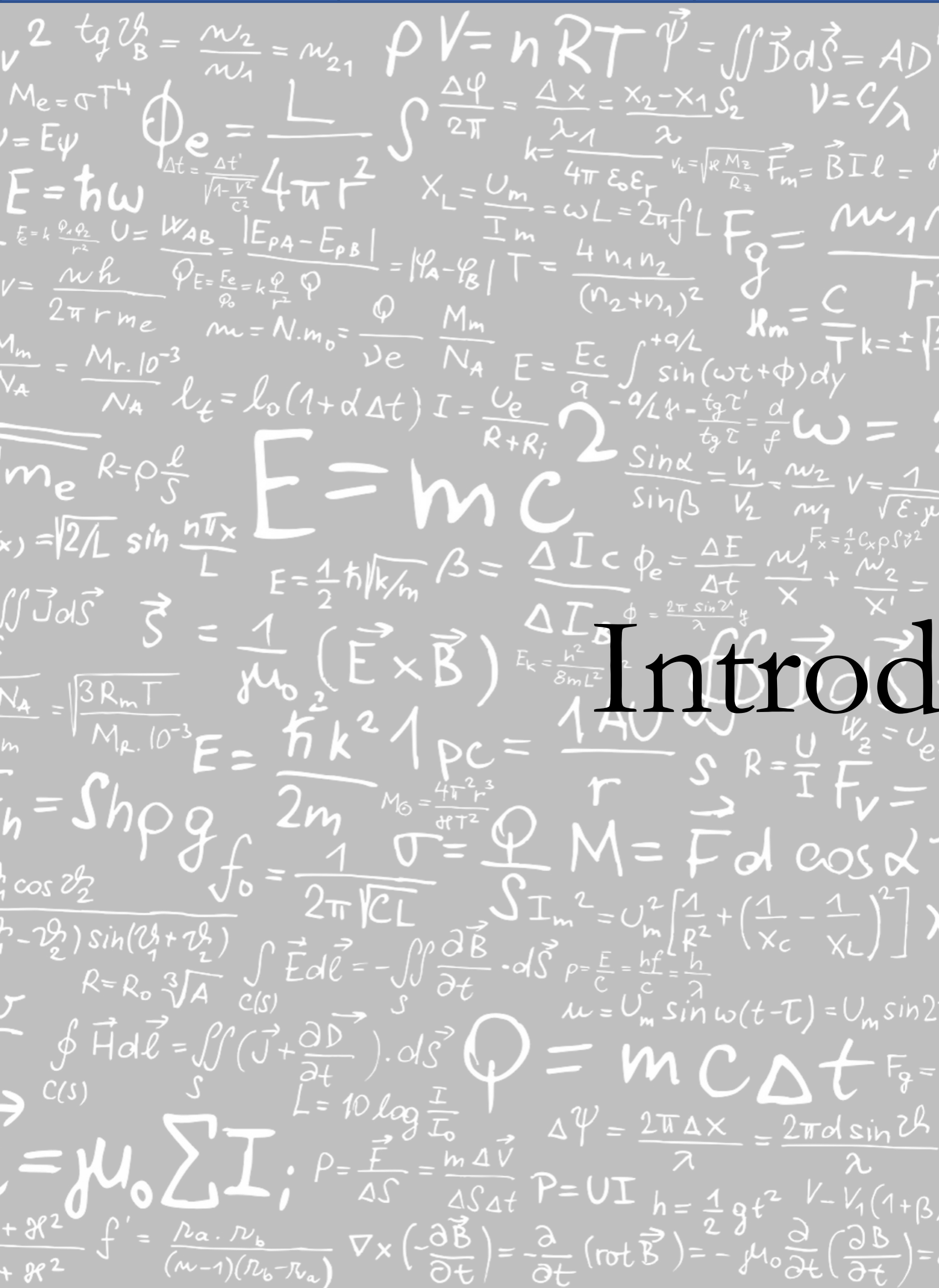
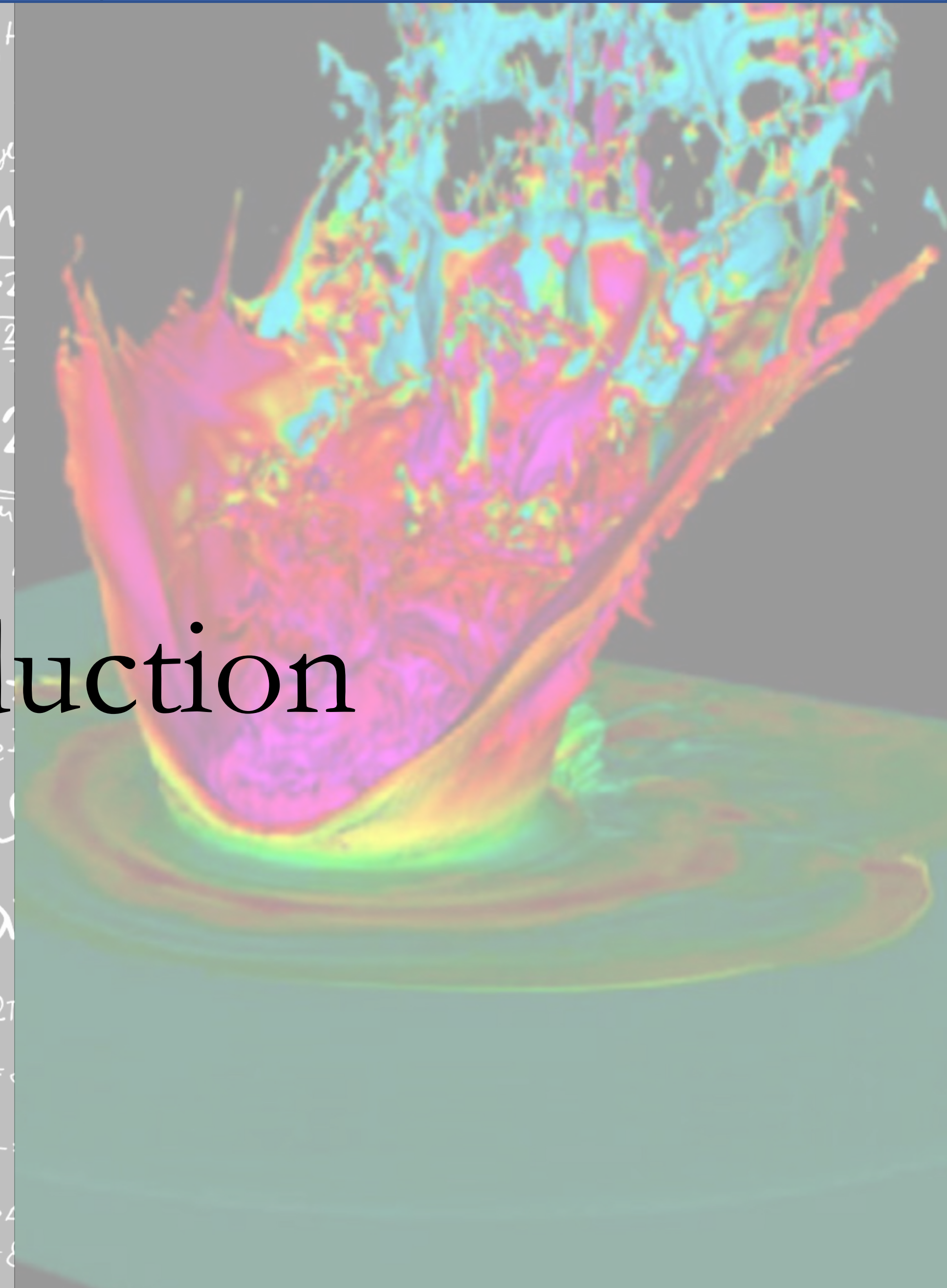
Méthodes Numériques

Cours 2 :

Résolution d'équations différentielles ordinaires (EDO)

Dr. Barbara PERRI
barbara.perri@universite-paris-saclay.fr





Introduction

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

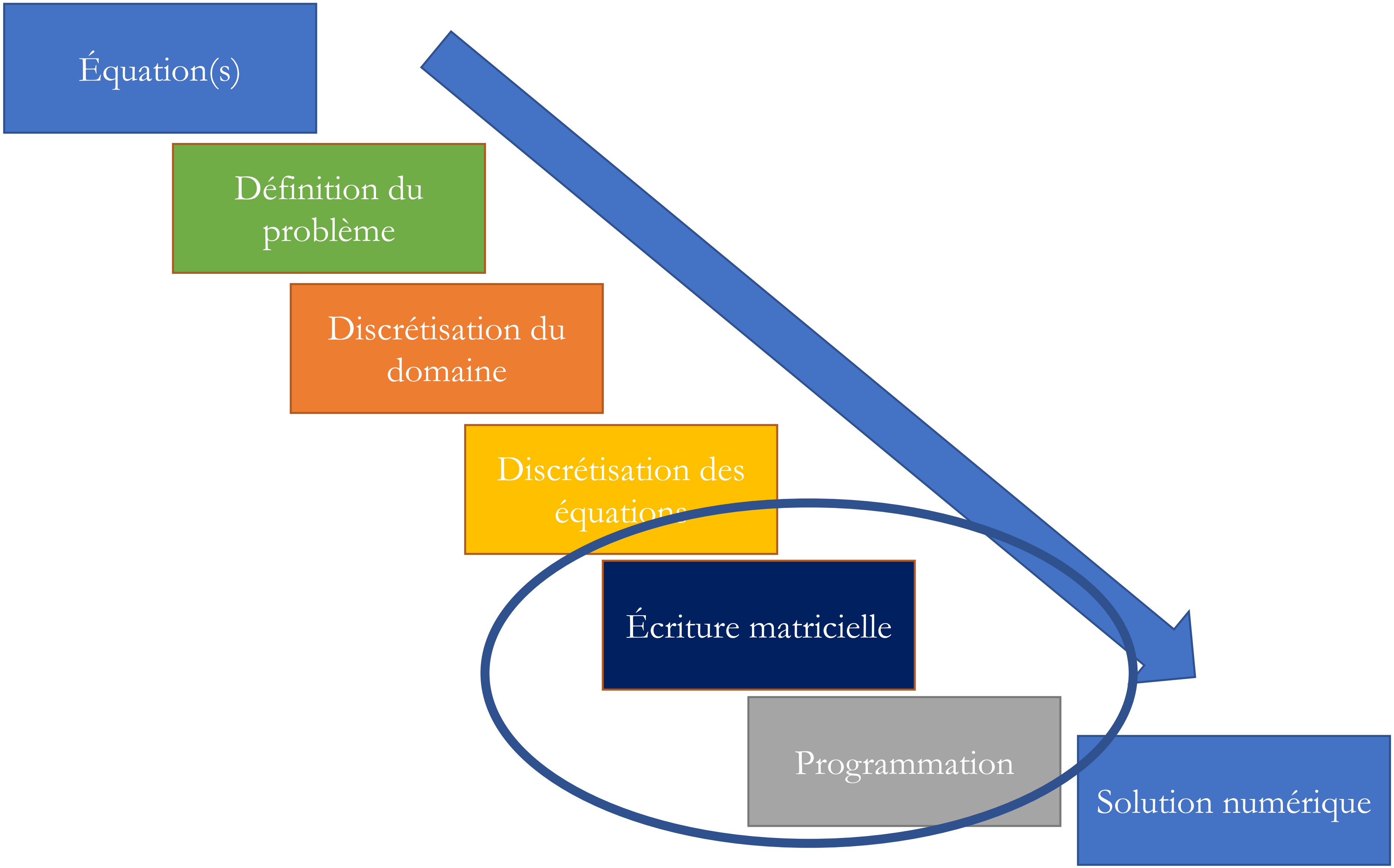
Tous les jeudi matin (8h45-12h45) au bâtiment 625

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

Modalités d'évaluation :

TPs + examen oral (question de cours + exercice)

Cycle des méthodes numériques



Rappel : Définition des EDOs

En méthodes numériques, on s'intéresse à deux grandes familles :

Équations différentielles ordinaires (EDO)

= les inconnues ne dépendent que d'une seule variable

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

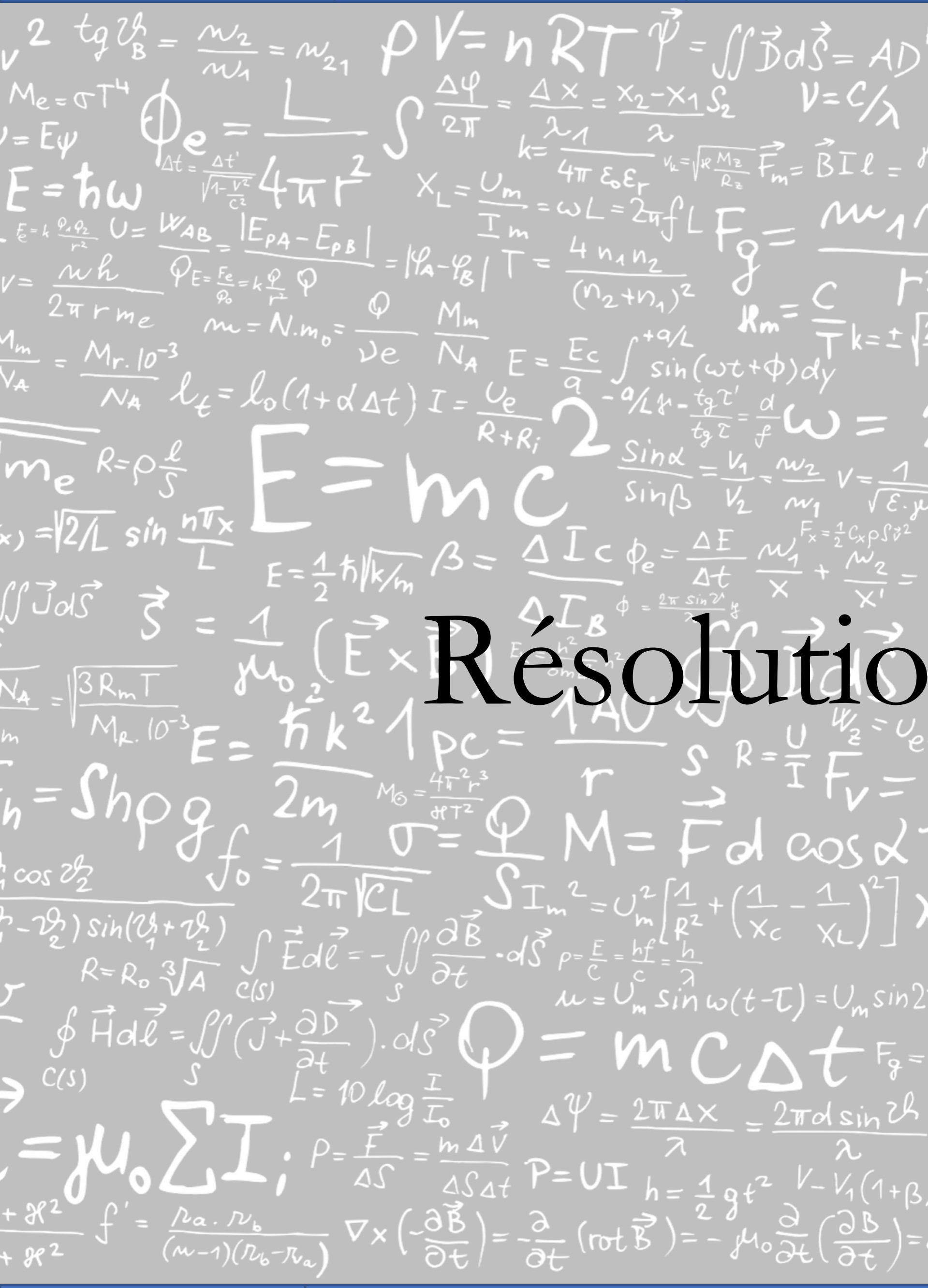
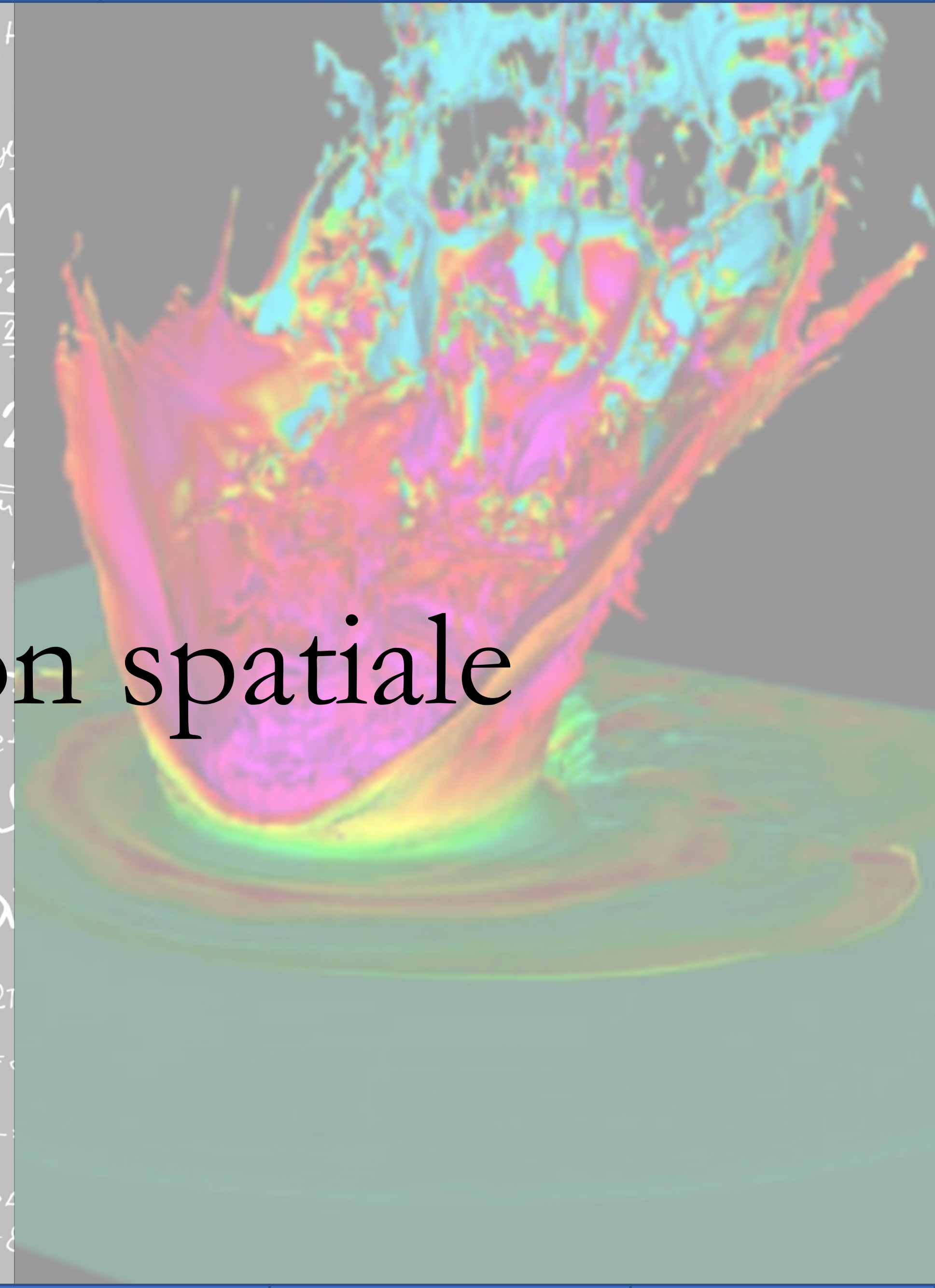
Équations aux dérivées partielles (EDP)

= les inconnues peuvent dépendre de plusieurs variables

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

→ On va d'abord voir les méthodes qu'on peut appliquer pour les EDOs (qui sont souvent plus simples)

→ Ceci va nous permettre d'introduire un certain nombre d'outils pour la suite

Résolution spatiale

Formulation du problème

1 2 3

On s'intéresse à une EDO qui dépend de l'espace (pour l'instant 1D)

→ système de p EDO d'ordre 1 :

$$\begin{cases} \dfrac{du_1}{dx} = f_1(x, u_1, u_2, \dots, u_p) \\ \dfrac{du_2}{dx} = f_2(x, u_1, u_2, \dots, u_p) \\ \dots \\ \dfrac{du_p}{dx} = f_p(x, u_1, u_2, \dots, u_p) \end{cases}$$

$x \in [L_0, L_f]$

$$\frac{d\vec{u}}{dx} = f(x, \vec{u}), \vec{u} = (u_1, u_2, \dots, u_p)$$

Discrétisation : $L_0 < x_1 < \cdots < x_N = L_f$

$$\vec{u}^k = \vec{u}(x_k), f^k = f(x_k, \vec{u}^k)$$

Conditions aux limites :

$\vec{u}^0 = \vec{U}^0, \vec{u}^N = \vec{U}^N$ (Dirichlet)

$\dfrac{d\vec{u}^0}{dx} = \dfrac{d\vec{U}^0}{dx}, \dfrac{d\vec{u}^N}{dx} = \dfrac{d\vec{U}^N}{dx}$ (Neuman)

→ on connaît les p fonctions $f_i(x, u_1, u_2, \dots, u_p)$

→ on cherche à identifier les solutions $u_1(x), u_2(x), \dots, u_p(x)$ passant par l'état initial

On peut écrire le problème sous forme intégrale :

$$\vec{u}^{k+1} = \vec{u}^k + \int_{x_k}^{x_{k+1}} f(x', \vec{u}(x'))dx'$$

→ Dans certains cas, l'intégrale peut être résolue analytiquement

Méthode de Newton : Théorie

Cas d'une EDO qu'on peut mettre sous la forme (intégration) :

$$F(x) = 0$$

→ le problème est alors ramené à trouver le zéro d'une fonction

→ Méthode de Newton (ou Newton-Raphson) :

On définit la tangente à la fonction $\mathrm{F}(x)$ en un point x_0 :

$$F'(x_0) \approx \frac{F(x) - F(x_0)}{x - x_0} \Longrightarrow F(x) \approx F(x_0) + F'(x_0)(x - x_0)$$

On a alors une approximation très simple de la fonction $\mathrm{F}(x)$

→ on va trouver le zéro de cette approximation plutôt (plus simple) :

$$0 \approx F(x_0) + F'(x_0)(x_1 - x_0) \Longrightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

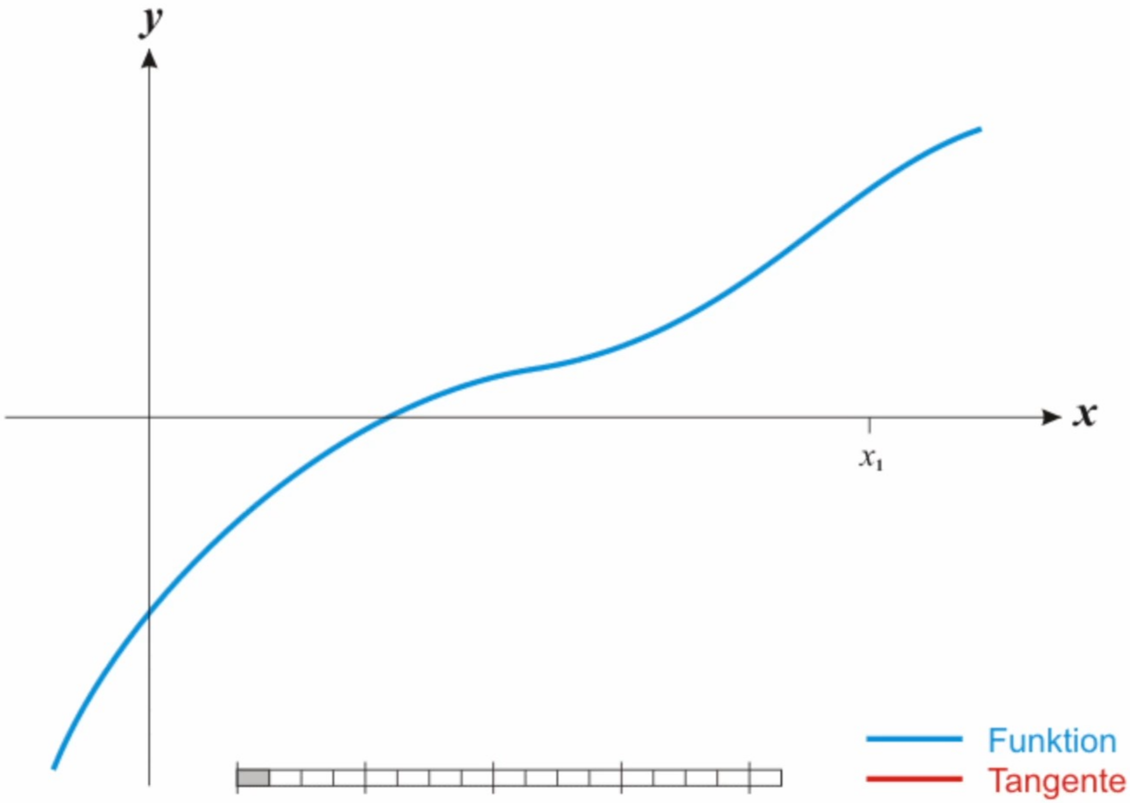
→ x_1 a alors plus de chance d'être un 0 de F que x_0

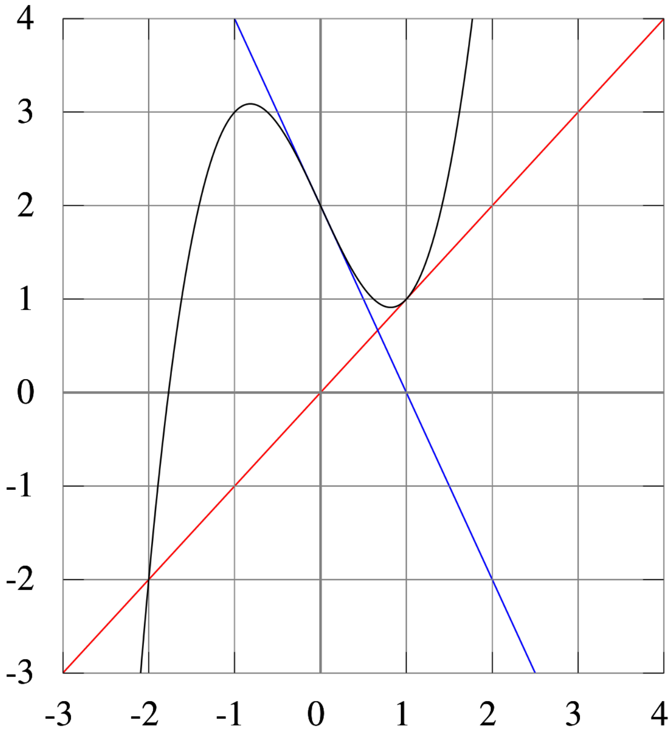
→ on répète la procédure de manière itérative jusqu'à atteindre la précision désirée

Méthode de Newton : Illustration

[Wikipedia]

Animation de la procédure pour un cas 1D simple :





Avantages :

Rapide, simple à coder, applicable dès que F est dérivable

Inconvénients :

Que certains problèmes, dérivée doit être non nulle, supposition initiale, plusieurs zéros ??

Méthode de Newton : Algorithme

<u>Condition initiale :</u>

Choix de x_0, choix de la précision ϵ

<u>Condition :</u>

Tant que $F(x_k)$ n'est pas en dessous de la précision souhaitée ϵ
= tant que x_k n'est pas un assez bon zéro

<u>Itération :</u>

Calcul du nouveau zéro avec la formule :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

[Réville 2016]

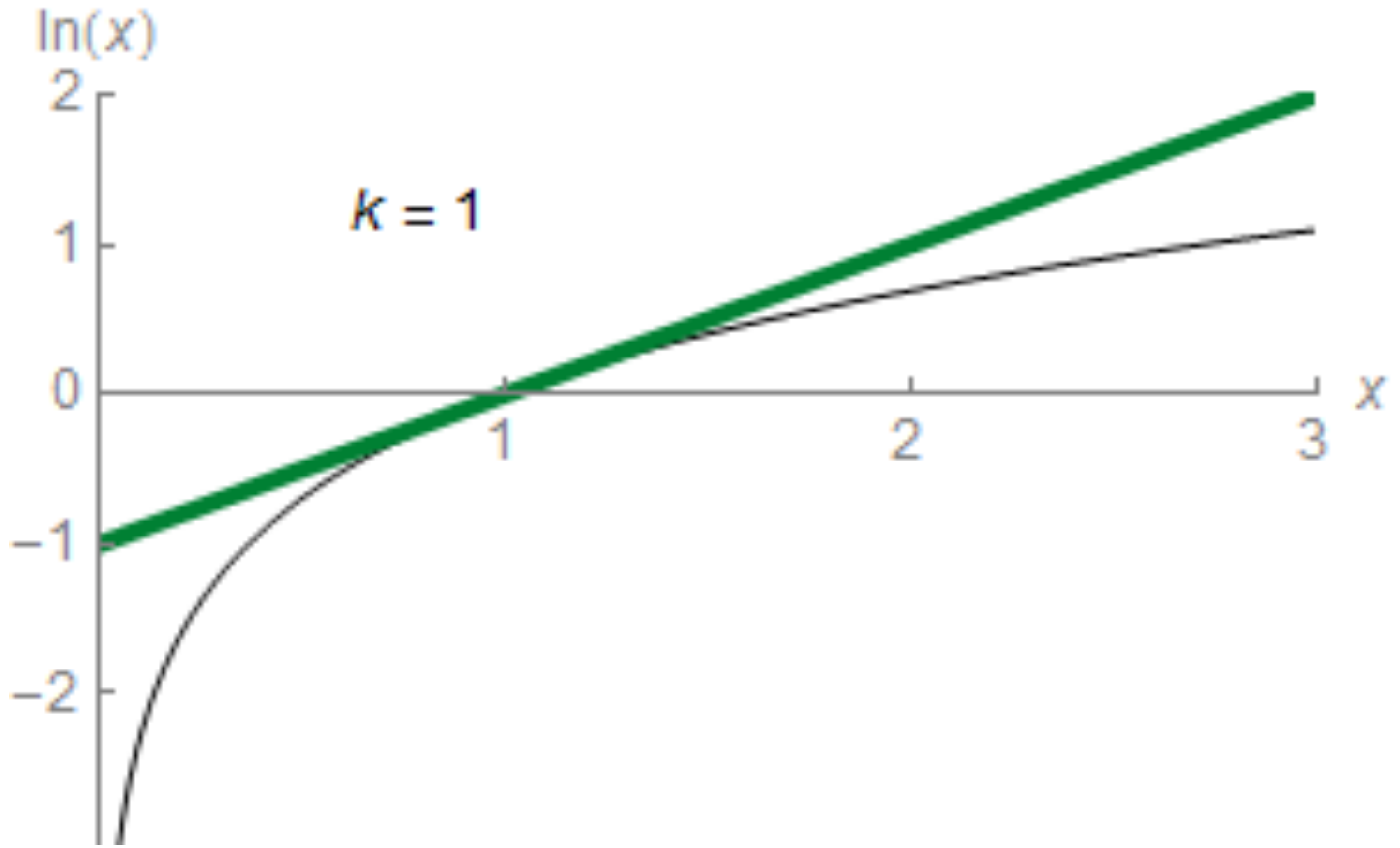
Algorithm 3 Méthode de Newton-Raphson

```
procedure NEWTON-RAPHSON(f, x_0)
    f_k ← f(x_0)
    while ||f_k|| ≥ ε do                          ▷ Début de l'itération
        x_{k+1} ← x_k − f(x_k)/f'(x_k)
        f_k ← f(x_{k+1})
    return x_k                                    ▷ Position du zéro à la précision voulue
```

Formule de Taylor (II)

[Wikipedia]

Visualisation :

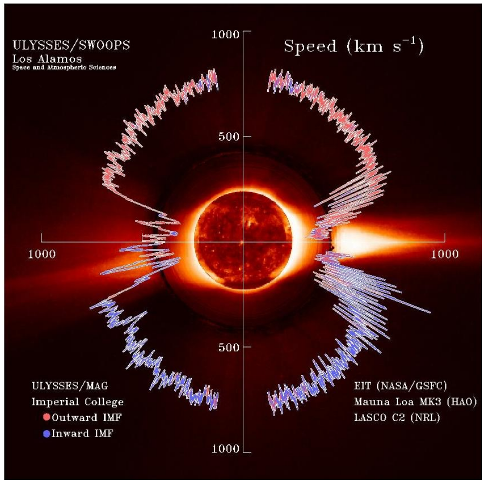


→ Plus on augmente n, plus on se rapproche de la solution exacte en ce point

1

Exemple d'EDO intégrable

Calcul d'une solution de vent solaire hydrodynamique en 1D (cas isotherme) :



[McComas+2008]

[Parker 1958]

$$\rho(r) = \frac{A}{ur^2} \qquad p(r) = c_s^2 \frac{A}{ur^2}$$

$$\frac{\partial}{\partial r}(\rho u r^2) = 0,$$

$$u\frac{\partial u}{\partial r} = -\frac{1}{\rho}\frac{\partial p}{\partial r} - \frac{GM_O}{r^2},$$

$$p = c_s^2 \rho.$$

$$\Rightarrow \quad \frac{\partial u}{\partial r}\left(u - \frac{c_s^2}{u}\right) = c_s^2 \frac{2}{r} - \frac{GM_\odot}{r^2}$$

$$(\vec{u} = u(r))$$

Changement de variable : $M = \frac{u}{c_s^2}, r_c = \frac{GM_\odot}{2c_s^2}$

$$\Rightarrow \quad \frac{\partial M}{\partial r}\left(M - \frac{1}{M}\right) = \frac{2}{r} - \frac{GM_\odot}{c_s^2 r^2}$$

On peut alors intégrer : $\Rightarrow \quad \frac{M^2}{2} - \ln M - 2\ln x - \frac{2}{x} + C = 0$

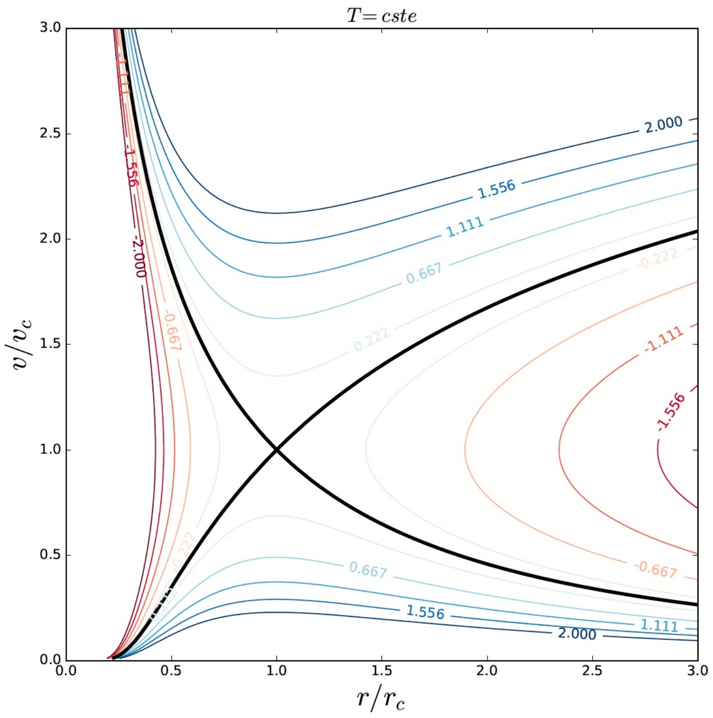
→ On a ramené notre système d'EDO à résoudre $F(x) = 0$

Méthode de Newton : Exemple

Calculer une solution de vent solaire hydrodynamique en 1D (cas isotherme) :

$$\frac{\partial}{\partial r}(\rho u r^2) = 0,$$
$$u\frac{\partial u}{\partial r} = -\frac{1}{\rho}\frac{\partial p}{\partial r} - \frac{GM_O}{r^2},$$
$$p = c_s^2 \rho.$$

$$\frac{M^2}{2} - ln\, M - 2ln\, x - \frac{2}{x} = C,$$
$$M = \frac{u}{c_s}, x = \frac{r}{r_c}, r_c = \frac{GM_\odot}{2c_s^2}$$
$$F(M,x) = \frac{M^2}{2} - ln\, M - 2ln\, x - \frac{2}{x}.$$



En traçant la fonction, on trouve que la solution qu'on cherche correspond à $C = -3/2$

Méthode de Newton en multi-D

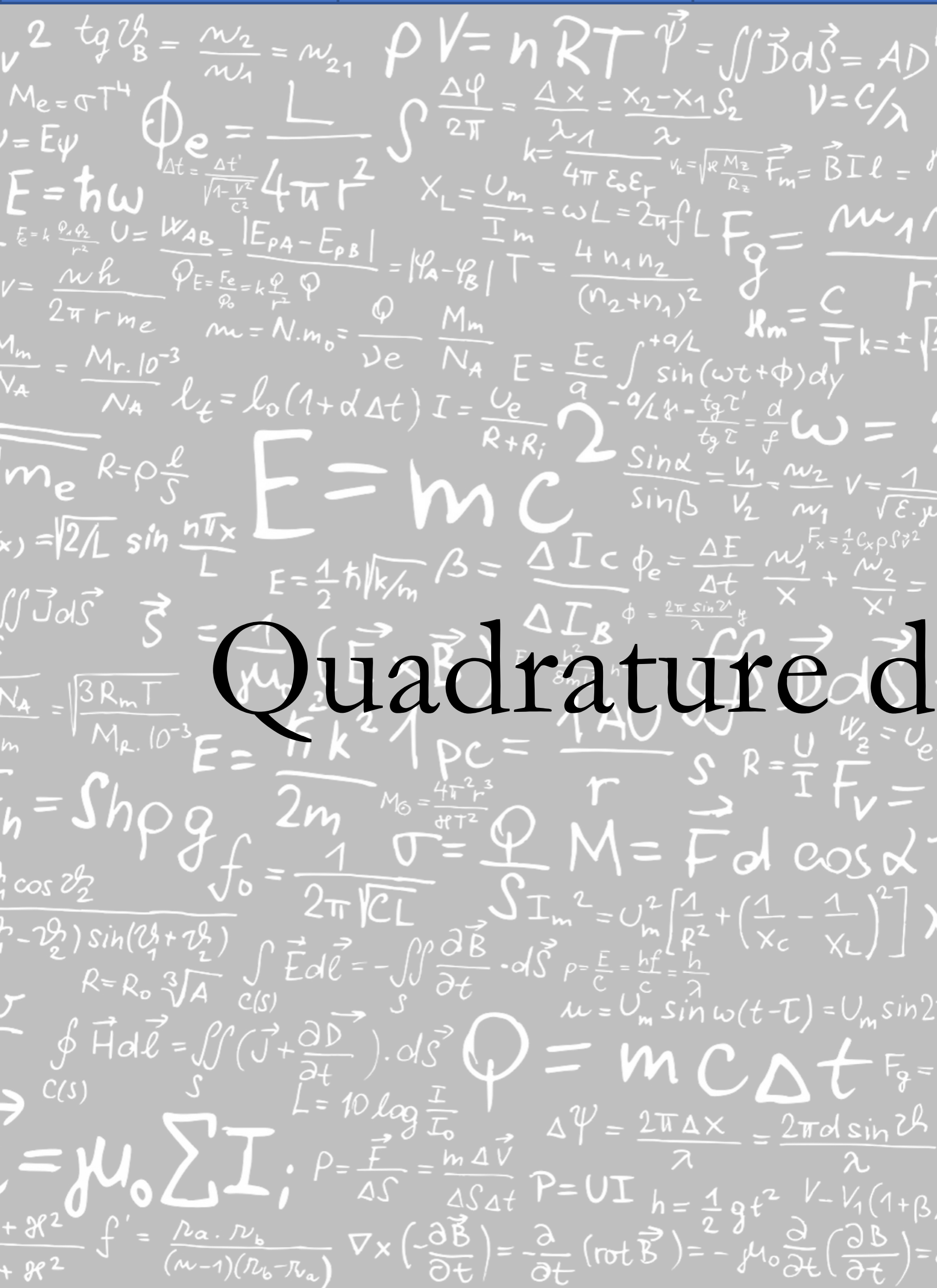
Pour une dimension n quelconque,
on remplace la dérivée de la fonction f par son Jacobien :

$$f = (f_1, f_2, \ldots, f_n)^T \quad \Longrightarrow \quad J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

On garde la même procédure algorithmique, on remplace simplement :

$$x_{k+1} \leftarrow x_k - \frac{f(x_k)}{f'(x_k)} \quad \Longrightarrow \quad x_{k+1} \leftarrow x_k - J^{-1}(x_k) f(x_k).$$

→ la difficulté devient alors d'inverser le Jacobien




Quadrature d'interpolation

Formulation du problème

Cette fois-ci, on s'intéresse à une EDO qui dépend du temps

→ système de p EDO d'ordre 1 :

$$\left\{\begin{array}{l} \frac{du_1}{dt} = f_1(t, u_1, u_2, \dots, u_p) \\ \frac{du_2}{dt} = f_2(t, u_1, u_2, \dots, u_p) \\ \dots \\ \frac{du_p}{dt} = f_p(t, u_1, u_2, \dots, u_p) \end{array}\right.$$

$t \in [t_0, t_f]$

$$\frac{d\vec{u}}{dt} = f(t, \vec{u}), \vec{u} = (u_1, u_2, \dots, u_p)$$

Discrétisation : $t_0 < t_1 < \cdots < t_N = t_f$

$$\vec{u}^n = \vec{u}(t_n), f^n = f(t_n, \vec{u}^n)$$

Conditions initiales : $\vec{u}^0 = \vec{U}$

→ on connaît les p fonctions $f_i(t, u_1, u_2, \dots, u_p)$

→ on cherche à identifier les solutions $u_1(t), u_2(t), \dots, u_p(t)$ passant par l'état initial

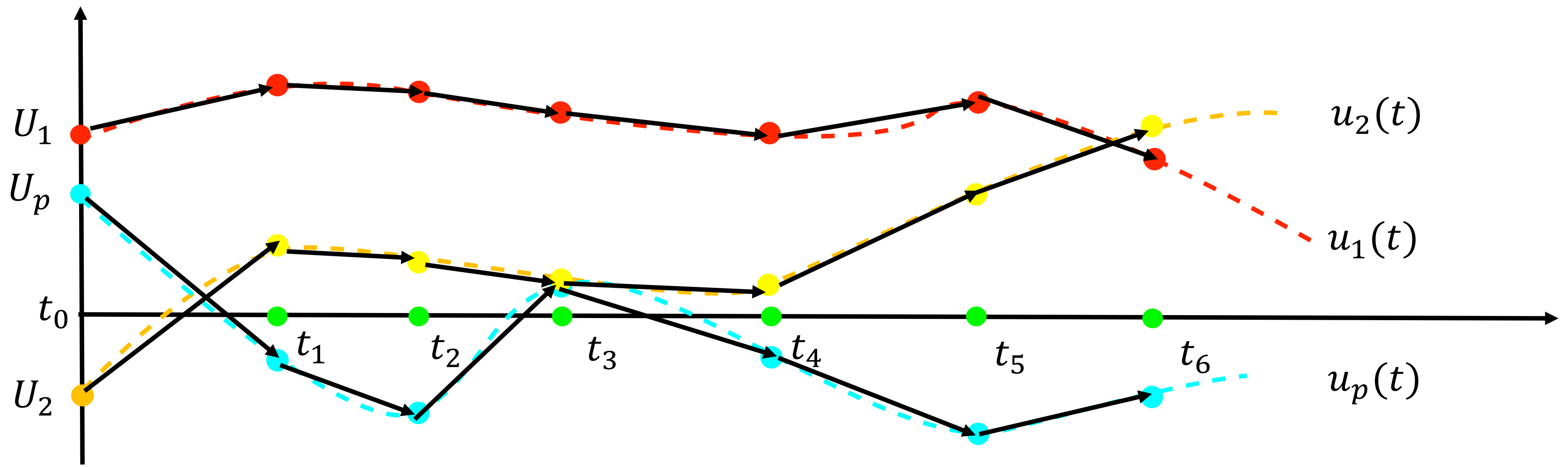
On peut écrire le problème sous forme intégrale :

$$\vec{u}^{n+1} = \vec{u}^n + \int_{t_n}^{t_{n+1}} f(t', \vec{u}(t'))dt'$$

→ Pour résoudre le problème, il s'agit alors de trouver comment approximer l'intégrale

Visualisation du problème

L'idée générale d'une intégration temporelle numérique peut être illustrée ainsi :



→ on calculera donc la solution pas à pas :

$$\underbrace{\begin{pmatrix} U_1 \\ U_2 \\ \dots \\ U_p \end{pmatrix}}_{\vec{u}^0} \xrightarrow[?]{t_1 - t_0} \underbrace{\begin{pmatrix} u_1^1 \\ u_2^1 \\ \dots \\ u_p^1 \end{pmatrix}}_{\vec{u}^1} \xrightarrow[?]{t_2 - t_1} \underbrace{\begin{pmatrix} u_1^2 \\ u_2^2 \\ \dots \\ u_p^2 \end{pmatrix}}_{\vec{u}^2} \xrightarrow[?]{} \dots \xrightarrow[?]{t_N - t_{N-1}} \underbrace{\begin{pmatrix} u_1^N \\ u_2^N \\ \dots \\ u_p^N \end{pmatrix}}_{\vec{u}^N}$$

?

avec quelle méthode ?

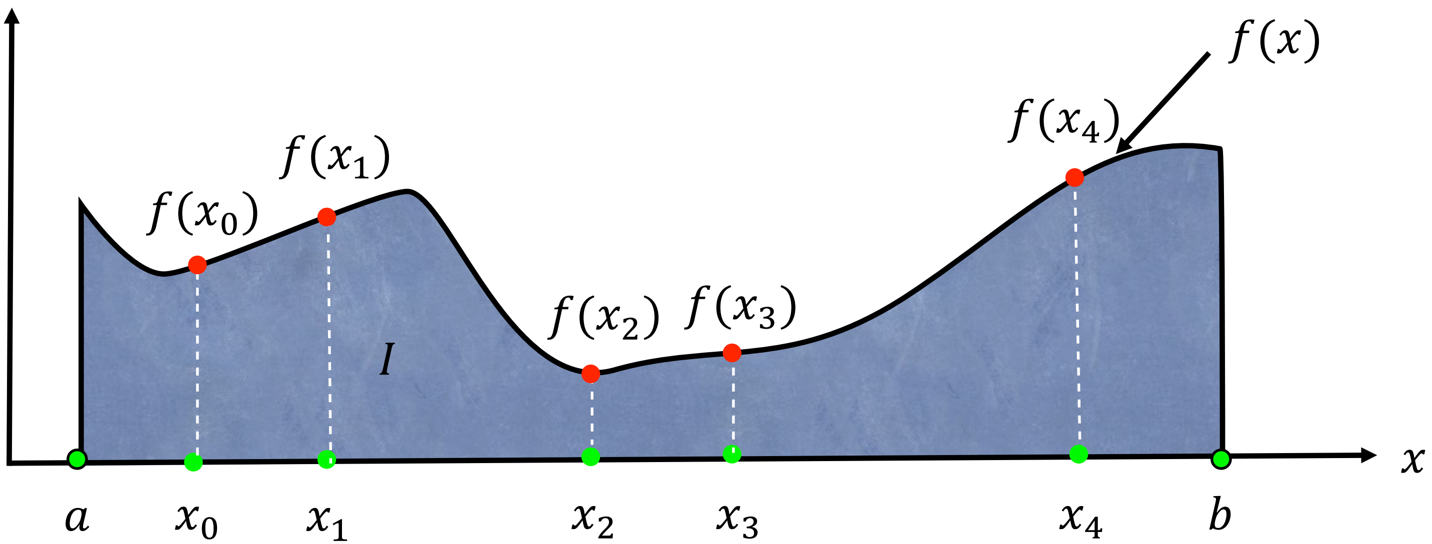
Intégration numérique

On a notre problème sous forme intégrale :

$$\vec{u}^{n+1} = \vec{u}^{n} + \int_{t_n}^{t_{n+1}} f\big(t', \vec{u}(t')\big) dt'$$

→ Approximer l'intégrale = calculer l'aire sous la courbe f :

$$I = \int_a^b f(x)dx$$



→ Problème : dans le cas numérique, on a un problème discret, pas une courbe continue !

→ Il faut être capable de réaliser une interpolation de la fonction et de l'intégrale

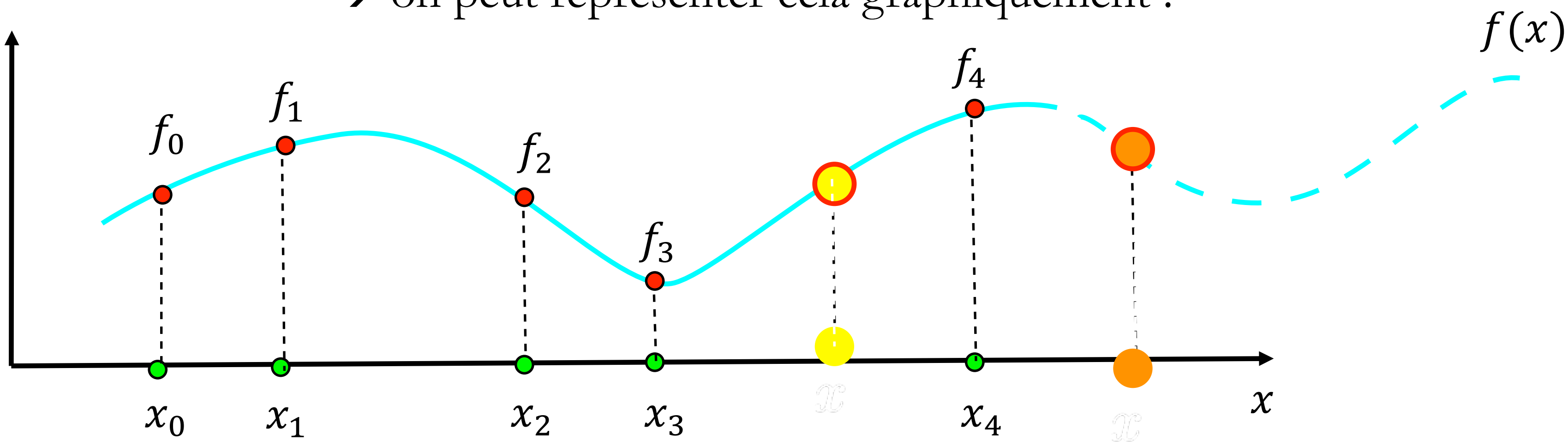
Interpolation (I)

Qu'est-ce que l'interpolation ?

Ex : un astronome au XVème siècle mesure la position des étoiles chaque nuit

instant	x_0	x_1	x_2	…	x_n
position	f_0	f_1	f_2	…	f_n

→ on peut représenter cela graphiquement :



→ avec ces mesures, on voudrait déterminer la position de l'étoile n'importe quand

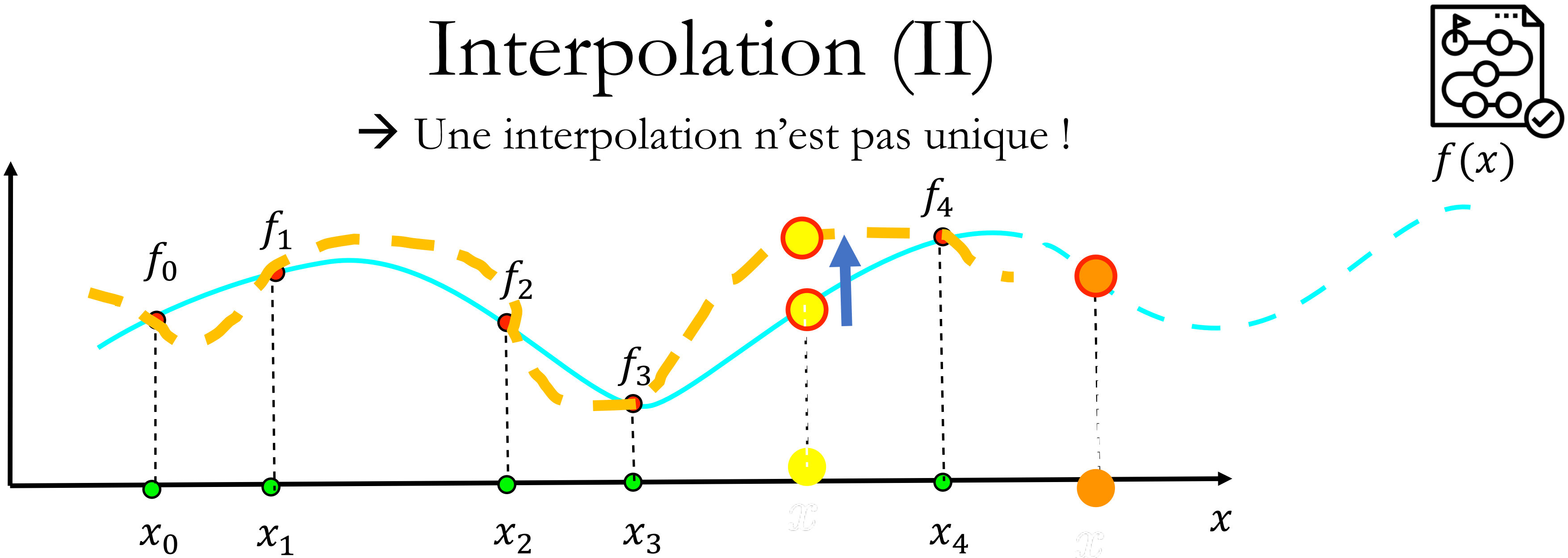
→ Interpolation : $x \in [x_0, x_n]$

→ Extrapolation : $x < x_0, x > x_n$

→ pour cela, il faut construire une fonction d'interpolation

Interpolation (II)

→ Une interpolation n'est pas unique !



→ Une interpolation possède une précision ! (cf. exemple plus tard)

→ Une interpolation se construit à l'aide d'une base de décomposition :

$$f(x) \approx \begin{cases} \sum_k c_k x^k & \text{(base polynomiale simple)} \\ \sum_k c_k e^{ikx} & \text{(base de Fourier)} \\ \sum_k c_k \phi_k(x) & \text{(base générale)} \end{cases}$$

→ Le but est toujours d'arriver à calculer les coefficients d'expansion c_k

Interpolation et EDP

L'interpolation ne sert pas juste à trouver une fonction

$$f(x) \approx \sum_k c_k \phi_k(x)$$

→ on peut s'en servir pour :

une dérivée :

$$f'(x) \approx \sum_k c_k \phi'_k(x)$$

une intégrale :

$$\int_0^1 f(x)dx \approx \sum_k c_k \left(\int_0^1 \phi_k(x)dx \right)$$

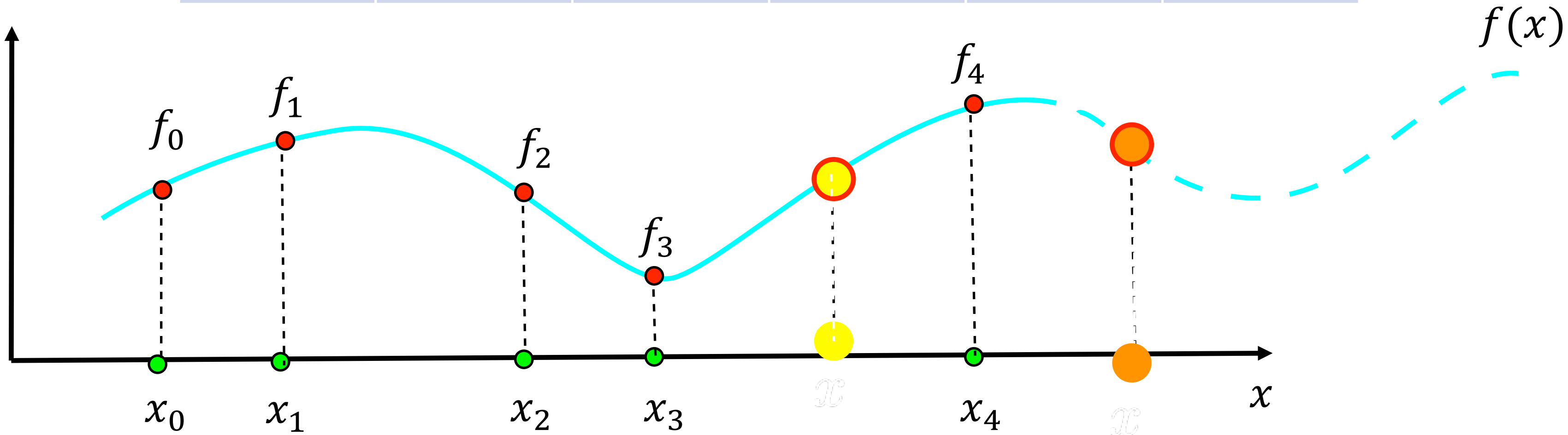
→ Ceci nous donne des formules très utiles,
qui constituent le point de départ d'un grand nombre de méthodes numériques !

<u>NB :</u> Dans ce cours, on se limitera à l'interpolation polynomiale de Lagrange

Polynôme de Lagrange : Problème

Reprenons notre problème initial :

instant	x_0	x_1	x_2	…	x_n
position	f_0	f_1	f_2	…	f_n



Le polynôme de Lagrange est le polynôme unique d'ordre n qui passe exactement par les $n+1$ points considérés → commençons avec une base polynomiale :

$$f(x) \approx P_L(x) = \sum_k c_k x^k \,(x)$$

Le problème devient alors :

comment trouver les $n+1$ coefficients c_k pour reconstruire le polynôme ?

Polynôme de Lagrange : Définition

En discrétisant le domaine, on peut mettre le polynôme sous forme d'un système linéaire :

$$i \in \{0,1,\dots,n\}$$

$$f(x_i) \approx f_i = P_L(x_i) = \sum_k c_k x_i^k\,(x) \quad \Rightarrow \quad \underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}}_{\boldsymbol{M}} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \dots \\ c_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \dots \\ f_n \end{bmatrix}$$

→ $\det(\boldsymbol{M}) \neq 0 \Longrightarrow$ il existe une unique solution au système

Lagrange a alors introduit le polynôme suivant à cause de ses propriétés :

$$l_k(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}$$

$$= \prod_{i=0, i\neq k}^{n} \frac{(x-x_i)}{(x_k-x_i)} \quad \Rightarrow \quad \begin{cases} i \neq k : l_k(x_i) = 0 \\ i = k : l_k(x_k) = 1 \end{cases} \qquad (l_k(x_i) = \delta_{ik})$$

→ On obtient alors la formulation unique du polynôme d'interpolation de Lagrange :

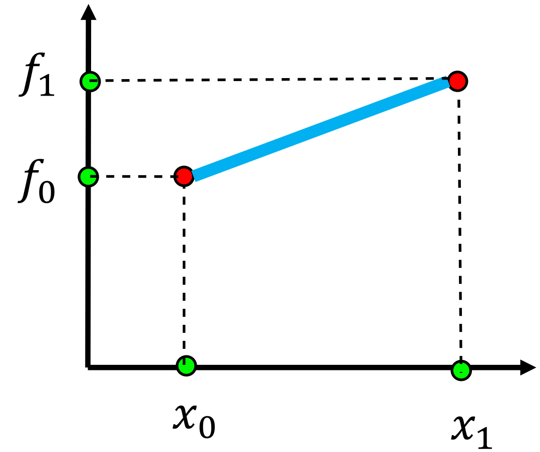
$$f(x) \approx P_L(x) = \sum_k f_k l_k\,(x) \quad \Rightarrow \quad P_L(x_i) = \sum_k f_k l_k\,(x_i) = \sum_k f_k \delta_{ik} = f_i$$

Polynôme de Lagrange : Exemples

Polynôme de Lagrange passant par 2 points = droite :

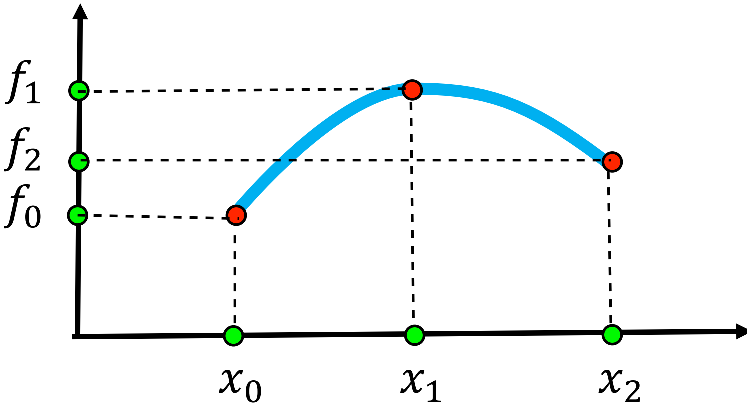
$$P_L(x) = \sum_k f_k l_k\,(x) \qquad l_k(x) = \prod_{i=0, i\neq k}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

$$P_L(x) = f_0 \frac{(x - x_1)}{(x_0 - x_1)} + f_1 \frac{(x - x_0)}{(x_1 - x_0)} = \frac{(f_1 - f_0)}{(x_1 - x_0)}(x - x_0) + f_0$$



Polynôme de Lagrange passant par 3 points = parabole :

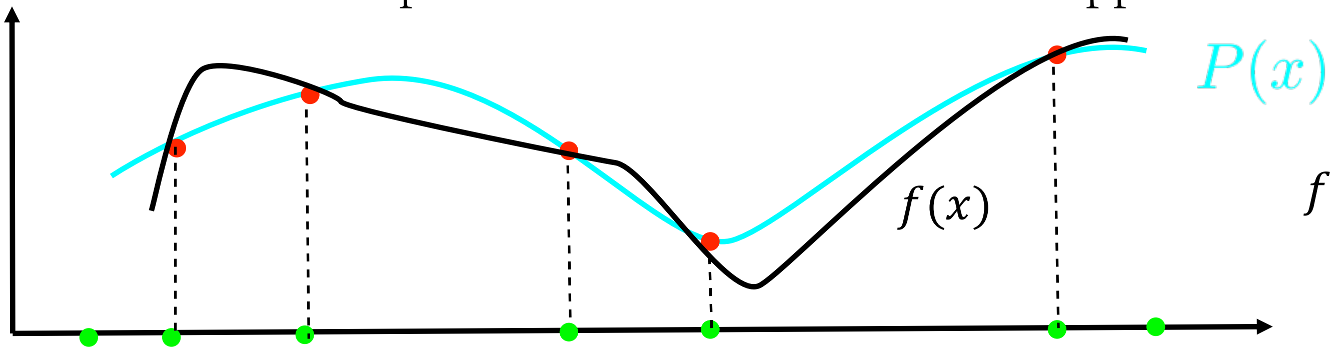
$$P_L(x) = f_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$
$$+ f_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + f_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$



→ Question-piège : et le polynôme de Lagrange passant par 1 point ?

Polynôme de Lagrange : Erreur

On voudrait pouvoir mesurer l'erreur due à l'approximation en polynôme :



$$f(x) = P_L(x) + erreur$$

Théorème utile :

Soit $f(x)$ une fonction n+1 dérivable sur l'intervalle [a,b]

Soient n+1 points : $a \leq x_0 < x_1 < \cdots < x_n \leq b$ où la fonction prend les valeurs $f_0, f_1, \ldots, f_n$

$$f(x) - P_L(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_n)}{(n+1)!} f^{(n+1)}(\xi), a \leq \min(x, x_0) < \xi < \max(x, x_n) \leq b$$

Si f(x) est un polynôme d'ordre $p \leq n$
→ l'erreur s'annule ($f^{(n+1)}(x) = 0$)

Que se passe-t-il si on augmente n ?
Si x est proche de x_i ?
Si f varie doucement ?

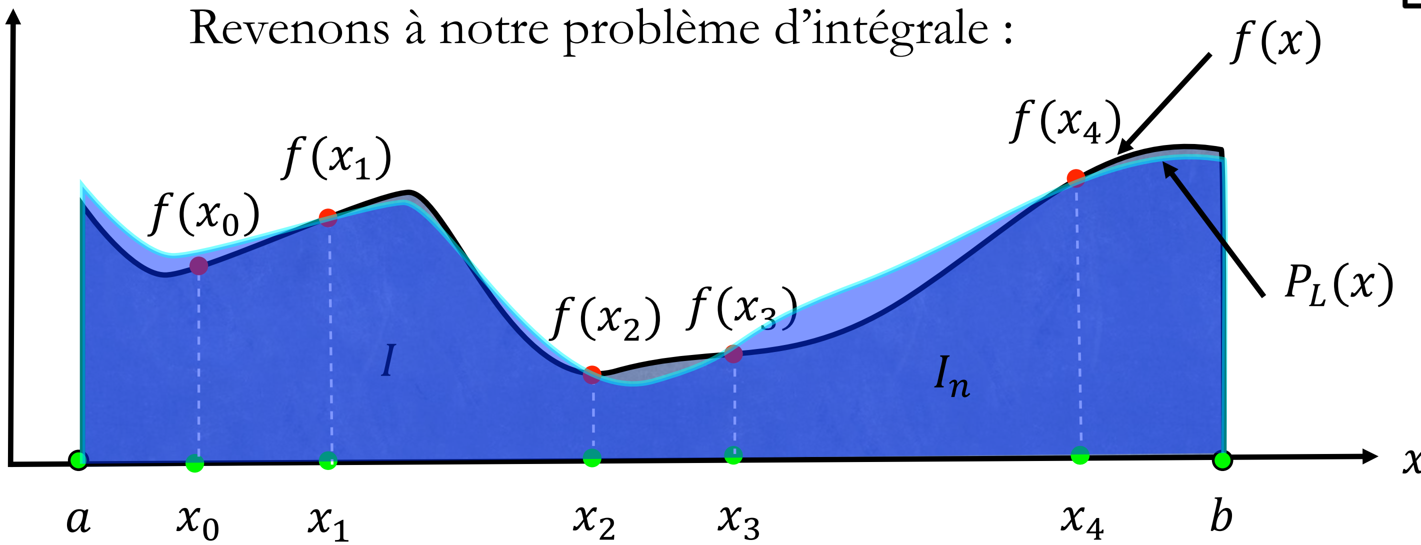
→ On peut alors borner l'erreur :

$$|f(x) - P_L(x)| \leq \frac{|(x - x_0)(x - x_1) \ldots (x - x_n)|}{(n+1)!} \max_{x \in [a,b]} |f^{(n+1)}(x)|$$

Polynôme de Lagrange et interpolation

Revenons à notre problème d'intégrale :

$$I = \int_a^b f(x)dx$$



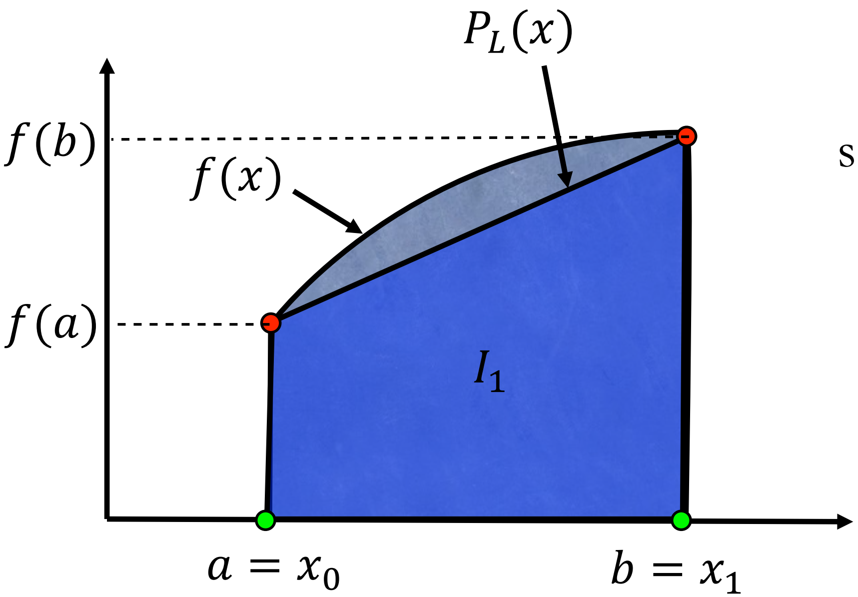
→ On remplace donc la fonction par le polynôme d'interpolation de Lagrange :

$$I \approx I_n = \int_a^b P_L(x)dx \quad \Rightarrow \quad I_n = \sum_{k=0}^{n} A_k^{(n)} f_k$$

→ On obtient alors une formule pour les coefficients de quadrature d'interpolation :

$$A_k^{(n)} = \int_a^b l_k(x)dx$$

Exemple : Méthode du trapèze



<u>Principe de la méthode :</u>

Pour approcher l'intégrale, on approxime l'aire sous la courbe par l'aire du trapèze correspondant :

$$I \approx \frac{(f(b) + f(a))(b-a)}{2}$$

<u>Lien avec le polynôme de Lagrange :</u>

Cela revient à utiliser un polynôme de Lagrange qui passe par deux points :

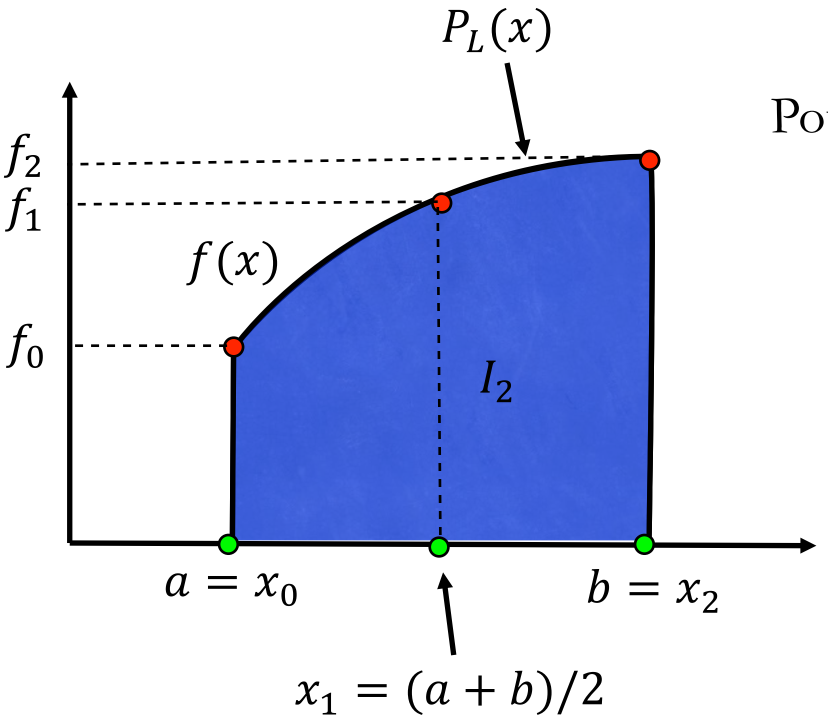
$$I = \int_a^b f(x)dx$$

$$I_1 = \int_a^b \left[f(a)\frac{(x-b)}{(a-b)} + f(b)\frac{(x-a)}{(b-a)}\right] dx$$

$$= f(a)\left(\frac{b-a}{2}\right) + f(b)\left(\frac{b-a}{2}\right)$$

$$= \left(\frac{f(a)+f(b)}{2}\right)(b-a)$$

Méthode de Simpson



Principe de la méthode :

Pour approcher l'intégrale, on approxime l'aire sous la courbe par l'aire sous une parabole

Lien avec le polynôme de Lagrange :

Cela revient à utiliser un polynôme de Lagrange qui passe par trois points :

$$I = \int_a^b f(x)dx$$

$$I_2 = \int_a^b \left[f_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \right] dx$$

$$= \left(\frac{b-a}{6}\right) [f(a) + 4f(x_1) + f(b)]$$

Analyse d'erreur

Avec le polynôme de Lagrange, on peut facilement quantifier l'erreur :

$$R = I - I_n = \int_a^b \big(f(x) - P_L(x)\big)dx = \int_a^b \frac{v(x)}{(n+1)!} f^{(n+1)}(\xi)\,,\, v(x) = (x - x_0)(x - x_1)\dots(x - x_n)$$

Méthode du trapèze (2 points, n=1) :

$$R = I - I_1 = \frac{f''(\xi)}{2} \int_a^b (x-a)(x-b)dx = -\frac{(b-a)^3}{12} f''(\xi)$$

$$|R| \leq \frac{(b-a)^3}{12} \max_{x \in [a,b]} |f''(x)|$$

Méthode de Simpson (3 points, n=2) :

$$R = I - I_2 = \frac{f'''(\xi)}{6} \int_a^b (x-a)\left(x - \frac{a+b}{2}\right)(x-b)dx$$

$$|R| \leq \frac{(b-a)^5}{90} \max_{x \in [a,b]} |f'''(x)|$$

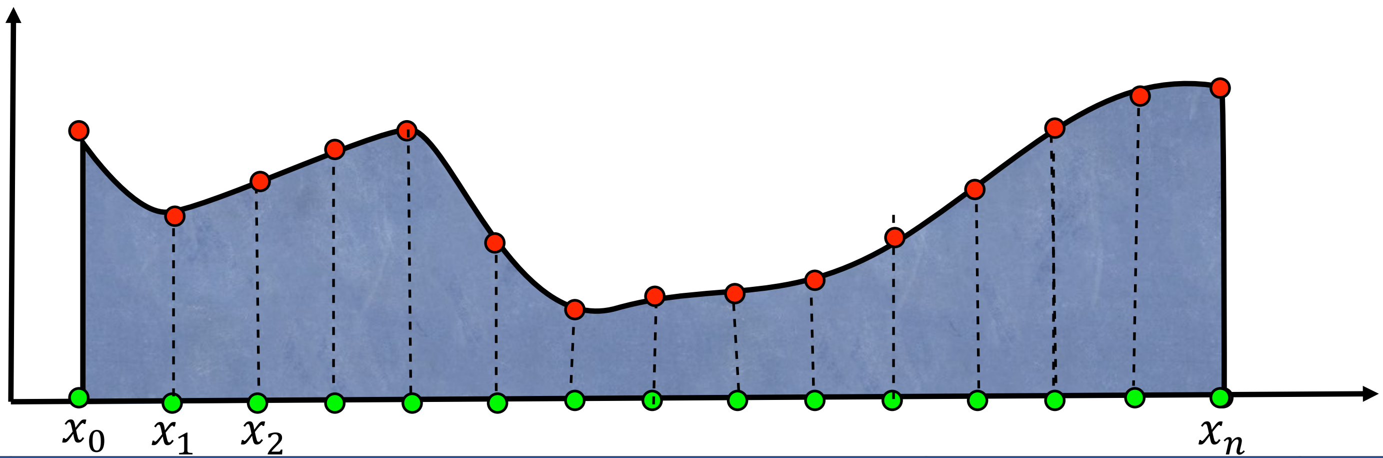
Formules de Newton-Cotes

On peut utiliser des polynômes d'interpolation d'ordres de plus en plus élevé :

Degré	Nom commun	Formule	Terme d'erreur
1	Méthode des trapèzes	$\frac{b-a}{2}(f_0+f_1)$	$-\frac{(b-a)^3}{12}f^{(2)}(\xi)$
2	Méthode de Simpson 1/3	$\frac{b-a}{6}(f_0+4f_1+f_2)$	$-\frac{(b-a)^5}{2880}f^{(4)}(\xi)$
3	Méthode de Simpson 3/8	$\frac{b-a}{8}(f_0+3f_1+3f_2+f_3)$	$-\frac{(b-a)^5}{6480}f^{(4)}(\xi)$
4	Méthode de Boole-Villarceau	$\frac{b-a}{90}(7f_0+32f_1+12f_2+32f_3+7f_4)$	$-\frac{(b-a)^7}{1935360}f^{(6)}(\xi)$
6	Méthode de Weddle-Hardy	$\frac{b-a}{840}(41f_0+216f_1+27f_2+272f_3+27f_4+216f_5+41f_6)$	$-\frac{(b-a)^9}{1567641600}f^{(8)}(\xi)$

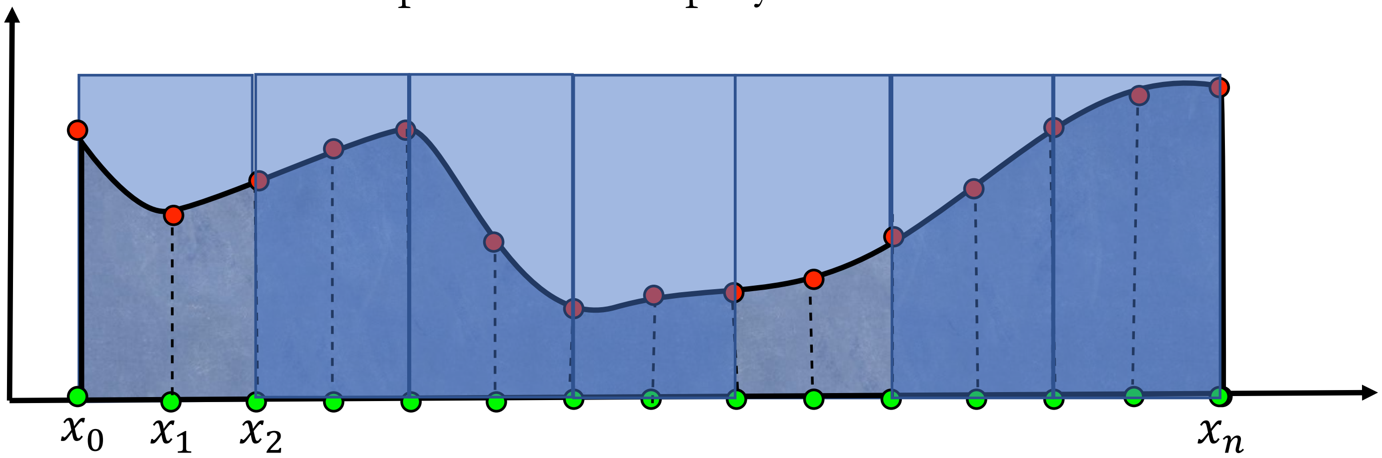
MAIS ATTENTION !

Sur de grands intervalles, ce n'est pas forcément plus précis !



Méthodes composites

À la place, on préfère diviser l'intervalle en sous-intervalles où on interpole avec des polynômes de bas ordre :



Exemple avec la méthode des trapèzes :

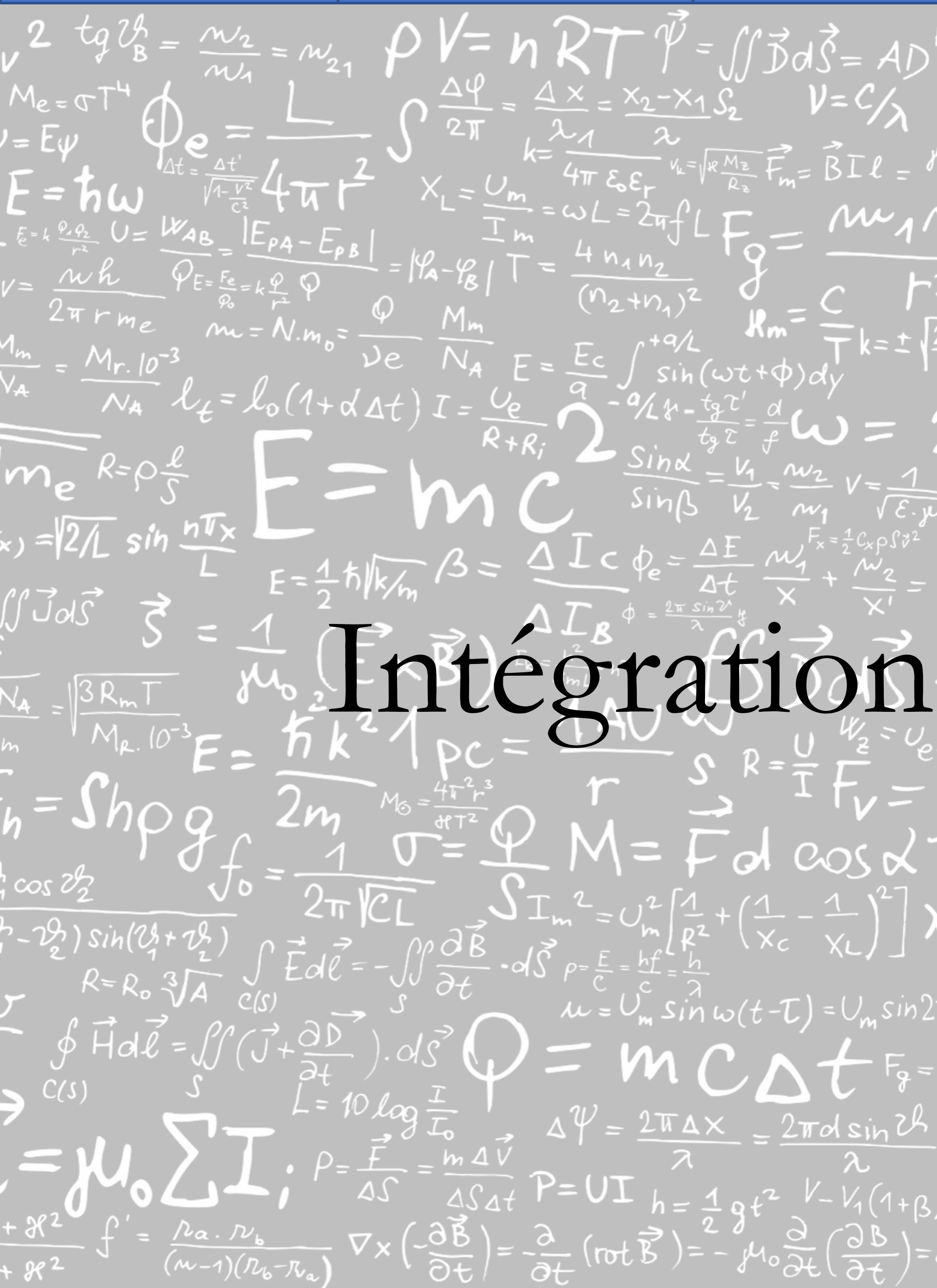
$$I = \int_a^b f(x)dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{i=0}^{n-1} I_i \,, I_i = (x_{i+1} - x_i)\left(\frac{f_i}{2} + \frac{f_{i+1}}{2}\right) + R$$

L'erreur diminue avec le nombre de sous-intervalle qu'on considère :

$$|R| \leq \frac{(b-a)^3}{12n^2} \max_{x\in[a,b]} |f''(x)|$$

Polynôme de Lagrange : Exercice

Petite exercice : interpolons un trajet de RER B (bien connu !)



Intégration temporelle

Formulation du problème

Cette fois-ci, on s'intéresse à une EDO qui dépend du temps

→ système de p EDO d'ordre 1 :

$$\left\{\begin{array}{l} \dfrac{du_1}{dt} = f_1(t, u_1, u_2, \dots, u_p) \\ \dfrac{du_2}{dt} = f_2(t, u_1, u_2, \dots, u_p) \\ \qquad\qquad \dots \\ \dfrac{du_p}{dt} = f_p(t, u_1, u_2, \dots, u_p) \end{array}\right.$$

$t \in [t_0, t_f]$

$$\frac{d\vec{u}}{dt} = f(t, \vec{u}), \vec{u} = (u_1, u_2, \dots, u_p)$$

Discrétisation : $t_0 < t_1 < \cdots < t_N = t_f$

$$\vec{u}^n = \vec{u}(t_n), f^n = f(t_n, \vec{u}^n)$$

Conditions initiales : $\vec{u}^0 = \vec{U}$

→ on connaît les p fonctions $f_i(t, u_1, u_2, \dots, u_p)$

→ on cherche à identifier les solutions $u_1(t), u_2(t), \dots, u_p(t)$ passant par l'état initial

On peut écrire le problème sous forme intégrale :

$$\vec{u}^{n+1} = \vec{u}^n + \int_{t_n}^{t_{n+1}} f(t', \vec{u}(t'))dt'$$

→ Pour résoudre le problème, il s'agit alors de trouver comment approximer l'intégrale

Lien avec le polynôme de Lagrange

$$\vec{u}^{n+1} = \vec{u}^{n} + \int_{t_n}^{t_{n+1}} f\big(t', \vec{u}(t')\big)dt'$$

En remplaçant $f(t', \vec{u}(t'))$ par un polynôme de Lagrange,
on peut donc calculer la solution

MAIS ATTENTION
il reste des choix à faire ! qui auront un impact sur la solution finale !
→ le choix du schéma numérique

Ordre du schéma
= degré du polynôme

Explicite vs. implicite
= choix des points d'interpolation

Formulation explicite vs. implicite

En fonction du schéma choisi, on a la possibilité d'exprimer le problème sous forme explicite ou implicite :

Explicite

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

$$f(t+\delta t) = F[f(t), f(t-\delta t), \ldots]$$

ex : $\frac{f(t_{n+1})-f(t_n)}{\delta t} = -f^2(t_n)$

→ La résolution est itérative

Avantages :

Plus intuitif, modélisation plus fine

Inconvénients :

Gourmande en ressources, limitée par la propagation de l'information

Implicite

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

$$G[\ldots, f(t-\delta t), f(t), f(t+\delta t)] = 0$$

ex : $\frac{f(t_{n+1})-f(t_n)}{\delta t} = -f^2(t_{n+1})$

→ La résolution est matricielle

Avantages :

Plus simple à formuler, moins instable

Inconvénients :

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

Méthodes de Adams-Bashfort

Les méthodes d'Adams-Bashfort sont des méthodes explicites
= on interpole des points qui sont déjà connus (présent et passé)

Adams-Bashfort 1 = Euler explicite

$$f(t', \vec{u}(t')) \rightarrow P(t') = f_n$$

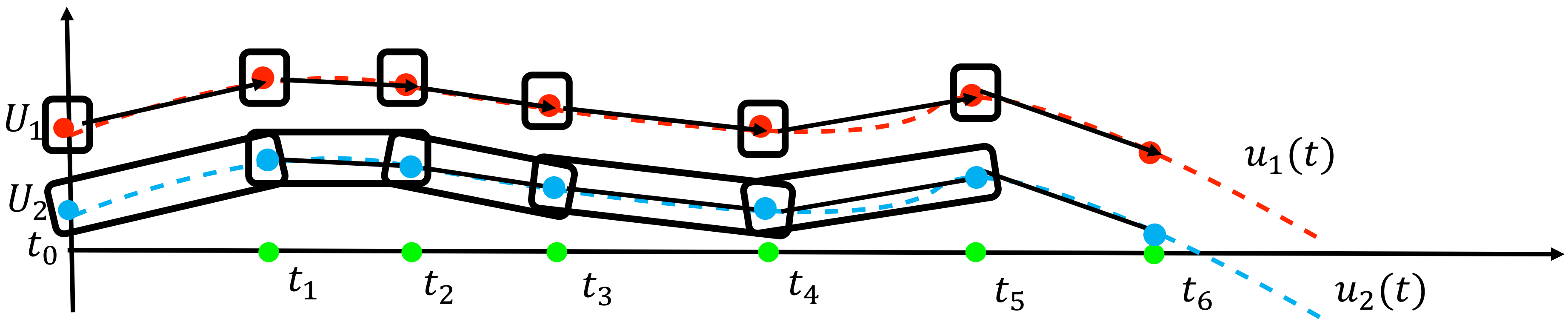
$$\vec{u}^{n+1} = \vec{u}^n + \int_{t_n}^{t_{n+1}} [f_n + O(\delta t')] dt'$$

$$\vec{u}^{n+1} = \vec{u}^n + \delta t f_n + O(\delta t^2)$$

Adams-Bashfort 2

$$f(t', \vec{u}(t')) \rightarrow P(t') = f_{n-1} \frac{t' - t_n}{t_{n-1} - t_n} + f_n \frac{t' - t_{n-1}}{t_n - t_{n-1}}$$

$$\vec{u}^{n+1} = \vec{u}^n + \delta t \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) + O(\delta t^3)$$



Méthodes de Adams-Moulton

Les méthodes d'Adams-Moulton sont des méthodes implicites
= on interpole des points qui sont ne sont pas encore connus (futur)

Adams-Moulton 1
= Euler implicite

$$f(t',\vec{u}(t')) \to P(t') = f_{n+1}$$

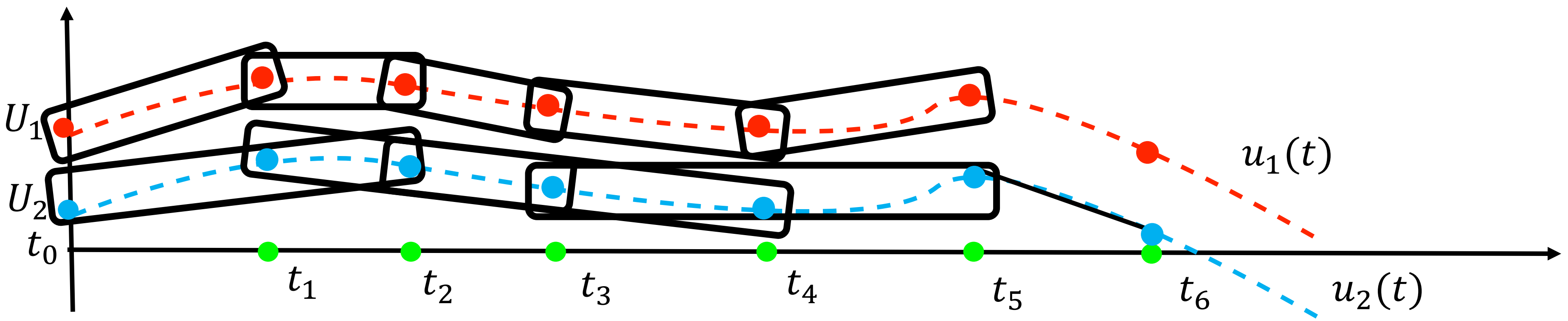
$$\vec{u}^{n+1} = \vec{u}^n + \int_{t_n}^{t_{n+1}} [f_{n+1} + O(\delta t')]dt'$$

$$\vec{u}^{n+1} = \vec{u}^n + \delta t f_{n+1} + O(\delta t^2)$$

Adams-Moulton 2
= Crank-Nicolson

$$f(t',\vec{u}(t')) \to P(t') = f_n \frac{t'-t_{n+1}}{t_n-t_{n+1}} + f_{n+1} \frac{t'-t_n}{t_{n+1}-t_n}$$

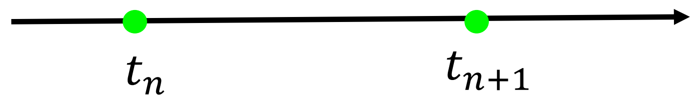
$$\vec{u}^{n+1} = \vec{u}^n + \frac{1}{2}\delta t(f_{n+1}+f_n) + O(\delta t^3)$$



Méthodes de Runge-Kutta

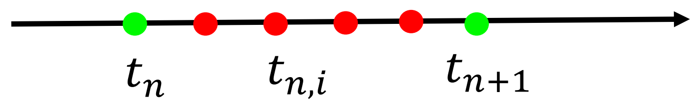
Enfin, les méthodes de Runge-Kutta sont parmi les plus utilisées en physique
→ méthodes itératives où on introduit des points intermédiaires :

$$\vec{u}^{n+1} = \vec{u}^{n} + \int_{t_n}^{t_{n+1}} f\big(t', \vec{u}(t')\big)dt'$$



$$\vec{u}^{n,i} = \vec{u}^{n} + \int_{t_n}^{t_{n,i}} f\big(t', \vec{u}(t')\big)dt' = \vec{u}^{n} + \delta t_n \int_0^{c_i} f\big(t_n + h\delta t_n, \vec{u}(t_n + h\delta t_n)\big)dh$$

$$t_{n,i} = t_n + c_i \delta t_n$$



$$\vec{u}^{n+1} = \vec{u}^{n} + \int_{t_n}^{t_{n+1}} f\big(t', \vec{u}(t')\big)dt' = \vec{u}^{n} + \delta t_n \int_0^{1} f\big(t_n + h\delta t_n, \vec{u}(t_n + h\delta t_n)\big)dh$$

La méthode de Runge-Kutta est donc donnée par :

$$\forall i = 1, \dots \boxed{q}, \begin{cases} t_{n,i} &= t_n + \boxed{c_i} h_n, \\ y_{n,i} &= y_n + h_n \sum_{k=1}^{i-1} \boxed{a_{ik}} p_{n,k} \\ p_{n,i} &= f(t_{n,i}, y_{n,i}) \end{cases}$$

$$y_{n+1} = y_n + h_n \sum_{k=1}^{q} \boxed{b_k} p_{n,k}.$$

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_q & a_{q1} & a_{q2} & \cdots & a_{q,q-1} & \\ \hline & b_1 & b_2 & \cdots & b_{q-1} & b_q \end{array}$$

(tableau de Butcher)

Exemples de Runge-Kutta (I)

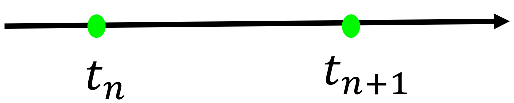
Quelques exemples :

Runge-Kutta d'ordre 1 = RK1

$h_n = \delta t$

$$\forall i = 1, \ldots, q, \begin{cases} t_{n,i} & = t_n + c_i h_n, \\ y_{n,i} & = y_n + h_n \sum_{k=1}^{i-1} a_{ik} p_{n,k} \\ p_{n,i} & = f(t_{n,i}, y_{n,i}) \end{cases}$$

$$y_{n+1} = y_n + h_n \sum_{k=1}^{q} b_k p_{n,k}.$$



$q = 1$

$b_1 = 1$

$c_1 = 0$

$a_{11} = 0$

$$\boxed{\vec{u}^{n+1} = \vec{u}^n + \delta t f(t_n, \vec{u}^n)}$$

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_q & a_{q1} & a_{q2} & \cdots & a_{q,q-1} & \\ \hline & b_1 & b_2 & \cdots & b_{q-1} & b_q \end{array}$$

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

Exemples de Runge-Kutta (II)

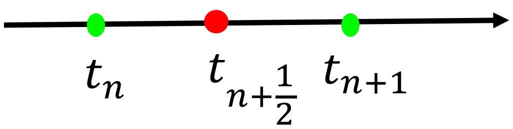
Quelques exemples :

Runge-Kutta d'ordre 2 = RK2

$$h_n = \delta t$$

$$\forall i = 1, \ldots, q, \begin{cases} t_{n,i} & = t_n + c_i h_n, \\ y_{n,i} & = y_n + h_n \sum_{k=1}^{i-1} a_{ik} p_{n,k} \\ p_{n,i} & = f(t_{n,i}, y_{n,i}) \end{cases}$$

$$y_{n+1} = y_n + h_n \sum_{k=1}^{q} b_k p_{n,k}.$$



$$\vec{u}^{n+1/2} = \vec{u}^n + \frac{\delta t_n}{2} f(t_n, \vec{u}^n)$$

$$\vec{u}^{n+1} = \vec{u}^n + \delta t_n f(t_{n+1/2}, \vec{u}^{n+1/2})$$

$$q = 2$$

$$b_1 = 0, b_2 = 1$$

$$c_1 = 0, c_2 = 1/2$$

$$a_{11} = 0, a_{21} = 1/2, a_{12} = 0, a_{22} = 0$$

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_q & a_{q1} & a_{q2} & \cdots & a_{q,q-1} & \\ \hline & b_1 & b_2 & \cdots & b_{q-1} & b_q \end{array}$$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}$$

Exemples de Runge-Kutta (III)

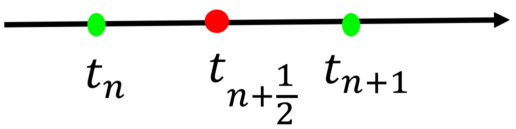
Quelques exemples :

Runge-Kutta d'ordre 4 = RK4

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

$$q = 4$$

$$c_1 = 0, c_2 = \frac{1}{2}, c_3 = \frac{1}{2}, c_4 = 1$$



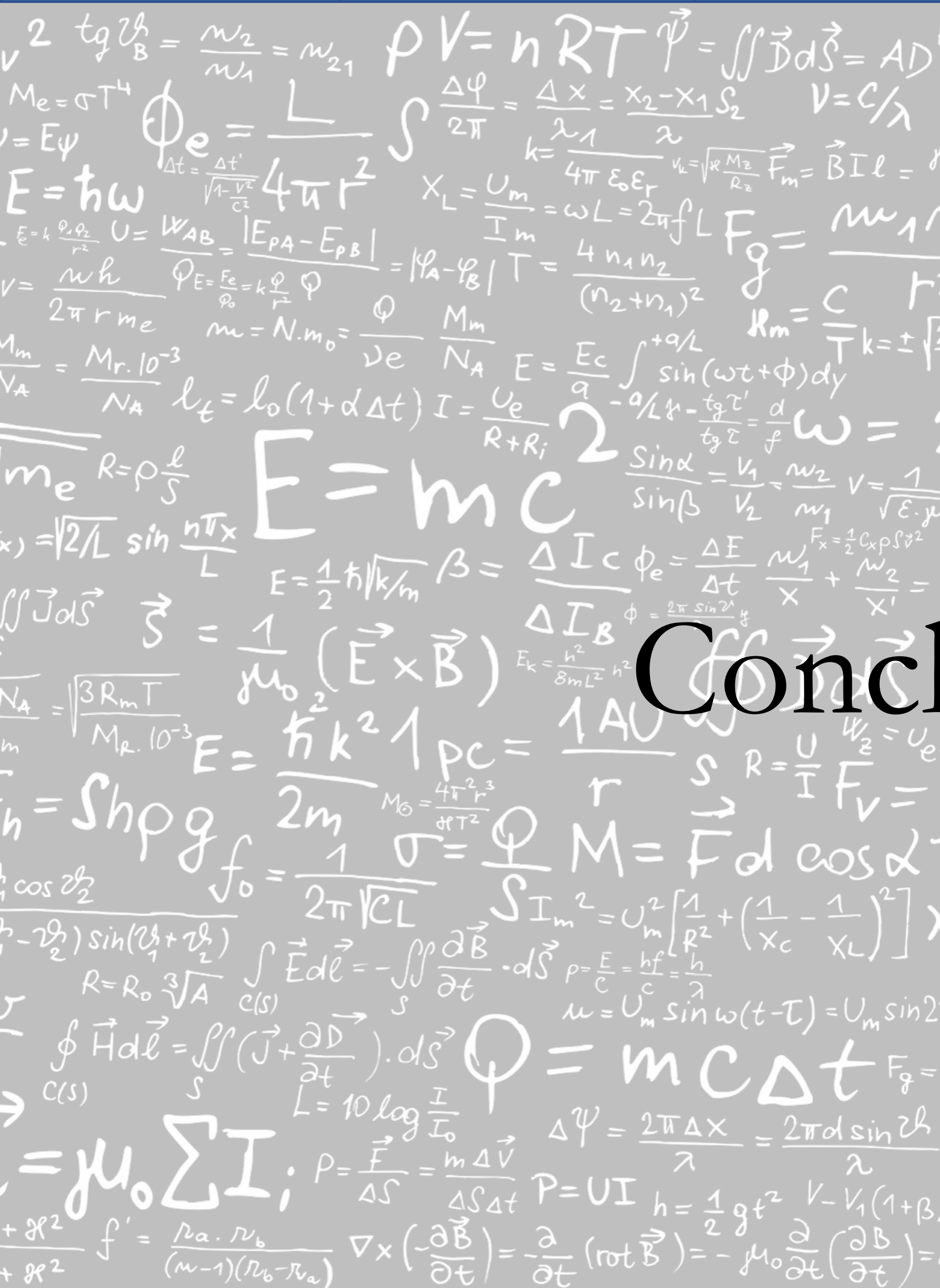
$$k_1 = f(t_n, \vec{u}^n)$$

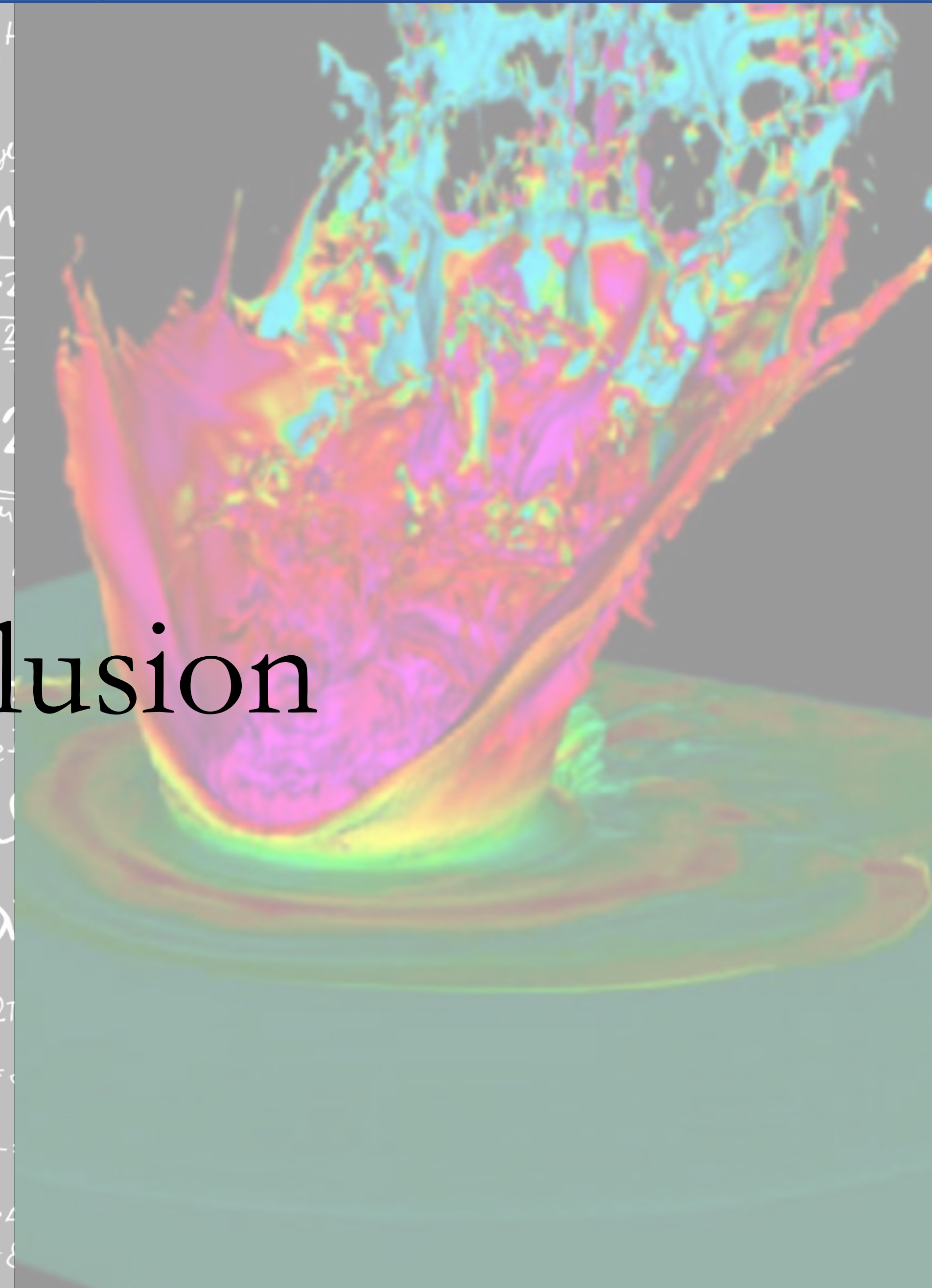
$$k_2 = f(t_n + \frac{\delta t_n}{2}, \vec{u}^n + \frac{\delta t_n}{2} k_1)$$

$$k_3 = f(t_n + \frac{\delta t_n}{2}, \vec{u}^n + \frac{\delta t_n}{2} k_2)$$

$$k_4 = f(t_n + \delta t_n, \vec{u}^n + \delta t_n k_3)$$

$$\vec{u}^{n+1} = \vec{u}^n + \frac{\delta t_n}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$





Conclusion

Récapitulatif

Les équations aux dérivées ordinaires ont l'avantage d'être plus facilement intégrables

On obtient une expression analytique de l'intégrale

Le problème revient à trouver le zéro d'une fonction

Méthodes de Newton

Le problème reste sous forme intégrale

Le problème revient à trouver une quadrature d'interpolation

Polynôme de Lagrange (interpolation + quadrature)

Application à l'UE

Résolution d'EDO

Polynôme de Lagrange

Quadrature d'interpolation

Premiers schémas (Adams-Bashfort, Euler, Adams-Moulton, Crank-Nicholson, Runge-Kutta)

TP1

Méthodes de Newton

TP2

Méthodes de Runge-Kutta

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

Modalités d'évaluation :

TPs + examen oral (question de cours + exercice)