Méthodes Numériques

Cours 5 :

Problèmes instationnaires 1D

Dr. Barbara PERRI

barbara.perri@universite-paris-saclay.fr

Plan de l'UE

Idée générale :

Au premier semestre, on va introduire les notions de base, et s'intéresser en détails à une méthode numérique précise

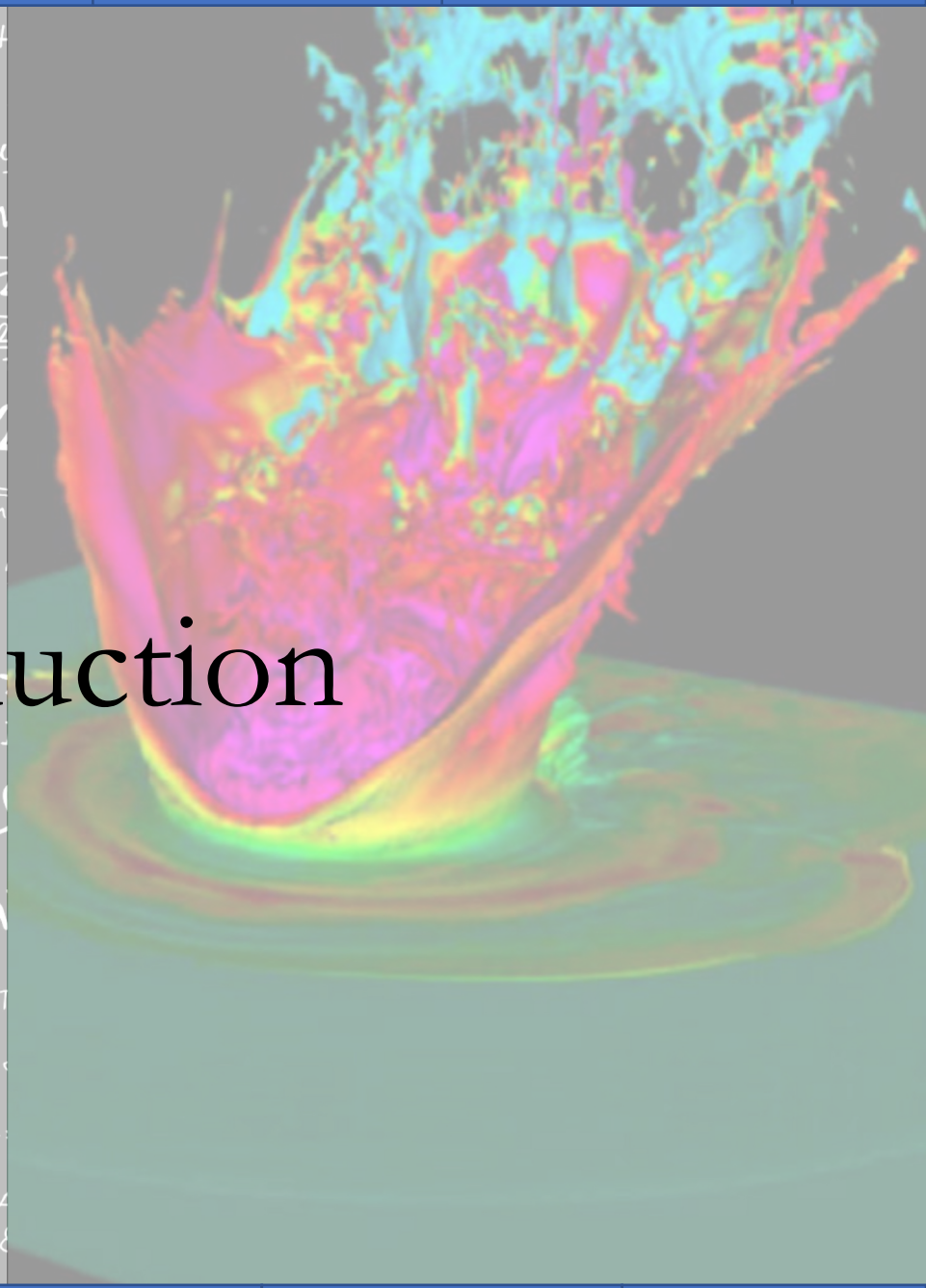
Tous les jeudi matin (8h45-12h45) au bâtiment 625

21 Novembre : Cours 1 + Cours 2
4 Décembre : Cours 3 + TP
5 Décembre : Cours 4 + TP
12 Décembre : **Cours 5** + TP
19 Décembre : Cours 6 + TP
9 Janvier : Cours 7 + TP
16 Janvier : Cours 8 + TP
23 Janvier : TP
30 Janvier : Examen

Modalités d'évaluation :

TPs + examen oral (question de cours + exercice)

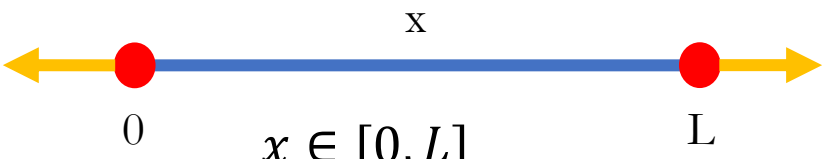
Introduction



Hypothèse : 1D

Pour cette première application, on ne va s'intéresser qu'à des problèmes à 1 dimension :

Domaines sous forme de segments



$x \in [0, L]$

δD : 2 points

Gauche : $x = 0$

Droite : $x = L$

Cela permet de simplifier l'écriture du problème dans un premier temps

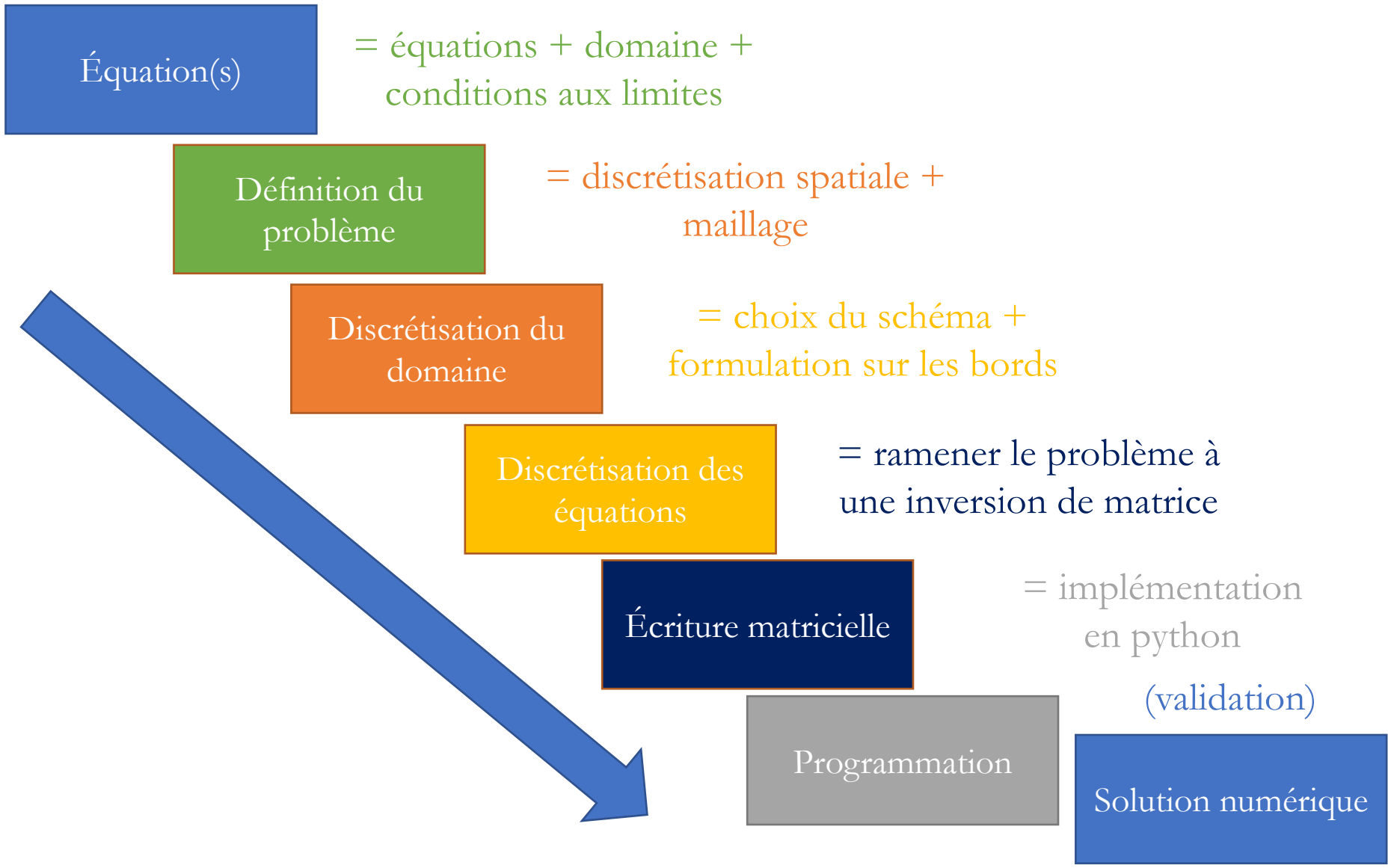
Équations différentielles ordinaires (EDO)

= les inconnues ne dépendent que d'une seule variable

$$\frac{df(x)}{dx} + f(x) = g(x)$$

MAIS la même méthode sera généralisable aux EDPs

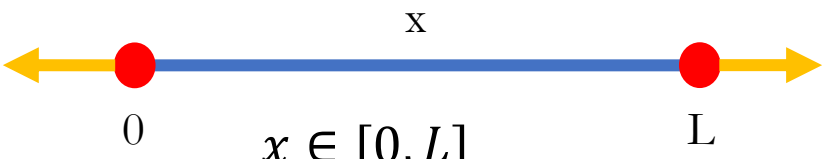
Cycle des méthodes numériques : Stationnaire



Hypothèse : 1D instationnaire

Pour cette première application, on ne va s'intéresser qu'à des problèmes à 1 dimension **spatiale** :

Domaines sous forme de segments



$x \in [0, L]$

δD : 2 points

Gauche : $x = 0$

Droite : $x = L$

Cela permet de simplifier l'écriture du problème dans un premier temps

Équations différentielles ordinaires (EDP)

= les inconnues dépendent de 2 variables : **spatiale** + **temps**

$$\frac{\partial f}{\partial t}(x,t) + \frac{\partial f}{\partial x}(x,t) = g(x,t)$$

On ne peut plus appliquer les méthodes propres aux EDOs !

Objectif

En 1D stationnaire, on a établi une méthode qui permet de calculer une solution numérique pour un domaine spatial donné

$$x \mapsto f(x)$$

→ En 1D **instationnaire**, on veut désormais une solution numérique pour un domaine spatial donné **et pour un temps donné**

→ La solution va évoluer dans le temps !

$$(x, t) \mapsto f(x, t)$$

→ Comment doit-on alors modifier la méthode vue précédemment ?

→ Comment représenter la solution et la valider ?

Évolution temporelle

Étape 1 : Définition du problème

1

Avant de passer à la partie numérique, il faut s'assurer
d'avoir bien défini le problème physique/mathématique continu

Premier exemple :
Équation d'advection

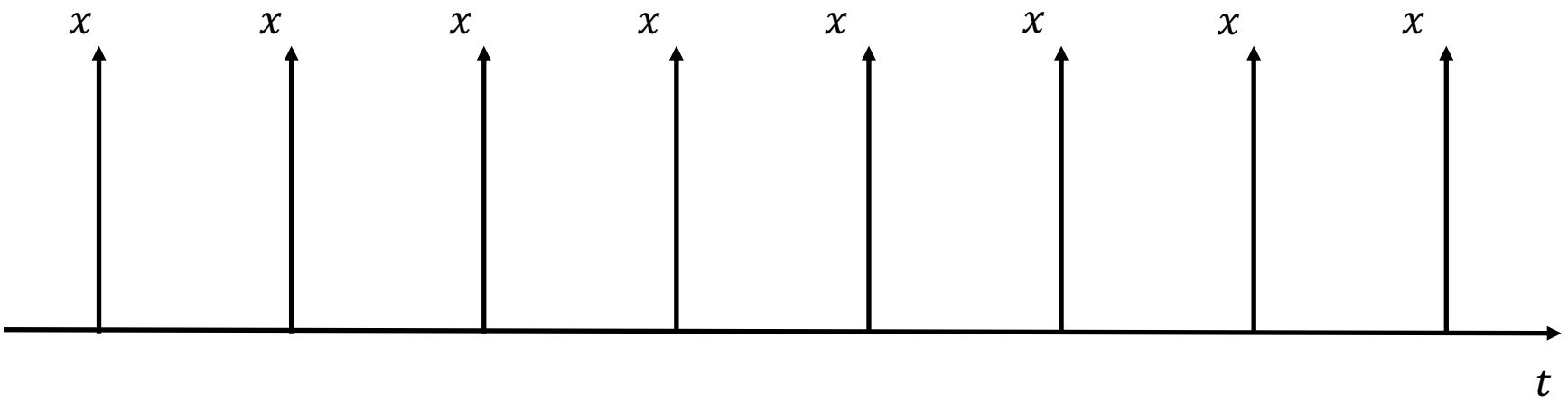
Équation :	$\frac{\partial f}{\partial t}(x,t) + a\frac{\partial f}{\partial x}(x,t) = 0$	(problème d'**advection**)
Inconnue :	$f(x,t)$	(fonction réelle 1D **+ temps**)
Domaines :	$x \in [0,L]$ $\quad$ $\boldsymbol{t \in [0,T]}$	(segments 1D cartésiens)
Condition aux limites :	$f(\mathbf{0},\boldsymbol{t}) = F(-at)$	(CL de type Dirichlet)
Condition initiale :	$\boldsymbol{f(x,0) = F(x)}$	**(fonction d'initialisation temporelle)**

Visualisation des domaines

1

Domaine spatial 1D :

MAIS pour chaque pas de temps
→ Conjugué à un domaine temporel 1D :



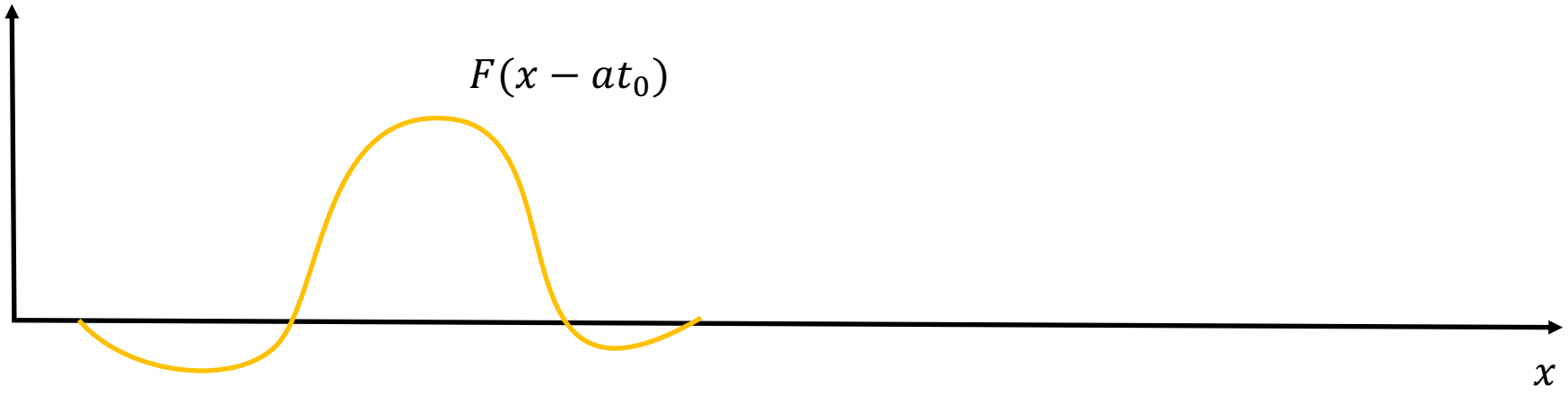
Solution de l'équation d'advection

1

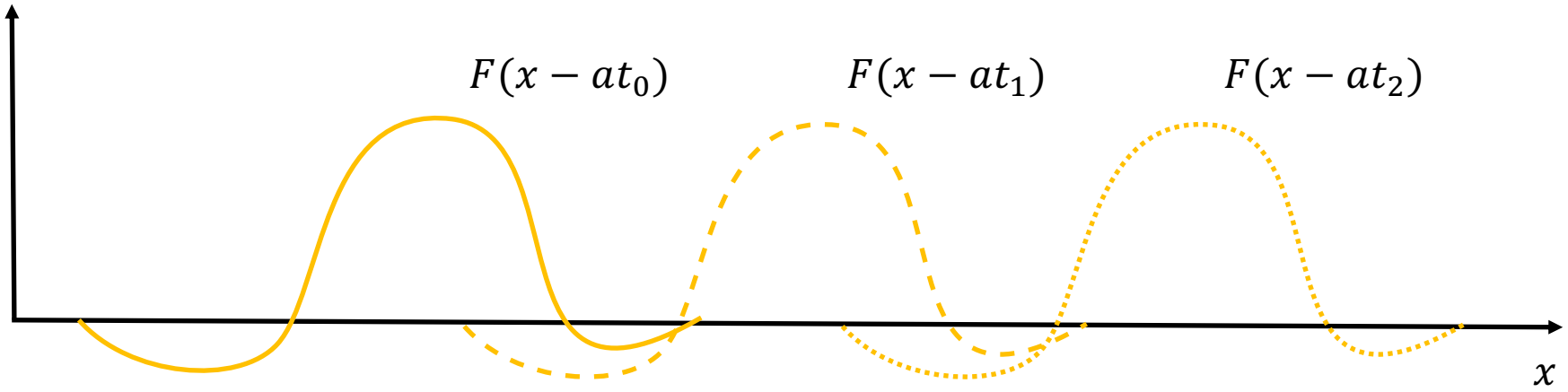
Solution : $f(x,t) = F(x - at)$

→ La fonction initiale est advectée sans déformation

Visualisation spatiale :



Visualisation temporelle :



Étape 2 : Discrétisation du domaine

2

On va maintenant passer d'un milieu continu à discret à l'aide d'un maillage :

Premier exemple :

Maillage uniforme

Nombres de points : $M+1$ en espace, $\boldsymbol{N+1}$ **en temps** (résolution)

Pas d'espace : $\delta x = L/M$ $\quad \boldsymbol{\delta t = T/N}$

Numérotation : $0 = x_0 < x_1 < x_2 < \cdots < x_M = L$, $x_i = i\delta x, \forall i \in [0, M]$ (noeuds du maillage spatial)

$\boldsymbol{0 = t_0 < t_1 < t_2 < \cdots < t_N = T}$, $\boldsymbol{t_n = n\delta t, \forall n \in [0, N]}$ (noeuds du maillage temporel)

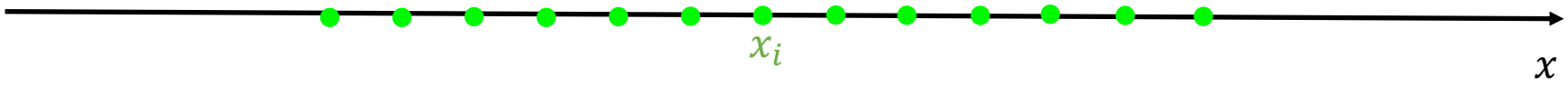
Équivalence discret/continu : $\boldsymbol{f(x_i, t_n) = f_i^n}$ (notation)

Valeurs nodales inconnues : $f_0^1, f_1^1, f_2^1, \ldots, f_M^1$ ... $f_0^N, f_1^N, f_2^N, \ldots, f_M^N$ (valeurs aux noeuds)

Visualisation des maillages

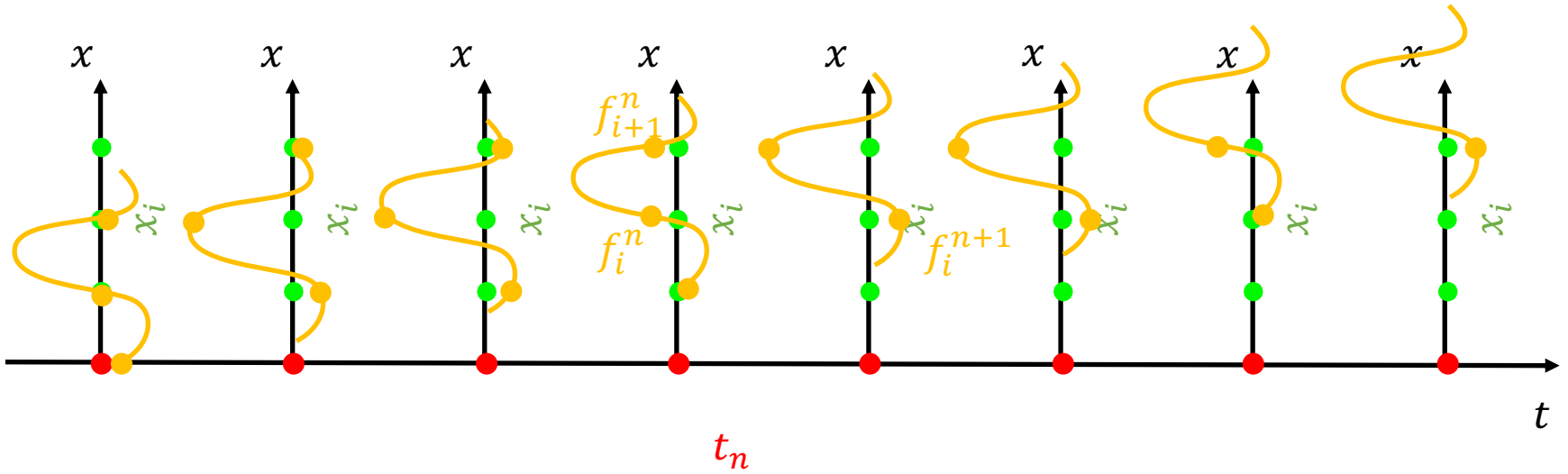
2

Discrétisation 1D :



MAIS pour chaque pas de temps

→ Conjugué à un maillage temporel 1D :



Étape 3 : Discrétisation des équations (principe)

3

Une fois qu'on a discrétisé le domaine de calcul et les quantités physiques,
il faut encore choisir comment approximer les dérivées = choix du schéma numérique
→ On a deux choix à faire : dérivées spatiales **+ dérivées temporelles**

$$\frac{\partial^k f_i^n}{\partial x^k} \approx \sum_m a_m f_{i+m}^n \qquad \frac{\partial^k f_i^n}{\partial t^k} \approx \sum_l b_l f_i^{n+l}$$

(discrétisation spatiale) (discrétisation temporelle)

$$\sum_{j \in J} c_j f_j^{n+1} = \sum_{k \in K} d_k f_k^n + \cdots$$

(schéma complet)

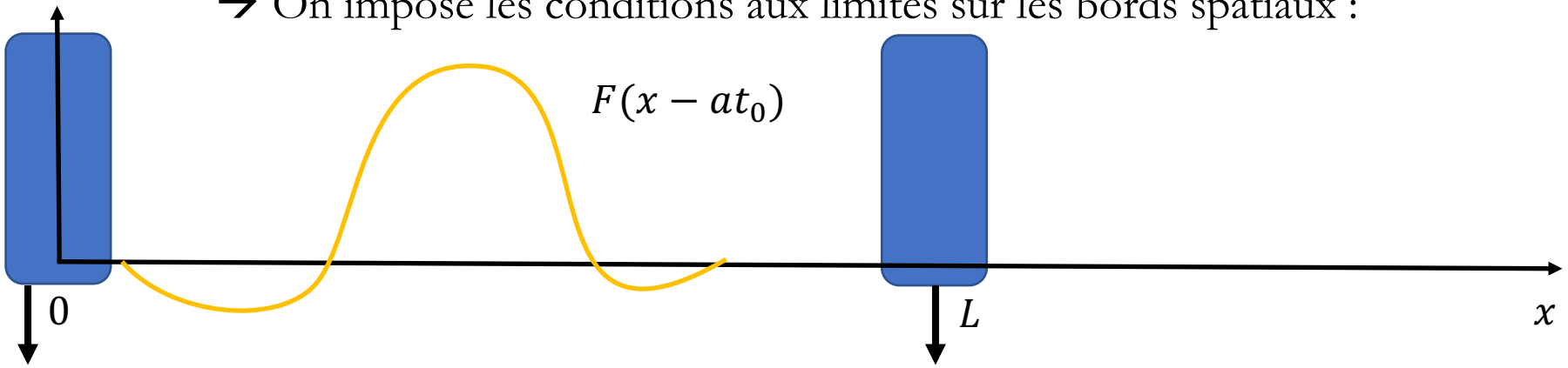
La version discrétisée de l'EDP permet alors de gérer l'avancement temporel
= chercher des équations algébriques pour tous les points qui permettent
de trouver le champ à tous les temps suivants
(on verra plus tard comment le représenter graphiquement)

Étape 3 : Discrétisation des équations (bords)

3

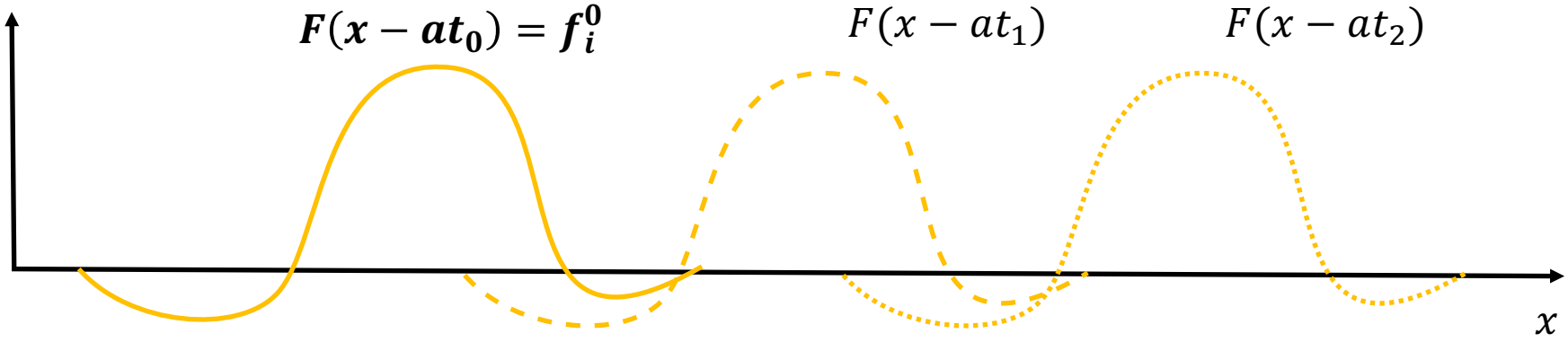
À nouveau, on fait la distinction entre l'intérieur du domaine et les bords

→ On impose les conditions aux limites sur les bords spatiaux :



Dirichlet (f_0^n) ou Neuman $(\partial_x f_0^n)$ Dirichlet (f_M^n) ou Neuman $(\partial_x f_M^n)$

→ Pour les bords temporels, il faut imposer une condition initiale :



→ Il faut autant de dérivées que l'ordre de la dérivée la plus élevée de l'équation – 1 !

Étape 3 : Discrétisation des équations (domaine)

3

Premier exemple :

Schéma UPWIND (Forward en temps, Backward en espace)

On cherche à discrétiser l'équation suivante :

$$\frac{\partial f}{\partial t}(x,t) + a\frac{\partial f}{\partial x}(x,t) = 0 \quad \Rightarrow \quad \frac{\partial f_i^n}{\partial t} + a\frac{\partial f_i^n}{\partial x} = 0$$

Discrétisation spatiale : (Backward)

$$\frac{\partial f_i^n}{\partial x} \approx \frac{f_i^n - f_{i-1}^n}{\delta x}$$

(maillage uniforme)

Discrétisation temporelle : (Forward)

$$\frac{\partial f_i^n}{\partial t} \approx \frac{f_i^{n+1} - f_i^n}{\delta t}$$

$$\frac{f_i^{n+1} - f_i^n}{\delta t} + a\frac{f_i^n - f_{i-1}^n}{\delta x} = 0 \quad \Rightarrow \quad f_i^{n+1} = f_i^n - \frac{a\delta t}{\delta x}(f_i^n - f_{i-1}^n)$$

$$\boxed{\forall i \in [1, M], \forall n \in [0, N-1]}$$

Étape 3 : Discrétisation des équations (graphique)

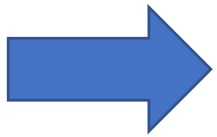
3

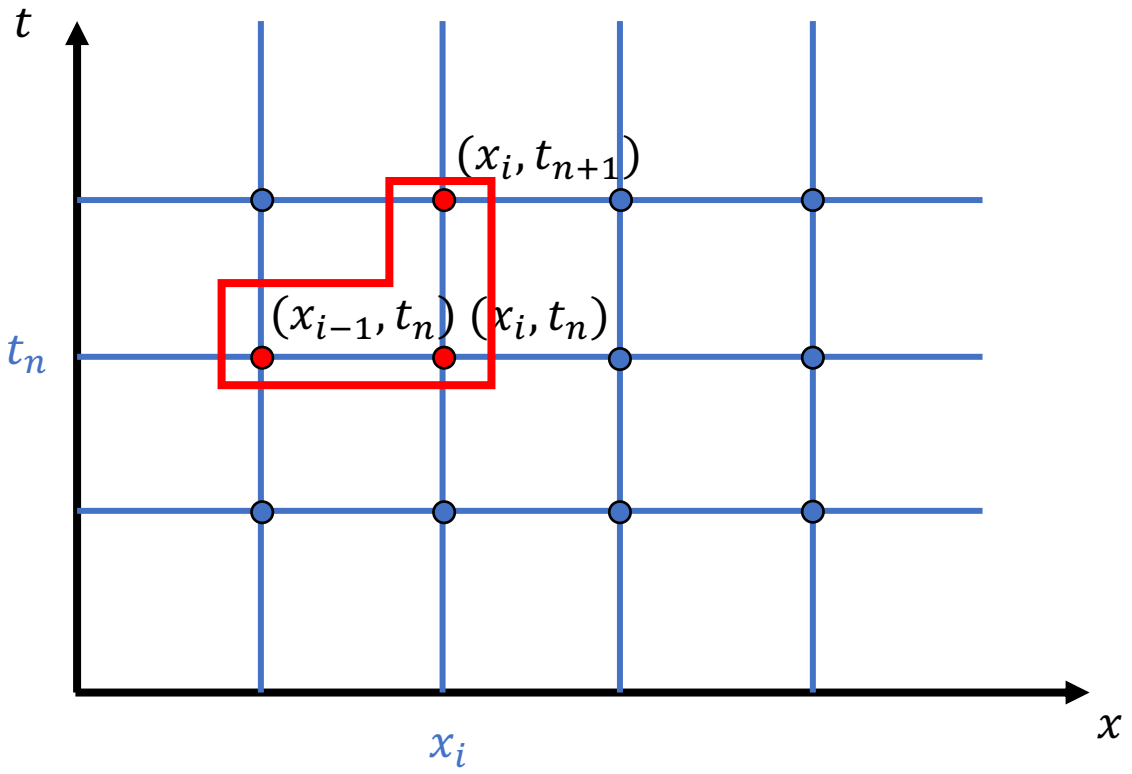
Premier exemple :

Schéma UPWIND (Forward en temps, Backward en espace)

On peut représenter un schéma de manière graphique de la façon suivante :

$$f_i^{n+1} = f_i^n - \frac{a\delta t}{\delta x}(f_i^n - f_{i-1}^n)$$





→ Cela permet de visualiser immédiatement les points impliqués et les niveaux du schéma

Étape 3 : Discrétisation des équations (bords)

3

Premier exemple :

Schéma UPWIND (Forward en temps, Backward en espace)

Condition aux limites :

→ Condition de Dirichlet à l'entrée du domaine :

$$f_0^{n+1} = F(-at_{n+1}), \forall n \in [0, N-1]$$

Condition initiale :

→ La dérivée la plus élevée étant d'ordre 1,
on a besoin de 1 condition portant sur la dérivée d'ordre 0 :

$$f_i^0 = F(x_i), \forall i \in [0, M]$$

→ Le problème final est donc sous la forme :

$$\begin{cases} f_i^0 = F(x_i) & \forall n \in [0, N-1] \\ f_0^{n+1} = F(-at_{n+1}) & \forall i \in [0, M] \\ f_i^{n+1} = (1-C) f_i^n + C\, f_{i-1}^n \end{cases} \qquad \left(C = \frac{a\delta t}{\delta x}\right)$$

Étape 4 : Écriture matricielle (principe)

4

$$f_i^{n+1} = (1-C)f_i^n + C\, f_{i-1}^n$$ → Schéma explicite ! (itératif)

→ On va quand même utiliser la forme matricielle pour traiter la partie en espace :

x_0 $f_0^{n+1} = F(-at_{n+1})$

x_1 $f_1^{n+1} = (1-C)f_1^n + Cf_0^n$

x_2 $f_2^{n+1} = (1-C)f_2^n + Cf_1^n$

… …

x_i $f_i^{n+1} = (1-C)f_i^n + Cf_i^n$

… …

x_{M-1} $f_{M-1}^{n+1} = (1-C)f_{M-1}^n + Cf_{M-2}^n$

x_M $f_M^{n+1} = (1-C)f_M^n + Cf_{M-1}^n$

$(M+1)\times(M+1)$

$AF = G$

$$\begin{bmatrix} f_0 \\ \dots \\ f_i \\ \dots \\ f_M \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{C_0} \\ \dots \\ g_i \\ \dots \\ \boldsymbol{C_L} \end{bmatrix}$$

$F^{n+1} = \varepsilon F^n + G$

$$\begin{bmatrix} f_0^{n+1} \\ \dots \\ f_i^{n+1} \\ \dots \\ f_M^{n+1} \end{bmatrix} \quad (M+1)\times(M+1) \quad \begin{bmatrix} f_0^n \\ \dots \\ f_i^n \\ \dots \\ f_M^n \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{C_0} \\ \dots \\ g_i \\ \dots \\ \boldsymbol{C_L} \end{bmatrix}$$

Étape 4 : Écriture matricielle (exemple)

Premier exemple :

Schéma UPWIND (Forward en temps, Backward en espace)

$$
\begin{array}{ll}
x_0 & f_0^{n+1} = F(-at_{n+1}) \\
x_1 & f_1^{n+1} = (1-C)f_1^n + Cf_0^n \\
x_2 & f_2^{n+1} = (1-C)f_2^n + Cf_1^n \\
\dots & \dots \\
x_i & f_i^{n+1} = (1-C)f_i^n + Cf_i^n \\
\dots & \dots \\
x_{M-1} & f_{M-1}^{n+1} = (1-C)f_{M-1}^n + Cf_{M-2}^n \\
x_M & f_M^{n+1} = (1-C)f_M^n + Cf_{M-1}^n
\end{array}
$$

$\mathcal{E}$

$$
\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{M-1} \\ x_M \end{array}
\begin{array}{c}
\begin{array}{cccccc} f_0^n & f_1^n & f_2^n & \dots & f_{M-1}^n & f_M^n \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & \dots & 0 & 0 \\
C & 1-C & 0 & \dots & 0 & 0 \\
0 & C & 1-C & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & \dots & 1-C & 0 \\
0 & 0 & 0 & 0 & C & 1-C
\end{bmatrix}
\end{array}
$$

G

$$
\begin{bmatrix} F(-at_{n+1}) \\ \dots \\ 0 \\ \dots \\ 0 \end{bmatrix}
$$

Condition initiale

$$
\begin{bmatrix} f_0^0 \\ \dots \\ f_i^0 \\ \dots \\ f_M^0 \end{bmatrix} = \begin{bmatrix} F(x_0) \\ \dots \\ F(x_i) \\ \dots \\ F(x_M) \end{bmatrix}
$$

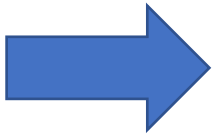
Étape 5 : Programmation

5

Il ne reste plus qu'à demander à l'ordinateur de résoudre le problème !

→ Sous forme matricielle :

$$F^0 = G^0$$
$$F^{n+1} = \varepsilon F^n + G$$



On n'a plus qu'à boucler sur n

Étapes informatiques :

1. Définir le vecteur G,
2. Définir la matrice ε,
3. Écrire la condition initiale,
4. Boucler sur l'indice temporel.

<u>Premier exemple :</u>

En Python, on utilisera les fonctions optimisées de calcul vectoriel (np.dot)

Au-delà : Validation

✓

Premier exemple :

Équation d'advection

Schéma UPWIND (Forward en temps, Backward en espace)

On choisit la fonction initiale :

$$F(x) = \cos(x)$$

→ Ma fonction solution doit être un transport de mon cosinus sans déformation

→ On n'a plus qu'à choisir les paramètres du domaine :

$L = 10$ $T = 10$ $\delta x = 0.01$ $\delta t = 0.01$

→ On fait un premier exemple avec $a = 0.9$

Exemple avec code (I)

1

Étape 1 : Définition du problème

Domaines : $x \in [0, L]$ $t \in [0, T]$

```
#taille du domaine/temps final
L=10
T=10
```

Conditions aux limites :

Condition initiale :

```
#5. initialisation  F(x) = cos (x) à l'instant t=0
f=np.cos(x)
```

Exemple avec code (II)

2

Étape 2 : Discrétisation du domaine

Nombre de points : $M+1, N+1$

```
#nombre de points et temps discrets (-1)
M=100
N=100
```

Pas d'espace et de temps : $\delta x = \frac{L}{M}, \delta t = \frac{T}{N}$

```
#pas de temps, pas d'espace
dx=L/M
dt=T/N
```

Numérotation :

$$0 = x_0 < x_1 < x_2 < \cdots < x_M = L$$

$$0 = t_0 < t_1 < t_2 < \cdots < t_N = T$$

```
#maillage/temps discrets
x=np.linspace(0,L,M+1)
t=np.linspace(0,T,N+1)
```

Exemple avec code (III)

3 4

Étape 3 & 4 : Discrétisation des équations et Écriture matricielle
(la discrétisation des équations se retrouve dans la forme de la matrice)

Définition du vecteur G :

```
#mise à jour du vecteur
F[0]=np.cos(-a*t[n+1])
```

Définition de la matrice d'évolution :

```
C=a*dt/dx
```

```
#4.matrice E d'évolution et vecteur F
E=np.zeros([M+1,M+1])
for j in range(1,M+1):    #pour j allant de 1 à M
    E[j,j-1]=C
    E[j,j]=1-C
```

Exemple avec code (IV)

5 ✓

Étape 5 : Programmation et Validation

Boucle temporelle :

```
#6. boucle d'avancement temporel
for n in range(N):   #pour n allant de 0 a N-1
```

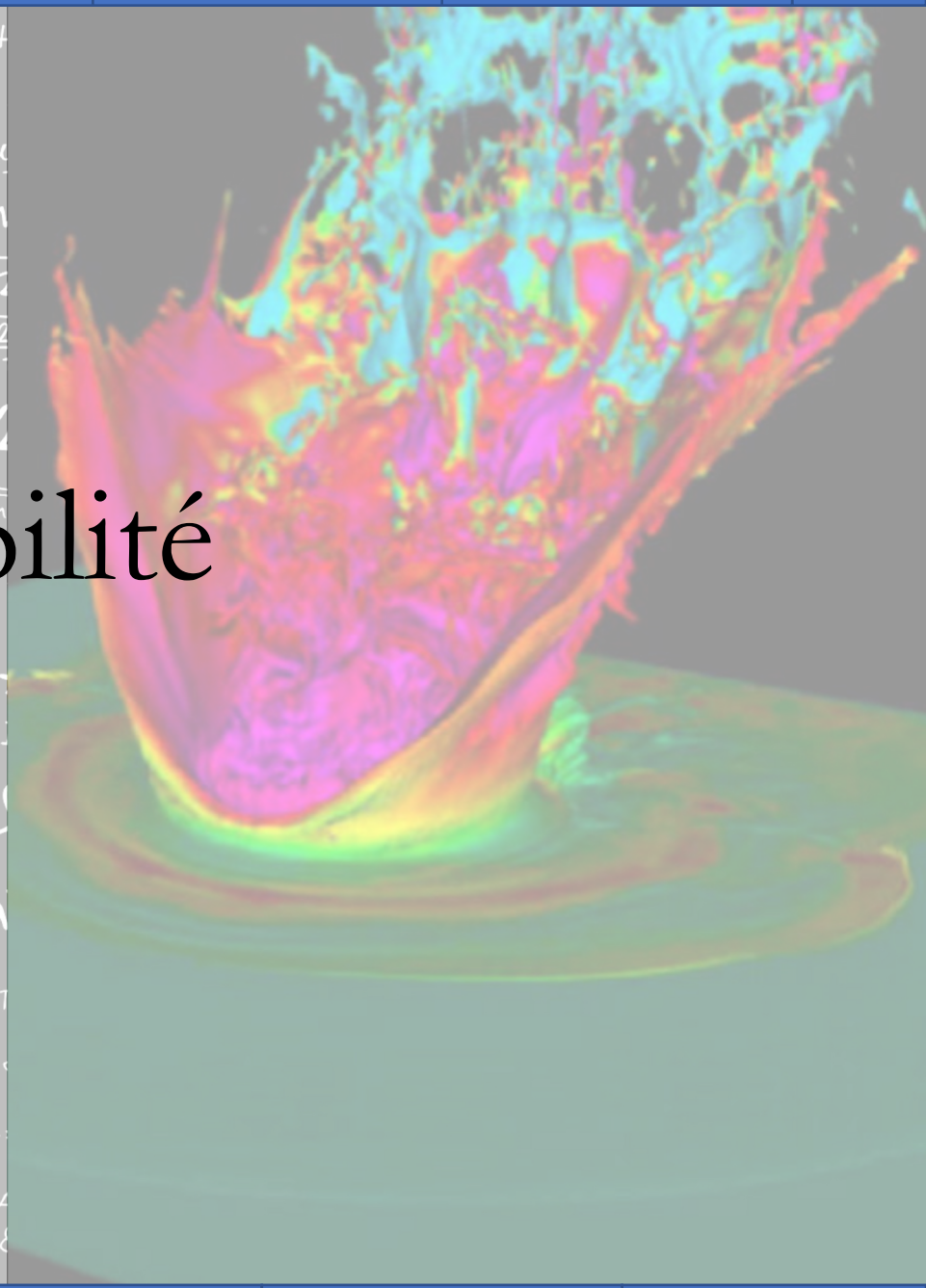
```
#mise à jour du vecteur F
F[0]=np.cos(-a*t[n+1])

#mise à jour du vecteur f
f=np.dot(E,f)+F
```

Comparaison avec solution analytique :

```
#solution exacte
f_ex=np.cos(x-a*t[n+1])
```

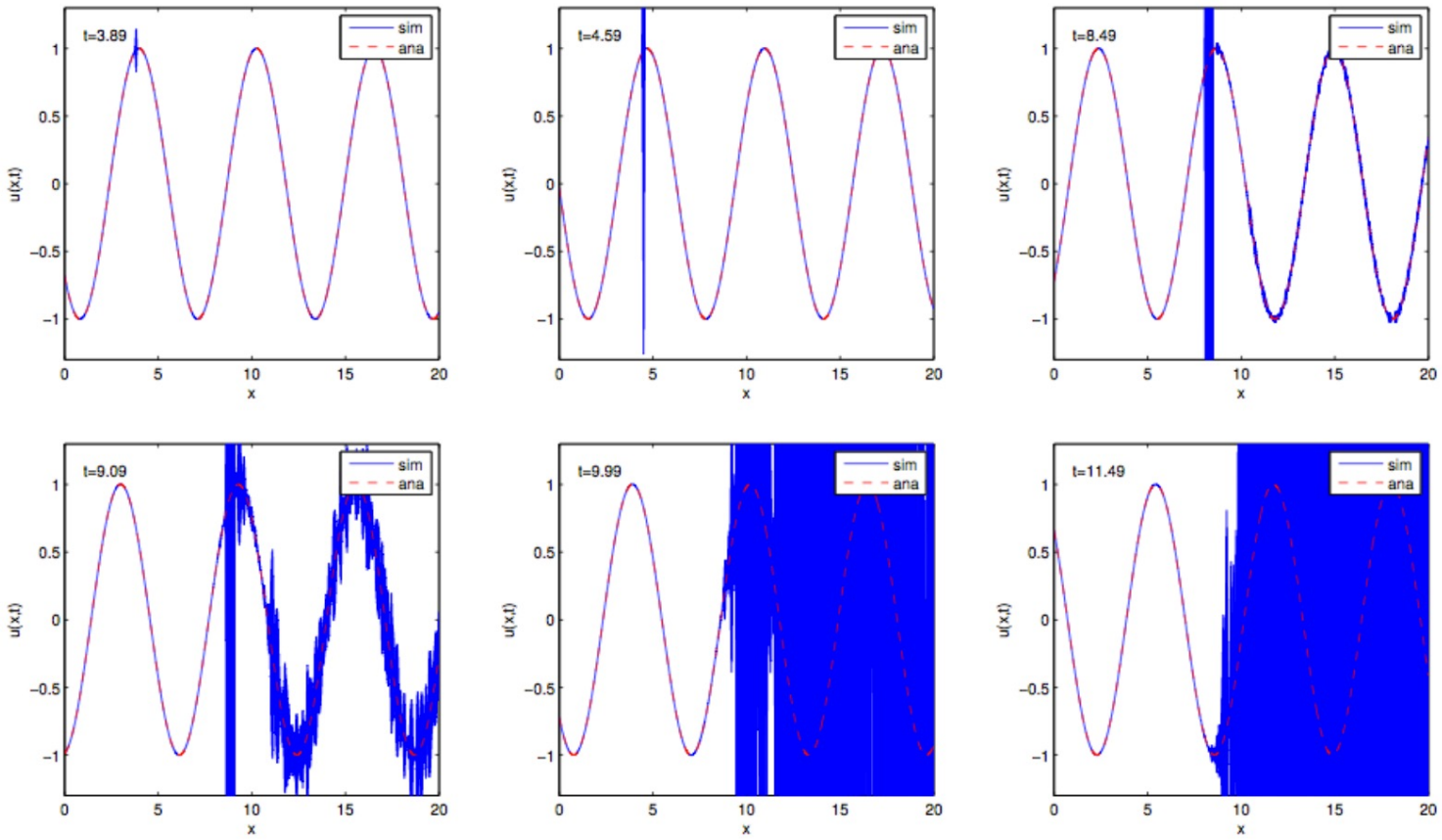
Question : Est-ce que notre méthode fonctionne pour tous les paramètres a ?

Stabilité



Exemple d'instabilité numérique

Si on augmente a au-delà de 1, on voit que les erreurs s'accumulent jusqu'à faire exploser la solution = instabilité numérique :



→ Comment empêcher que cela se produise ?

Comment garantir la stabilité ?

Le schéma UPWIND est un schéma stable sous la condition :

$$C = \frac{a\delta t}{\delta x} \leq 1$$

Pour garantir la stabilité, on peut donc jouer sur 3 paramètres :

→ Réduire a (si le problème le permet et à une résolution donnée)

→ Réduire δt (mais cela ralentit la résolution du problème)

→ Augmenter δx (pour introduire de la diffusion numérique mais moins précis)

Question : Laquelle de ces 3 méthodes vous paraît la plus efficace dans ce cas précis ?

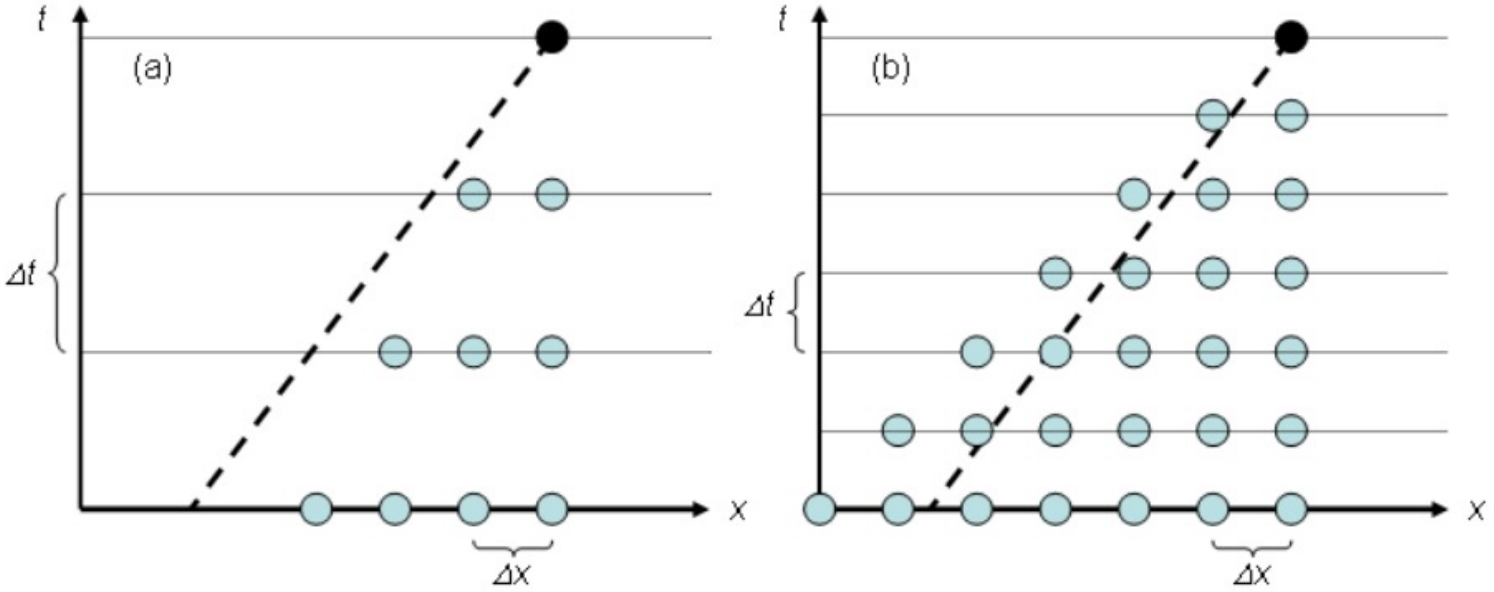
Représentation graphique de la stabilité

Représentons une solution instable vs. stable dans le plan du schéma :

Solution instable

Solution stable

[Durran 1999]



C'est la manière « physique » de comprendre la condition de stabilité :

$$C = \frac{a\delta t}{\delta x} \leq 1 \Rightarrow v_{num} = \frac{\delta x}{\delta t} \geq a = v_{phys} \Rightarrow t_{num} = \delta t \leq \frac{\delta x}{a} = t_{phys}$$

Condition CFL

CFL = Courant-Friedrichs-Lewy

→ C'est une condition nécessaire pour la stabilité dans le cas d'un schéma explicite :

1D : $$C = \frac{a\delta t}{\delta x} \leq C_{max} = 1$$

2D : $$C = \frac{a_x \delta t}{\delta x} + \frac{a_y \delta t}{\delta x} \leq C_{max} = 1$$

Cas général : $$C = \delta t \left(\sum_{i=1}^{k} \frac{a_{x_i}}{\delta x_i} \right) \leq C_{max} = 1$$

→ Attention !
C'est une condition nécessaire, mais pas forcément suffisante !

Stabilité et schéma implicite (I)

1

Étape 1 : Définition du problème

Deuxième exemple :

Équation de diffusion

Équation : $\frac{\partial f}{\partial t}(x,t) - \alpha \frac{\partial^2 f}{\partial x^2}(x,t) = s(x,t)$ (problème de **diffusion**)

Inconnue : $f(x,t)$ (fonction réelle 1D + temps)

Domaines : $x \in [0,L]$ $t \in [0,T]$ (segments 1D cartésiens)

Condition aux limites : $f(0,t) = G(t), f(L,t) = H(t)$ (CL de type Dirichlet)

Condition initiale : $f(x,0) = g(x)$ (fonction d'initialisation temporelle)

Stabilité et schéma implicite (II)

2

Étape 2 : Discrétisation du domaine

Deuxième exemple :

Équation de diffusion + maillage uniforme

Nombres de points : $M+1$ en espace, $N+1$ en temps

Pas d'espace : $\delta x = L/M \qquad \delta t = T/N$ (résolution)

Numérotation : $0 = x_0 < x_1 < x_2 < \cdots < x_M = L$
$x_i = i\delta x, \forall i \in [0, M]$ (noeuds du maillage spatial)

$0 = t_0 < t_1 < t_2 < \cdots < t_N = T$
$t_n = n\delta t, \forall n \in [0, N]$ (noeuds du maillage temporel)

Équivalence discret/continu : $f(x_i, t_n) = f_i^n$ (notation)

Valeurs nodales inconnues : $f_0^1, f_1^1, f_2^1, \ldots, f_M^1$
...
$f_0^N, f_1^N, f_2^N, \ldots, f_M^N$ (valeurs aux noeuds)

Stabilité et schéma implicite (III)

3

Étape 3 : Discrétisation des équations (domaine)

Deuxième exemple :

Équation de diffusion + schéma de Crank-Nicolson

$$\frac{\partial f}{\partial t}(x,t) - \alpha \frac{\partial^2 f}{\partial x^2}(x,t) = s(x,t) \quad \Rightarrow \quad \frac{\partial f_i^n}{\partial t} - \alpha \frac{\partial^2 f_i^n}{\partial x^2} = s_i^n \qquad \forall i \in [0,M] \quad \forall n \in [0,N]$$

FTCS en n :

$$\frac{f_i^{n+1} - f_i^n}{\delta t} - \alpha \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{\delta x^2} = 0$$

BTCS en $n+1$:

$$\frac{f_i^{n+1} - f_i^n}{\delta t} - \alpha \frac{f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}}{\delta x^2} = 0$$

→ La moyenne des deux donne : $\left(S = \frac{\alpha \delta t}{\delta x^2}\right)$

$$f_i^{n+1} = f_i^n + \frac{S}{2}\left[\left(f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}\right) + \left(f_{i+1}^n - 2f_i^n + f_{i-1}^n\right)\right] + \frac{\delta t}{2}\left(s_i^{n+1} + s_i^n\right)$$

→ On a donc à la fin :

$$f_i^{n+1} - \frac{S}{2}\left(f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}\right) = f_i^n + \frac{S}{2}\left(f_{i+1}^n - 2f_i^n + f_{i-1}^n\right) + \frac{\delta t}{2}\left(s_i^{n+1} + s_i^n\right)$$

Stabilité et schéma implicite (IV)

3

Étape 3 : Discrétisation des équations (bords)

Deuxième exemple :

Équation de diffusion + schéma de Crank-Nicolson

Condition aux limites :

→ Conditions de Dirichlet :

$$f_0^{n+1} = G^{n+1}, f_M^{n+1} = H^{n+1}, \forall n \in [0, N-1]$$

Condition initiale :

→ La dérivée la plus élevée étant d'ordre 1 (temporelle !), on a besoin de 1 condition portant sur la dérivée d'ordre 0 :

$$f_i^0 = g(x_i) = g_i, \forall i \in [0, M]$$

→ Le problème final est donc sous la forme :

$$\begin{cases} f_i^0 = g_i & \forall n \in [0, N-1] \\ f_0^{n+1} = G^{n+1}, f_M^{n+1} = H^{n+1} & \forall i \in [0, M-1] \\ f_i^{n+1} - \frac{S}{2}(f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}) = f_i^n + \frac{S}{2}(f_{i+1}^n - 2f_i^n + f_{i-1}^n) + \frac{\delta t}{2}(s_i^{n+1} + s_i^n) \end{cases} \quad \left(S = \frac{\alpha \delta t}{\delta x^2}\right)$$

Stabilité et schéma implicite (V)

4

Étape 4 : Écriture matricielle

Deuxième exemple :

Équation de diffusion + schéma de Crank-Nicolson

$$
\begin{array}{ll}
x_0 & f_0^{n+1} = G^{n+1} \\
x_1 & f_1^{n+1} - \frac{S}{2}(f_2^{n+1} - 2f_1^{n+1} + f_0^{n+1}) = f_1^n + \frac{S}{2}(f_2^n - 2f_1^n + f_0^n) + \frac{\delta t}{2}(s_1^{n+1} + s_1^n) \\
\dots & \dots \\
x_i & f_i^{n+1} - \frac{S}{2}(f_{i-1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}) = f_i^n + \frac{S}{2}(f_{i+1}^n - 2f_i^n + f_{i-1}^n) + \frac{\delta t}{2}(s_i^{n+1} + s_i^n) \\
\dots & \dots \\
x_{M-1} & f_{M-1}^{n+1} - \frac{S}{2}(f_M^{n+1} - 2f_{M-1}^{n+1} + f_{M-2}^{n+1}) = f_{M-1}^n + \frac{S}{2}(f_M^n - 2f_{M-1}^n + f_{M-2}^n) + \frac{\delta t}{2}(s_{M-1}^{n+1} + s_{M-1}^n) \\
x_M & f_M^{n+1} = H^{n+1}
\end{array}
$$

$$F^{n+1} = \varepsilon F^n + G$$

$$\begin{bmatrix} f_0^{n+1} \\ \dots \\ f_i^{n+1} \\ \dots \\ f_M^{n+1} \end{bmatrix} \quad (M+1) \times (M+1) \quad \begin{bmatrix} f_0^n \\ \dots \\ f_i^n \\ \dots \\ f_M^n \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{C_0} \\ \dots \\ g_i \\ \dots \\ \boldsymbol{C_L} \end{bmatrix}$$

$$O_g F^{n+1} = O_d F^n + G$$

$$\begin{bmatrix} f_0^{n+1} \\ \dots \\ f_i^{n+1} \\ \dots \\ f_M^{n+1} \end{bmatrix} \quad (M+1) \times (M+1) \quad \begin{bmatrix} f_0^n \\ \dots \\ f_i^n \\ \dots \\ f_M^n \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{C_0} \\ \dots \\ g_i \\ \dots \\ \boldsymbol{C_L} \end{bmatrix}$$

Stabilité et schéma implicite (VI)

4

Étape 4 : Écriture matricielle

Deuxième exemple :

Équation de diffusion + schéma de Crank-Nicolson

$$O_g F^{n+1} = O_d F^n + G$$

O_g

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{M-1} \\ x_M \end{array} \begin{array}{c} \begin{array}{cccccc} f_0^{n+1} & f_1^{n+1} & f_2^{n+1} & \dots & f_{M-1}^{n+1} & f_M^{n+1} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -S/2 & 1+S & -S/2 & \dots & 0 & 0 \\ 0 & -S/2 & 1+S & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1+S & -S/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

O_d

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{M-1} \\ x_M \end{array} \begin{array}{c} \begin{array}{cccccc} f_0^{n} & f_1^{n} & f_2^{n} & \dots & f_{M-1}^{n} & f_M^{n} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ S/2 & 1-S & S/2 & \dots & 0 & 0 \\ 0 & S/2 & 1-S & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1-S & S/2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

G

$$\begin{bmatrix} G^{n+1} \\ (s_1^{n+1} + s_1^n)\delta t/2 \\ \dots \\ (s_i^{n+1} + s_i^n)\delta t/2 \\ \dots \\ (s_{M-1}^{n+1} + s_{M-1}^n)\delta t/2 \\ H^{n+1} \end{bmatrix}$$

Condition initiale

$$\begin{bmatrix} f_0^0 \\ \dots \\ f_i^0 \\ \dots \\ f_M^0 \end{bmatrix} = \begin{bmatrix} g_0 \\ \dots \\ g_i \\ \dots \\ g_M \end{bmatrix}$$

Stabilité et schéma implicite (VII)

5

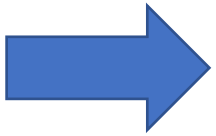
Étape 5 : Programmation

Il ne reste plus qu'à demander à l'ordinateur de résoudre le problème !

→ Sous forme matricielle :

$$F^0 = G^0$$

$$O_g F^{n+1} = O_d F^n + G$$



On n'a plus qu'à boucler sur n :

$$F^{n+1} = \underbrace{O_g^{-1}(O_d}_{\varepsilon} F^n + G)$$

Étapes informatiques :

1. Définir le vecteur G,
2. Définir les matrices O_g et O_d,
3. Écrire la condition initiale,
4. Boucler sur l'indice temporel et inverser la matrice O_g.

Deuxième exemple :

En Python, on utilisera la librairie linalg et sa fonction solve

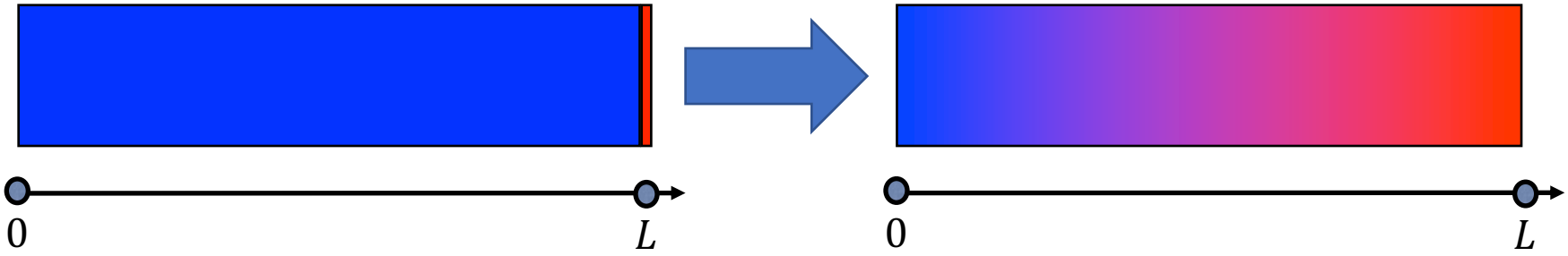
Stabilité et schéma implicite (VIII)

✓

Étape 6 : Validation

Pour la validation, on va prendre un exemple physique concret :

On considère un matériau solide, initialement à basse température, qui est mis en contact avec une source chaude au temps $t = 0$



Conditions aux limites : $f(0,t) = 0, f(L,t) = 1$

Condition initiale : $f(x,0) = 0$

Terme source : $s(x,t) = 0$

→ On n'a plus qu'à choisir les paramètres du domaine :

$L = 10$ $T = 20$ $\delta x = 0.01$ $\delta t = 0.01$

→ On fait un premier exemple avec $\alpha = 1.0$

Stabilité et schéma implicite (IX)

Exemple avec code :

Définition des matrices O_g, O_d et du vecteur G

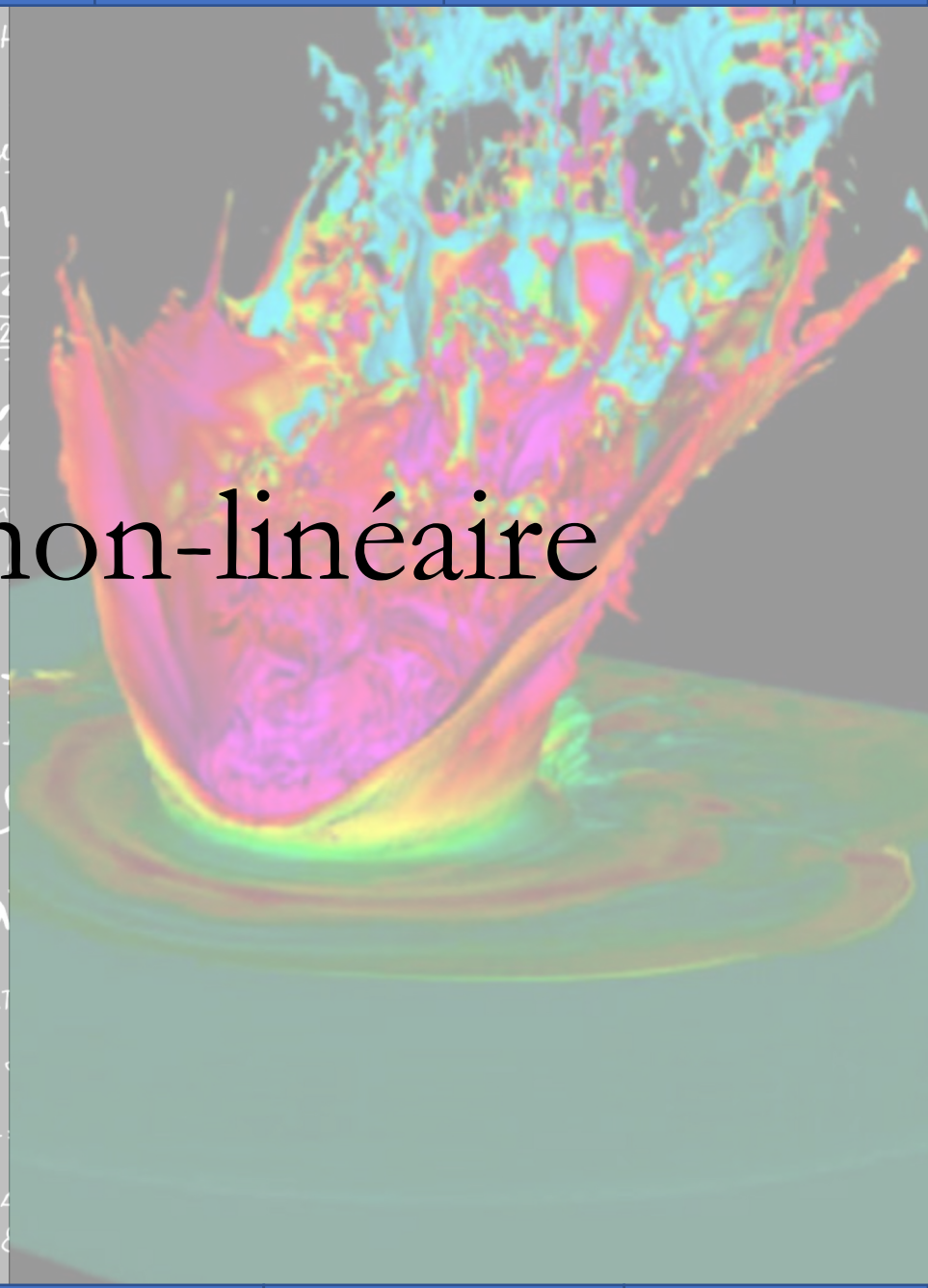
```
#4. Matrice Og et Od et vecteur g
Og=np.zeros([M+1,M+1])
Od=np.zeros([M+1,M+1])
g=np.zeros(M+1)

#corriger les lignes des pts de bords (0 et M)
Og[0,0]=1
Og[M,M]=1
g[M]=1
```

```
#boucle pour les points int
for j in range(1,M):   #pour j allant de 1 à M-1
    Og[j,[j-1,j,j+1]]=[-S/2,1+S,-S/2]
    Od[j,[j-1,j,j+1]]=[ S/2,1-S, S/2]
```

Boucle temporelle

```
#5. boucle temporel
for n in range(N):  #pour n allant de 0 a N-1

    #trouver la solution de Og .dot f_new = Od .dot f_old + g
    membrededroite=np.dot(Od,f)+g
    f=np.linalg.solve(Og,membrededroite)
```

Problème non-linéaire



Définition d'un problème non-linéaire

On appelle **problème non-linéaire** un problème dont le comportement n'est pas linéaire

→ La sortie n'est pas proportionnelle à l'entrée

→ On peut même avoir des systèmes avec des propriétés chaotiques !

Ces problèmes ne respectent pas le principe de superposition

→ Ils ne peuvent pas être étudiés à l'aide d'analyse par décomposition

On peut les identifier facilement, car ce sont des problèmes :

- où la fonction inconnue apparaît sous forme polynômiale
- OU où la fonction inconnue est multipliée par elle-même ou ses dérivées

Le problème, c'est que la majorité des problèmes en physique sont en fait non-linéaires

→ Qu'est-ce que cela change en pratique pour la résolution numérique ?

Étape 1 : Définition du problème

1

Exemple général :

Problème non-linéaire

Équation : $\boldsymbol{\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \alpha\frac{\partial^2 u}{\partial x^2}}$ **(équation de Burgers)**

Inconnue : $u(x,t)$ (fonction réelle 1D + temps)

Domaine : $x \in [0, L]$ $t \in [0, T]$ (segments 1D cartésiens)

Condition aux limites : **(CL périodique de type Dirichlet)**

Condition initiale : $u(x,0) = U(x)$ (fonction d'initialisation temporelle)

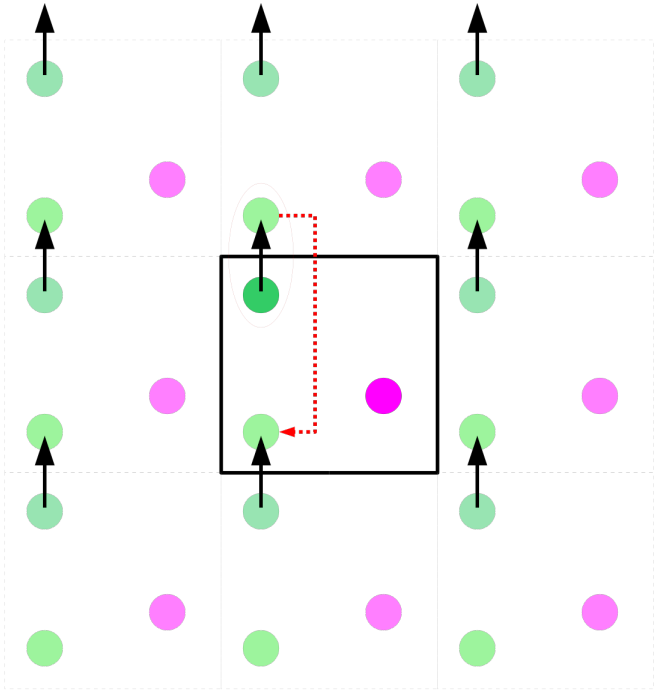
Conditions aux limites périodiques

Ici, on va utiliser un cas particulier des conditions de Dirichlet

On impose bien une valeur de la fonction sur les bords,
mais cette valeur est celle de l'autre bord du domaine
→ c'est comme si le domaine bouclait sur lui-même !

$$u(x,t) = u(x+L,t)$$

$$u_M^n = u_0^n$$



→ C'est en fait une manière artificielle de créer un domaine quasi-infini !

Étape 2 : Discrétisation du domaine

2

Exemple général :

Problème non-linéaire

Maillage uniforme :

Nombres de points : $M+1$ en espace, $N+1$ en temps (résolution)

Pas d'espace : $\delta x = L/M \qquad \delta t = T/N$

Numérotation : $0 = x_0 < x_1 < x_2 < \cdots < x_M = L$
$x_i = i\delta x, \forall i \in [0, M]$ (noeuds du maillage spatial)

$0 = t_0 < t_1 < t_2 < \cdots < t_N = T$
$t_n = n\delta t, \forall n \in [0, N]$ (noeuds du maillage temporel)

Équivalence discret/continu : $u(x_i, t_n) = u_i^n$ (notation)

Valeurs nodales inconnues : $\boldsymbol{u_0^1, u_1^1, u_2^1, \ldots, u_{M-1}^1}$
$\ldots$
$\boldsymbol{u_0^N, u_1^N, u_2^N, \ldots, u_{M-1}^N}$ (valeurs aux noeuds)

Étape 3 : Discrétisation des équations

3

Exemple général :

Problème non-linéaire

On cherche à discrétiser l'équation suivante :

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \alpha\frac{\partial^2 u}{\partial x^2} \quad \Rightarrow \quad \frac{\partial u_i^n}{\partial t} + u_i^n\frac{\partial u_i^n}{\partial x} = \alpha\frac{\partial^2 u_i^n}{\partial x^2}$$

Discrétisation temporelle : (Forward)

$$\frac{\partial u_i^n}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\delta t}$$

Discrétisation spatiale : (Centré + CN)

$$\frac{\partial u_i^n}{\partial x} \approx \frac{u_{i+1}^n - u_{i-1}^n}{2\delta x} \qquad \frac{\partial^2 u_i^n}{\partial x^2} \approx \frac{1}{2\delta x^2}\left[\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) + \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)\right]$$

$$\Rightarrow \quad \frac{u_i^{n+1} - u_i^n}{\delta t} + u_i^n\frac{u_{i+1}^n - u_{i-1}^n}{2\delta x} = \frac{\alpha}{2\delta x^2}\left[\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) + \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)\right]$$

$$\Rightarrow$$

$$u_i^{n+1} - \frac{S}{2}\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) = u_i^n + \frac{S}{2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) + \frac{\boldsymbol{\delta t}}{\boldsymbol{2\delta x}}\boldsymbol{u_i^n}\left(\boldsymbol{u_{i+1}^n - u_{i-1}^n}\right)$$

$$\left(S = \frac{\alpha\delta t}{\delta x^2}\right)$$

Étape 4 : Écriture matricielle

4

Deuxième exemple :

Problème non-linéaire

$$O_g F^{n+1} = O_d F^n + G$$

O_g

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{M-1} \\ x_M \end{array} \begin{array}{c} \begin{array}{ccccccc} u_0^{n+1} & u_1^{n+1} & u_2^{n+1} & \dots & u_{M-1}^{n+1} & u_M^{n+1} \end{array} \\ \begin{bmatrix} 1+S & -S/2 & 0 & \dots & 0 & -S/2 \\ -S/2 & 1+S & -S/2 & \dots & 0 & 0 \\ 0 & -S/2 & 1+S & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1+S & -S/2 \\ -S/2 & 0 & 0 & \dots & -S/2 & 1+S \end{bmatrix} \end{array}$$

O_d

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{M-1} \\ x_M \end{array} \begin{array}{c} \begin{array}{ccccccc} u_0^{n} & u_1^{n} & u_2^{n} & \dots & u_{M-1}^{n} & u_M^{n} \end{array} \\ \begin{bmatrix} 1-S & S/2 & 0 & \dots & 0 & S/2 \\ S/2 & 1-S & S/2 & \dots & 0 & 0 \\ 0 & S/2 & 1-S & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1-S & S/2 \\ S/2 & 0 & 0 & \dots & S/2 & 1-S \end{bmatrix} \end{array}$$

G

$$\left(\delta t \begin{bmatrix} 0 & -\frac{1}{2\delta x} & 0 & \dots & 0 & \frac{1}{2\delta x} \\ \frac{1}{2\delta x} & 0 & -\frac{1}{2\delta x} & \dots & 0 & 0 \\ 0 & \frac{1}{2\delta x} & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 \\ -\frac{1}{2\delta x} & 0 & 0 & \dots & 0 & -\frac{1}{2\delta x} \\ -\frac{1}{2\delta x} & 0 & 0 & \dots & \frac{1}{2\delta x} & 0 \end{bmatrix} \begin{bmatrix} u_0^0 \\ \dots \\ u_i^0 \\ \dots \\ u_M^0 \end{bmatrix} \right) * \begin{bmatrix} u_0^0 \\ \dots \\ u_i^0 \\ \dots \\ u_M^0 \end{bmatrix}$$

Condition initiale

$$\begin{bmatrix} u_0^0 \\ \dots \\ u_i^0 \\ \dots \\ u_{M-1}^0 \end{bmatrix} = \begin{bmatrix} U(x_0) \\ \dots \\ U(x_i) \\ \dots \\ U(x_{M-1}) \end{bmatrix}$$

Étape 5 : Programmation

5

Exemple général :

Problème non-linéaire

Il ne reste plus qu'à demander à l'ordinateur de résoudre le problème !

→ Sous forme matricielle :

$$F^0 = G^0$$

$$O_g F^{n+1} = O_d F^n + G$$

On n'a plus qu'à boucler sur n :

$$F^{n+1} = \underbrace{O_g^{-1}(O_d F^n + G)}_{\varepsilon}$$

Étapes informatiques :

1. Définir le vecteur G,
2. Définir les matrices O_g et O_d,
3. Écrire la condition initiale,
4. Boucler sur l'indice temporel et inverser la matrice O_g.

Deuxième exemple :

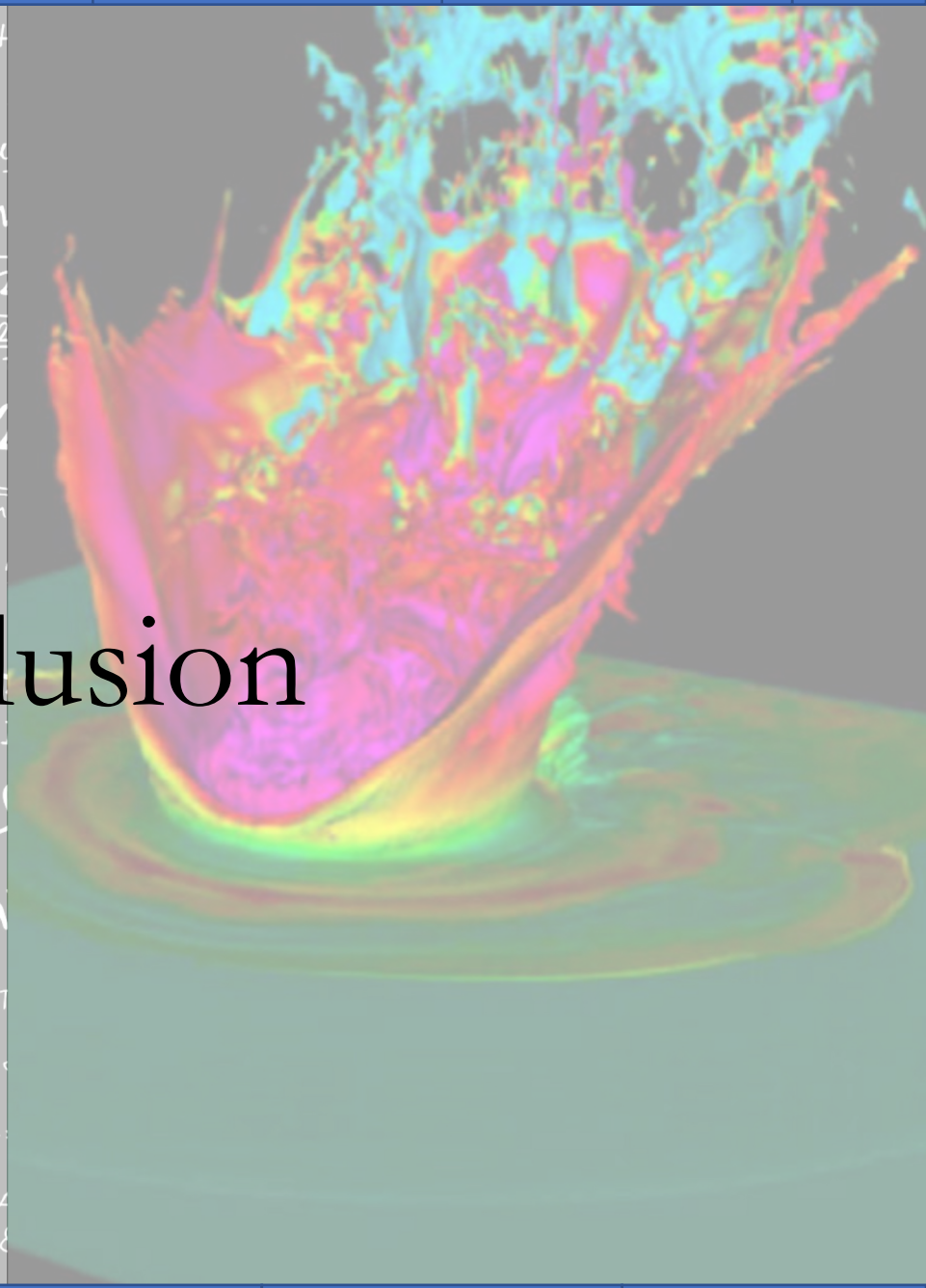
En Python, on utilisera la librairie linalg et sa fonction solve

Exemple avec code

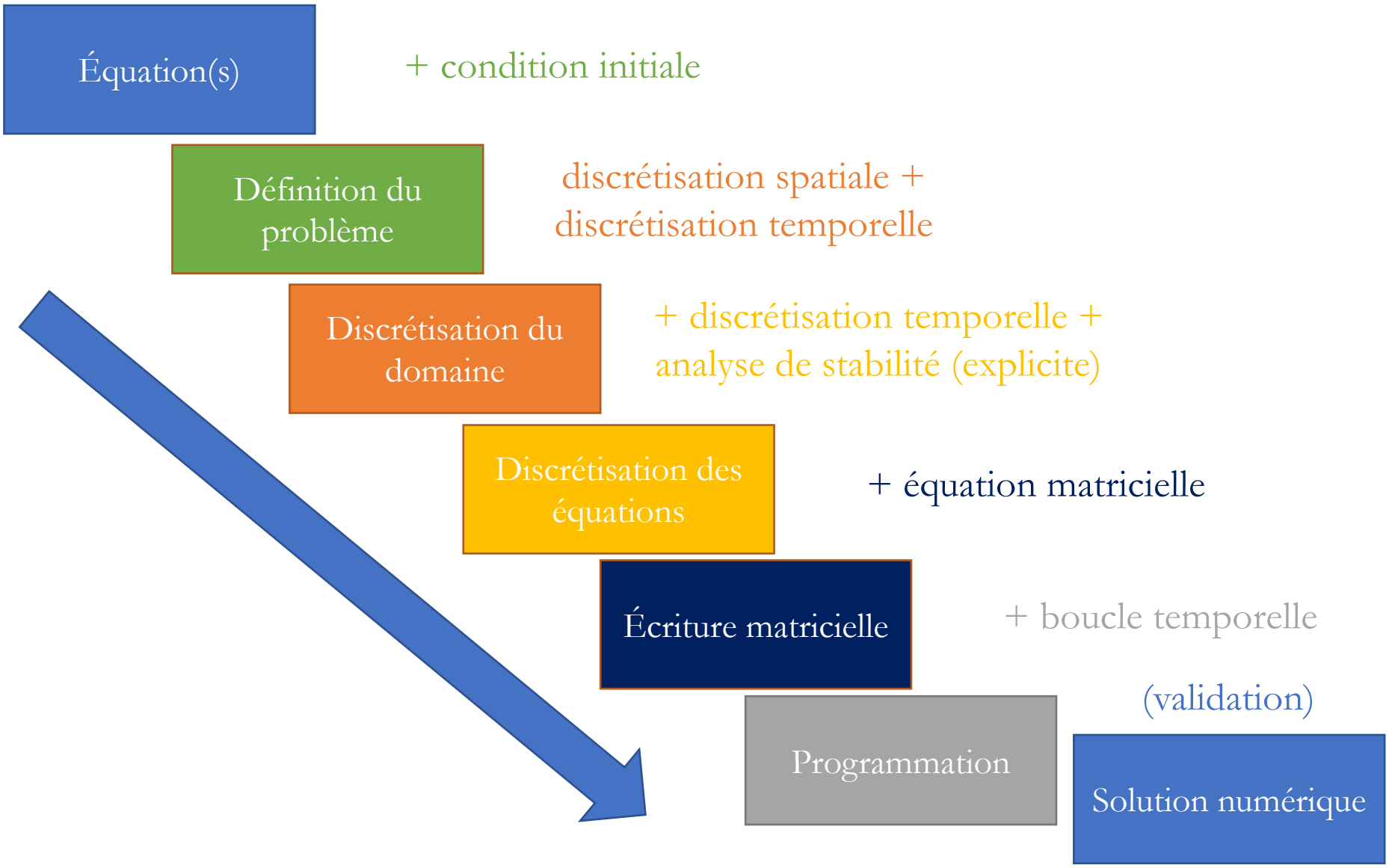
```
#operateurs de gauche et de droite
Og=np.zeros([M,M])  # un point en moins ici car la $x_M = x_0$.
Od=np.zeros([M,M])
A=np.zeros([M,M])

for j in range(M): #pour i de 0 à M-1

    #indice voisin à gauche
    jG=j-1
    if j==0:  #le voisin de gauche du premier point est le dernier (périodicité)
        jG=M-1

    #indice voisin à droite
    jD=j+1
    if j==M-1: #le voisin de gauche du premier point est le dernier (périodicité)
        jD=0

    Og[j,[jG,j,jD]]=[-S/2,1+S,-S/2]
    Od[j,[jG,j,jD]]=[S/2,1-S,S/2]
    A[j,[jG,jD]]=[dt/(2*dx),-dt/(2*dx)]
```

```
#initialisation
f=np.sin(x)
g=np.zeros(M)

figure(1)
#boucle d'avancement temp
for n in range(N):   #pour n=0 a N-1

    #calcul de -(du/dx)*(u)
    g=A.dot(f)*f                    #A.dot(f) = meme chose que np.dot(A,f)

    #solution du systeme linéaire Og*fnp1 = Od*fn +gnp1
    f=np.linalg.solve(Og,Od.dot(f)+g)
```

Conclusion



Cycle des méthodes numériques : Instationnaire



Schémas explicites/implicites et stabilité

Une fois le schéma mis sous forme générale,
on voit tout de suite s'il est explicite ou implicite

Explicite

= on exprime l'état futur du système en fonction des états passés

$$f^{n+1} = F[f^n, f^{n-1}, \dots]$$

→ La résolution est itérative

Avantages :

Plus facile à écrire et à résoudre

Inconvénients :

Pas toujours stable,
besoin de respecter la condition CFL

Implicite

= on exprime l'état futur du système en fonction de l'état futur également

$$G[\dots, f^{n-1}, f^n, f^{n+1}] = 0$$

→ La résolution est matricielle

Avantages :

Toujours stable, pas de limitation

Inconvénients :

Pas toujours facile d'inverser la matrice,
plus difficile à implémenter

→ Il faut trouver le meilleur compromis en fonction de chaque problème !

Application à l'UE

Définition du problème

- Poser une condition initiale,
- Identifier un problème non-linéaire.

Discrétisation du domaine :

- Savoir gérer une discrétisation temporelle en plus de la discrétisation spatiale.

Discrétisation des équations :

- Savoir combiner les formulations,
- Savoir manipuler un schéma de Crank-Nicolson,
- Savoir tester la stabilité.

Écriture matricielle :

- Boucle temporelle,
- Cas d'un schéma explicite vs. implicite.

Programmation :

- Programmation Python de chaque étape,
- Librairies Python utiles.

Validation :

- Effectuer une validation temporelle.

Plan de l'UE

Idée générale :

Au premier semestre, on va introduire les notions de base, et s'intéresser en détails à une méthode numérique précise

Tous les jeudi matin (8h45-12h45) au bâtiment 625

21 Novembre : Cours 1 + Cours 2
4 Décembre : Cours 3 + TP
5 Décembre : Cours 4 + TP
12 Décembre : Cours 5 + **TP**
19 Décembre : Cours 6 + TP
9 Janvier : Cours 7 + TP
16 Janvier : Cours 8 + TP
23 Janvier : TP
30 Janvier : Examen

Modalités d'évaluation :

TPs + examen oral (question de cours + exercice)