$ celerity of flow (m/s)
$D =$ diffusion coefficient (m^2/s)

Upwind

$$\frac{\partial c}{\partial t} \approx \frac{c_{i,j+1} - c_{i,j}}{\Delta t}, \qquad \frac{\partial^2 c}{\partial x^2} \approx \frac{c_{i-1,j} - 2c_{i,j} + c_{i+1,j}}{\Delta x^2}, \qquad \frac{\partial c}{\partial x} \approx \frac{c_{i,j} - c_{i-1,j}}{\Delta x} \text{ or } \frac{\partial c}{\partial x} \approx \frac{c_{i+1,j} - c_{i,j}}{\Delta x}$$

$O(\Delta t)$ $O(\Delta x^2)$ If $u > 0$ $O(\Delta x)$ If $u < 0$

Courant number, $\rho = u\dfrac{\Delta t}{\Delta x}$, Fourier number, $r = D\dfrac{\Delta t}{\Delta x^2}$

$\boxed{\text{If } u > 0}$

$$\frac{c_{i,j+1} - c_{i,j}}{\Delta t} + u\left(\frac{c_{i,j} - \boxed{c_{i-1,j}}}{\Delta x}\right) = D\left(\frac{c_{i-1,j} - 2c_{i,j} + c_{i+1,j}}{\Delta x^2}\right)$$

$$c_{i,j+1} = c_{i,j} + r\left(c_{i-1,j} - 2c_{i,j} + c_{i+1,j}\right) - \rho\left(c_{i,j} - c_{i-1,j}\right)$$

$$= (r + \boxed{\rho})c_{i-1,j} + (1 - 2r - \rho)c_{i,j} + rc_{i+1,j}$$

Upwind explicit scheme ($u > 0$)

Advection and Diffusion

Consider the transport and diffusion of a dissolved substance in one spatial dimension $$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = D\frac{\partial^2 c}{\partial x^2}$$

with: $c =$ concentration of dissolved substance (mg/m³)
$u =$ celerity of flow (m/s)
$D =$ diffusion coefficient (m²/s)

Upwind

$$\frac{\partial c}{\partial t} \approx \frac{c_{i,j+1} - c_{i,j}}{\Delta t}, \qquad \frac{\partial^2 c}{\partial x^2} \approx \frac{c_{i-1,j} - 2c_{i,j} + c_{i+1,j}}{\Delta x^2}, \qquad \frac{\partial c}{\partial x} \approx \frac{c_{i,j} - c_{i-1,j}}{\Delta x} \text{ or } \frac{\partial c}{\partial x} \approx \frac{c_{i+1,j} - c_{i,j}}{\Delta x}$$

$O(\Delta t)$ $\qquad$ $O(\Delta x^2)$ $\qquad$ If $u > 0$ $\quad$ $O(\Delta x)$ $\quad$ If $u < 0$

Courant number, $\rho = u\frac{\Delta t}{\Delta x}$, Fourier number, $r = D\frac{\Delta t}{\Delta x^2}$

$\boxed{\text{If } u < 0}$

$$\frac{c_{i,j+1} - c_{i,j}}{\Delta t} + u\left(\frac{\boxed{c_{i+1,j}} - c_{i,j}}{\Delta x}\right) = D\left(\frac{c_{i-1,j} - 2c_{i,j} + c_{i+1,j}}{\Delta x^2}\right)$$

$$c_{i,j+1} = c_{i,j} + r(c_{i-1,j} - 2c_{i,j} + c_{i+1,j}) - \rho(c_{i+1,j} - c_{i,j})$$

$$= rc_{i-1,j} + (1 - 2r + \rho)c_{i,j} + (r + \boxed{\rho})c_{i+1,j}$$

Upwind explicit scheme ($u < 0$)

Advection and Diffusion

Consider the transport and diffusion of a dissolved substance in one spatial dimension $\dfrac{\partial c}{\partial t} + u\dfrac{\partial c}{\partial x} = D\dfrac{\partial^2 c}{\partial x^2}$

$$\boxed{\frac{\partial^2 c}{\partial x^2} \approx \frac{c_{i-1,j+1} - 2c_{i,j+1} + c_{i+1,j+1}}{\Delta x^2}, \quad \frac{\partial c}{\partial x} \approx \frac{c_{i,j+1} - c_{i-1,j+1}}{\Delta x}} \qquad [\times \theta\,]$$

weighting

$$\frac{\partial c}{\partial t} \approx \frac{c_{i,j+1} - c_{i,j}}{\Delta t}, \qquad \frac{\partial^2 c}{\partial x^2} \approx \frac{c_{i-1,j} - 2c_{i,j} + c_{i+1,j}}{\Delta x^2}, \qquad \frac{\partial c}{\partial x} \approx \frac{c_{i,j} - c_{i-1,j}}{\Delta x} \qquad [\times (1-\theta)\,]$$

If $u > 0$

Courant number, $\rho = u\dfrac{\Delta t}{\Delta x}$, Fourier number, $r = D\dfrac{\Delta t}{\Delta x^2}$

$$\frac{c_{i,j+1} - c_{i,j}}{\Delta t} + u\left((1-\theta)\left(\frac{c_{i,j} - c_{i-1,j}}{\Delta x}\right) + \theta\left(\frac{c_{i,j+1} - c_{i-1,j+1}}{\Delta x}\right)\right) = D\left(\begin{array}{c} (1-\theta)\left(\dfrac{c_{i-1,j} - 2c_{i,j} + c_{i+1,j}}{\Delta x^2}\right) \\ +\theta\left(\dfrac{c_{i-1,j+1} - 2c_{i,j+1} + c_{i+1,j+1}}{\Delta x^2}\right) \end{array}\right)$$

Upwind implicit scheme ($u > 0$): $\quad -\theta(r+\rho)c_{i-1,j+1} + \big(1 + \theta(2r+\rho)\big)c_{i,j+1} - \theta r c_{i+1,j+1} = \alpha_{i,j}$

where $\quad \alpha_{i,j} = c_{i,j} + (1-\theta)\big((r+\rho)c_{i-1,j} - (2r+\rho)c_{i,j} + rc_{i+1,j}\big)$

Boundary conditions

Assume $u > 0$.

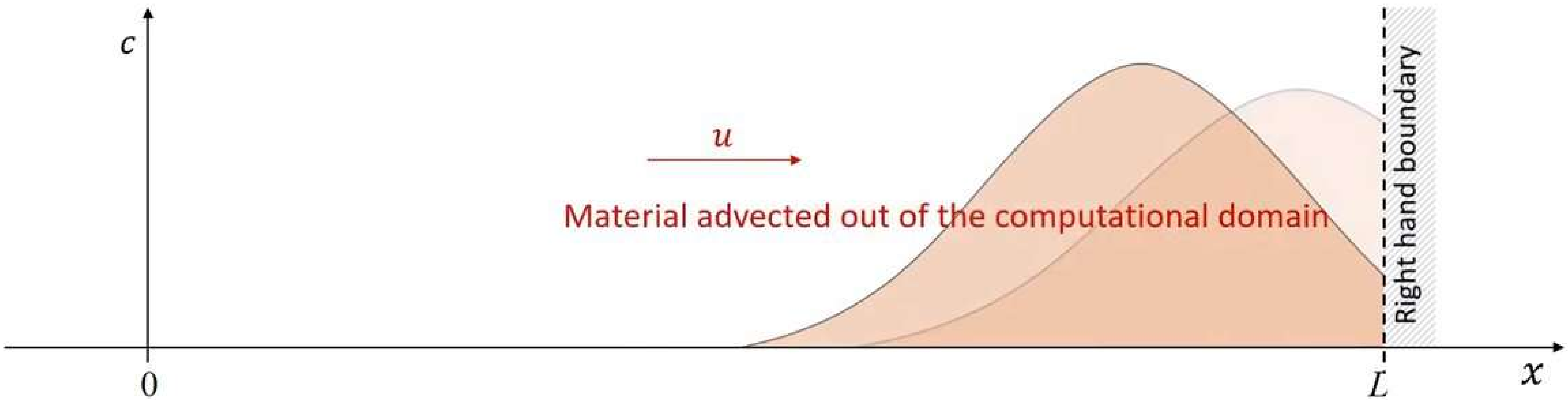
Left hand boundary ($x = 0$): 1. Dirichlet condition $c(0,t) = f(t), t \geq 0$

2. Neumann condition $\dfrac{\partial c}{\partial x}(0,t) = g(t), t \geq 0$

Right hand boundary ($x = L$): $\boxed{\text{\underline{\textit{Absorbing boundary condition}}}}$ = Negligible diffusion at downstream BC

$$\frac{\partial c}{\partial t}(L,t) + u\frac{\partial c}{\partial x}(L,t) = 0$$

[Advection equation]



Boundary conditions

Assume $u > 0$. Absorbing boundary condition at $x = L$ $\qquad \dfrac{\partial c}{\partial t}(L,t) + u\dfrac{\partial c}{\partial x}(L,t) = 0$

Explicit upwind scheme: $c_{i,j+1} = (r+\rho)c_{i-1,j} + (1-2r-\rho)c_{i,j} + rc_{i+1,j}$ where $\rho = u\dfrac{\Delta t}{\Delta x}$, $r = D\dfrac{\Delta t}{\Delta x^2}$

Downstream boundary condition: $\boxed{c_{N,j+1} = \rho c_{N-1,j} + (1-\rho)c_{N,j}}$

```
# Dirichlet upstream and Absorbing BC downstream
yOUT[0] = yUpstream
yOUT[-1] = Cr*yIN[-2] + (1.-Cr)*yIN[-1]
```

Weighted average implicit upwind scheme:

$$-\theta(r+\rho)c_{i-1,j+1} + \big(1+\theta(2r+\rho)\big)c_{i,j+1} - \theta r c_{i+1,j+1} =$$
$$c_{i,j} + (1-\theta)\Big((r+\rho)c_{i-1,j} - (2r+\rho)c_{i,j} + rc_{i+1,j}\Big)$$

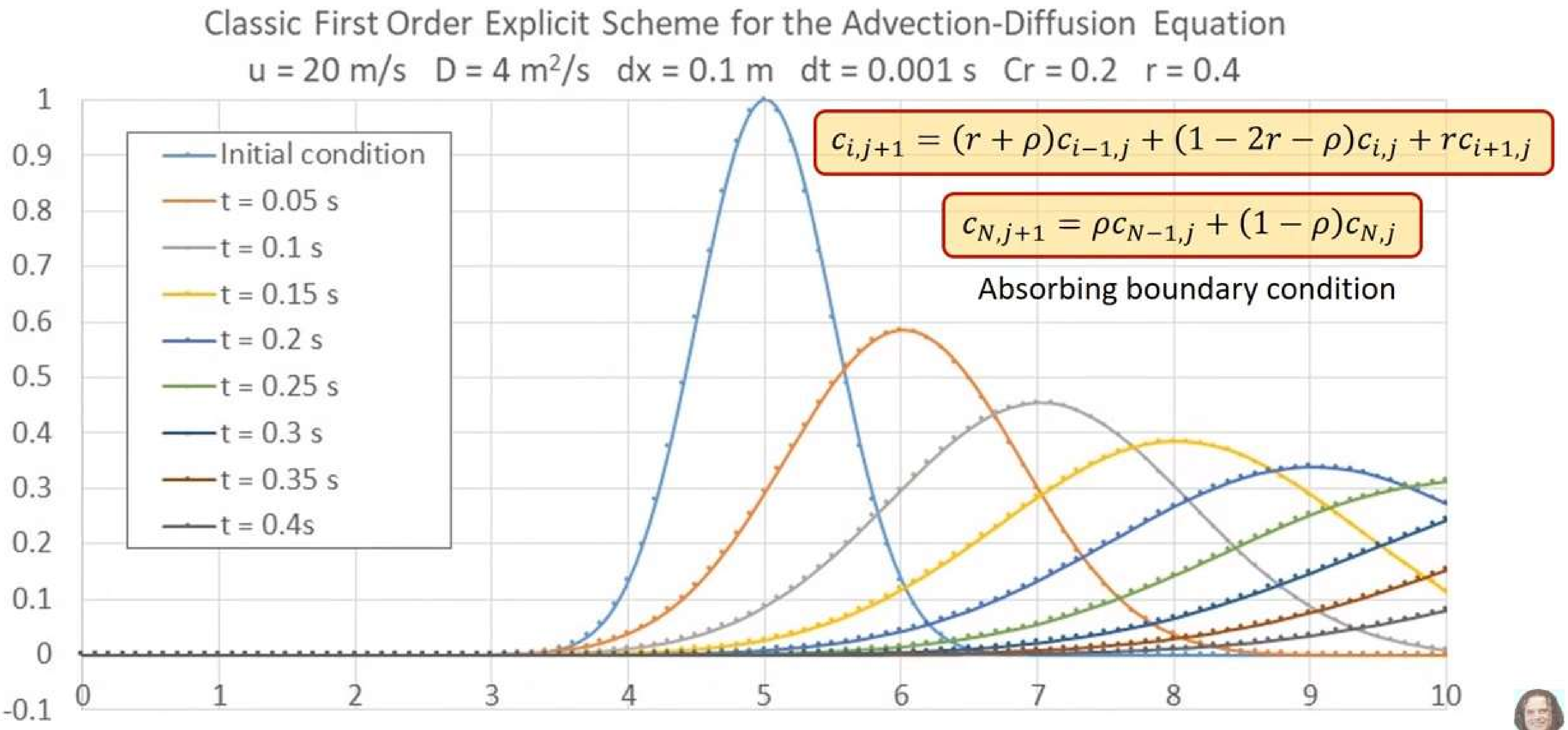
Downstream boundary condition:

$$\boxed{-\rho\theta c_{N-1,j+1} + (1+\rho\theta)c_{N,j+1} = \rho(1-\theta)c_{N-1,j} + \big(1-\rho(1-\theta)\big)c_{N,j}}$$

Exercise: Derive an implicit upwind scheme for the advection-diffusion equation for the case $u < 0$, including the downstream boundary condition.

Finite Difference Schemes for Advection and Diffusion

Explicit scheme for the diffusion equation



Finite Difference Schemes for Advection and Diffusion

```python
import numpy as np
from fipy import *

# space discretization
xL = 0.
xR = 10.
dx = 0.1
delX = xR - xL
nx = int(delX/dx)
mesh = Grid1D(nx=nx, dx=dx) + xL
```

```python
# time discretization
# Advection
Cr = 0.2 # Courant Number: Cr = u*dt/dx
u = 20.
dtAdv = abs(Cr*dx/u)
# Diffusion
Fr = 0.4 # Fourier Number: Fr = D*dt/dx**2
D = 4.
dtDiff = abs(Fr*dx**2/D)
# Advection-Diffusion
dt = min(dtAdv, dtDiff)
delT = 0.2
stepsNbr = int(delT/dt)
```

```python
# unkown's initialization
yMin = 0.
yMax = 1.
yInit = CellVariable(mesh=mesh, value=yMin)
y = CellVariable(mesh=mesh, value=yInit)
```

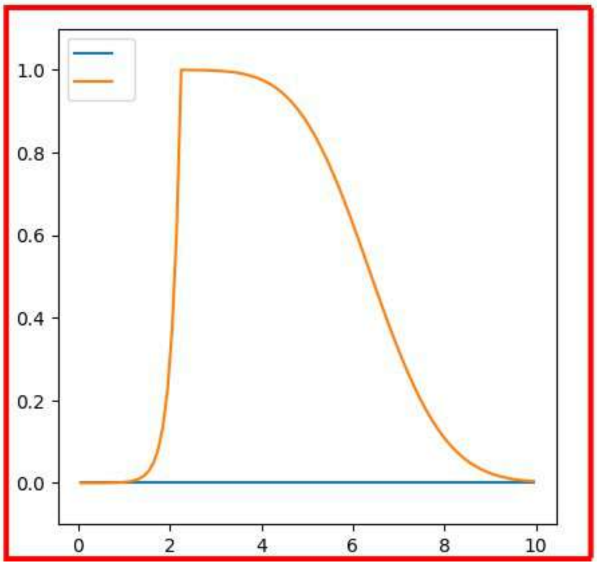
```python
# prepare SourceTerm (at the closest grid cell to xs1)
xs1 = 2.21
ds1 = DistanceVariable(mesh = mesh,
                       value = numerix.sqrt((mesh.x-xs1)**2))
s1Mask = (ds1 == ds1.min())
s1Value = yMax
largeValue = 1e+10
SourceTerm1 = - ImplicitSourceTerm(s1Mask * largeValue) \
              + s1Mask * largeValue * s1Value
```

```python
# Applying noFlow BC
y.faceGrad.constrain((0,), where=mesh.exteriorFaces)

# Defining the discritized equation
eq = (TransientTerm(coeff=1.0)
    + PowerLawConvectionTerm(coeff=(u,))
    == DiffusionTerm(coeff=D) + SourceTerm1)
```

```python
# Iterating in time
for step in range(stepsNbr):
    eq.solve(var=y, dt=dt)

# Ploting the results
viewer = Viewer(vars = (yInit, y),
                datamin = min(yInit.min(), y.min())-.1,
                datamax = max(yInit.max(), y.max())+.1)
viewer.plot()
```

```python
print('dx = ', dx, '\n'
      'delX = ', delX, '\n'
      'nx = ', nx, '\n'
      'Cr = ', Cr, '\n'
      'u = ', u, '\n'
      'dtAdv = ', dtAdv, '\n'
      'Fr = ', Fr, '\n'
      'D = ', D, '\n'
      'dtDiff = ', dtDiff, '\n'
      'dt = min(dtAdv,dtDiff) = ', dt, '\n'
      'delT = ', delT, '\n'
      'stepsNbr = ', stepsNbr)
```

```
dx =  1.00e-01
delX =  10.0
nx =  100
Cr =  0.2
u =  20.0
dtAdv =  1.00e-03
Fr =  0.4
D =  4.0
dtDiff =  1.00e-03
dt = min(dtAdv,dtDiff) =  1.00e-03
delT =  0.2
stepsNbr =  199
```



```python
import numpy as np
from fipy import *

# space discretization
w = h = 1000 # m
dx = dy = 10 # m
nx, ny = int(w/dx), int(h/dy)
mesh = Grid2D(nx=nx, dx=dx, ny=ny, dy=dy)
print('cellsNbr = ', mesh.globalNumberOfCells)

```

```python
# time discretization
# Advection
vx = vy = 10. # m/s
Cr = 0.5
dtAdv = abs(Cr*dx/max(vx,vy))
# Diffusion
D = 80.
Fr = 0.5 # Courant Number: Cr = u*dt/dx
dtDiff = abs(Fr*dx/max(vx,vy))
dt = min(dtAdv,dtDiff)
stepsNbr = 100

```

```python
def sourceMask(x, y, xcs, ycs, width, isRectangular = False):
    # rectangular
    if isRectangular:
        return ((xcs-width<x) & (x<xcs+width) \
            & (ycs-width<y) & (y<ycs+width))
    # circular
    else:
        ds = DistanceVariable(mesh=mesh,
                              value=numerix.sqrt((x-xcs)**2+(y-ycs)**2))
        return (ds <= width)

```

```python
# prepare SourceTerm (at the closest grid cell to xs1)
xs1, ys1 = 250, 250
ds1 = DistanceVariable(mesh=mesh,
                       value=numerix.sqrt((mesh.x-xs1)**2+(mesh.y-ys1)**2))
s1CellMask = (ds1 == ds1.min())
xcs1, *_ = mesh.x[s1CellMask]
ycs1, *_ = mesh.y[s1CellMask]
s1Width = 40
s1Mask = sourceMask(x=mesh.x,
                    y=mesh.y,
                    xcs=xcs1,
                    ycs=ycs1,
                    width=s1Width,
                    isRectangular=False)
zs1 = 1200.
```

```python
# unkown's initialization
zInit = CellVariable(mesh=mesh, value=200.)
zInit.setValue(value=zs1, where=s1Mask)
z = CellVariable(mesh=mesh, value=zInit)

```

General Conservation Equation

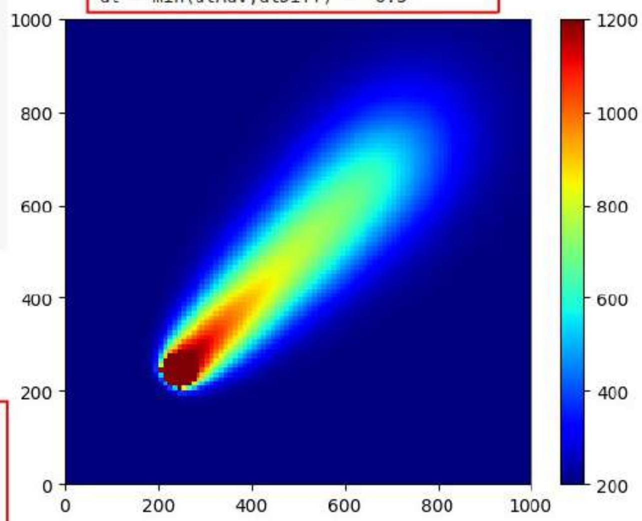
$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\text{transient}} + \underbrace{\nabla\cdot(\vec{u}\phi)}_{\text{convection}} = \underbrace{\left[\nabla\cdot(\Gamma_i\nabla)\right]^n\phi}_{\text{diffusion}} + \underbrace{S_\phi}_{\text{source}}$$

```python
# closed BC (Neumann), i.e. grad(y) = 0
z.faceGrad.constrain((0.,), where=mesh.exteriorFaces)

# Defining the discritized equation
convCoeff = (vx, vy)
eq = (TransientTerm(coeff=1.0)
    + PowerLawConvectionTerm(coeff=convCoeff)
    == DiffusionTerm(coeff=D)
    - ImplicitSourceTerm(s1Mask*1e10) + s1Mask*1e10*zs1)

```

```python
# Ploting the results
viewer = Viewer(vars = (z),
                datamin = z.min()-.1,
                datamax = z.max()+.1)

# Iterating in time
for step in range(stepsNbr):
    eq.solve(var=z, dt=dt)
    viewer.plot()

```

```python
print('dx = ', dx, '\n'
      'dy = ', dy, '\n'
      'nx = ', nx, '\n'
      'ny = ', ny, '\n'
      'vx = ', vx, '\n'
      'vy = ', vy, '\n'
      'Cr = ', Cr, '\n'
      'dtAdv = ', dtAdv, '\n'
      'D = ', D, '\n'
      'Fr = ', Fr, '\n'
      'dtDiff = ', dtDiff, '\n'
      'dt = min(dtAdv,dtDiff) = ', dt, '\n'
      'stepsNbr = ', stepsNbr, '\n'
      'xcs1 = ', xcs1, '\n'
      'ycs1 = ', ycs1, '\n'
      'zs1 = ', zs1, '\n')
```

```
cellsNbr =  10000
dx =  10
...
dt = min(dtAdv,dtDiff) =  0.5
```



```python
import numpy as np
from fipy import *

# space discretization
dx = 60.
geoFileName = '/content/drive/MyDrive/lago.geo.txt'
mesh = Gmsh2D(geoFileName)
print('cellsNbr = ', mesh.globalNumberOfCells)

```

...

```python
# prepare SourceTerm (at the closest grid cell to xs1)
xs1, ys1 = 1500, 200
```

...

```python
print('dx = ', dx, '\n'
      'vx = ', vx, '\n'
      'vy = ', vy, '\n'
      'Cr = ', Cr, '\n'
      'dtAdv = ', dtAdv, '\n'
      'D = ', D, '\n'
      'Fr = ', Fr, '\n'
      'dtDiff = ', dtDiff, '\n'
      'dt = min(dtAdv,dtDiff) = ', dt, '\n'
      'stepsNbr = ', stepsNbr, '\n'
      'xcs1 = ', xcs1, '\n'
      'ycs1 = ', ycs1, '\n'
      'zs1 = ', zs1, '\n')
```

```
mesh.globalNumberOfCells =  1466
dx =  60.0
vx =  10.0
vy =  10.0
Cr =  0.5
dtAdv =  3.0
D =  80.0
Fr =  0.5
dtDiff =  3.0
dt = min(dtAdv,dtDiff) =  3.0
stepsNbr =  10
xcs1 =  1511.5147564586898
ycs1 =  196.30208451099128
zs1 =  1200.0
```