UNIVERSITÉ DE LORRAINE

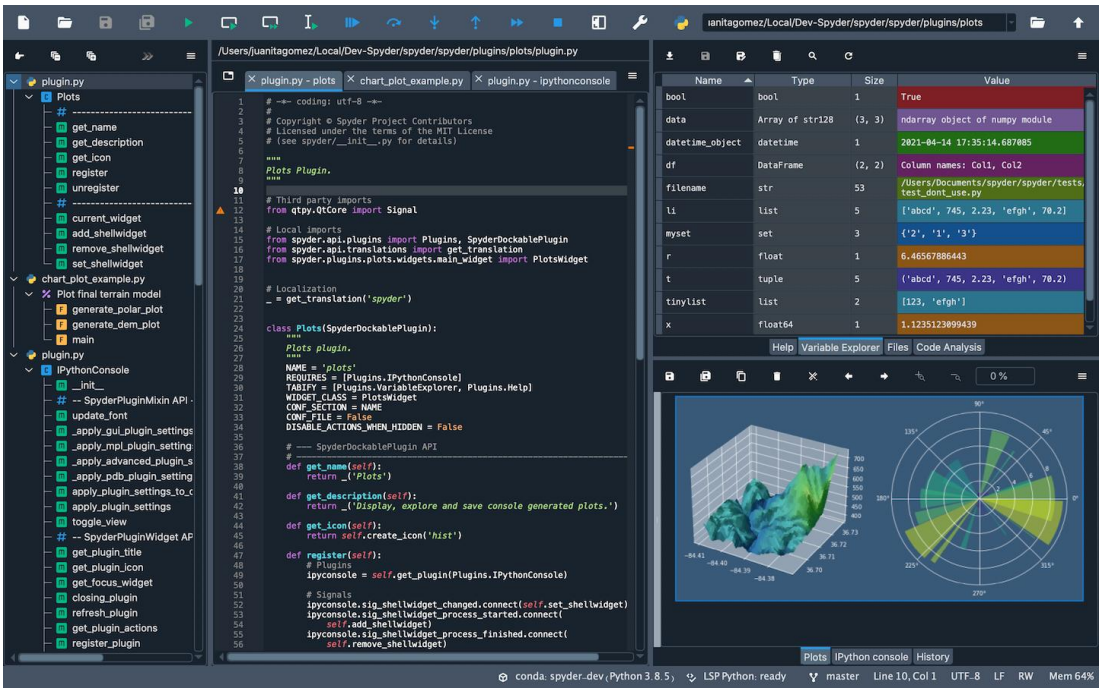
FST FACULTÉ DES SCIENCES ET TECHNOLOGIES Nancy

UE 503.1 ANALYSE NUMERIQUE - L3 MFE

POLYCOPE DE COURS, TD, TP

python™

spyder The Scientific Python Development Environment



Olivier BOTELLA

LEMTA – Université de Lorraine - CNRS
2 avenue de la Forêt de Haye, TSA 60604
54518 Vandoeuvre (France)

Email : Olivier.Botella@univ-lorraine.fr

Lecture web page : https://arche.univ-lorraine.fr/course/view.php?id=6384

Interpolation

I) Exemple "naïf"

Les observations scientifiques de la concentration d'ozone dans la ville de Nancy a produit le tableau suivant :

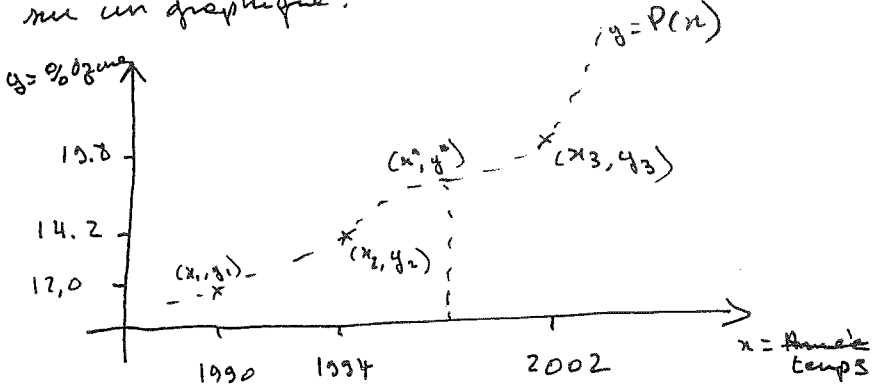
ozone (%)	$y_1 = 12.0$	$y_2 = 14.2$	?	$y_3 = 19.8$	?
Année	$x_1 = 1990$	$x_2 = 1994$	$x_0 = 1998$	$x_3 = 2002$	$x_{new} = 2006$

On souhaite maintenant :

* estimer la concentration d'ozone en $x_0 = 1998$
  ⇒ Problème d'interpolation.
* Prédire la ——— en $x_{new} = 2006$
  ⇒ Problème d'extrapolation.

La résolution de ces 2 problèmes fait appel à des techniques d'approximation polynomiale.

Naïvement, cela consiste à reporter les données du tableau sur un graphique :



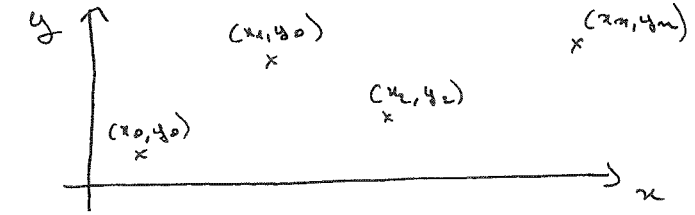
Ensuite, on cherche à une fonction "suffisamment simple" qui passe par ces données (par ex. un polynôme $y = P(x)$). Il suffit alors de calculer $y_0 = P(x_0)$ avec points précédents (ici $x_0 = 1998$ ou 2002).

II Interpolation polynômiale

Rappels sur les polynômes :

- On note $\mathbb{P}_N$ l'ensemble des polyn. de degré $\leq N$.
  (ex. : $\mathbb{P}_1$: polynômes de la forme $ax+b$; $\mathbb{P}_2$: ——— ax^2+bx+c, $bx+c$, ou c
- $P(x) \in \mathbb{P}_N$ possède au plus N racines complexes.
- $\mathbb{P}_N$ est un espace vectoriel de dimension $N+1$. Sa base canonique est $\{1, x, x^2, \dots, x^N\}$.

II-1 Formulation du problème



Supposons qu'on dispose d'un ensemble de $n+1$ données

$$\{(x_i, y_i),\ i = 0 \dots n\}$$

que l'on connaît sous forme tabulaire

x_0	x_1	---	x_i	---	x_n
y_0	y_1		y_i		y_n

On cherche le polynôme $p(x)$ de degré n tel que

$$p(x_i) = y_i \quad , \quad \text{pour } 0 \le i \le n. \qquad (1)$$

Théorème (Propriété importante)
Le polynome d'interpolation $p(x)$ est unique.

exemple : - par 2 points passe une seule droite ($n=1$)
- par 3 —— —————— parabole ($n=2$)
- par $n+1$ —————— polyn. de degré n.

Démonstration : Supposons qu'il existe un autre polynome q qui vérifie (1). Alors, le polynome

$$r(x) = p(x) - q(x)$$

possède $n+1$ racines $x_0, \dots, x_n$. C'est seulement possible si $\forall x \; r(x) \equiv 0$.

II-2 Une méthode naïve

Pour résoudre le problème d'interpolation (1), une première possibilité est d'écrire $p(x) \in \mathbb{P}_n$ dans sa base canonique :

$$p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n = \sum_{j=0}^{n} a_j x^j .$$

Le problème (1) revient à chercher les $n+1$ coefficients $\{a_j\}$ à partir des $n+1$ données $\{x_i, y_i\}$, soit :

$$a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_n x_i^n = y_i \quad , \quad 0 \le i \le n$$

$$\Leftrightarrow \quad \sum_{j=0}^{n} (x_i)^j a_j = y_i \quad , \quad 0 \le i \le n. \qquad (2)$$

Le calcul des $\{a_j\}$ par (2) revient à résoudre le système linéaire de taille $(n+1) \times (n+1)$:

$$\underline{\underline{A}} \; \underline{X} = \underline{B}, \qquad (3)$$

avec $\underline{X} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$, $\underline{B} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$

$$\underline{\underline{A}} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}$$

Résoudre "efficacement" un système linéaire n'est pas une tâche triviale (cf : plus loin dans la année).
Dans la suite de ce cours, nous allons utiliser une technique d'interpolation plus astucieuse, qui utilise la base de Lagrange $\{L_j(x), j = 0, \dots n\}$ plûtot que la base canonique $\{x^j, j = 0, \dots, n\}$ pour représenter le polynome $p(x)$.

II 3 Interpolation de Lagrange

En premier lieu, considérons le problème d'interpolation des données $\{(x_i, y_i),\ i = 0 \dots n\}$ représentées sur le tableau

Tab2

0	0	- - -	0	1	0	- - -	0
x_0	x_1		x_{j-1}	x_j	x_{j+1}		x_n

On cherche donc le pol. d'interpolation $L_j(x) \in \mathbb{P}_n$ qui vérifie

$$\text{Pour } i = 0, \dots, n : \quad L_j(x_i) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \qquad (4)$$

(Rq : Notation de Kronecker $L_j(x_i) = \delta_{ij}$).

Ce polynôme de degré n possède n racines $\{x_0, x_1, \dots, x_{j-1}, x_{j+1}, \dots\}$ il est donc forcément de la forme :

$$L_j(x) = \alpha_j (x - x_0)(x - x_1) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)$$
$$= \alpha_j \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i),$$

où α_j est une constante. Pour la déterminer, il suffit de rq :

$$L_j(x_j) = 1 \iff \alpha_j = \frac{1}{\prod_{\substack{i=0 \\ j \neq i}}^{n} (x_j - x_i)}.$$

Définition : Le polynôme

$$L_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{(x - x_i)}{(x_j - x_i)} \qquad (5)$$

est appelé le $j^{\text{ème}}$ polynôme de Lagrange.

Maintenant, pour interpréter le problème :

0	0		y_j		0
x_0	x_1		x_j		x_n

l'interpolant s'écrit :

$$p(x) = y_j L_j(x).$$

Pour le problème plus général :

y_0	y_1	- - -	y_j	- - -	y_n
x_0	x_1	- - -	x_j	- - -	x_n

l'interpolant est une combinaison linéaire des polynômes de Lagrange :

$$p(x) = \sum_{j=0}^{n} y_j L_j(x). \qquad (6)$$

Remarque (Approximation de fonctions)

Un des problèmes de la théorie mathématique de l'approximation de fonctions est de chercher une approximation "simple" (p. ex. polynomiale) d'une fonct. $f : \mathbb{R} \to \mathbb{R}$ quelconque.

Si on suppose que les valeurs $\{f(x_i)\}$ de la fonction sont connues en certains points $\{x_i, i = 0, \dots, n\}$, alors le polynôme d'interpolation de Lagrange de $f(x)$ s'écrit

$$\sum_{j=0}^{n} f(x_j) L_j(x).$$

On note généralement $\Pi_n f$ le polynôme d'interpolation

* L'interpolation de Lagrange est une interpolation compacte, mieux adaptée au calcul numérique que l'interpolation "naïve" dans la base $\{x^i\}$, car elle ne nécessite pas la résolution de systèmes linéaires (cf II-

* Lorsque le nombre de données n est grand, l'interpolation polynomiale peut donner de mauvais résultat (cf TD, TP : phénomène de Runge). Bien que le polynome "adhère" aux données par construction, il peut osciller fortement entre ces points.
(Math, cela se traduit par le fait que $\Pi_n f$ ne converge pas uniformément vers f.).

Remèdes contre ces oscillations :

- Si on peut choisir les noeuds x_i : ne pas les prendre équidistants, mais plutôt choisir les points de Tchébychev, qui sont définis dans $\mathcal{U} =]a,b[$ par :

$$x_j = a + \frac{b-a}{2}\left[1 + \cos\frac{(2j+1)\pi}{2n+2}\right], \quad j = 0, 1, \dots,$$

- Utiliser l'interpolation basée sur les polynomes par morceaux.

$$x_i = \frac{a+b}{2} - \frac{(b-a)}{2}\cos(i\pi/n), \quad i = 0, \dots, n.$$

III Interpolation par intervalles

Rappel: Espaces fonctionnels $C^k(\mathcal{U})$.

$C^k(\mathcal{U}) = \{ f$, tels que : f est continue sur $\mathcal{U}$,
• $f^{(i)}$ est continue sur $\mathcal{U}$ pour $1 \le i \le$

exple: $C^0(\mathcal{U})$: ensemble des fonctions continues.

$C^1(\mathcal{U})$: fonctions continues à dérivée première co

III-1 Notion de polynômes par morceaux (piecewise polynomes)

On considère l'intervalle $\mathcal{U} =]a,b[$, et ~~la partition de~~ des noeuds répartis sur $\mathcal{U}$:

$$\mathcal{U}_h = \{ x_i, \ i = 0, \dots, n \}$$

A partir de ces noeuds, on définit les intervalles $\mathcal{U}_i =]x_i,$ de longueur $h_i = x_{i+1} - x_i$.

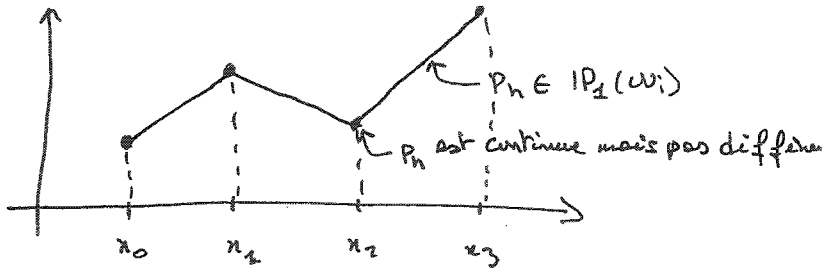
Déf: Un polynome par morceaux $P_h(x)$ sur $\mathcal{U}_h$ est tel que
- P_h est un polynome de degré k sur $\mathcal{U}$;
- $P_h \in C^{k-1}(\mathcal{U})$.

rq: En particulier, on remarque que la dérivée $k^{ième}$ de P_h est discontinue sur les noeuds x_i.

Les polyn. p.m. les plus utilisés sont les polynômes p.m $\mathbb{P}_1$, qui sont tels que :

- $P_h(x) \in \mathbb{P}_1$ sur ω_i (de la forme $ax+b$)
- $P_h(x)$ est seulement continue aux noeuds x_i.

Un exemple de polyn. p.m est :



III-2 Base de Lagrange des polyn. par morceaux $\mathbb{P}_1$

Pour pouvoir représenter $P_h(x)$ de façon commode, on définit sa base de Lagrange :

$$\{ \varphi_j(x), \quad j = 0, \dots, n \},$$

et $P_h(x)$ s'écrit alors :

$$P_h(x) = \sum_{j=0}^{n} \alpha_j \varphi_j(x) \qquad (\alpha_j \in \mathbb{R}).$$

Les fonctions de base sont les "fonctions chapeaux" définies,

$$\varphi_j(x) = \begin{cases} 0 & \text{si } x \leq x_{j-1} \\ \dfrac{x - x_{j-1}}{x_j - x_{j-1}} & \text{si } x \in \omega_{j-1} = [x_{j-1}, x_j] \\ \dfrac{x_{j+1} - x}{x_{j+1} - x_j} & \text{si } x \in \omega_j = [x_j, x_{j+1}] \\ 0 & \text{si } x \geqslant x_{j+1} \end{cases}$$

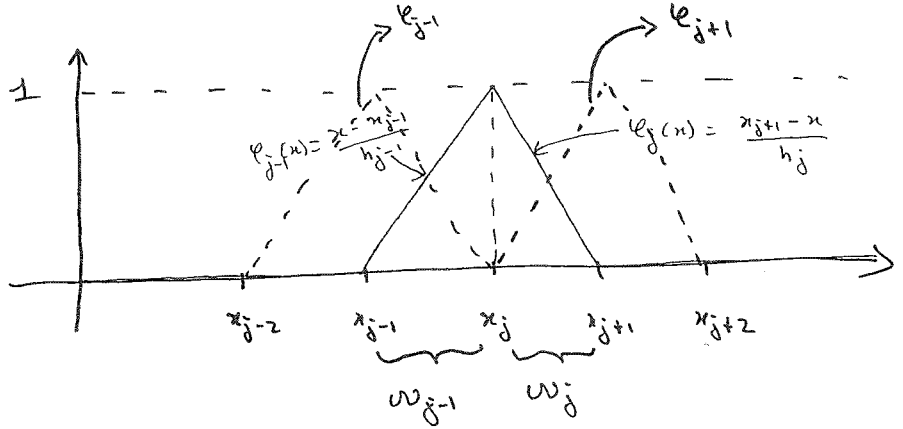


Fig: 1 Représentation des fonctions de base $\varphi_j(x)$

* Dans l'intervalle $\omega_i = [x_i, x_{i+1}]$, on sait que seules 2 fonctions de bases φ_{i-1} et φ_i sont $\neq 0$, donc

$$P_h(x)\Big|_{\omega_i} = \alpha_i \varphi_i(x) + \alpha_{i+1} \varphi_{i+1}(x)$$

$$= \alpha_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + \alpha_{i+1} \frac{x - x_i}{x_{i+1} - x_i}$$

On retrouve bien le fait que $P_h(x)$ est un polynome de degré 1 dans les intervalles ω_i.

* Les fonctions $\varphi_j(x)$ sont continues aux noeuds x_i, mais pas dérivable.

$$\Rightarrow P_h(x) \in \mathcal{C}^0(\omega)\,.$$

* Il est important de remarquer que $\varphi_j(x)$ s'annule sur tous les noeuds sauf x_j :

$$\varphi_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

III-3 Application à l'interpolation de fonctions

Soit $f(x)$ une fonction à interpoler aux noeuds x_i.
On veut la représenter par un polynome p.m. $\mathbb{P}_1$ tel que :

$$f(x_i) = P_h(x_i) \quad \text{pour } i = 0, \dots, n$$

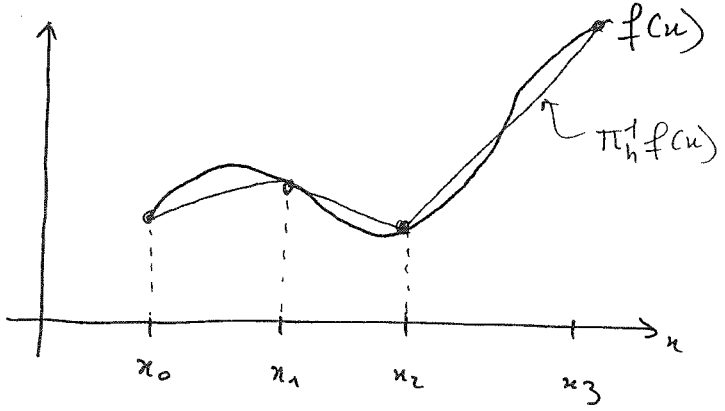
$$= \sum_{j=0}^{n} \alpha_j \varphi_j(x_i)$$

$$= \alpha_i \quad \text{puisque } \varphi_j(x_i) = \delta_{ij}\,.$$

Donc l'interpolant de $f(x)$ s'écrit simplement :

$$\sum_{j=0}^{n} f(x_j)\, \varphi_j(x)\,.$$

On note généralement $\Pi_1^h f(x)$ cet interpolant.



Ce type d'interpolation est couramment appellée "interpolation polynomial par morceaux" (morceaux = intervalles).

Remarques :

* L'avantage de l'interpolation polynomiale par intervalle est de supprimer les oscillations parasites de l'interpol polynomiale globale (cf TP).

* Les polynomes p.m. $\mathbb{P}_1$ ont de nombreuses applications dans la résolution des eqts de la mécanique (méth. des elements finis en méca des solides).

* Le principal défaut des polyn. p.m. $\mathbb{P}_1$ est leur manque de régularité (continuité mais pas C^1) aux noeuds x_i. Pour y remédier, on utilise généralement l'interpolant $\Pi_h^3 f$ qui a les propriétés :

- $\pi_h^3 f(x_i) = f(x_i)$ pour $i = 0, \dots, n$ (1

- $\pi_3^h f \in \mathbb{P}_3(\omega_i)$

- $\pi_3^h f \in C^2(\omega)$: il est 2 fois différentiable en x_i.

Ce type d'interpolation est aussi connue sous le nom de splines cubiques. Il a été conçu au début des années 1950 dans les bureaux de Design de Citroën. Depuis, l'interpolation par splines a trouvé d'innombrables applications en CAO.

Intégration numérique

I) Exemples et motivation

Dans les problèmes de mécanique, on trouve souvent des intégrales dont le calcul est difficile ou impossible à effectuer analytiquement (pas de primitives).

Par exemple :

$$\int_0^1 e^{-x^2}\,dx, \qquad \int_0^1 \cos x^2\,dx.$$

Dans ce chapitre, on va décrire un certain nombre de méthodes de quadrature numérique, pour calculer de façon approchée une intégrale donnée. Ces méthodes s'appliquent aussi au cas où la fonction est seulement connue en quelques points du domaine d'intégration (formule pratique).

II) Formules élémentaires

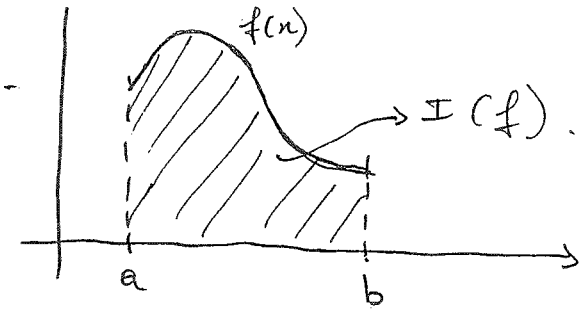
On veut calculer l'intégrale

$$I(f) = \int_a^b f(x)\,dx. \qquad (1)$$

On va ici décrire quelques formules de quadrature, notées $I_0(f)$, telles que

$$I_0(f) \overset{?}{=} I(f). \qquad (2)$$

$\overset{?}{=}$ signifie : on approche



I) Formule des rectangle (ou du point milieu).

On remplace f par

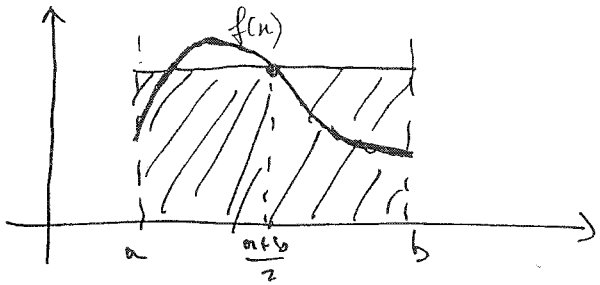
$$\Pi_0(f) = f\left(\frac{a+b}{2}\right),$$

la valeur constante prise par f au milieu de l'intervalle $I = [a, b]$. On obtient

$$\int_a^b \Pi_0(f) = \int_a^b f\left(\frac{a+b}{2}\right) dx = (b-a) f\left(\frac{a+b}{2}\right).$$

On définit donc

$$\boxed{I_0(f) = (b-a) f\left(\frac{a+b}{2}\right)} \overset{?}{=} I(f)$$



2) Formule du trapèze

On remplace $f(x)$ par $\Pi_1(f)$, le polynôme d'interpolation de degré 1 aux points a et b :

$$\Pi_1(f) = \frac{(x-a)}{b-a} f(b) + \frac{(x-b)}{(a-b)} f(a),$$

($\Pi_1 f(x)$)

par application de l'interpolation de Lagrange.

On définit alors

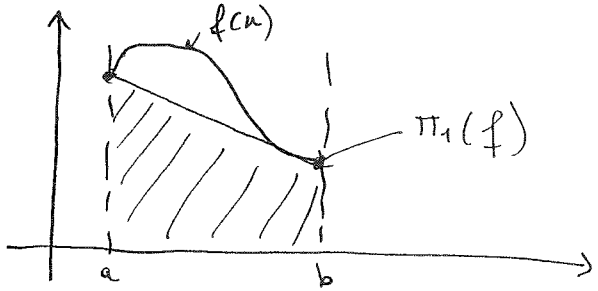
$$I_1(f) = \int_a^b \Pi_1(f)\,dx \overset{?}{=} I(f).$$

Calculons $I_1(f)$:

$$\int_a^b \Pi_1(f) = \frac{f(b)}{b-a}\int_a^b (x-a)\,dx + \frac{f(a)}{a-b}\int_a^b (x-b)\,dx,$$

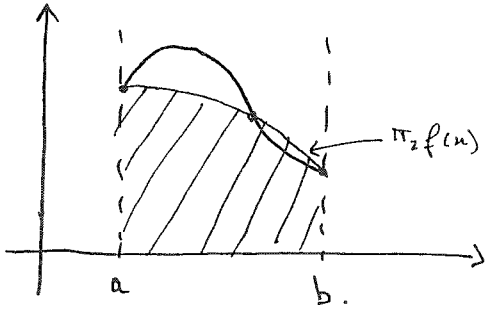
etc..., après calcul :

$$\boxed{I_1(f) = \frac{(b-a)}{2} f(b) + \frac{b-a}{2} f(a)} \qquad (7)$$



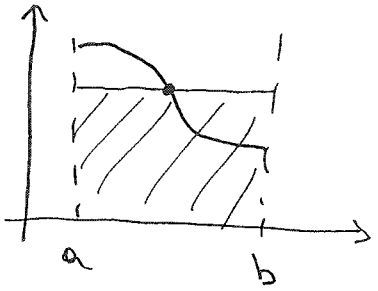
3) Formule de Simpson

On remplace $f(x)$ par son interpolant de degré 2 $\Pi_2 f(x)$ aux points d'interpolation $x_0 = a$, $x_1 = \frac{a+b}{2}$, $x_2 = b$.
(déjà vu en TD).

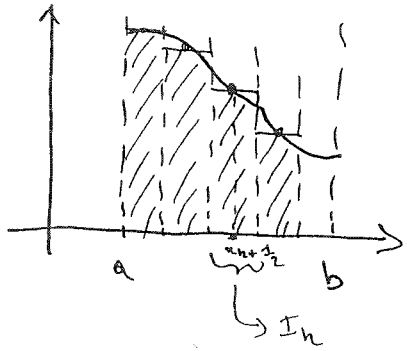


$$I_2(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \qquad (8$$

III Formules composites



formule du rectangle



formule composite du rectangle.

1) généralités

Pour obtenir une meilleure approximation numérique, on pourra utiliser des polyn. de Lagrange de degré de + en + grand. Néanmoins, on se souvient que l'interpolation de Lagrange peut donner des résultats oscillants: en pratique, on utilise rarement des quadratures utilisant des interpolants de Lagr - q de degré ≥ 2.

Une meilleure stratégie est de construire des formules de quadrature composites. Pour cela, on décompose le domaine $\mathcal{D} =]a,b[$ en n sous-domaines

$$\mathcal{D}_h =]x_h, x_{h+1}[, \qquad h = 0, \dots, n-1$$

telle que :

- Leur longueur est $h_h = x_{h+1} - x_h$
- Le milieu de $\mathcal{D}_h$ est noté $x_{h+\frac{1}{2}} = \dfrac{x_h + x_{h+1}}{2}$
- On note $x_0 = a$, $x_n = b$

L'intégrale à calculer ~~devient~~ se décompose comme :

$$\int_a^b f(x)\,dx = \sum_{h=0}^{n-1} \int_{\mathcal{D}_h} f(x)\,dx \qquad (9)$$

Dans chaque sous-domaine, on peut maintenant appliquer une formule de quadrature élémentaire, p. ex. la formule du rectangle :

$$\int_{\mathcal{D}_h} f(x)\,dx \simeq h_h\, f(x_{h+\frac{1}{2}}) \qquad (10)$$

On obtient alors la <u>formule de quadrature composite du rectangle</u>:

$$\int_a^b f(x)\,dx \simeq \sum_{h=0}^{n-1} h_h\, f(x_{h+\frac{1}{2}}). \qquad (11)$$

<u>Cas particulier important</u> :

Dans le cas où chaque intervalle est de longueur h constante. On appellera $I_{a,h}$ la formule composite du rectangle, telle

$$I_{a,h} = h \sum_{h=0}^{n-1} f(x_{h+\frac{1}{2}}).$$

La longueur h est appelée <u>pas de discrétisation</u>.

2) Autres formules composites

- <u>Formule du Trapèze</u> (notée $I_{1,h}$)

Dans chaque intervalle $\mathcal{D}_h$, on applique la formule du trapèze

$$\int_{\mathcal{D}_h} f(x)\,dx \simeq \frac{h_h}{2}\left(f(x_h) + f(x_{h+1})\right)$$

Pour $h_h = h$ constant, on obtient finalement :

$$I_{1,h} = \frac{h}{2}\left[f_0 + 2\sum_{h=1}^{n-1} f(x_h) + f(x_n)\right] \quad (1$$

- <u>Formule de Simpson</u> (notée $I_{2,h}$) :

$$(13) \qquad \int_{\mathcal{D}_h} f(x)\,dx \simeq \frac{h_h}{6}\left[f(x_h) + 4 f(x_{h+\frac{1}{2}}) + f(x_{h+1})\right]$$

et donc, pour $h_h = h$, on obtient :

$$(14) \qquad \dots = \frac{h}{6}\left[f(x_0) + 2\sum f(x_h) + 4\sum f(x_{h+\frac{1}{2}}) + f(x_n\right.$$

IV) Analyse des erreurs

1) Généralités

But de cette partie : Quantifier la précision d'une formule numérique, c'est-à-dire mesurer l'erreur :

$$E_h = |I(f) - I_h(f)| . \qquad (15)$$

Cette erreur doit tendre vers zéro lorsque $h \to 0$.
(Sinon, problème : la formule n'est pas consistante).

Dans la suite de cette partie, on verra que les formules de quadrature de la partie II sont telles que :

$$E_h \leq C h^q \ (= O(h^q)) \ (\text{avec } C \text{ : constante})$$

où $q \geq 0$ est l'ordre de la méthode. Plus q est grand, plus l'erreur sera petite $\Rightarrow$ une méthode d'ordre élevé est (en général) plus précise qu'une méthode du premier ordre ($q = 1$).

Cette analyse des erreurs est effectuée en utilisant les développements de Taylor :

$$f(a+h) = f(a) + h f'(a) + \frac{h^2}{2} f''(a) + \dots + \frac{h^n}{n!} f^{(n)}(a) + \underbrace{O(\dots)}_{\to 0}$$

Définition (petit "oh^n" et grand "Oh^n")

(utilisés dans la suite du cours)

- $f = O(g)$ s'il existe $K \geq 0$ t.q. :
$$\left|\frac{f}{g}\right| \leq K \quad \text{lorsque } h \to 0 .$$
- $f = o(g)$ si :
$$\left|\frac{f}{g}\right| \to 0 \quad \text{lorsque } h \to 0 .$$

2) Analyse des erreurs pour la formule du rectangle :

Sur l'intervalle $\omega_n = [x_n, x_{n+1}]$, la formule du rectangle (pas h constant) s'écrit :

$$\int_{x_n}^{x_{n+1}} f(x)\,dx \simeq h\, f(x_{n+\frac{1}{2}}) .$$

On utilise un D.L. de $f(x)$ au point $x_{n+\frac{1}{2}}$:

$$f(x) = f(x_{n+\frac{1}{2}}) + (x - x_{n+\frac{1}{2}}) f'(x_{n+\frac{1}{2}}) + \frac{(x - x_{n+\frac{1}{2}})^2}{2} f''(x_{n+\frac{1}{2}}) + \dots$$

On intègre ce D.L. dans $[x_n, x_{n+1}]$:

$$\int_{x_n}^{x_{n+1}} f(x)\,dx = \int_{x_n}^{x_{n+1}} f(x_{n+\frac{1}{2}})\,dx + f'(x_{n+\frac{1}{2}}) \underbrace{\int_{x_n}^{x_{n+1}} (x - x_{n+\frac{1}{2}})\,dx}_{=0 \text{ (fonction impaire sur } \omega_n)} + \frac{f''(x_{n+\frac{1}{2}})}{2} \int_{x_n}^{x_{n+1}} (x - x_{n+\frac{1}{2}})^2\,dx + \dots$$

On remarque que les termes de puissance impaires ont une intégrale nulle, et il reste donc :

$$\underbrace{\int_{x_n}^{x_{n+1}} f(x)\,dx}_{\text{intégrale exacte}} = \underbrace{h\, f(x_{n+\frac{1}{2}})}_{\substack{\text{formule} \\ \text{du} \\ \text{rectangle}}} + \underbrace{\frac{h^3}{24} f''(x_{n+\frac{1}{2}}) + \frac{h^5}{1920} f^{(4)}(x_{n+\frac{1}{2}}) + \dots}_{\text{terme d'erreur } E_h \text{ sur } \omega_n} \quad ($$

~~• Si f admet des dérivées d'ordre ≥ 2~~

- Si toutes les dérivées d'ordre ≥ 2 de $f(x)$ sont nulles, alors $E_h = 0 \quad \forall h$. La formule des rectangle est exacte pour les polynomes de degré 1.

- Le terme dominant de l'erreur est telle que $E_h = \mathcal{O}$. Donc la formule des rectangle est précise au troisième ordre sur un intervalle I_h.

Pour obtenir l'ordre d'une formule composite, il faut sommer (16) sur tous les intervalles :

$$\underbrace{\int_a^b f(x)\,dx}_{I(f)} = \underbrace{\sum_{k=0}^{n-1} h f\left(x_{k+\frac{1}{2}}\right)}_{I_{1,h}(f)} + \underbrace{\sum_{k=0}^{n-1} \frac{h^3}{24} f''\left(x_{k+\frac{1}{2}}\right) + \dots}_{E_h}$$

Pour simplifier la démonstration, on écrit que le terme dominant de l'erreur est :

$$E_h = \sum_{k=0}^{n-1} \mathcal{O}(h^3) = h^2 \mathcal{O}(h^2) \sum_{k=0}^{n-1} h = (b-a)\,\mathcal{O}(h^2)$$

$$= \mathcal{O}(h^2).$$

La formule du rectangle composite est donc précise au second-ordre sur l'intervalle tout entier $\Omega =]a,b[$.

3) Formule du trapèze

$$\int_{x_n}^{x_{n+1}} f(x)\,dx \overset{?}{=} \frac{h}{2}\left(f(x_n) + f(x_{n+1})\right)$$

Pour déterminer l'ordre de cette formule, on va utiliser au maximum ce qu'on a vu précédemment.

On considère les D.L. suivants (autour de $x_{n+\frac{1}{2}}$) :

$$f(x_n) = f\left(x_{n+\frac{1}{2}}\right) - \frac{h}{2} f'(x_n) + \frac{1}{2}\left(-\frac{h}{2}\right)^2 f''(x_n) + \frac{1}{6}\left(-\frac{h}{2}\right)^3 f'''(x_n) + \mathcal{O}(h^4)$$

$$f(x_{n+1}) = f\left(x_{n+\frac{1}{2}}\right) + \frac{h}{2} f'\left(x_{n+\frac{1}{2}}\right) + \frac{1}{2}\left(\frac{h}{2}\right)^2 f''\left(x_{n+\frac{1}{2}}\right) + \dots$$

La $\frac{1}{2}$ somme de ces 2 expression conduit à :

$$f\left(x_{n+\frac{1}{2}}\right) = \frac{1}{2}\left(f(x_n) + f(x_{n+1})\right) - \frac{1}{8} h^2 f''\left(x_{n+\frac{1}{2}}\right) - \frac{1}{384} h^4 f^{(4)}\left(x_{n+\frac{1}{2}}\right) + \dots$$

On injecte cette expression dans l'estimation d'erreur (16) du rectangle, pour obtenir :

$$\int_{x_n}^{x_{n+1}} f(x)\,dx = \frac{h}{2}\left(f(x_n) + f(x_{n+1})\right) - \frac{h^3}{12} f''\left(x_{n+\frac{1}{2}}\right) - \frac{h^5}{480} f^{(4)}(x_i) + \dots$$

Conclusions :

- Elle est précise au 3ème ordre
- Son degré d'exactitude est 1
- Dans le cas de la formule composite, on montre comme précédemment que :

$$E_h = (b-a)\,\mathcal{O}(h^2),$$

3) Formule de Simpson

- cf. TD et ci-dessous.

IV) Résumé

Pour un intervalle $I_k = [x_k, x_{k+1}]$, toutes les formules de quadrature peuvent se mettre sous la forme :

$$\int_{x_k}^{x_{k+1}} f(u)\,du \overset{?}{=} I(f) = \sum_{i=0}^{m} \alpha_i f(\zeta_i)$$

avec :
- les $\{\zeta_i\}$ sont les noeuds de la formule
- Les $\{\alpha_i\}$ ——— poids ———

(ex : Rectangle : $m = 0$, $\alpha_i = h_k$, $\zeta_i = x_{k+\frac{1}{2}}$).

Pour obtenir une formule composite, on somme sur les intervalles I_k, $k = 0, \dots, n-1$.

formule	Poids	Noeuds	r	Q	Q_{cm}
Rectangle	h_k	$x_{k+\frac{1}{2}}$	1	3	
Trapèze	$\{\frac{h_k}{2}, \frac{h_k}{2}\}$	$\{x_k, x_{k+1}\}$	1	3	
Simpson	$\{\frac{h_k}{6}, \frac{2}{3}h_k, \frac{h_k}{6}\}$	$\{x_k, x_{k+\frac{1}{2}}, x_{k+1}\}$	3	5	

- r : degré d'exactitude
  (intègre exactement les polynômes de degré r ou plus)
- Q : ordre de précision.

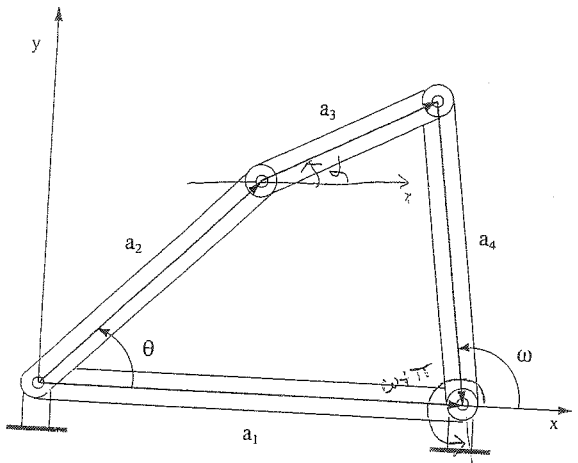


Fig. 1 : Configuration de 4 tiges articulées. On remarque que la tige $\vec{a_1}$ est immobile (ses 2 extrémités sont en appui simple), tandis que les 3 autres sont mobiles.

(I) Exemple et motivation

On considère le système mécanique de la fig. 1 : on cherche à connaître déterminer l'angle θ entre la tige $\vec{a_1}$ et $\vec{a_4}$ lorsque la tige $\vec{a_4}$ est positionnée avec un angle $\omega + \pi = \text{mes}(\vec{ox}, \vec{a_4})$ avec l'horizontale.

Dans une première étape, on met en équation le problème : On part de l'identité vectorielle :

$$\vec{a_1} = \vec{a_2} + \vec{a_3} + \vec{a_4},$$

qu'on projette sur les axes (Ox) et (Oy), soit :

$$a_1 = a_2 \cos\theta + a_3 \cos\alpha + a_4 \cos(\omega+\pi) \qquad (1a)$$

$$0 = a_2 \sin\theta + a_3 \sin\alpha + a_4 \sin(\omega+\pi) \qquad (1b)$$

avec $\alpha = \text{mes}(\vec{\ldots}, \vec{a_3})$

pour éliminer α dans le système (1), ce qui entraîne :

$$(a_1 - a_2\cos\theta + a_4\cos\omega)^2 + (a_4\sin\omega - a_2\sin\theta)^2 = a_3^2$$

Avec un peu de trigonométrie, cette équation devient :

$$\underbrace{\frac{a_1}{a_2}\cos\omega - \frac{a_1}{a_4}\cos\theta - \cos(\omega-\theta) + \frac{a_1^2+a_2^2-a_3^2+a_4^2}{2a_2a_4}}_{f_\omega(\theta)} = 0$$

Le problème de mécanique s'écrit donc sous la forme :

Trouver $\theta \in \mathbb{R}$ (↳ $]-\pi, \pi]$?) tel que : $f_\omega(\theta) = 0$, (2)

où $\omega \in \mathbb{R}$ est un paramètre donné. On remarque que la fonction $f_\omega : \mathbb{R} \to \mathbb{R}$ est non linéaire :

$$\forall \lambda, \beta \quad f_\omega(\lambda\theta_1 + \beta\theta_2) \neq \lambda f_\omega(\theta_1) + \beta f_\omega(\theta_2).$$

Pour ω quelconque, on ne peut trouver une solution analytique au problème (2). Il faut donc utiliser des méthodes numériques de résolution d'équation (ou système d'eqts) non-linéaire.

I) Fonctions/équations linéaires

Une équation linéaire :

$$f(x) = 0. \qquad (1)$$

est une équation qui vérifie le **superposition** des solutions :

si x_1 et x_2 vérifient (1), alors
leur combinaison linéaire $\alpha x_1 + \beta x_2$ ($\alpha, \beta \in \mathbb{R}$)
~~vérifie (1)~~ (autant de (1)) :

(2) $$f(\alpha x_1 + \beta x_2) = \alpha \underbrace{f(x_1)}_{=0} + \beta \underbrace{f(x_2)}_{=0} = 0.$$

- Exemples de fct linéaires :

en 1D : $f(x) = 3x$.

en nD : $\underline{x} = \begin{pmatrix} x \\ y \end{pmatrix}$, $\begin{cases} 3x + 2y = 1 \\ -x + y = 4 \end{cases}$
(système linéaire).

- **Propriétés importantes :**

— Par le principe de superposition de solutions, on peut toujours calculer les solutions analytiques des équations linéaires, par des méthodes systématiques (ex : cours d'EDO du 1er et 2nd ordre).
(ex : cas $a\ddot{y} + b\dot{y} + cy = 0$.

— Malheureusement, la plupart des équations de la physique sont **non-linéaires**.

II) Fnch / equations linéaires :

Une fonction non linéaire ne vérifie pas la superposition de solutions (2).

Exemple : $f(x) = x^2$, $f(x) = \sin(x)$

on a bien sur $f(x_1+x_2) \neq f(x_1) + f(x_2)$.

En conséquence, il n'existe pas de méthodes systématiques pour résoudre les equations non-linéaires, sauf dans des cas très particuliers, p. ex :

- polynômes du 2nd ordre : $ax^2+bx+c=0$ et 3ème ordre, pour les ordres supérieurs ?
- EDO : méthodes de Ricatti, etc ...

Pour résoudre des eqts non-linéaires issus de modèles physiques complexes (niveau Master) il faut utiliser des méthodes numériques.

~~INTÉGRATION NUMÉRIQUE~~
ÉQTS NON-LINÉAIRES

du sept. 2005 NL

II) Généralités

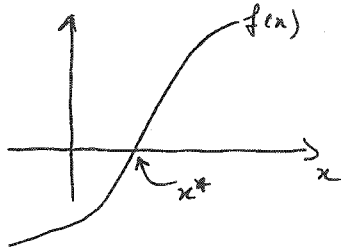
II-1 Positionnement du problème

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction continue. On cherche le(s) zéro(s) x^*, tel que :

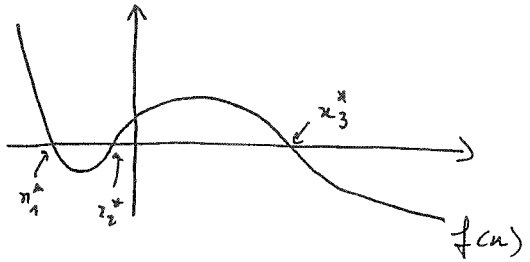
$$f(x^*) = 0.$$

II-2 Exemples rencontrés

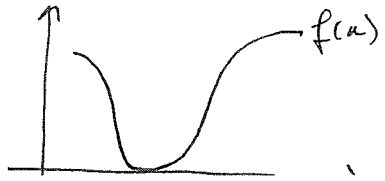
a) Le plus simple : un seul zéro



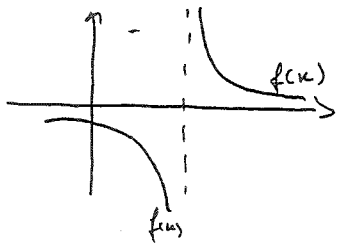
b) + sieurs zéros



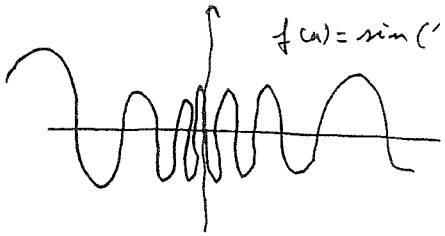
c) Zéros multiples



d) Exemples pathologiques (à éviter)



fonctions discontinues



fonction avec de nombreuses infinité de racines.

II-3 Stratégie à suivre :

a) Localiser grossièrement (ex : graphiquement le(s) zéro(s) x^* : on notera x_0 cette évaluation qu

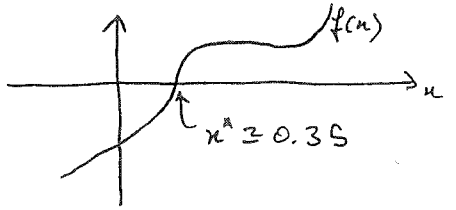
b) Construire à partir de x_0 une suite $(x_1, x_2, \ldots)$ telle que :

$$\lim_{n \to +\infty} x_n = x^*$$

II-4 Exemple test :

Dans la suite des cours, on va considérer l'exemple :

Trouver le zéro de $f(x) = \sin(2x) - 1 + x$ (1



On voit graphiq^t que $x^* \approx 0.35$ est unique. On peut donc raisonnablement utiliser $x_0 \in [0, 0.5]$

Il reste maintenant à définir la suite :

III Méthode de la bissection (ou dichotomie) NL

Une 1ère méthode purement "géométrique" pour définir $\{x_n, n \geq 0\}$ est de remarquer que si on trouve un intervalle $I_0 = [a_0, b_0]$ t.q :

$$f(a_0) f(b_0) < 0 \qquad (3)$$

alors f change de signe dans I_0, et donc $x_\infty \in I_0$ puisque f est continue.

On peut alors subdiviser (faire la dichotomie) I_0 en intervalles plus petits $I_n = [a_n, b_n]$, en utilisant le test (3), de façon à encadrer la valeur x_∞ par des intervalles de + en plus petits: ainsi le centre $x_n = \frac{a_n + b_n}{2} \to x_\infty$ lorsque $n \to$

L'algo de la bissection prend la forme successive:

1) Initialisation : On choisit $I_0 = [a_0, b_0]$, tel que (3) est vérifié, on définit

$$x_0 = \frac{a_0 + b_0}{2}.$$

(Pour le pb test (1), on peut choisir $I_0 = [0, 1] \Rightarrow x_0 = 1/2$).

L'erreur commise s'écrit:

$$e_0 = |x_0 - x_\infty| \leq \frac{b_0 - a_0}{2}.$$

2) Détermination de $I_1 = [a_1, b_1]$ (et donc $x_1 = \frac{a_1 + b_1}{2}$)

Suivant le signe du produit:

$$f(x_0) f(a)$$

cas 1) $f(x_0) f(a) < 0$: alors nécessairement $x^* \in [a, x_0]$. et on définit $I_1 = [a, x_0]$ dont le centre est:

$$x_1 = \text{milieu}(I_1) = \frac{a + x_0}{2}$$

cas 2: $f(x_0) f(a) > 0$: nécessairement $x^* \in I_1 = [x_0, b]$, NL

- dont le centre est :

$$x_1 = \text{milieu}(I_1) = \frac{x_0 + b}{2}.$$

3) Test de convergence: On vérifie si $f(x_1) = 0$.

Si ce n'est pas le cas, on sait quand même que l'erreur est telle que:

$$e_1 = |x_1 - x_0| \leq \frac{b - a}{4}.$$

et on revient à l'itération 2) pour définir x_2 et I_2

De proche en proche, on définit la suite $\{x_n, n \geq 0\}$ qui est telle que:

$$e_n = |x_n - x_\infty| \leq |x_{n-1} - x_\infty| \leq \frac{b - a}{2^{n+1}} \qquad (4)$$

Donc $\varepsilon_n \to 0$ lorsque $n \to \infty$. Clairement, si on a trouvé un bon intervalle de départ I_0, la méthode de bissection converge toujours.

Bien on le verra en TD, la cvgce de la méthode est très lente. Dans la suite des cours, on va définir des méth. plus performantes.

$$|\phi(x_k) - \phi(x_*)| \leq C |x^k - x^*| \quad , \quad 0 \leq C < 1$$

Les fonct. d'itération qui vérifient cette inégalité sont appellées <u>fonctions contractantes</u>. C'est en général difficile à prouver qu'une fonction est contractante. A la place, nous allons utiliser le thm suivant pour obtenir un critère pratique assurant la convergence d'un algo. du point fixe :

<u>Thm (Convergence)</u>. Soit ϕ une fonction d'itération cvt. Si ϕ vérifie la condition de <u>stabilité</u> :

$$|\phi'(x_*)| < 1, \qquad (11)$$

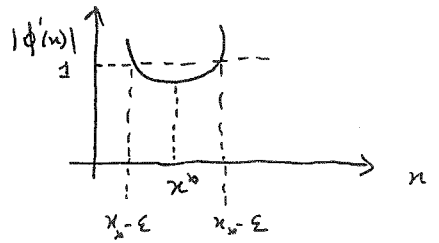
alors pour x_0 "suffisamment" proche de x_*, l'itération du point fixe va converger.

<u>Idée de la démo</u> : $x^{k+1} - x_* \simeq \phi'(x_*)(x_k - x_*)$ (D.L. en x_*)

$$x^{k+1} - x_* = \phi(x_k) - \phi(x_*) = \phi'(\eta)(x_k - x_*)$$

pour $\eta \in [x_k, x_*]$, en utilisant un D.L. avec reste écrite en $x = x_*$.

on suppose x_k proche de x_*, donc si $\phi'(x_*) < 1$, alors $\phi'(\eta) < C$



On a donc :

$$|x^{k+1} - x_*| \leq \max_{x \in [x_k, x_*]} |\phi'(x)| \; |x^k - x_*|$$

or $|\phi'(x_*)| < 1$, alors par continuité : ↗

$$\exists \varepsilon > 0 \text{ tq } \quad |\phi'(x)| < 1, \quad \forall x \in]x^* - \varepsilon, x^* + \varepsilon[.$$

Si x_k est suffisamment proche de x^*, alors

$$[x_k, x_*] \subset]x^* - \varepsilon, x^* + \varepsilon[,$$

et donc

$$\max_{x \in [x_k, x_*]} |\phi'(x)| = C < 1, \text{ soit :}$$

$$|x^{k+1} - x_*| \leq C |x^k - x| \quad \text{avec } 0 < C < 1 .$$

▨.

<u>Définition (l'ordre d'une méthode)</u>

Une méthode est convergente d'ordre q si elle vérifie :

$$|x_{k+1} - x_*| \leq C |x_k - x_*|^q$$

- Si $q = 1$ et $0 < C < 1$: Convergence linéaire.
- Si $q = 2$: Convergence quadratique.
- Si $q = 3$: ——— cubique.

<u>Remarques</u> :

* D'après Eq. (4), la méthode de bissection est convergente au 1er ordre.

* D'après le <u>thm</u> pcdt :

- Si $0 < |\phi'(x_*)| < 1$: une méthode du point fixe a une CV. linéaire.
- Si $\phi'(x_*) = 0$: on verra que la CV est quadratic.

Page (8) du Cours :

Définition (Stabilité)

Une fonction d'itération ϕ est stable si :

$$|\phi'(x_*)| < 1 \qquad (11).$$

Théorème (Convergence)

Si ϕ est stable et consistante, alors pour x_0 "suffisamment" proche de x_*, la méthode du point fixe va converger.

Démo simplifiée

On fait un DL en x_* :

$$\underbrace{\phi(x_n)}_{x_{n+1}} \underset{2}{=} \underbrace{\phi(x_*)}_{x_*} + \phi'(x_*)(x_n - x_*)$$

$$\Rightarrow x_{n+1} - x_* \underset{2}{=} \phi'(x_*)(x_n - x_*)$$

$$\Rightarrow E_{n+1} \underset{2}{=} |\phi'(x_*)| E_n \quad \text{avec } E_n = |x_n - x_*|$$

Si on pose $C \equiv |\phi'(x_*)| < 1$

alors la suite géométrique

$$E_{n+1} \underset{2}{=} C E_n = C^2 E_{n-1} = \dots = C^{n+1} E_0$$

converge vers 0.

Synonymes: Newton-Raphson, meth. des tangentes.

On revient au pbème original :

$$f(x) = 0$$

dont la solution est x_*.

On initie l'algorithme par la condition initiale x_0 et, si elle est suffisamment proche de x_*, alors l'approximation linéaire suivante est valide : (D.L. en x_0)

$$f(x) \overset{?}{=} L(x) = f(x_0) + f'(x_0)(x - x_0). \quad (12)$$

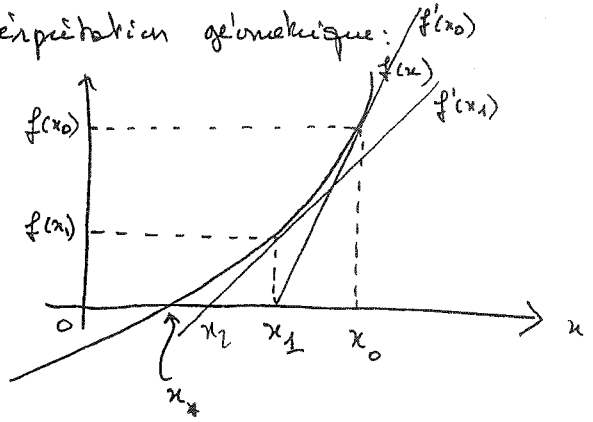
On définit x_1 comme la racine de $L(x) = 0$, soit :

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

De proche en proche, on définit l'algo de Newton :

$$\begin{cases} \bullet\ x_0 \in \mathbb{R} \text{ donné, tq } f'(x_0) \neq 0. \\ \bullet\ x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)}. \end{cases} \quad (13)$$

Interprétation géométrique :



Selon (12) : x_{k+1} est l'intersection entre
- l'axe Ox
- la droite passant par $f(x_k)$ de coeff. directeur $f'(x_k)$.

D'après (13), la méthode de Newton est une méthode du point fixe avec :

$$\phi(x) = x - \frac{f(x)}{f'(x)}. \quad (14)$$

- Vérifions sa consistance :

$$x^* = x^* - \frac{f(x^*)}{f'(x^*)} \iff f(x^*) = 0 \quad \text{si } f'(x^*) \neq 0.$$

On en déduit (provisoirement) que :

1) l'algo de Newton est consistant. D'après le thm 1, il converge si $|\phi'(x_*)| < 1$.

2) si $f'(x_*) = 0$ (x^* : zéro double) $\Rightarrow$ pbème de convergence : il faut utiliser des meth. de Newton modifiées (cf T.D).

- Examinons maintenant la cv de l'algo de Newton dans le cas d'un zéro simple ($f(x^*) = 0$, $f'(x_*) \neq 0$).

Après calcul, on a :

$$\phi'(x) = \frac{f(x) f''(x)}{[f'(x)]^2} \Rightarrow \phi'(x_*) = 0.$$

Donc la meth. de Newton converge, au moins linéairement. Quel est son ordre réel ? On reprend les grandes lignes de la démo du thm 1 : En utilisant un D.L. avec reste exacte en $x = x_*$:

$$x_{k+1} - x_* = \phi(x_k) - \phi(x_*) = \underbrace{\phi'(x_*)}_{=0}(x_k - x_*) + \frac{\phi''(\eta)}{2}(x_k - x_*)^2$$

pour $\eta \in [x_k, x_*]$.

IV-1 Généralités

On revient au pbme test (1), soit :

Trouver x^* tq. $\underbrace{\sin(2x^*) - 1 + x^*}_{f(x^*)} = 0$. (5)

On peut écrire ce pbme comme :

Trouver x^* tq $x^* = \underbrace{1 - \sin(2x^*)}_{\phi_1(x^*)}$ (6)

On peut aussi écrire :

Trouver x^* tq $\sin(2x^*) = 1 - x^*$

$\Leftrightarrow x^* = \underbrace{\frac{1}{2}\arcsin(1-x^*)}_{\phi_2(x^*)}$ (7)

Dans les 2 cas, il semble naturelle de former la suite :

$$x^{k+1} = \phi_i(x^k)$$

Si la suite converge, on aura trouvé x_* tel que :

$$x^* = \phi_i(x^*) \Leftrightarrow f(x_*) = 0.$$

On aura alors transformé la recherche du zéro de f en la recherche du point fixe de $\phi_i(x)$.

Définition : Une méthode qui cherche les racines du problème ($) avec les itérations

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = \phi(x_k) \end{cases} \quad (8)$$

est appelée méthode du point fixe. La fonction $\phi(x)$ est appelée fonction d'itération.

Comme on l'a vu pour l'exple test (1), il y a d'innombrables façons de définir $\phi(x)$.

Le plus simple est de poser :

$$\phi(x) = x - f(x) \quad (9)$$

c'est la méthode de Picard. Plus généralement, on pose :

$$\phi(x) = x + q f(x) \quad (10)$$

avec $q \in \mathbb{R}^*$. On peut même choisir $q = q(x)$ tant que $q(x)$ ne s'annule pas.

Pour une fonction d'itération quelconque $\phi(x)$, il faut s'assurer de la cohérence avec le problème de base (1) :

Définition (cohérence)

On dit que la méth. du point fixe (8) est cohérente avec le pbme (1) si :

$$x = \phi(x) \Leftrightarrow f(x) = 0.$$

exple : Les itérations (6) et (7) sont cohérentes avec le pbme test (1), mais pas $\phi_3(x) = \frac{1}{2}[1 - \sin(2x)]$

IV-2 Analyse de la convergence

Une fois que la cohérence d'une méth. du point fixe est établie, il faut s'assurer de sa convergence : Il faut que l'erreur $e_k = |x_k - x_*|$ diminue à chaque itération, soit :

$$e^{k+1} = |x^{k+1} - x_*| \leq c\,|x^k - x_*|, \text{ avec } \boxed{0 \leq c < 1}$$

$\xi \in [x_n, x_*]$

$$|x_{n+1} - x_*| \leq C |x_n - x_*|^2 \quad (15)$$

la meth. de Newton a une convgce quadratique pour un zéro simple.

IV-4 Résumé

Méthode du point fixe : $x^{n+1} = \phi(x_n)$ pour résoudre $f(x) = 0$ dans l'intervalle I.

1) Etudier sa consistance : $x_* = \phi(x_*) \Leftrightarrow f(x_*) = 0$.

2) Etudier sa stabilité : Pour x_0 "suffisamment" proche de x_*, on a :

- Si $0 < |\phi'(x_*)| < 1$: cv d'ordre 1.
  estimation d'erreur : $e_{n+1} \leq C e_n$, avec $C = \max_{x \in I} |\phi'(x)|$
- Si $\phi'(x_*) = 0$: cv d'ordre 2.
  $e_{n+1} \leq C e_n^2$, avec $C = \frac{1}{2} \max_{x \in I} |\phi''(x)|$.

Cas particuliers :

- méthode de Picard :
$$\phi(x) = x - f(x)$$
- méthode de Newton-corde (cf TD) : $\phi(x) = x - \frac{f(x)}{f'(x_0)}$
- méthode de Newton :

$$\phi(x) = x - \frac{f(x)}{f'(x)}$$

rq : • ordre 2 pour racine simple ($f(x_*) = 0$, $f'(x_*) \neq 0$).
• — 1 — double (cf TD) ($f(x_*) = 0$, $f'(x_*) = 0$, $f''(x_*) \neq 0$).

On va généraliser les algos. vus dans le cas 1D au cas multi-D, par ex :

Trouver le(s) zéro(s) des syst. N-L :
$$\begin{cases} x^2 + y^2 - 1 = 0 & (1) \\ 2x + y - 1 = 0 . & (2) \end{cases}$$

Graphiquement, on voit 2 racines $\underline{x}_1^* = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ et $\underline{x}_2^* = \begin{pmatrix} 4/5 \\ -3/5 \end{pmatrix}$ qui sont l'intersection du cercle (Eq (1)) et de la droite (Eq (2)).

V-1 Rappels de maths

a) Analyse vectorielle / matricielle

Soit $\underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$.

On définit le produit scalaire :
$$(\underline{x}, \underline{y}) = \sum_{i=1}^{n} x_i y_i$$
qui induit une norme vectorielle euclidienne :
$$\|\underline{x}\|_2 = \sqrt{(\underline{x}, \underline{x})}^{1/2}$$

Soit $\underline{\underline{A}} \in \mathbb{R}^{n,n}$ (n lignes, n colonnes) (matrice carré de taille $n \times n$)

On définit la norme matricielle euclidienne :
$$|||\underline{\underline{A}}|||_2 = \sup_{\substack{\underline{x} \in \mathbb{R}^n \\ \underline{x} \neq \underline{0}}} \frac{\|\underline{\underline{A}}\,\underline{x}\|_2}{\|\underline{x}\|_2} \quad (16)$$

Cette norme matricielle est peu pratique à utiliser. A la place, nous allons souvent utiliser la ρ-ke :
$$|||\underline{\underline{A}}|||_2 = \max_{i=1,\dots,n} |\lambda_i| \quad (17)$$

où λ_i sont les val. propres de $\underline{\underline{A}}$.

$$\rho(\underline{\underline{A}}) = \max_{i=1,\dots,n} |\lambda_i|, \qquad (18)$$

Il y a une autre relation très utile : D'après (16), on a

$$\| \underline{\underline{A}}\, \underline{x} \|_2 \leq |||A|||_2 \, \| \underline{x} \|_2 \qquad (19)$$

b) Calcul différentiel

* fonction $f : \mathbb{R} \to \mathbb{R}$

Formule de Taylor : $f(x+h) = f(x) + h \underbrace{f'(x)}_{\text{Dérivée 1ère}} + \dots$

* fonction $f : \mathbb{R}^n \to \mathbb{R}$

$\underline{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ et $f(\underline{x})$ est un réel.

Formule de Taylor : $f(\underline{x} + \underline{h}) = f(\underline{x}) + \underbrace{(\underline{\nabla} f(\underline{x}), \underline{h})}_{\text{différentielle}} + \dots \qquad (20)$

aussi notée $D(f)(\underline{x})$.

avec $$\underline{\nabla} f(\underline{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\underline{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\underline{x}) \end{pmatrix} \qquad (21)$$

Exple : Dans $\mathbb{R}^2$, $\underline{x} = \begin{pmatrix} x \\ y \end{pmatrix}$, $f(\underline{x}) = f(x,y) = x^2 + y^2 - 1$

$$\underline{\nabla} f(\underline{x}) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

* fonction $\underline{f} : \mathbb{R}^n \to \mathbb{R}^p$ (17)

$\underline{x} \in \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ et $\underline{f}(\underline{x}) = \begin{pmatrix} f_1(\underline{x}) \\ \vdots \\ f_p(\underline{x}) \end{pmatrix} \in \mathbb{R}^p$

Formule de Taylor : $\underline{f}(\underline{x} + \underline{h}) = \underline{f}(\underline{x}) + [\underline{\underline{\nabla}} \underline{f}(\underline{x})]\, \underline{h} + \dots \qquad (22)$

La différentielle est cette fois une matrice $\in \mathbb{R}^{p,n}$

$$[\underline{\underline{\nabla}} \underline{f}(\underline{x})] = \begin{pmatrix} \underline{\nabla} f_1(\underline{x}) \\ \underline{\nabla} f_2(\underline{x}) \\ \vdots \\ \underline{\nabla} f_p(\underline{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & & & \\ \vdots & & & \\ \frac{\partial f_p}{\partial x_1} & \dots & \dots & \frac{\partial f_p}{\partial x_n} \end{pmatrix} = \left(\frac{\partial f_i(\underline{x})}{\partial x_j} \right)_{i,j} \qquad (23)$$

Rq : En math, on appelle cette matrice la matrice jacobienne (ou simplement le jacobien), notation équivalentes :

$$[\underline{\underline{\nabla}} \underline{f}(\underline{x})], \quad J_{\underline{f}}(\underline{x}), \quad D\underline{f}(\underline{x}).$$

exemple : dans $\mathbb{R}^2$:

$$\underline{f}(\underline{x}) = \begin{cases} f_1(x,y) = x^2 + y^2 - 1 \\ f_2(x,y) = 2x + y - 1 \end{cases}$$

$$\underline{\underline{\nabla}} \underline{f}(\underline{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x & 2y \\ 2 & 1 \end{pmatrix}$$

V-3 Généralisation des algos. aux systèmes multi-D

a) Positionnement du problème

On cherche le(s) zéro(s) $\underline{x}^*$ du système :

$$\underline{f}(\underline{x}) = 0 \qquad \text{avec } \underline{f} : \mathbb{R}^n \to \mathbb{R}^n .$$

b) Itérations du point fixe

Comme en 1D, l'algo. du point fixe s'écrit :

$$\begin{cases} \bullet\ \underline{x}^0 \in \mathbb{R}^n \text{ donné} \\ \bullet\ \underline{x}^{k+1} = \underline{\Phi}(\underline{x}^k) \end{cases}$$

Il faut alors s'assurer de sa consistance :

$$\underline{x}^* = \underline{\Phi}(\underline{x}^*) \iff \underline{f}(\underline{x}^*) = 0 \qquad (25)$$

et aussi sa stabilité, il faut que :

$$\rho\left(\underline{\underline{\nabla}}\,\underline{\phi}(x^*)\right) \leq 1 . \qquad \text{(en 1D : } |\phi'(x^*)| \leq 1) \qquad (26)$$

Preuve : (simplifiée)

$$\underline{x}^{k+1} - \underline{x}^* = \underline{\phi}(x^k) - \underline{\phi}(x^*)$$

$$\simeq \left[\underline{\underline{\nabla}}\,\underline{\phi}(x^*)\right](x^k - x^*) \quad \text{par D.L. en } x^*$$

ce qui implique :

$$\|\underline{x}^{k+1} - \underline{x}^*\| \leq \underbrace{\left|\left|\left| \underline{\underline{\nabla}}\,\underline{\phi}(x_*) \right|\right|\right|}_{\simeq\, \rho\left(\underline{\underline{\nabla}}\,\underline{\phi}(x_*)\right)} \|x_k - x_*\|$$

donc CV linéaire si $\rho\left(\underline{\underline{\nabla}}\,\underline{\phi}(x^*)\right) \leq 1$.

Concl : Stab. Consist + stab. $\Rightarrow$ point fixe CV linéairement (pour $\underline{x}_0$ suffisamment proche de $\underline{x}_*$ (i.e. $\|x_0 - x_*\|$ pet

c) Algorithme de Newton

Pour établir cet algo., on reprend le principe vu en 1D :

Par un D.L. de Taylor en $\underline{x}_k$, on établit l'approx. linéaire

$$\underline{f}(\underline{x}_{k+1}) \simeq L(\underline{x}) = \underline{f}(x_k) + \left[\underline{\underline{\nabla}}\,\underline{f}(x_k)\right](\underline{x} -$$

On définit $\underline{x}_{k+1}$ comme solution du système linéaire :

$$L(x_{k+1}) = 0$$

soit :

$$\left[\underline{\underline{\nabla}}\,\underline{f}(x_k)\right](\underline{x}_{k+1} - \underline{x}_k) = -\underline{f}(x_k) . \quad (7$$

Ainsi, pour calculer $\underline{x}_{k+1}$, il faut résoudre un système linéaire.

L'algo. de Newton s'écrit de façon pratique :

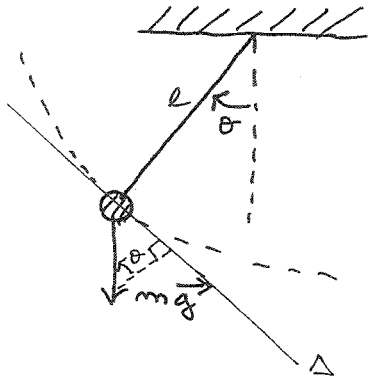
- $\underline{x}_0 \in \mathbb{R}^n$ donné, t.q. $\left[\underline{\underline{\nabla}}\,\underline{f}(x_0)\right]$ est inversible.
- Itérations, $k \geq 0$:
  - Calculer la matrice $\underline{\underline{A}}_k = \underline{\underline{\nabla}}\,\underline{f}(x_k)$
  - Inverser le syst. linéaire $\underline{\underline{A}}_k\, \delta\underline{x} = -\underline{f}(x_k)$
  - Effectuer : $\underline{x}_{k+1} = \underline{x}_k + \delta\underline{x}$.

Thm (Admis) Si $\left[\underline{\underline{\nabla}}\,\underline{f}(x_*)\right]$ est inversible alors, pour x_0 suffisamment proche de x_*, la meth. de Newton admet une CV. quadratique.

Equations Différentielles Ordinaires

1) Motivation : Le pendule oscillant

Mouvement d'un pendule de masse m attaché à une tige de longueur l :



$\theta(t)$: angle du pendule à l'instant t.

Le mouvement est régi par le principe fondamental de la dynamique :

$$m\vec{g} = m\vec{a}$$

On projette cette eqt sur la tangente (Δ) à la trajectoire :

$$mg\sin\theta = m\underbrace{\ddot{\theta} l}_{\text{accél. tangentielle}}$$

L'eqt du mvt est donc l'Eqt. diff. ordinaire (EDO) :

$$\ddot{\theta} = -\omega_*^2 \sin\theta \quad , \text{ avec } \omega_* = \sqrt{\frac{g}{l}} \qquad (1)$$

Pour définir complètement le mvt, on leur adjoint les conditions initiales à $t=0$:

$$\begin{cases} \theta(0) = \theta_0 : \text{position initiale de la masse} \\ \dot{\theta}(0) = \dot{\theta}_0 : \text{vitesse} \end{cases} \qquad (2)$$

Le système (1) - (2) est un problème de Cauchy pour les EDO. Malgré son apparence simple, il est fortement non-linéaire et il n'existe pas de solution analytique à ce problème. Ce chapitre du cours est dédié à l'approximation numérique de ce type de problèmes.

Remarque : Le système (1) - (2) est une EDO du 2nd ordre. Il est possible de la transformer en système d'EDO du premier ordre :

$$\underline{\dot{Y}}(t) = \underline{F}(\underline{Y}, t) \quad , \quad \underline{Y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad , \quad \underline{F} : \mathbb{R}^2 \to \mathbb{R} \qquad (3a)$$

$$\underline{Y}(0) = Y_0 \qquad (3b)$$

en posant $y_1 = \theta$ et en définissant la nouvelle inconnue $y_2 = \dot{\theta}$. De cette façon, l'EDO (1) devient :

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -\omega_*^2 \sin y_1 \end{cases}$$

qui est de la forme (3a) avec :

$$\underline{F}(\underline{Y}) = \begin{cases} f_1(y_1, y_2) = y_2 \\ f_2(y_1, y_2) = -\omega_*^2 \sin y_1 . \end{cases}$$

De même, la cond. initiale (2) s'écrit :

$$\underline{Y}(0) = \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} \theta_0 \\ \dot{\theta}_0 \end{pmatrix} .$$

II) Généralités

(Notation : $\dot{y}(t) = \frac{dy}{dt}(t)$.)

On s'intéresse aux équations diff. ordinaires (EDO) du 1er ordre, de la forme :

$$\dot{y}(t) = f(t, y(t)), \qquad (1)$$

où $f : \mathbb{R}^2 \to \mathbb{R}$ est une fonction donnée.

~~Dans le chap. VI, on verra comment transformer les E.D.O d'ordre plus élevé en systèmes d'E.D.O du premier ordre. → Exemple du pendule.~~

Exemple 1 : On considère $f(y,t) = 3y - 3t$, l'E.D.O s'écrit donc :

$$\dot{y}(t) = 3y(t) - 3t \qquad (2)$$

dont les solutions sont de la forme $y_\alpha = (\alpha - \frac{1}{3})e^{3t} + t + \frac{1}{3}$

Cette E.D.O. est linéaire, car on peut superposer 2 solutions y_{α_1} et y_{α_2}.

Exemple 2 $f(y,t) = 1 - y^2$, donc

$$\dot{y}(t) = 1 - y^2 \qquad (3)$$

C'est un exemple d'E.D.O non-linéaire. De plus cette E.D.O est dite autonome ($f(y,t)$ ne dépend que de y)

Pour déterminer complètement la solution, on lui adjoint une condition initiale, p. ex :

$$y(t=0) = y_0. \qquad (4).$$

Le système (1), (4) est appelé problème de Cauchy, soit :

$$(5) \quad \begin{cases} \dot{y}(t) = f(t, y(t)) & \text{pour } t \in [0,T] \\ y(0) = y_0. \end{cases}$$

III) Schémas numériques de base

1) Construction

Il existe de nombreuses méthode de construction. On va donner ici des méthodes basées sur les formules de quadrature numérique.

On effectue préalablement une subdivision du domaine de calcul $\Omega = [0, T]$, de la forme

$$t_n = n\,h, \quad n = 0, \dots, N,$$

avec $h = \frac{T}{N}$ le pas de temps.

$t_0 \quad t_1 \quad t_2 \quad - - - - - \quad t_N.$

$t=0 \qquad\qquad\qquad\qquad t=T$

On intègre l'E.D.O (1) entre t_n et t_{n+1} :

$$\int_{t_n}^{t_{n+1}} \dot{y}\, dt = \int_{t_n}^{t_{n+1}} f(y(t), t)\, dt,$$

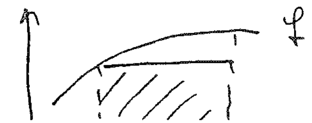
soit

$$y^{n+1} - y^n = \int_{t_n}^{t_{n+1}} f(y(t), t)\, dt. \qquad (6)$$

(notation : $y^n = y(t_n)$).

Il faut maintenant évaluer le RHS de (6) par quadrature numérique.

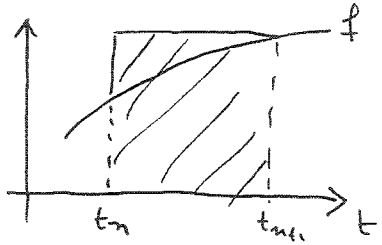
La méthode la plus simple consiste à approcher l'intégrale en utilisant la valeur au point t_n :



$$\int_{t_n}^{t_{n+1}} f\, dt \simeq h\, f(t_n, y^n)$$

$$\frac{y^{n+1} - y^n}{h} = f(t_n, y^n). \qquad (7)$$

On peut aussi utiliser la valeur au point final t_{n+1} :

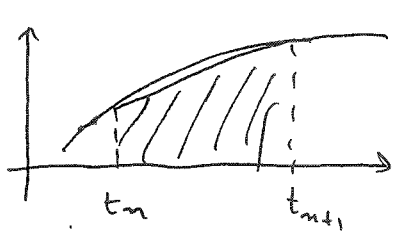


$$\int_{t_n}^{t_{n+1}} f\,dt \simeq h\, f(t_{n+1}, y^{n+1})$$

On obtient alors la méthode d'Euler rétrograde ER1 :

$$\frac{y^{n+1} - y^n}{h} = f(t_{n+1}, y^{n+1}) \qquad (8).$$

On peut aussi utiliser la méthode du trapèze :



$$\int_{t_n}^{t_{n+1}} f\,dt \simeq \frac{h}{2}\left[f(t_{n+1}, y^{n+1}) + f(t_n, y^n\right.$$

On obtient alors la méthode de Crank-Nicholson :

$$\frac{y^{n+1} - y^n}{h} = \frac{1}{2}\left[f(t_{n+1}, y^{n+1}) + f(t_n, y^n)\right]($$

Des schémas plus sophistiqués peuvent être construits avec les quadratures numériques vues dans les cours précédents. On les verra en temps voulu en TD.

A chacun de ces schémas, on leur adjoint la C.I (4) :

$$y^0 = y_0 \quad . \qquad (10)$$

III-2| Utilisation

Concrètement, tous ces schémas sont utilisés par le programme d'avancement en temps à partir de la C.I :

$$y^0 = y(0).$$

Par exemple, l'application du schéma EP1 pour $n=1$ donne :

$$y^1 = y_0 + h\, f(t_0, y_0)$$

qui est une approximation de $y(t_1)$, puis :

$$y^2 = y_1 + h\, f(t_1, y_1)$$

etc ... jusqu'à $t_N = T$. Le schéma EP1 est qualifié d'explicite car il n'y a pas d'eqt à résoudre pour calculer les y^{n+1}.

Examinons l'utilisation du schéma ER1. Au temps $t=t_1$ on a :

$$y^1 = y^0 + h\, f(t_1, y^1),$$

et il n'y a pas de formule de récurrence simple qui donne facilement y_1 lorsque f n'est pas linéaire. Par exemple pour l'ex 2 (Eq. (3)), alors on obtient :

$$y^1 + h(y^1)^2 = y_0 + h.$$

On est amené à résoudre une eqt non linéaire pour calculer y^1. Pour cette raison, le schéma ER1 est qualifié d'implicite.

Rq : Dans le cas où f est linéaire, i.e. l'exemple (1), alors $f = 3y - 3t$ et le schéma ER1 donne :

$$y^1 = y^0 + h(3y^1 - 3t_1)$$

soit $(1-3h)\,y^1 = y^0 - 3t_1$.

III Analyse des schémas

Le schéma explicite EP1 semble préférable à ER1 car il nécessite moins d'efforts pour avancer la solution en temps.

Néanmoins, il y a un prix à payer à cette facilité de calcul : la stabilité du schéma.

III-1 Stabilité

Déf: C'est la propriété qui permet d'obtenir une solution numérique dont l'erreur ne s'amplifie pas lorsque $t \to +\infty$.

Considérons par exemple le problème modèle suivant pour $\lambda < 0$:

$$\begin{cases} \dot{y} = \lambda y, & t \in [0, +\infty[. \\ y(0) = y_0. \end{cases} \qquad (11)$$

La solution exacte est

$$y(t) = y_0 e^{\lambda t},$$

qui a la ppté de s'amortir au cours du temps :

$$\lim_{t \to \infty} y(t) = 0$$

puisque $\lambda < 0$.

Lorsqu'on applique EP1 à (13), on obtient :

$$\begin{aligned} y^n &= (1 + \lambda h) y^{n-1} \\ &= (1 + \lambda h)^n y_0 \\ &= \sigma^n y_0 \end{aligned}$$

avec $\sigma = 1 + \lambda h$. Le réel σ est appelé facteur d'ampli

Pour que la solution numérique y^n reste bornée lorsque $n \to \infty$, on doit avoir nécessairement :

$$|\sigma| < 1,$$

ce qui conduit à

$$-1 < 1 + \lambda h < 1$$

$$0 < h < -2/\lambda = \frac{2}{|\lambda|}.$$

Pour obtenir une solution bornée, le schéma EP1 doit être utilisé avec un pas de temps h vérifiant la condition

$$h \leq \frac{2}{|\lambda|}. \qquad (12).$$

On dit alors que le schéma est conditionnellement stable

Que se passe-t'il pour le schéma ER1 ? Lorsqu'il est appliqué au problème test (11), on obtient :

$$y^{n+1} = y^n + \lambda h y^{n+1}$$

soit

$$y^n = \sigma^n y_0 \quad \text{avec} \quad \sigma = \frac{1}{1 - \lambda h}.$$

La condition de stabilité s'écrit :

$$|\sigma| \leq 1 \iff |1 - \lambda h| \geq 1, \qquad (13)$$

qui est bien vérifiée pour tout $h > 0$. Le schéma ER1 est stable sans condition.

III-2 - Stabilité dans le cas général

Dans le cas d'un rhs quelconque $f(t,y)$, une analyse de stabilité analogue peut être réalisée, sous quelques restrictions.

Rappel : Formule de Taylor pour fonctions à 2 variables (forme symbolique)

$$f(x+h, y+k) = \sum_{n\geq 0} \frac{1}{n!} \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^n f(x,y)$$

$$= f(x,y) + \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) f(x,y) + \frac{1}{2!}\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^2 f(x$$

$$+ \cdots$$

Première restriction : l'analyse de stabilité s'effectue pour un E.D.O linéaire.

Il faut donc linéariser $f(y,t)$ autour du point (y_n, t_n). En ne gardant que les termes linéaires du D.L. de f autour de (y_n, t_n), on obtient :

$$f(t,y) \simeq f(y_n,t_n) + (t-t_n)\frac{\partial f}{\partial t}(y_n,t_n) + (y-y_n)\frac{\partial f}{\partial y}(y_n,$$

$$= \lambda y + \alpha + \beta t$$

avec :

$$\lambda = \frac{\partial f}{\partial y}(y_n,t_n)$$

$$\alpha = f(y_n,t_n) - t_n\frac{\partial f}{\partial t}(y_n,t_n) - y_n\frac{\partial f}{\partial y}(y_n,t_n$$

$$\beta = \frac{\partial f}{\partial t}(y_n,t_n)$$

L'E.D.O linéarisé est donc :

$$\dot{y} = \lambda y + \alpha + \beta t .$$

La solution s'écrit :

$$y(t) = \underline{Cte}\, e^{\lambda t} - \left(\frac{\alpha}{\lambda} + \frac{\beta}{\lambda^2} + \frac{\beta}{\lambda} t\right),$$

la partie la plus dangereuse de l'instabilité est la partie exponentielle, qui est solution de l'problème test :

$$\dot{y} = \lambda y .$$

2ème restriction : l'analyse de stabilité concerne la partie exponentielle de la solution : On ne retient donc que la partie autonome de l'E.D.O linéarisée.

Donc, au temps (t_n, y_n), la stabilité du schéma EP1 est :

$$h \leq \frac{2}{-\lambda} \quad , \quad \lambda = \frac{\partial f}{\partial y}(y_n,t_n) .$$

Lors d'une intégration en temps dans $t \in [0,T]$, il faut donc que :

$$h \leq \frac{2}{-\frac{\partial f}{\partial y}(y_n,t_n)} \quad , \quad \forall n = 0, \cdots, N.$$

3- Stabilité - Résumé

Considérons le pbme de Cauchy

$$\begin{cases} \dot{y} = f(t, y(t)) \\ y(0) = y_0 \end{cases} \qquad (15)$$

à résoudre par schéma numérique. On peut définir 3 classes de schémas pour ce problème :

- Schémas stables sans conditions :

La sol. numérique reste bornée (ie ne croît pas vers $\pm\infty$) quelle que soit la valeur du pas de temps h.
Un schéma implicite (ex: EP1) fait généralement partie de cette classe. Le prix à payer est de résoudre une equation (qui peut être non-linéaire) pour calculer y^{n+1}.

- Schémas instables : La solution numérique explose quelle que soit la valeur de h. $\Rightarrow$ Schéma inutilisable.

- Schémas conditionnellement stables:

Avec certains choix du paramètre h, la solution reste bornée. Les schémas explicites (ex: EP1) appartiennent généralement à cette catégorie.

Pour déterminer dans quelle classe appartient un schéma, il faut réaliser une analyse de stabilité sur le problème modèle :

$$\dot{y} = \lambda y,$$

qui est + simple que l'EDO originale (1S), tout en gardant ses caractéristiques principales (p.ex: en effectuant une linéarisation de (1S)).

L'analyse de stabilité sur le problème modèle donne une condition nécessaire de stabilité, qui évite les instabilités de forme exponentielles (les plus dangereuses).

Introduction

Outre la propriété de stabilité du schéma, on cherche à connaître sa propriété de précision (ou convergence), c.a.d savoir si à un instant $t_n = nh$ la solution numérique y^n vérifie :

$$|y^n - y(t_n)| \longrightarrow 0 \quad \text{lorsque } h \to 0.$$

On veut aussi déterminer l'ordre de précision q du schéma, telle que

$$|y^n - y(t_n)| = O(h^q).$$

Il y a essentiellement 2 méthodes basées pour déterminer l'ordre de précision q :

* méthode basé sur les D.L. de Taylor.
* ——— sur le problème modèle (11).

On va les expliciter pour le schéma EP1.

a) Précision basée sur le problème modèle

La solution exacte du problème (11) est :

$$y(t_n) = e^{\lambda t_n} y_0 = e^{\lambda n h} y_0 = (e^{\lambda h})^n y_0.$$

Le D.L. de $e^{\lambda h}$ est :

$$e^{\lambda h} = 1 + \lambda h + \frac{(\lambda h)^2}{2} + \frac{(\lambda h)^3}{3!} + \cdots \quad (16)$$

Par contre, la solution numérique donnée par EP1 est :

$$y^n = \sigma^n y^0,$$

avec le facteur d'amplification :

$$\sigma = 1 + \lambda h.$$

On observe que σ représente le D.L. du facteur d'amplification "exacte" $e^{\lambda h}$ jusqu'à l'ordre 1 inclus

à l'ordre q si son facteur d'amplification représente le D.L. de $e^{\lambda h}$ jusqu'à l'ordre $\frac{(\lambda h)^q}{q!}$ inclus.

On a alors la propriété fondamentale :

Théorème : (Admis)

Consistance à l'ordre q + stabilité $\Rightarrow$ convergence à l'ordre q :

$$|y^n - y(t_n)| = O(h^q).$$

B) **Consistance basée sur les D.L. de Taylor.**

Le schéma EP1 s'écrit :

$$\frac{y^{n+1} - y^n}{h} = f(y_n, t_n). \quad (17)$$

On effectue le D.L. de y^{n+1} au temps t_n :

$$y^{n+1} = y^n + h\dot{y}^n + \frac{h^2}{2}\ddot{y}^n + O(h^3)$$

que l'on reporte dans le schéma (17), pour obtenir :

$$\dot{y}^n = \underbrace{f(y_n, t_n)}_{\text{E.D.O au temps } t^n} + \underbrace{\frac{h}{2}\ddot{y}^n + O(h)}_{\hookrightarrow \tau_n : \text{reste ou erreur de Troncature}}$$

Pour ce schéma, l'erreur de Troncature est :

$$\tau_n = \frac{h}{2}\ddot{y}^n = O(h)$$

Le schéma EP1 est donc consistant à l'ordre 1.

---

* Pour le schéma EP1, en résumé :
  - il est stable pour $h < 2/|\lambda|$
  - Il est consistant à l'ordre 1.

Il est donc convergent lorsqu'on fixe $h < 2/|\lambda|$.

* Pour le schéma EP2

$$\frac{y^{n+1} - y^n}{h} = f(y^{n+1}, t_{n+1}).$$

Son facteur d'amplification est

$$\sigma = \frac{1}{1 - \lambda h} = 1 + \lambda h + (\lambda h)^2 + \dots$$

Il représente donc le D.L. de $e^{\lambda h}$ jusqu'à l'ordre 1 :
il est consistant à l'ordre 1.
$\Rightarrow$ Il est convergent $\forall h > 0$.

IV-1 Motivations

Dans la pratique, on utilise des schémas plus sophistiqués que ER1 et EP1, qui ont l'avantage d'être plus précis (on vient au moins $q \geq 2$) et possèdes une stabilité équivalente ou parfois supérieure.

Nous allons ici présenter les méthodes les plus populaires : R-K.

Considérons la forme intégrale (6) de l'EDO :

$$y^{n+1} - y^n = \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt .$$

Pour augmenter la précision, une stratégie est de considérer des formules de quadrature + précises, p. ex. la formule du trapèze donne :

$$y^{n+1} - y^n = \frac{h}{2}\left[f(t_n, y^n) + f(t_{n+1}, y^{n+1}) \right] \quad (18).$$

C'est le schéma C-N, en $O(h^2)$, stable sans condition (cf. TD).

Malheureusement, il est implicite.

Bricolons-le pour rendre explicite : On peut essayer de "prédire" y^{n+1} par y^* en appliquant ER1 :

$$y^* = y^n + h f(t_n, y_n) \quad (19)$$

~~et de le corriger~~ "La prédiction $y^* \simeq y(t_{n+1})$ est au 1er ordre.

On peut maintenant essayer de la corriger en la reportant dans C-N, soit :

$$y^{n+1} = y^n + \frac{h}{2}\left[f(t_n, y^n) + f(t_{n+1}, y^*) \right] . \quad (20).$$

Le schéma explicite à 2 étapes (19)-(20) est connu sous le nom de schéma "prédicteur-correcteur" (ou Sch. de Heun), et on montrera qu'il est en $O(h^2)$.

Le schéma (19)-(20) peut s'écrire de façon équivalente (15) sous la forme :

$$k_1 = h f(t_n, y^n) \quad (21)$$

$$k_2 = h f(t_n + h, y^n + k_1) \quad (22)$$

$$y_{n+1} = y_n + \frac{1}{2}\left[k_1 + k_2\right] . \quad (23)$$

Si on le compare au schéma EP1, qui s'écrit :

$$y_{n+1} = y_n + k_1$$

le schéma prédicteur-correcteur augmente la précision de EP1 en évaluant f à un point supplémentaire dans $[t_n, t_{n+1}]$.

Cette construction est typique des méthodes de Runge-Kutta, où de judicieuses évaluation de f en q points de $[t_n, t_{n+1}]$ donne un schéma d'ordre q.

IV-2 Les schémas R-K d'ordre 2

La forme générale des schémas R-K explicites à 2 pas est :

$$k_1 = h f(y_n, t_n) \quad (24)$$

$$k_2 = h f(t_n + \alpha h, y_n + \beta k_1) \quad (25)$$

$$y^{n+1} = y^n + \gamma_1 k_1 + \gamma_2 k_2 , \quad (26)$$

où $\gamma_1, \gamma_2, \alpha$ et β sont des coefficients à déterminer. (On voit bien que le schéma pred/cor. (21)-(23) est un cas particulier).

Ces coefficients sont déterminés de façon que le schéma soit le + précis possible.

pour un schéma (26), nous allons comparer le D.L. de ce schéma à celui de la solution exacte au temps $t = t_n$.

La solution exacte donne :

$$y_{n+1} = y_n + h\,\dot{y}_n + \frac{h^2}{2}\,\ddot{y}_n + \dots \quad (27)$$

De plus,

$$\dot{y} = f(y,t) \Rightarrow \dot{y}_n = f(y_n, t_n) = f_n \quad \text{(notation)}$$

et :

$$\ddot{y} = \frac{d}{dt}\dot{y} = \frac{d}{dt} f(t, y(t)) = \frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y} \Rightarrow \ddot{y}_n = \frac{\partial}{\partial t} f_n + f_n \frac{\partial}{\partial y}$$

Le D.L. de la sol. exacte s'écrit donc :

$$y_{n+1} = y_n + h f_n + \frac{h^2}{2}\left(\frac{\partial}{\partial t} f_n + f_n \frac{\partial}{\partial y} f_n\right) + \dots \quad (28$$

Effectuons maintenant le D.L. du schéma de R-K (24)-(26). Le D.L. de k_2 en (t_n, y_n) s'écrit :

$$k_2 = h\left[f_n + \alpha h \frac{\partial}{\partial t} f_n + \beta k_1 \frac{\partial}{\partial y} f_n \right]$$

$$= h f_n + \alpha h^2 \frac{\partial}{\partial t} f_n + \beta h^2 f_n \frac{\partial}{\partial y} f_n,$$

On en déduit que le D.L. du schéma (26) s'écrit :

$$y_{n+1} = y_n + h(\gamma_1 f_n + \gamma_2 f_n) + h^2 \gamma_2 \left(\beta f_n \frac{\partial}{\partial y} f_n + \alpha \frac{\partial}{\partial t} f\right.$$
$$\left. + \dots \right. \quad (29).$$

En comparant (28) et (29), une condition nécessaire pour que le schéma R.K. soit en $O(h^2)$ est que :

$$\begin{cases} \gamma_1 + \gamma_2 = 1 \\ \gamma_2 \beta = \frac{1}{2} \\ \gamma_2 \alpha = \frac{1}{2} \end{cases}$$

Il y a 4 coeff. pour 3 eq^{ts} (non linéaires). On utilise, p. ex., α comme paramètre pour écrire :

$$\gamma_2 = \frac{1}{2\alpha}, \quad \beta = \alpha, \quad \gamma_1 = 1 - \frac{1}{2\alpha}.$$

Finalement, le schéma RK2-α (24)-(26) s'écrit :

$$k_1 = h f(y_n, t_n) \quad (29)$$

$$k_2 = h f(t_n + \alpha h, y_n + \alpha k_1) \quad (30)$$

$$y^{n+1} = y^n + \left(1 - \frac{1}{2\alpha}\right) k_1 + \frac{1}{2\alpha} k_2. \quad (31)$$

Dans la pratique, il y a 2 formes couramment utilisées :

* $\alpha = 1$ correspond au schéma de Heun (19)-
* $\alpha = \frac{1}{2}$ correspond au schéma "pred./correct" :

$$y^*_{n+\frac{1}{2}} = y_n + \frac{h}{2} f(y_n, t_n) \quad (32)$$

$$y_{n+1} = y_n + h f(y^*_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}) \quad (33).$$

Effectuons l'analyse du schéma RK2-α pour le problème modèle $\dot{y} = \lambda y$: on obtient

$$k_1 = \lambda h y^n$$

$$k_2 = \dots = \lambda h y^n + \alpha \lambda^2 h^2 y^n$$

$$y^{n+1} = y^n + \left(1 - \frac{1}{2\alpha}\right)\lambda h y^n + \frac{1}{2\alpha}(\lambda h y^n + \alpha \lambda^2 h^2 y^n$$

$$= y^n \underbrace{\left(1 + \lambda h + \frac{\lambda^2 h^2}{2}\right)}_{\sigma}$$

Le facteur d'amplification représente les 2 premiers termes du D.L de $e^{\lambda h}$, ce qui confirme la précision en $O(h^2)$ de RK2-α

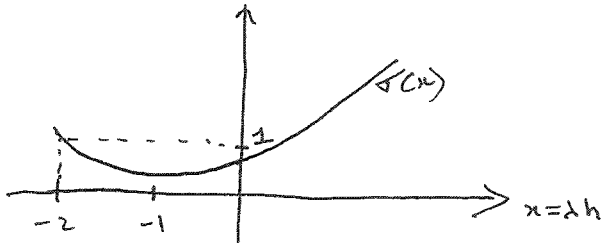
La stabilité requiert que

$$-1 \leq \sigma(\lambda h) \leq 1 .$$

Pour déterminer cette condition, le plus simple est de tracer le graphe de

$$x \longmapsto \sigma(x) = 1 + x + \frac{x^2}{2}$$

et d'examiner les valeurs de $x < 0$ pour lesquelles $|\sigma(x)| < 1$.



La stabilité requiert: $-2 \leq x \leq 0$

$$-2 \leq \lambda h \leq 0$$

$$\Rightarrow \boxed{h \leq -\frac{2}{\lambda}} .$$

c'est la même condition de stabilité que EP1.

Concl: les schémas RK2-α sont:

- explicites
- précis en $\mathcal{O}(h^2)$
- Aussi stables que EP1.

Rq: Par un procédé similaire (augmenter le nombre d'étapes K1, K2, K3, etc...) il est possible de construire des schémas R-K explicites de + en + précis. Le grand intérêt de ces schémas R-K est que leur stabilité s'améliore lorsque la précision $\nearrow$. En TD on étudiera le schéma RK4

V) Systèmes d'E.D.O.

On peut appliquer les schémas numériques scalaires des sections précédentes aux systèmes d'E.D.O, de la forme :

$$\dot{\underline{X}}(t) = N(t, \underline{X}(t)), \qquad \underline{X}(0) = \underline{X}_0 \tag{34}$$

avec $\underline{X}(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix} \in \mathbb{R}^n$.

Pour la clarté de ce chapitre, on va considérer un système d'EDO linéaire à coefficients constants, tels que N a la forme:

$$N(t, X) = \underline{\underline{A}}\,\underline{X} + \underline{F}(t), \tag{35}$$

où $\underline{\underline{A}}$ est une matrice de $\mathbb{R}^{n,n}$ à coefficients constants.

Le cas où N a une forme générale (ie, non-linéaire) sera effectué dans l'exo5 du TD3.

L'application des schémas numériques de la partie II et IV est evidente, par exemple le schéma EP1 s'écrit:

$$\frac{1}{h}(\underline{X}^{n+1} - \underline{X}^n) = \underline{\underline{A}}\,\underline{X}^n + \underline{F}(t_n), \tag{36a}$$

qui s'écrit sous forme itérative:

$$\underline{X}^{n+1} = (\underline{\underline{I}} + h\,\underline{\underline{A}})\,\underline{X}^n + h\,\underline{F}(t_n) \tag{36b}$$

L'Analyse de ce schéma reprend les grandes lignes de ce qui a été fait au chap. III pour les E.D.O scalaires. En particulier, l'analyse de stabilité pour le système (34)-(35)

$$\dot{X} = A X \qquad (37)$$

Afin de pouvoir appliquer les résultats de la Sect. III-2 il faut diagonaliser la matrice $\underline{\underline{A}}$, c-a-d :

$$\underline{\underline{A}} = \underline{\underline{S}}\ \underline{\underline{D}}\ \underline{\underline{S}}^{-1} \qquad (38)$$

où $\underline{\underline{D}}$ est la matrice diagonale des valeurs propres de $\underline{\underline{A}}$, et $\underline{\underline{S}}$ la matrice de passage (ie, la $i^{ème}$ colonne de $\underline{\underline{S}}$ contient le vecteur propre associé à la $i^{ème}$ val. propre de $\underline{\underline{A}}$).

En effectuant le changement de base $\underline{Y} = \underline{\underline{S}}^{-1} \underline{X}$, le syst. (13) s'écrit :

$$\underline{\underline{S}}^{-1} \dot{\underline{X}} = \underline{\underline{S}}^{-1} \underline{\underline{A}}\, \underline{\underline{S}}\, \underline{\underline{S}}^{-1} \underline{X} + \underline{\underline{S}}^{-1} \underline{F},$$

Pour la variable Y, ce système prend la forme diagonale

$$\dot{\underline{Y}} = \underline{\underline{D}}\, \underline{Y} + \underline{G}, \qquad \underline{G} = \underline{\underline{S}}^{-1} \underline{F}$$

qui s'écrit composante par composante :

Pour $i=1,\dots,n$: $\dot{y}_i = \lambda_i y_i + g_i(t), \qquad (39)$

où λ_i est une v.p. de $\underline{\underline{A}}$. L'avantage de ce changement de variable est donc bien au système d'EDO totalement découplé (39).

On peut appliquer le schéma EP1 à cette équation, i

$$y_i^{n+1} = (1 + \lambda_i h)\, y_i^n + h\, g_i(t). \qquad (40)$$

D'après l'analyse de stabilité de la section III-2, ce schéma est stable si :

EDO 21

$$h \leq 2 / -\lambda_i, \quad i = 1, \dots, n. \quad (4$$

Ainsi, pour assurer la stabilité du schéma EP1 (3 il faut que toutes les conditions de stabilité (41) soient vérifiée, i.e :

$$h \leq \frac{2}{\max_i \{-\lambda_i\}}.$$

Résumé : Pour effectuer l'analyse de stabilité d'un schéma numérique au système d'EDO :

$$\dot{X} = A X + F(t),$$

il faut :

1) Calculer les v.p. λ_i de $\underline{\underline{A}}$

2) Effectuer l'étude de stabilité du schéma numérique appliqué à l'EDO scalaire :

$$\dot{y} = \lambda_i y, \qquad i = 1, \dots, n$$

3) La cond. de stabilité est alors donnée par la plus restrictive des cond. de stabilité pour chaque EDO scalaire.

Remarques

1) Même si $\underline{\underline{A}}$ est à coefficients réels, ses v.p. ne sont pas nécessairement réelles (il faudrait en outre que $\underline{\underline{A}}$ soit symétrique). On peut énoncer que :

- Si $\lambda_i \leq 0$, on utilise l'analyse de stab. du chap III-1.
- Si $\lambda_i > 0$, on ne peut rien dire sur la stabilité du schéma (cf chap III-2).
- Si $\lambda_i \in \mathbb{C}$, avec une partie réelle négative, la "mode d'emploi" de l'analyse de stabilité est donnée dans l'exo 5. du TD 3.

2) Le cas général $\dot{X} = N(t, X)$ où N est non-linéaire se ramène au cas (34)-(35) après linéarisation de $N(t, X)$ (d'une façon analogue à ce qui a été fait dans la sect. III-2 et l'exo 2 du TD 3 pour une EDO scalaire).

Un exemple est donné dans l'exo 5.

Appendix A: Methods for Ordinary Differential Equations (ODEs)

We consider the initial value problem:

$$\left.\begin{array}{ll} (1a) & \dot{y} = f(t, y(t)) \quad , \quad t \in [0,T] = \mathcal{V} \\ (1b) & y(t=0) = y_0 \end{array}\right\}$$

often called a Cauchy problem.

I) Building of basic numerical schemes

the basic methods used in CFD (Euler scheme, Crank-Nicolson method) will be built here by means of numerical quadrature.

First, the time domain $\mathcal{V} = [0,T]$ is subdivided as $\{t_n = n\,\Delta t, \; n = 0, \dots, N\}$

where $\Delta t = \frac{T}{N}$ is the time-step.

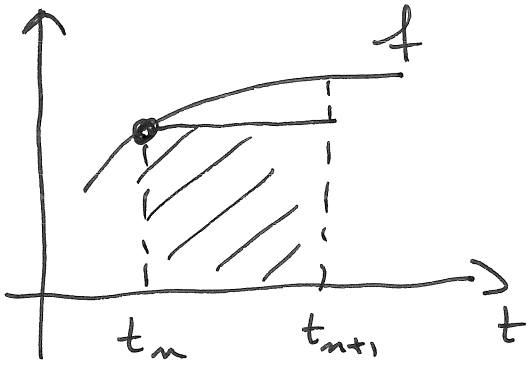
The EDO (1a) is first integrated in $[t_n, t_{n+1}]$ to yield:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt \quad (2)$$

No discrete approximation has yet been performed. Now, the RHS of (2) will be approximated with numerical quadrature based on Lagrange polynomials.

I-1| Forward Euler method (FE1)

The simplest method stems from the approximation of (2) by using values at time t_n:



$$\int_{t_n}^{t_{n+1}} f\,dt \overset{\Delta t}{=} f(t_n, y^n)$$

where we used the compact notation:

$$\underbrace{y^n}_{\text{approx. solution}} \equiv \underbrace{y(t_n)}_{\text{exact solution}}$$

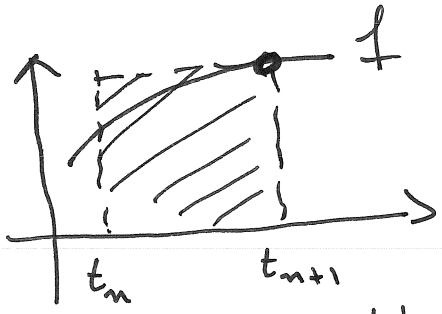
the forward Euler scheme reads then:

$$\frac{y^{n+1} - y^n}{\Delta t} = f(t_n, y^n) \quad (3)$$

Its acronym is FE1, because we shall see later that the method is first order in time.

I-2| Backward Euler Method (BE1)

If the RHS of (2) is approximated with end point t_{n+1}:



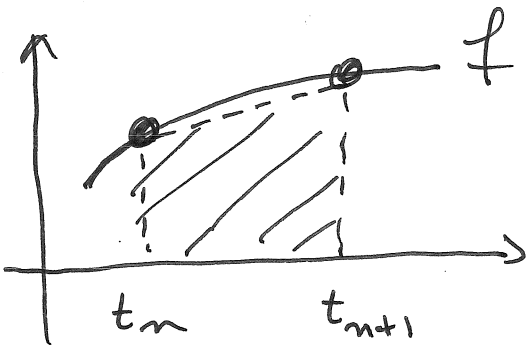
$$\int_{t_n}^{t_{n+1}} f\,dt \overset{\Delta t}{=} f(t_{n+1}, y^{n+1})$$

We obtain the backward Euler method (BE1):

$$\frac{y^{n+1} - y^n}{\Delta t} = f(t_{n+1}, y^{n+1}) \quad (4)$$

I-3| Trapezoidal method

The RHS of (2) can be approximated by the Trapezoidal method (use of Lagrange polynomial of order 2)



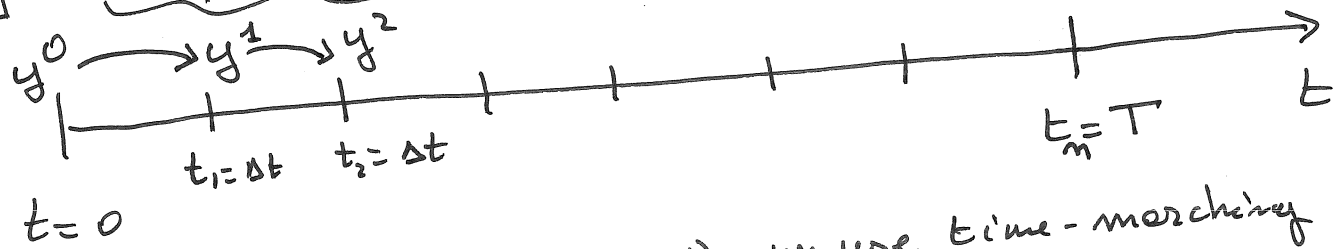
$$\int_{t_n}^{t_{n+1}} f\,dt \simeq \frac{\Delta t}{2}\left[f(t_{n+1}, y^{n+1}) + f(t_n, y^n) \right]$$

The resulting trapezoidal method:

$$\frac{y^{n+1} - y^n}{\Delta t} = \frac{1}{2}\left[f(t_{n+1}, y^{n+1}) + f(t_n, y^n) \right] \quad (5)$$

is often called the Crank-Nicolson method (CN) when used for time integration of PDEs.

II| Use of time integration scheme. Implicit or explicit method



For ~~integrating~~ solving ODE (1), we use time-marching by starting with the initial condition at $t_0 = 0$:

$$y^0 = y(0),$$

Then, y^1 at $t = t_1$ is calculated with FE1, for example, written in its iterative form:

$$\underbrace{y^1}_{\text{Unknown at } t = t_1} = \underbrace{y^0 + \Delta t\, f(t_0, y^0)}_{\text{known at } t = t_1} \quad (6)$$

This expression is ready to be programed in any language, such as FORTRAN or MATLAB. Since there is no equation to solve to obtain y^1 from values at previous time-level, the FE1 scheme is called explicit. Furthermore, Eq. (6) involves the solution at only time levels t_0 and t_1, thus FE1 is called a two-level scheme.

By contrast, the BE1 scheme leads to the iterative equation:

$$y^1 - \Delta t\, f(t_1, y^1) = y^0, \qquad (7)$$

the solution of an equation is needed to obtain y^1, resulting in more computional work. The BE1 is called an implicit scheme, and is also a two-level scheme.

The CN (trapezoidal) schem is an implicit two-level scheme.

III) Truncation error of time-integrate scheme.

The local truncation error of a scheme (LTE) is the difference between the numerical scheme and the ODE at time t_n. It can be summarized as:

$$\underset{\text{at } t_n}{\text{scheme}} = \underset{\text{at } t_n}{\text{ODE}} + \tau^n$$

where τ^n is the LTE at time t_n. It is usually performed by using Taylor expansion of the solution, i.e.:

$$y^{n+1} = y^n + \Delta t\, \dot{y}^n + \frac{\Delta t^2}{2} \ddot{y}^n + O(\Delta t^3) \qquad (8)$$

Let us calculate the LTE of backward-Euler scheme by introducing this Taylor expansion in (3):

$$\underbrace{\frac{y^{n+1}-y^n}{\Delta t} - f(t_n, y_n)}_{\text{FBE1 scheme at } t_n} = \frac{y^n + \Delta t\, \dot{y}^n + \frac{\Delta t^2}{2}\ddot{y}^n + O(\Delta t^3) - y^n}{\Delta t} - f(t_n, y_n)$$

$$= \underbrace{\dot{y}^n - f(t_n, y_n)}_{\text{ODE at } t_n} + \tau^n$$

where the LTE is $\tau^n = \frac{\Delta t}{2}\ddot{y}^n + O(\Delta t^2)$.

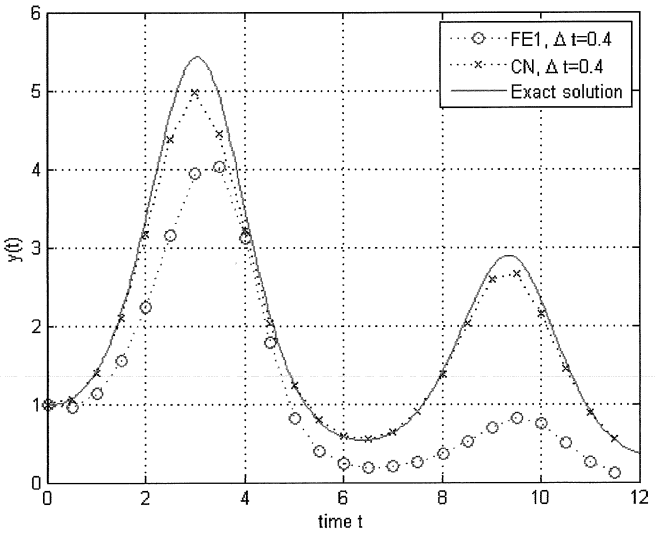
Hence, the FBE1 scheme is first-order accurate.

In a similar fashion, we can show the the LTE of BE1 is:

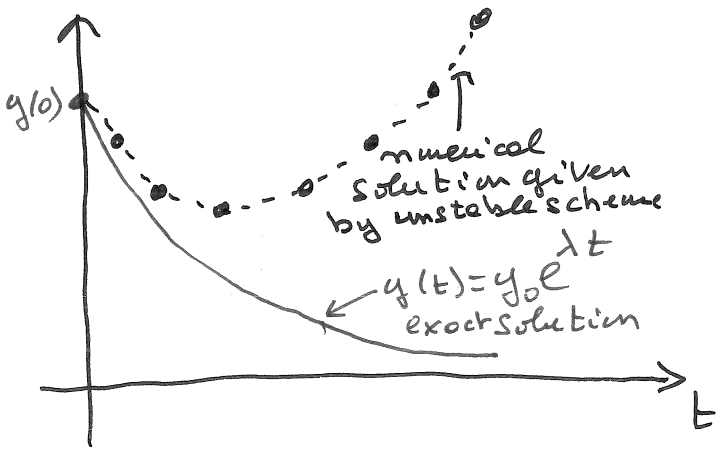
$$\tau^n = -\frac{\Delta t}{2}\ddot{y}^n + O(\Delta t^2)$$

and the LTE of CN scheme is:

$$\tau^n = \frac{\Delta t^2}{12}\dddot{y}^n + O(\Delta t^3).$$



IV Stability of a numerical scheme



The stability (also called absolute stability) is the property of the scheme to yield a numerical solution who does not diverge to ∞ when t is large.

Let us consider the model problem for $\lambda < 0$:

$$\begin{cases} \dot{y} = \lambda y, & t \in [0, +\infty[\\ y(0) = y_0 \end{cases} \quad (9)$$

Its exact solution is:

$$y(t) = y_0 e^{\lambda t},$$

and since $\lambda = 0$ the solution decays with time ($\lim_{t\to\infty} y(t) = 0$).

When BE1 is applied to model problem (9), the iterative equation reads:

$$y^{n+1} = y^n + \Delta t \lambda y^n \Rightarrow y^n = \sigma y^{n-1} = \dots = (\sigma)^n y^0$$

where $\sigma = 1 + \lambda \Delta t$ is the <u>amplification factor</u> of the scheme. In order to have a solution that does not blow up in time, we must have:

$$|\sigma| < 1 \quad (10)$$

which leads to $0 < \Delta t < -2/\lambda$.

Hence, the BE1 scheme is stable if we use a time-step that verifies the stability condition:

$$\Delta t < \frac{2}{-\lambda}. \quad (11)$$

We say that BE1 is <u>conditionnally</u> stable.

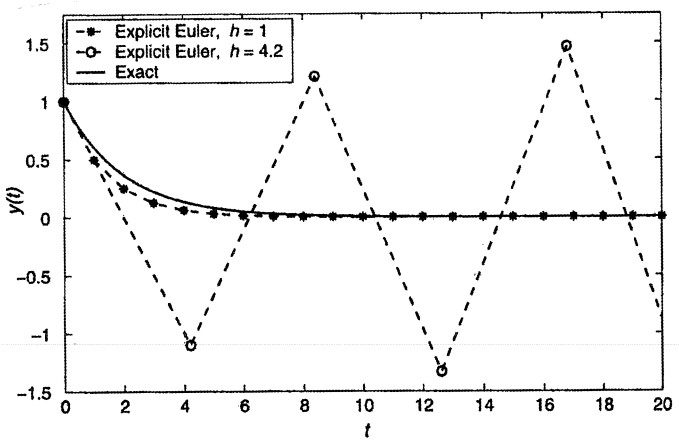
- If we use BE1 to solve model problem (9), the amplification factor is $\sigma = \dfrac{1}{1-\lambda\Delta t}$, and stability condition $|\sigma|<1$ leads to $|1-\lambda\Delta t|>1$, which is always verified for $\Delta t>0$. Hence, the BE1 is stable with no conditions.

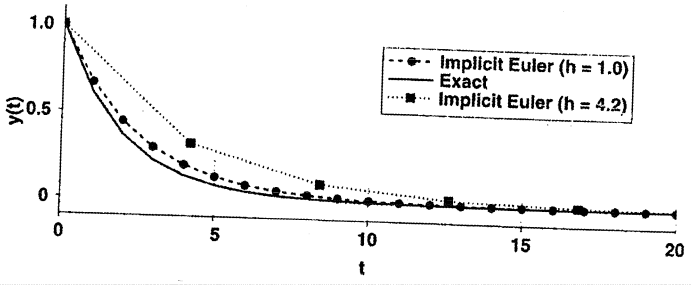
- The amplification factor for CN scheme is

$$\sigma = \frac{1+\lambda\Delta t/2}{1-\lambda\Delta t/2}$$

and stability condition (10) is verified for all $\Delta t>0$. As the BE1 schem, the CN scheme is always stable.

Application: We consider the model problem $\dot{y}+\frac{1}{2}y=0$. the stability condition of FE1 is $\Delta t<4$, while BE1 and CN are stable for all $\Delta t>0$. Fig.1 shows computations with FE with various values of Δt, the one with $\Delta t=4.2$ is unstable. Fig.2 shows that BE1 is stable for all values of Δt.

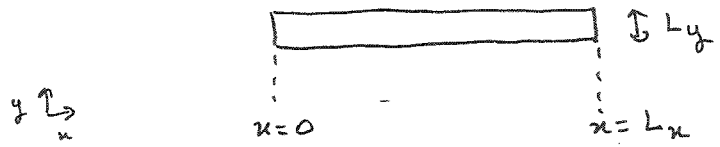




Résolution de Systèmes linéaires I : Généralités et méthodes directes

I) Motivation

On souhaite connaître l'équilibre thermique d'un barreau



tel que :

- Le barreau est long : $L_x \gg L_y$
- On impose une source de chaleur uniforme répartie dans le barreau : $\dot{w}(x,y)$ (en W/m^3)
- Aux extrémités $x=0$ et $x=L_x$ on impose la température nulle : $T=0$.

On cherche à connaître la distribution de température $T(x,y)$ dans ce barreau. Pour cela, on considère que dans un volume élémentaire dV du barreau, il existe :

- $\vec{q}(x,y)$: flux de chaleur au travers de la surface $\Gamma = \partial V$ (en W/m^2)
- $\dot{w}(x,y)$, la source de chaleur.

Pour que le volume de V soit en équilibre thermique, il faut que le flux de chaleur au travers de Γ soit égal à la quantité de chaleur reçue :

$$\int_{\partial V} \vec{q} \cdot \vec{n} \, d\Gamma = \int_V \dot{w} \, dV$$

soit, en utilisant le théorème de la divergence :

$$-\int_V (\nabla \cdot \vec{q} - \dot{w}) \, dV = 0. \quad (1)$$

La loi de Fourier énonce que $\vec{q} = -k \nabla T$, où k est la conductivité thermique du métal, ce qui conduit à

$$\int_V [-k \nabla^2 T - \dot{w}] \, dV = 0,$$

en supposant que k est une constante. Puisque cette intégrale doit être nulle pour tout volume V du barreau, alors nécessairement :

$$-k \nabla^2 T = \dot{w}. \quad (1)$$

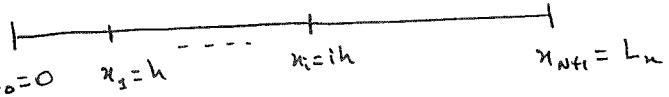
Cette équation aux dérivées partielles est connue sous le nom d'équation de Laplace.

On la simplifie en considérant que $L_x \gg L_y \Rightarrow T(x,y) = T(x)$ et on obtient le problème aux limites (avec $f = \dot{w}/k$)

$$(2a) \quad -\frac{d^2T}{dx^2} = f(x) \quad \text{pour } x \in]0, L_x[$$

$$(2b) \quad T(0) = 0, \quad T(L_x) = 0.$$

Il est possible de résoudre cette équation par la méthode des différences finies. Pour cela, on introduit le maillage du barreau :



qui est tel que $x_i = ih$ pour $i = 0, \dots, N+1$ avec $h = \frac{L_x}{N+1}$

On cherche maintenant à calculer $T_i = T(x_i)$ en ces points du maillage.

conduisent à :

$$T_0 = 0 \quad , \quad T_{N+1} = 0 .$$

Aux points $\{x_i, i=1,\dots,N\}$ situés à l'intérieur du maillage, l'Eq. (2a) s'écrit :

$$-T_i'' = f_i \quad , \quad \text{pour } i=1,\dots,N. \quad (3)$$

Pour obtenir une approximation de T_i'', on utilise la formule des TD 1 :

$$T_i'' \overset{?}{=} \frac{T_{i-1} - 2T_i + T_{i+1}}{h^2} . \quad (4)$$

En introduisant cette approxim. dans Eq. (3), on obtient le système linéaire de taille $N \times N$:

en $i=1$: $-\left(\frac{T_0 - 2T_1 + T_2}{h^2}\right) = f_1$ (5a) ($T_0 = 0$ d'après les C.L.)

pour $i=1,\dots,N-1$: $-\left(\frac{T_{i-1} - 2T_i + T_{i+1}}{h^2}\right) = f_i$ (5b)

pour $i=N$: $-\left(\frac{T_{N-1} - 2T_N + T_{N+1}}{h^2}\right) = f_N$ (5c) ($T_{N+1} = 0$)

Sous forme matricielle, il s'écrit comme :

$$\underline{\underline{K}}_n \, \underline{T} = \underline{F} \quad (6)$$

avec :

$$\underline{T} = \begin{pmatrix} T_1 \\ \vdots \\ T_N \end{pmatrix}, \quad \underline{F} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix},$$

$\underline{\underline{K}}_n$ = [illegible] pour :

$$\underline{\underline{K}}_n = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & (0) \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ (0) & & & -1 & 2 \end{bmatrix} \quad (7)$$

matrice tridiagonale

Conclusion : La discrétisation des équations de la mécanique donne lieu à des systèmes linéaires, qu'il faut maintenant résoudre. Pour des systèmes de grande taille ($n \gg 1$), il faut utiliser des méthodes qui soient :

(lorsqu'elles sont utilisées sur un ordinateur)

* efficaces (i.e., rapides)
* peu sensibles aux erreurs d'arrondi
* robustes : elle doivent fournir un résultat pour un grand nombre de problèmes différents.

II Généralités sur les systèmes linéaires

Un système linéaire d'ordre n s'écrit sous forme algébrique :

$$\text{pour } i=1,\dots,n \quad \sum_{j=1}^{n} A_{ij} x_j = b \quad (8)$$

(i : indice des lignes, j : indice des colonnes).

Il s'écrit aussi sous la forme matricielle :

$$\underline{\underline{A}} \, \underline{X} = \underline{B}$$

avec $\underline{X}$ et $\underline{B}$ des vecteurs de taille n, et $\underline{\underline{A}}$ une matrice de taille $n \times n$ telle que :

$$\underline{\underline{A}} = (A_{ij})_{i,j=1,\dots,n} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & & & \\ \vdots & & & n \end{bmatrix}$$

* La matrice $\underline{\underline{A}}$ est triangulaire supérieure si elle est telle que $A_{ij} = 0$ pour $i > j$, soit pour $n=3$:

$$\underline{\underline{A}} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & A_{33} \end{bmatrix}$$

* Elle est dite triangulaire inférieure si elle est telle que $A_{ij} = 0$ pour $j > i$, soit p. ex $n=3$:

$$\underline{\underline{A}} = \begin{bmatrix} A_{11} & 0 & 0 \\ A_{21} & A_{22} & 0 \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

* Elle est dite symétrique si $A_{ij} = A_{ji}$, par exemple :

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 0 & 4 \\ 3 & 4 & 2 \end{bmatrix}$$

* Un type de matrice très courant sont les matrices tridiagonales, de la forme :

$$\underline{\underline{A}} = \begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \quad \text{pour } n = 4.$$

Dans ce type de matrices, seuls les éléments diagonaux (les b_i), sur la 1ère diagonale supérieure (les c_i) et inférieure (les a_i) peuvent être non nuls.

On rencontre souvent ce type de matrices dans la méthode des différences finies. En mécanique, un exemple classique est la matrice k_n (Eq. (7)) pour laquelle, quel que soit n, la matrice a seulement 3 éléments non nuls par ligne. De plus, on remarque que k_n est symétrique.

III Méthodes de résolution directe.

On souhaite donc résoudre le système d'ordre n :

$$\underline{\underline{A}}\ \underline{X} = \underline{B} \qquad (S).$$

Dans ce cours, on considérera toujours des systèmes non singuliers ($\det \underline{\underline{A}} \neq 0$), pour lesquels la solution unique de (S) s'écrit :

$$\underline{X} = \underline{\underline{A}}^{-1}\ \underline{B}.$$

Pour des raisons liées aux erreurs d'arrondi (cf chap 1) on ne doit jamais résoudre un syst. lin. sur ordinateur en calculant $\underline{\underline{A}}^{-1}$ et en la multipliant avec $\underline{B}$. On utilise plus généralement le procédé d'élimination de Gauss. C'est l'exemple le plus populaire de méthodes de résolution directe, qui fournissent la solution en un nombre fini d'étapes.

(A l'opposé : les méthodes de résolution itératives, qui fournissent la solution en un nombre infini (théoriquement) d'étapes, seront vu dans le chap. suivant).

Avant d'aborder le cas général, considérons l'exemple du syst. triang. sup. d'ordre 3 défini par :

$$\underline{\underline{A}} = \begin{bmatrix} 4 & 8 & 12 \\ 0 & 2 & 4 \\ 0 & 0 & 2 \end{bmatrix}, \quad \underline{b} = \begin{pmatrix} 4 \\ 2 \\ 4 \end{pmatrix}$$

qui s'écrit :

$$\begin{cases} 4x_1 + 8x_2 + 12x_3 = 4 \\ 2x_2 + 4x_3 = 2 \\ 2x_3 = 4 \end{cases} \qquad (10)$$

On le résout facilement en observant que la dernière ligne donne $x_3 = 2$, celle d'au dessus donne alors :

$$x_2 = \frac{1}{2}[2 - 4x_3] = -3$$

et la première ligne donne enfin :

$$x_1 = \frac{1}{4}[4 - 8x_2 - 12x_3] = 1.$$

Cette méthode est l'algorithme de substitution rétrograde ("backward sweep"). Pour un système d'ordre n, cet algo s'écrit :

$$x_n = b_n / A_{nn}, \qquad (11a)$$

et, pour $i = N-1$ jusqu'à 1 :

$$x_i = \frac{1}{A_{ii}} \left(b_i - \sum_{j=i+1}^{N} A_{ij} x_j \right) \qquad (11b)$$

Cet algorithme se programme en 5 lignes sur un ordi., et il est au coeur de la méthode de substitution de Gauss pour les matrices $\underline{\underline{A}}$ quelconques.

III - 4 Méthode d'élimination de Gauss (MEG)

La MEG a pour but de transformer le système $\underline{\underline{A}}\,\underline{x} = \underline{b}$ en un système équivalent (ie, ayant la même solution) de la forme

$$\underline{\underline{U}}\,\underline{x} = \underline{b}' \qquad (12)$$

où $\underline{\underline{U}}$ (ie : Upper) est une matrice triang. & supérieure. Le système (12) peut alors être aisément résolu par l'algo de substitution rétrograde de la section précédente.

Pour décrire la MEG, prenons l'exemple du système d'ordre 3 :

$$\underline{\underline{A}} = \begin{bmatrix} 4 & 8 & 12 \\ 3 & 8 & 13 \\ 2 & 9 & 18 \end{bmatrix} \qquad \underline{b} = \begin{pmatrix} 4 \\ 5 \\ 11 \end{pmatrix}$$

qui s'écrit :

$$\begin{cases} 4x_1 + 8x_2 + 12x_3 = 4 & r_1^{(1)} \\ \boxed{3}x_1 + 8x_2 + 13x_3 = 5 & r_2^{(1)} \\ \boxed{2}x_1 + \boxed{9}x_2 + 18x_3 = 11 & r_3^{(1)} \end{cases} \qquad (13)$$

et on notera $r_i^{(k)}$ la $i^{\text{ième}}$ ligne du système obtenu à la $k^{\text{ième}}$ étape de la MEG (pour le syst. (13), on a $k = 1$).

1ère partie de la MEG : mise sous forme triangulaire supérieure

Afin de rendre le syst. (13) triangul. sup., il faut annuler les coefficients encadrés dans (13) en effectuant des opérations algébriques sur les lignes $r_i^{(1)}$ de ce syst. En effectuant les opérations suivantes :

$$(14) \quad \begin{cases} r_1^{(2)} = r_1^{(1)} \\ r_2^{(2)} = r_2^{(1)} - m_{21} r_1^{(1)}, \text{ avec } m_{21} = 3/4 \end{cases}$$

le système (13) se transforme en :

$$\begin{cases} -4x_1 + 8x_2 + 12x_3 = 4 & r_1^{(2)} \\ 2x_2 + 4x_3 = 2 & r_2^{(2)} \\ \boxed{5}x_2 + 12x_3 = 9 & r_3^{(2)} \end{cases} \quad (1$$

Pour annuler le coefficient encadré dans (15), il faut faire les opérations

$$\begin{aligned} r_1^{(3)} &= r_1^{(2)} \\ r_2^{(3)} &= r_2^{(2)} \\ r_3^{(3)} &= r_3^{(2)} - m_{32} r_2^{(2)}, \quad m_{32} = 5/2. \end{aligned} \quad (16$$

On obtient alors le système triangulaire supérieur

$$\begin{cases} 4x_1 + 8x_2 + 12x_3 = 4 \\ 2x_2 + 4x_3 = 2 \\ 2x_3 = 4 \end{cases} \quad (17)$$

2ème Partie de la MEG : Résolution par substitution rétrograde

Le syst. triang. sup. peut alors être aisément résolu par l'algo de la section III-1. Dans notre exemple, on aura reconnu que (17) est l'exemple de la section précédente, dont la solution est :

$$x_1 = 1, \quad x_2 = -3, \quad x_3 = 2.$$

III.3 factorisation LU

Il est parfois utile de réinterpréter la MEG comme une méthode de factorisation de la matrice $\underline{\underline{A}}$ sous la forme

$$\underline{\underline{A}} = \underline{\underline{L}}\ \underline{\underline{U}}, \quad (18)$$

telle que $\underline{\underline{U}}$ est la matrice triang. sup. donnée par la première partie de la MEG ; pour notre exemple :

$$\underline{\underline{U}} = \begin{bmatrix} 4 & 8 & 12 \\ 0 & 2 & 4 \\ 0 & 0 & 2 \end{bmatrix}$$

et $\underline{\underline{L}}$ (Lower) est la matrice triang. inférieure de la forme :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix}$$

où les m_{ij}, $i > j$, sont les <u>multiplicateurs</u> utilisés par les opérations élémentaires (14) et (16) de la première partie de la MEG. Pour notre exemple :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 3/4 & 1 & 0 \\ 1/2 & 5/2 & 1 \end{bmatrix}.$$

Lorsque la factorisation LU de la matrice $\underline{\underline{A}}$ est connue il est aisé de résoudre le système $\underline{\underline{L}}\,\underline{\underline{U}}\,\underline{x} = \underline{b}$ par les 2 étapes successives :

puis -

$$\underline{\underline{U}}\ \underline{x} = \underline{y} \qquad (19\,b)$$

Puisque ces 2 systèmes sont triangulaires, on les résout aisément par un algorithme de substitution rétrograde (pour (19b)) et de substitution progressive (pour (19a)).

III-4 Complexité Algorithmique

Afin d'évaluer l'efficacité (i.e, le temps de calcul) d'un algorithme de résolution numérique, il est important de connaître sa complexité algorithmique, i.e le nombre d'op-rations arithmétiques scalaires (multiplications, additions, etc) qu'il nécessite. Ces opérations arithmétiques sont souvents appelées flops (floating points scalar operations).

Par exemple, comptons le nombre de flops que nécessite la multiplication matrice-vecteur $\underline{\underline{A}}\,\underline{x}$, qui s'écrit :

$$\text{pour } i = 1, \dots, n \qquad \sum_{j=1}^{n} A_{ij} x_j = A_{i1} x_1 + A_{i2} x_2 + \dots + A_{in} x_n \quad ($$

Pour chaque ligne i, on fait donc n multiplications et $n-1$ additions, soit $2n-1$ flops. La complexité de l'algo est donc :

$$n(2n-1) = \mathcal{O}(n^2) \quad \text{flops.}$$

Ainsi, lorsqu'on double la taille n de la matrice, le nombre d'opérations est multiplié par 4. Le coût de cet algorithme peut être réduit dans le cas particulier des matrices tridiagonales (p. ex. la matrice de diff. finies $K_n(7)$) l'opération (20) devient alors

$$\text{pour } i = 1, \dots, n \qquad \sum_{j=1}^{n} A_{ij} x_j = a_i x_{i-1} + b_i x_i + c_i x_{i+1}$$

et elle nécessite seulement $5n = \mathcal{O}(n)$ opérations. Le coût de cet algorithme varie seulement linéairement avec la taille n de la matrice.

Il est crucial de connaître la complexité d'un algorithme de résolution de syst. linéaire, car une méthode qui semble séduisante sur le plan théorique peut se révéler inutilisable en pratique. Prenons l'exemple des formules de Cramer qui donne la solution du système $A\underline{x} = \underline{b}$ sous la forme

$$\underline{x}_j = \frac{\Delta_j}{\det(A)}, \quad j = 1, \dots, n,$$

où Δ_j est le déterminant de la matrice obtenue en remplaçant la $j^{ème}$ colonne de $\underline{\underline{A}}$ par le second membre $\underline{b}$.

La complexité algorithmique de cette formule est représentée par le calcul (par récurrence) des déterminants, et on peut montrer que celle est de $(n+1)!$ flops, ce qui est gigantesque: pour résoudre un système linéaire d'ordre $n=50$, un P.C. actuel nécessiterait plusieurs milliards d'années de temps de calcul ("temps CPU") $\Rightarrow$ La méthode de Cramer n'est jamais utilisée en pratique.

Pour la MEG, on peut facilement montrer que: $\frac{2}{3}n^3 + 2n^2$

- La mise sous forme triang. supérieure nécessite $O(n^3)$ flop
- La substitution rétrograde nécessite n^2 flops.

Soit un total de $O(n^3) + O(n^2)$ flops.

Ainsi, pour résoudre un problème de taille $n=50$, la MEG nécessite environ 100000 flops, ce qui prend 10^{-4}s sur un PC de 200 Mhz : bien mieux que la meth. de Cramer !

Il faut aussi retenir que le coût dominant de la MEG est la mise sous forme triang. sup., soit $O(n^3)$ opérations. Ainsi, lorsque la taille de la matrice est multipliée par 2, le coût algorithmique est multiplié par 8. Pour les systèmes de grandes tailles, le temps de calcul de la MEG peut se révéler prohibitif. Les méthodes itératives (cf chap. suivant) sont alors utilisées dans l'espoir de diminuer les temps de calcul de la MEG.

): Un P.C. à 200 Mhz calcule environ 10^9 flops par secondes, soit 3.10^{18} flops par années. Pour réaliser $51! \geq 10^{67}$ opérations, le P.C. devrait calculer pendant 5.10^{49} années [illegible]

Lorsque'on représente des nombres réels sur un ordinateur (ou une calculatrice), il resulte généralement des erreurs d'arrondis. Par exemple, la représentation à 4 chiffs significatifs du rationnel $b = \frac{1}{3}$ est :

$$b + \delta b = 0,3333$$

ce qui génère une erreur relative d'arrondi $\frac{\delta b}{b} = 10^{-4}$.

Supposons que de telles erreurs d'arrondi sont introduites dans le second-membre du système linéaire $\underline{\underline{A}}\,\underline{X} = \underline{B}$ qui devient alors :

$$\underline{\underline{A}}\,\underline{X} = \underline{B} + \underline{\delta B}\,.$$

Dans ce cas, les opérations arithmétiques (addition, mult., .. de sa résolution par méthode de Gauss vont propager les erreurs d'arrondis jusqu'à la solution calculée : $\underline{X} +$

Dans certains cas "critiques", on observe qu'une faible perturbation $\left(\frac{\|\delta B\|}{\|B\|} << 1\right)$ va s'amplifier jusqu'à entraîner de larges erreurs $\left(\frac{\|\delta X\|}{\|X\|} >> 1\right)$ sur la solution calculée, ce qui est inacceptable : On dit alors que le système est mal conditionné.

Ce paragraphe a pour but de :

- Définir le nombre de condition $\chi(\underline{\underline{A}})$ d'un système linéaire.
- Estimer les erreurs de la méthode de Gauss pour les systèmes mal conditionnés.
- Vous constaterez que de nombreux problèmes de la mécanique sont mal conditionnés.

IV-1 Exemple de systèmes mal conditionnés

Considérons le système linéaire d'ordre 2 :

$$\underline{\underline{A}}\,\underline{X} = \underline{B}\,, \qquad (20a)$$

avec $\underline{\underline{A}} = \begin{bmatrix} 1 & 1 \\ 1 & \frac{10001}{10000} \end{bmatrix}$ et $\underline{B} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$. (20b)

L'unique solution de (20) est $\underline{X} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$.

Considérons maintenant une petite perturbation de ce système linéaire, soit :

$$\underline{\underline{A}}\,\underline{X}' = \underline{B} + \underline{\delta B}\,, \qquad (21)$$

avec $\underline{\delta B} = \begin{pmatrix} 0 \\ 10^{-4} \end{pmatrix}$. La solution de ce système est $\underline{X}' = \underline{X} + \underline{\delta x} =$

Pour cet exemple, on remarque avec effroi qu'un changement relatif du second membre, de l'ordre de :

$$\frac{\|\underline{\delta B}\|}{\|\underline{B}\|} = \frac{10^{-4}}{2\sqrt{2}} \simeq 0.04\,\%\,,$$

produit un changement relatif dans la solution :

$$\frac{\|\underline{\delta X}\|}{\|\underline{X}\|} = \frac{\sqrt{2}}{2} \simeq 71\,\%\,,$$

soit une amplification de $\dfrac{\|\delta X\|/\|X\|}{\|\delta B\|/\|B\|} \simeq 2.10^4$ de la perturbation du second-membre. Que se passe-t-il ?

explication géométrique

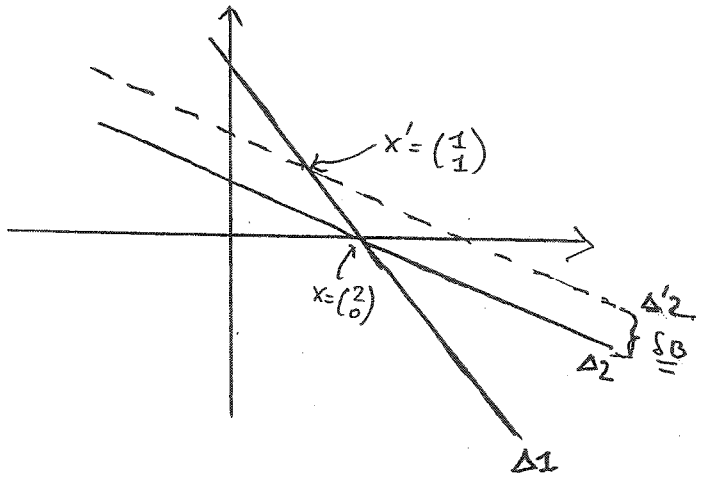
On peut faire une première estimation qualitative de ce phénomène en interprétant la solution du système soit :

$$\begin{cases} x + y = 2 & (\Delta 1) \\ \dots & (\Delta 2) \end{cases}$$

comme l'intersection $X = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ des deux droites $\Delta 1$ et $\Delta 2$. Ces 2 droites sont quasiment parallèles : une petite ~~chose~~ translation de $\Delta 2$, par exemple :

$$x + (1.0001)\, y = 2 + 10^{-4} \quad (\Delta 2')$$

va produire un grand changement sur l'intersection de ces deux droites, qui est maintenant $X' = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.



V-2. Nombre de conditionnement

On souhaite estimer l'erreur relative $\|\delta X\| / \|X\|$ que l'on commet sur la solution, en fonction des données du problème ($\underline{\underline{A}}$ et $\underline{B}$), et de la perturbation $\delta \underline{B}$. Pour cela, on dispose du théorème :

Théorème On appelle $\chi(\underline{\underline{A}}) \geq 1$ le **nombre de conditionnement** du système (20) défini par :

$$\chi(\underline{\underline{A}}) = |||\underline{\underline{A}}||| \; |||\underline{\underline{A}}^{-1}|||, \qquad (22)$$

où $|||\cdot|||$ est une norme matricielle. On dispose de l'estimation d'erreur :

$$0 \leq \frac{\|\delta X\| / \|X\|}{\|\delta B\| / \|B\|} \leq \chi(\underline{\underline{A}}) \qquad (23).$$

Démonstration : On rappelle qu'une norme matricielle $|||\cdot|||$ est définie par :

$$|||\underline{\underline{A}}||| = \sup_{\substack{\underline{X} \in \mathbb{R}^m \\ \underline{X} \neq 0}} \frac{\|\underline{\underline{A}}\underline{X}\|}{\|\underline{X}\|}.$$

En particulier, on en déduit que pour tout $\underline{X} \in \mathbb{R}^m$, on a l'inégalité bien utile :

$$\|\underline{\underline{A}}\underline{X}\| \leq |||\underline{\underline{A}}||| \; \|\underline{X}\|. \qquad (24)$$

A partir de la relation $\underline{B} = \underline{\underline{A}}\,\underline{X}$, on déduit de cette inégalité que

$$\|\underline{B}\| \leq |||\underline{\underline{A}}||| \; \|\underline{X}\|,$$

et de $\underline{X} = \underline{\underline{A}}^{-1}\underline{B}$, on déduit que :

$$\|\underline{X}\| \leq |||\underline{\underline{A}}^{-1}||| \; \|\underline{B}\|,$$

ces 2 relations conduisent à l'encadrement :

$$\frac{1}{|||\underline{\underline{A}}^{-1}||| \; \|\underline{B}\|} \leq \frac{1}{\|\underline{X}\|} \leq \frac{|||\underline{\underline{A}}|||}{\|\underline{B}\|} \qquad (25).$$

D'une manière analogue, à partir des relations $\delta \underline{B} = \underline{\underline{A}}\,\delta \underline{X}$ et $\delta \underline{X} = \underline{\underline{A}}^{-1}\delta \underline{B}$ on déduit l'encadrement :

$$\frac{\|\delta B\|}{\|\underline{\underline{A}}\|} \leq \|\delta \underline{X}\| \leq |||\underline{\underline{A}}^{-1}||| \; \|\delta \underline{b}\| \qquad (26).$$

A partir de (25) et (26), on obtient finalement :

$$\frac{1}{\chi(\underline{\underline{A}})} \frac{\|\delta \underline{B}\|}{\|\underline{B}\|} \leq \frac{\|\delta \underline{X}\|}{\|\underline{X}\|} \leq \chi(\underline{\underline{A}}) \frac{\|\delta \underline{B}\|}{\|\underline{B}\|},$$

avec $\chi(\underline{\underline{A}}) = |||\underline{\underline{A}}||| \; |||\underline{\underline{A}}^{-1}|||$, d'où on déduit (22).

l'estimation d'erreur (2.) montre que la solution peut être aussi bien "insensible" sur la solution peut être aussi bien "insensible"

$\frac{\|\delta x\|}{\|x\|} \geq 0$, que potentiellement dangereuse $\frac{\|\delta x\|}{\|x\|} \geq$

lorsque $\chi(A)$ est grand.

Comment calculer "efficacement" le nombre de conditionnement $\chi(A) = |||A||| \, |||A^{-1}|||$ sans calculer les normes matricielles ? Pour un certain type de matrices, très courant en méca-nique, il suffit de connaître les val. propres de A pour calculer $\chi(A)$:

Propriété : Une matrice symétrique A est dite définie positive (SDP) si toutes ses valeurs propres sont strictement positives. Le conditionnement d'une matrice SDP est simplement

$$\chi(A) = \frac{\lambda_{max}}{\lambda_{min}} \geq 1,$$

où λ_{max} et λ_{min} sont respectivement la plus grande et plus petite valeur propre de A.

Calculons $\chi(A)$ pour l'exemple (20). La matrice A est symétrique, ses val. propres sont $\lambda_1 \simeq 5.10^{-5}$ et $\lambda_2 \simeq 2$, elle est donc SPD et son conditionnement est :

$$\chi(A) = \frac{\lambda_2}{\lambda_1} \simeq 4.10^4.$$

L'estimation d'erreur (2.) devient alors :

$$0 \leq \frac{\|\delta x\|/\|x\|}{\|\delta B\|/\|B\|} \leq 4.10^4$$

et on avait observé avec $\|\delta x\|/\ldots \geq 2.10^4 = \frac{1}{2}\chi(A)$

pour l'exemple (20). Ainsi, sans faire de choix particulier sur la perturbation δB, on observe une erreur relative proche du cas extrême $\chi(A) \simeq 4.10^4$.

Désormais, on qualifiera les systèmes linéaires pour lesquels $\chi(A)$ est grand de systèmes mal conditionnés qui sont (potentiellement) très sensible aux erreurs d'arr

Les systèmes linéaires issus de la mécanique sont notoirement connus pour être mal conditionnés. Par exemple, le TD 4 nous montre que les val. propres de la matrice de diff. finies symétrique K_n (Eq. (7)) sont :

$$\lambda_k = 2 - 2\cos\frac{k\pi}{n+1}, \quad k = 1, \ldots, n.$$

La matrice K_n est donc SDP, et son conditionnement est :

$$\chi(K_n) = \frac{\lambda_{max}}{\lambda_{min}} = \frac{1+\cos\frac{\pi}{n+1}}{1-\cos\frac{\pi}{n+1}},$$

et, lorsque n est grand, un développement limité montre que

$$\chi(K_n) \simeq \frac{4}{\pi^2}(n+1)^2.$$

Cette matrice est donc mal conditionnée pour des problèmes de grande taille.

IV - 3 Conditionnement et erreurs d'arrondi des méthodes directes

Même si on ne connaît pas la solution exacte d'un système linéaire, il est possible d'évaluer l'importance des erreurs d'arrondis dues par une méthode directe.

Supposons que pour résoudre le système linéaire $\underline{\underline{A}}\,\underline{X} = \underline{B}$ on ait utilisé une méthode directe (p.ex. la ME G) pour obtenir une solution X', qui est entaché d'erreur d'arrondis : alors forcément la quantité

$$\underline{\underline{A}}\,\underline{X}' - \underline{B} \neq 0 .$$

On définit alors le résidu comme :

$$\underline{R} = \underline{B} - \underline{\underline{A}}\,\underline{X}', \qquad (27)$$

et forcément X' est la solution exacte du système :

$$\underline{\underline{A}}\,\underline{X}' = \underline{B} + \underline{R} ,$$

où $\underline{R}$ correspond à la petite perturbation $\delta\underline{B}$ du syst. (21) l'estimation d'erreur (23) appliquée à ce système donne, dans le cas présent :

$$\frac{\|\underline{X}' - \underline{X}\|}{\|\underline{X}\|} \leq \chi(\underline{\underline{A}}) \, \frac{\|\underline{R}\|}{\|\underline{B}\|}$$

Ce qui montre que l'erreur relative croît avec le nombre de conditionnement : lorsque $\chi(\underline{\underline{A}})$ est suffisamment grand les erreurs sur la solution peuvent être du même ordre que la solution elle-même : Très mauvais !

Exemple On considère le système linéaire $AX = B$ où $\underline{\underline{A}}$ est la matrice de Hilbert d'ordre n définie par :

$$\underline{\underline{A}}_{ij} = \frac{1}{i+j-1} , \qquad i,j = 1, \dots, n. \qquad (28)$$

On montre que son conditionnement se comporte comme $\chi(\underline{\underline{A}}) \simeq e^{3.5n}$ (c'est donc un système très mal conditionné, même pour des systèmes de taille modeste ($n \geq 10$). La Fig 1 montre que l'erreur obtenue par méthode directe croît avec le nombre de conditionnement $\chi(\underline{\underline{A}})$, et qu'elle devient inacceptable lorsque $\chi(\underline{\underline{A}})$ est grand ($n \geq 12$). Par contre, l'erreur donné par la méthode itérative reste acceptable.

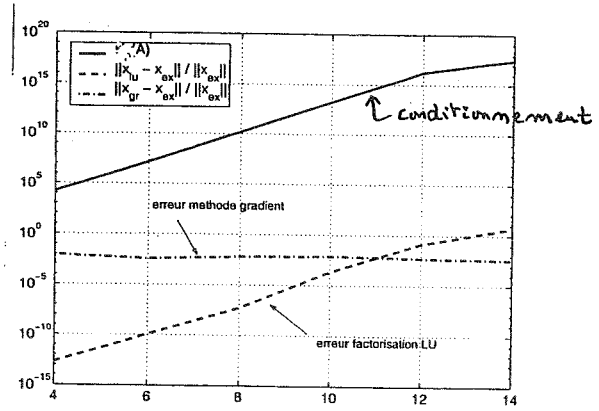


Fig : 1 : Résolution des systèmes de Hilbert (28) pour diverses valeurs de n : Erreur relative entre la solution exacte $\underline{X}_{ex}$ et la solution donnée par :

- Méthode directe (fact. LU) : $\underline{X}_{LU}$
- Méthode itérative ("gradient") : $\underline{X}_{GR}$

Résolution de systèmes linéaires II :
- Méthodes itératives

I Généralités

I-1 Définition du problème

On veut résoudre le système linéaire d'ordre n

$$\underline{\underline{A}}\,\underline{X} = \underline{F}, \qquad (1)$$

où $\underline{\underline{A}}$ est "régulière" ($\det \underline{\underline{A}} \neq 0$). Son unique solution est notée $\underline{X}^*$. Pour cela, on souhaite définir une suites $\{\underline{X}^k,\ k \geq 0\}$ telle que :

$$\lim_{k \to +\infty} \underline{X}^k = \underline{X}^*.$$

I-2 Motivation

Le développement de méthodes itératives est basé sur les 2 principales carences des méthodes directes, qui nous avons identifiées dans le chapitre précédent :

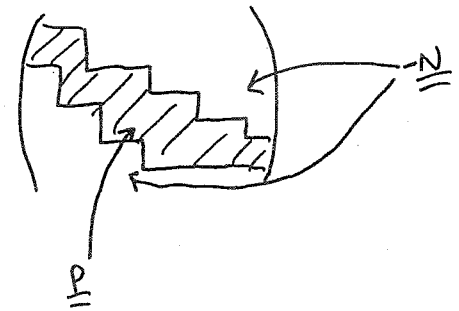
* Pour une matrice $\underline{\underline{A}}$ quelconque, l'élimination de Gauss (ou méthode LU) nécessite $\mathcal{O}(n^3)$ opérations arithmétiques (flops), ce qui peut être prohibitif pour des problèmes de grande taille.

  En contraste, le passage $\underline{X}^k \to \underline{X}^{k+1}$ pour une méth. itérative nécessite seulement $\mathcal{O}(n^2)$ flops (essentiellement l'évaluation du résidu $\underline{R}^k = \underline{B} - \underline{\underline{A}}\,\underline{X}^k$).

  Une méthode itérative peut donc devenir compétitive si elle converge en un nombre d'itérations indépendant de n.

* Si $\underline{\underline{A}}$ est mal conditionnée, une méthode directe peut entraîner de larges erreurs d'arrondis. Ce phénomène est beaucoup moins sensible avec les méthodes itératives (cf. Fig 1 du chap. précédent) car, en particulier, on peut contrôler la valeur du résidu $\underline{R}^k$ de la solution.

II Principes de base : les meth. de décomposition ("Splitting")



L'idée de base est de décomposer la matrice $\underline{\underline{A}}$ en :

$$\underline{\underline{A}} = \underline{\underline{P}} - \underline{\underline{N}}, \qquad (2)$$

le problème (1) s'écrit alors de façon équivalente :

$$\underline{\underline{P}}\,\underline{X} = \underline{\underline{N}}\,\underline{X} + \underline{F}. \qquad (3)$$

On définit alors la méthode itérative :

$$\underline{\underline{P}}\,\underline{X}^{k+1} = \underline{\underline{N}}\,\underline{X}^k + \underline{F} \qquad (4).$$

La matrice $\underline{\underline{P}}$, appelée matrice de préconditionnement

- $\underline{\underline{P}}$ est inversible (i.e., $\underline{\underline{P}}^{-1}$ existe)
- $\underline{\underline{P}}$ est "facile" à inverser.

Sous ces hypothèses, on peut formellement écrire l'itération sous la forme :

$$X^{k+1} = \underbrace{\underline{\underline{P}}^{-1}\underline{\underline{N}}}_{\underline{\underline{B}}} X^k + \underline{\underline{P}}^{-1}\underline{F} \quad . \qquad (5)$$

On peut remarquer que cette itération correspond à une méthode du point fixe $x^{k+1} = \phi(x^k)$ (cf cours sur les syst. non-linéaires). Par analogie à cet type de méthodes, la matrice $\underline{\underline{B}} = \underline{\underline{P}}^{-1}\underline{\underline{N}}$ est appellée matrice d'itération.

Il est aisé de vérifier que la méthode (4) est consistante

$$\underline{\underline{P}}\underline{X}^* = \underline{\underline{N}}\underline{X}^* + \underline{F} \iff (\underline{\underline{P}} - \underline{\underline{N}})\underline{X}^* = \underline{F}$$
$$\iff \underline{\underline{A}}\,\underline{X}^* = \underline{F} \quad .$$

Examinons maintenant la convergence de la méthode (4).

Définition (Convergence) : Une méthode itérative est convergente si l'erreur à la $k^{ième}$ itération $\underline{E}^k = \underline{X}^k - \underline{X}^*$ vérifie :

$$\lim_{k \to +\infty} \| \underline{E}^k \| = 0$$

En combinant (4) et (3), on obtient :

$$\underline{X}^k - \underline{X}^* = \underline{\underline{B}}\,(\underline{X}^{k-1} - \underline{X}^*)$$

d'où on déduit l'inéqalité :

$$\| \underline{E}^k \| \leq |||\underline{\underline{B}}||| \, \| \underline{E}^{k-1} \|$$

... sous. N.L., on a $|||\underline{\underline{B}}||| \leq \rho(\underline{\underline{B}})$, et on obtient finalement :

$$\| \underline{E}^k \| \leq [\rho(\underline{\underline{B}})]^k \, \| \underline{E}^0 \| .$$

Ainsi, l'itération (4), qui est consistante, converge si et seulement si ~~est~~ elle est stable :

$$\boxed{\rho(\underline{\underline{B}}) < 1 .} \qquad (6)$$

Rq : Contrairement aux meth. itér. pour syst. non lin., on peut remarquer que l'itération (4) converge pour toute solution initiale $\underline{X}^0 \in \mathbb{R}^n$.

III Les méthodes itératives de base

Dans cette partie, on va présenter les 3 méthodes "historiques" de Jacobi, Gauss-Seidel, relaxation (S.O.R.), qui reposent toutes sur la décomposition (2) :

$$\underline{\underline{A}} = \underline{\underline{P}} - \underline{\underline{N}} .$$

Elles diffèrent l'une de l'autre par le choix du préconditionneur $\underline{\underline{P}}$ qu'il faut inverser.

Leurs performances dépendent des valeurs propres de la matrice d'itération $\underline{\underline{B}} = \underline{\underline{P}}^{-1}\underline{\underline{N}}$, plus particulièrement de son rayon spectral :

$$\rho(\underline{\underline{B}}) = \max_{i=1,\dots,n} |\lambda_i(\underline{\underline{B}})| .$$

En effet, d'après l'estimation d'erreur :

$$\| \underline{E}^k \| \leq \rho(\underline{\underline{B}}) \, \| \underline{E}^{k-1} \| ,$$

on déduit que plus $\rho(\underline{\underline{B}})$ sera petit, plus la convergence sera rapide.

On comparera les performances de ces méthodes sur le problème test d'ordre 2 :

$$\underline{\underline{A}}\,\underline{X} = \underline{F}, \quad \text{avec} \quad \underline{\underline{A}} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \qquad (7)$$

qui correspond à la matrice de différence-finies $\underline{\underline{K}}_n$ avec $n=2$ du chapitre précédent. Le cas n quelconque sera examiné en TD.

III-1 Décompositions de la matrice $\underline{\underline{A}}$

Toutes les méthodes itératives de base reposent sur la décomposition :

$$\underline{\underline{A}} = \begin{bmatrix} \diagdown\!\underline{\underline{D}}\!\diagdown & -\underline{\underline{U}} \\ -\underline{\underline{L}} & \end{bmatrix}, \text{ soit } \quad \underline{\underline{A}} = \underline{\underline{D}} - \underline{\underline{L}} - \underline{\underline{U}} \qquad (8)$$

tel que :

* $\underline{\underline{D}}$ ("diagonal") est composée des éléments diagonaux de $\underline{\underline{A}}$:

$$\underline{\underline{D}}_{i,j} = \begin{cases} A_{ii} & \text{pour } i = 1, \dots, n \\ 0 & \text{pour } i \neq j \end{cases}$$

* $\underline{\underline{L}}$ ("Lower") : éléments triangulaires inférieurs de $\underline{\underline{A}}$:

$$\underline{\underline{L}}_{i,j} = \begin{cases} -A_{ij} & \text{si } i > j \ (\text{ie, pour } j=1,\dots,i-1) \\ 0 & \text{si } i \leq j \end{cases}$$

* $\underline{\underline{U}}$ ("Upper") : éléments triangulaires inférieurs de $\underline{\underline{A}}$:

$$\underline{\underline{U}}_{i,j} = \begin{cases} -A_{ij} & \text{si } i < j \ (\text{ie, pour } j=i+1,\dots,n \\ 0 & \text{si } i \geq i \end{cases}$$

Exemple : Pour la matrice diff. finies d'ordre 3 ($n=3$)

$$\underline{\underline{A}} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix},$$

la décomposition (8) correspond à :

$$\underline{\underline{D}} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad \underline{\underline{L}} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \underline{\underline{U}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

III-2 La méthode de Jacobi

La méthode la plus simple consiste à choisir le préconditionneur

$$\underline{\underline{P}}_J = \underline{\underline{D}}, \qquad (9)$$

car dans ce cas le préconditionneur est trivial à inverser :

$$\underline{\underline{D}}^{-1} = \begin{cases} \frac{1}{A_{ii}} & \text{pour } i = j \\ 0 & \text{sinon} \end{cases}.$$

Ce choix entraîne $\underline{\underline{N}}_J = \underline{\underline{P}}_J - \underline{\underline{A}} = \underline{\underline{L}} + \underline{\underline{U}}$, et l'itération (4) devient :

$$\underline{X}^{k+1} = \underline{\underline{D}}^{-1}(\underline{\underline{L}} + \underline{\underline{U}})\,\underline{X}^k + \underline{\underline{D}}^{-1}\underline{F}. \qquad (10a)$$

En posant $\underline{X}^k = (x_i^k,\ i=1\dots n)$ et $\underline{F} = (f_i,\ i=1,\dots,n)$, elle s'écrit pour les composantes scalaires :

$$x_i^{k+1} = \frac{1}{A_{ii}}\left(f_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} A_{ij}\, x_i^k \right), \quad i=1,\dots,n. \qquad (10$$

Cette méthode est connue sous le nom de meth. de Jacobi

Le préconditionneur est :

$$\underline{\underline{P}}_J = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \Rightarrow \underline{\underline{P}}_J^{-1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}$$

d'où on déduit :

$$\underline{\underline{N}}_J = \underline{\underline{P}}_J - \underline{\underline{A}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow \underline{\underline{B}}_J = \underline{\underline{P}}_J^{-1} \underline{\underline{N}}_J = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix}$$

L'itération (10a) s'écrit donc :

$$\underline{X}^{k+1} = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix} \underline{X}^k + \begin{pmatrix} f_1/2 \\ f_2/2 \end{pmatrix}. \quad (11)$$

Les valeurs propres de $\underline{\underline{B}}_J$ sont $1/2$ et $-1/2$, et le facteur de réduction est donc :

$$\rho(\underline{\underline{B}}_J) = \frac{1}{2}.$$

La méthode de Jacobi converge et, de plus :

$$\|\underline{E}^k\| \leq \frac{1}{2} \|\underline{E}^{k-1}\|,$$

ce qui montre que l'erreur est au moins divisée par 2 à chaque itération.

Pour ce problème simple, la méthode de Jacobi se révèle efficace. Comme on le verra en TD, son efficacité chute considérablement pour des systèmes mal conditionnés. Des améliorations s'imposent !

III-3 La méthode de Gauss-Seidel

Afin de présenter cette méthode, écrivons l'itération de Jacobi (11) pour les composantes :

$$2 x_1^{k+1} = x_2^k + f_1$$

puis,

$$2 x_2^{k+1} = x_1^k + f_2.$$

Une amélioration est d'utiliser dans cette dernière équation la dernière (et meilleure) estimation de x_1, soit x_1^{k+1}, ce qui donne

$$2 x_1^{k+1} = x_2^k + f_1$$

puis,

$$2 x_2^{k+1} = x_1^{k+1} + f_2.$$

En regroupant à gauche l'itéré $k+1$, cette nouvelle méthode s'écrit matriciellement :

$$\begin{bmatrix} 2 & 0 \\ -1 & 2 \end{bmatrix} \begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x_1^k \\ x_2^k \end{pmatrix} + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

Ce qui correspond à l'itération :

$$(-\underline{\underline{L}} + \underline{\underline{D}}) \underline{X}^{k+1} = \underline{\underline{U}} \underline{X}^k + \underline{F}. \quad (12)$$

C'est la méthode de Gauss-Seidel. Elle est de la forme (4) avec :

$$\underline{\underline{P}}_{GS} = -\underline{\underline{L}} + \underline{\underline{D}}, \quad \underline{\underline{N}}_{GS} = \underline{\underline{P}}_{GS} - \underline{\underline{A}} = \underline{\underline{U}}. \quad (1$$

Pour les composantes, elle s'écrit :

$$\text{Pour } i = 1, \ldots, m : \quad x_i^{k+1} = \frac{1}{A_{ii}} \left[f_i - \sum_{j=1}^{i-1} A_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} A_{ij} x_j^k \right] \quad ($$

Application aux poutres des (4)

$$\underline{\underline{P}}_{GS} = \begin{bmatrix} 2 & 0 \\ -1 & 2 \end{bmatrix}, \quad \underline{\underline{P}}_{GS}^{-1} = \begin{bmatrix} 1/2 & 0 \\ 1/4 & 1/2 \end{bmatrix}, \quad \underline{\underline{N}}_{GS} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\Rightarrow \underline{\underline{B}}_{GS} = \begin{bmatrix} 0 & 1/2 \\ 0 & 1/4 \end{bmatrix}$$

Les val. propres de $\underline{\underline{B}}_{GS}$ sont 0 et $\frac{1}{4}$, donc :

$$\rho(\underline{\underline{B}}_{GS}) = [\rho(\underline{\underline{B}}_J)]^2 = \frac{1}{4}.$$

La méthode de Gauss-Seidel divise l'erreur par 4 à chaque itération : on en déduit que une itération de G-S équivaut à une itération de Jacobi ; pour un coût algorithmique semblable : la méthode de Gauss-Seidel est donc plus performante.

III La méthode de relaxation (ou S.O.R)

(S.O.R. : "Successive Over Relaxation")

On rappelle que le préconditionneur de la méthode de Gauss-Seidel est (cf. Eq.(13)) :

$$\underline{\underline{P}} = \underline{\underline{D}} - \underline{\underline{L}}.$$

Une généralisation de la méthode G.S. est de considérer maintenant le préconditionneur :

$$\underline{\underline{P}}_\omega = \frac{1}{\omega}\underline{\underline{D}} - \underline{\underline{L}}, \quad \text{avec } \omega \in \mathbb{R}. \qquad (15$$

Le paramètre de relaxation (ou d'"accélération") ω devra être choisi de manière à obtenir le facteur de réduction ρ_ω le plus petit possible. On remarquera que le cas $\omega = 1$ correspond à la méthode de Gauss-Seidel

Pour obtenir une méthode consistante, on définit N_ω on

$$\underline{\underline{N}}_\omega = \underline{\underline{P}}_\omega - \underline{\underline{A}} = \underline{\underline{P}}_\omega - (\underline{\underline{D}} - \underline{\underline{L}} - \underline{\underline{U}}) = \left(\frac{1-\omega}{\omega}\right)\underline{\underline{D}} + \underline{\underline{U}}. \qquad (15$$

Avec ces définitions de P_ω et N_ω, l'itération (4) s'écrit :

$$\left(\frac{1}{\omega}\underline{\underline{D}} - \underline{\underline{L}}\right)\underline{x}^{k+1} = \left[\left(\frac{1-\omega}{\omega}\right)\underline{\underline{D}} + \underline{\underline{U}}\right]\underline{x}^k + \underline{F}.$$

Après multiplication de cette équation par ω, on obtient pour les composantes :

$$\text{pour } i=1,\dots,n: \quad x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{A_{ii}}\left(f_i - \sum_{j=1}^{i-1} A_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} A_{ij}\, x$$

Comment choisir le paramètre ω ? cela se fait en calculant, pour chaque problème $\underline{\underline{A}}\,\underline{x} = \underline{F}$, la matrice d'itération $\underline{\underline{B}}(\omega) = \underline{\underline{P}}_\omega^{-1}\underline{\underline{N}}_\omega$ et en cherchant la valeur de ω qui minimise son rayon spectral ρ_ω

On précisera le calcul de ω pour le problème test (7) plus loin dans le cours.

Le théorème suivant détermine les valeurs "permises" de ω :

Théorème : Le facteur de réduction ρ_ω de la méthode S.O. vérifie :

$$|1-\omega| \leq \rho_\omega.$$

Ainsi, pour que la méthode converge (i.e. $\rho_\omega < 1$), il faut nécessairement que :

$$0 < \omega < 2.$$

Démonstration : On note λ_i les valeurs propres de la matrice d'itération $\underline{\underline{B}}(\omega) = \underline{\underline{P}}_\omega^{-1}\underline{\underline{N}}_\omega$. Par définition du rayon spectral, on a :

$$\rho_\omega = \max_{i=1,\dots,N} |\lambda_i| \geq \sqrt[n]{\prod_{i=1}^{n} |\lambda_i|} \qquad (16)$$

Par ailleurs, on sait que :

$$\prod_{i=1}^{n} |\lambda_i| = |\det(\underline{\underline{B}}(\omega))| = \frac{|\det(\underline{\underline{N}}_\omega)|}{|\det(\underline{\underline{P}}_\omega)|}$$

Calculons ces déterminants :

- $\underline{\underline{P}}_\omega = \frac{1}{\omega}\underline{\underline{D}} - \underline{\underline{L}}$ est triangulaire inférieure, de diagonale $\frac{1}{\omega}\underline{\underline{D}}$, donc :

$$\det(\underline{\underline{P}}_\omega) = \frac{\det(\underline{\underline{D}})}{\omega^n}.$$

- De même, pour $\underline{\underline{N}}_\omega = \frac{1-\omega}{\omega}\underline{\underline{D}} + \underline{\underline{U}}$ triang. supérieure :

$$\det(N_\omega) = \left(\frac{1-\omega}{\omega}\right)^n \det(\underline{\underline{D}}).$$

Ainsi, l'inégalité (16) devient :

$$\rho_\omega \geq \sqrt[n]{|\det(\underline{\underline{B}}(\omega))|} = |1-\omega|.$$

Pour converger, il faut que $\rho_\omega < 1$, donc nécessairement ω doit être choisi tel que :

$$0 < \omega < 2.$$

Application au problème test (7)

Nous allons maintenant déterminer le choix "optimal" de $\omega \in]0,2[$ pour le problème test, qui correspond à minimiser le facteur de réduction ρ_ω de la méthode S.O.R. Avec ce choix de ω, la méthode S.O.R présentera un bien meilleur temps de convergence que la méthode de Gauss-Seidel (et donc de Jacobi).

Pour la matrice $\underline{\underline{A}} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$, on obtient : (12

$$P_\omega = \frac{1}{\omega}\underline{\underline{D}} - \underline{\underline{L}} = \begin{bmatrix} 2/\omega & 0 \\ -1 & 2/\omega \end{bmatrix}, \quad N_\omega = \left(\frac{1-\omega}{\omega}\right)\underline{\underline{D}} + \underline{\underline{U}} = \begin{bmatrix} \frac{2(1-\omega)}{\omega} & 1 \\ 0 & \frac{2(1-\omega)}{\omega} \end{bmatrix}$$

L'itération S.O.R s'écrit :

$$\begin{bmatrix} 2/\omega & 0 \\ -1 & 2/\omega \end{bmatrix} \begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \end{pmatrix} = \begin{bmatrix} \frac{2(1-\omega)}{\omega} & 1 \\ 0 & \frac{2(1-\omega)}{\omega} \end{bmatrix} \begin{pmatrix} x_1^k \\ x_2^k \end{pmatrix} + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

La matrice d'itération de la méthode s'écrit :

$$B_\omega = P_\omega^{-1} N_\omega = \begin{bmatrix} 1-\omega & \omega/2 \\ \frac{\omega(1-\omega)}{2} & \frac{1}{4}\omega^2 + 1 - \omega \end{bmatrix}$$

Les valeurs propres de la matrice B_ω sont :

$$\lambda_{+,-} = 1 - \omega + \frac{\omega^2}{8} \pm \frac{\omega}{8}\sqrt{p(\omega)}$$

avec $p(\omega) = \omega^2 - 16\omega + 16 = (\omega - \omega_0)(\omega - \omega_1)$

tel que $\omega_1 = 8 + 4\sqrt{3} > 2$ et $\omega_0 = 8 - 4\sqrt{3} \simeq 1{,}07$.

On remarque que $p(\omega)$ change de signe dans $]0, 2[$: les valeurs propres λ_+ et λ_- peuvent donc être complexes.

Ainsi, pour calculer $\rho_\omega = \max\{|\lambda_+|, |\lambda_-|\}$, il faut considérer 2 cas :

1er Cas : Pour $\omega \in]0, \omega_0[$, on a $p(\omega) > 0$, les valeurs propres sont donc réelles et on trouve facilement :

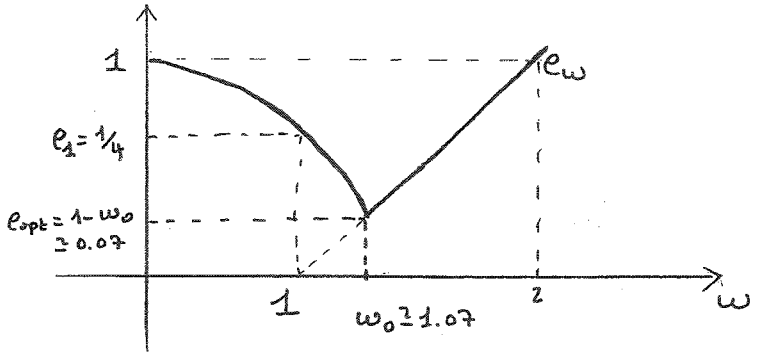
$$\rho_\omega = \frac{\omega^2}{8} - \omega + 1 + \frac{\omega}{8}\sqrt{\omega^2 - 16\omega + 16}$$

propres sont des complexes conjugués. On obtient :

$$\rho_\omega = |\lambda_+| = |\lambda_-| = \sqrt{\left(1-\omega+\frac{\omega^2}{8}\right)^2 + \frac{\omega^2}{64}(-p(\omega))}$$

$$= \omega - 1 \qquad \text{après calcul.}$$

On en déduit le graphe de ρ_ω (pas à l'échelle !)



En premier lieu, on retrouve bien que :

* Pour $\omega \leq 0$ et $\omega \geq 2$, on a $\rho_\omega \geq 1 \Rightarrow$ la méthode div

* Pour $\omega = 1$, on retrouve le facteur de réduction de la méthode de Gauss-Seidel :

$$\rho_1 = \rho_{GS} = \frac{1}{4}$$

Graphiquement, on observe que le minimum de ρ_ω est obtenu en $\omega = \omega_0 \simeq 1.07$. Le facteur de réduction optimal de la méthode S.O.R est donc :

$$\rho_{opt} = \rho_{\omega_0} = \qquad \omega_0 - 1 \simeq 0.07.$$

De plus, on remarque que :

$$\rho_{opt} \simeq (\rho_{GS})^2$$

ce qui montre qu'une itération de S.O.R avec $\omega = \omega_{opt}$:

- * équivaut à 2 itérations de G.-S.
- * équivaut à 4 itérations de Jacobi.

En conclusion de ce paragraphe :

* Par un choix intelligent du paramètre ω, et donc des valeurs propres de la matrice d'itération, on peut bâtir une méthode itérative performante.

* Dans le cas d'une matrice $\underline{\underline{A}}$ générale, le calcul de ω_{opt} peut être très compliqué. Heureusement, pour les systèmes lin. issus de la mécanique, il existe des théorèmes mathématiques pour déterminer ω_{opt} facilement (cf TD.).

IV Résultats de convergence pour les méthodes itératives de base

Supposons que vous souhaitiez résoudre le système $\underline{\underline{A}}\,\underline{X} = \underline{f}$, avec $\underline{\underline{A}}$ quelconque, en utilisant la meth. de Jacobi, de G.-S. ou S.O.R. Afin de s'assurer que la méthode que vous choisissez soit convergente, il faudrait calculer son facteur de réduction :

$$\rho(\underline{\underline{B}}) = \rho(\underline{\underline{P}}^{-1}\underline{\underline{N}}) = \rho(\underline{\underline{I}} - \underline{\underline{P}}^{-1}\underline{\underline{A}})$$

et vérifier qu'il est strictement inférieur à 1. Le calcul est, en général, très fastidieux. Heureusement, pour certaines matrices spéciales (que l'on retrouve en mécanique), nous disposons de théorèmes bien utiles qui assurent la convergence de nos méthodes itératives.

La première classe de matrices spéciales regroupe les matrices SDP (rappel: une matrice symétrique est définie positive si ses val. propres sont strictement positives). On dispose d'un critère bien pratique pour déterminer si une matrice est SPD:

Critère de Sylvester: Une matrice $\underline{\underline{A}}$ symétrique est définie positive si

$$\det(\underline{\underline{A}}_k) > 0, \quad k = 1, \dots, n$$

où $\underline{\underline{A}}_k = (\underline{\underline{A}}_{ij})_{i,j=1,\dots,k}$ est la sous matrice principale d'ordre k de $\underline{\underline{A}}$.

Exemple: Considérons la matrice $\underline{\underline{A}} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$.

On calcule ses sous-matrices principales:

$$\underline{\underline{A}}_1 = [2] \Rightarrow \det(\underline{\underline{A}}_1) = 2 > 0$$

$$\underline{\underline{A}}_2 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \Rightarrow \det(\underline{\underline{A}}_2) = 3 > 0$$

$$\underline{\underline{A}}_3 = \underline{\underline{A}} \Rightarrow \det(\underline{\underline{A}}_3) = 4 > 0.$$

Donc, d'après le critère de Sylvester, elle est SDP (on le savait déjà, cf. TD).

Pour les matrices SDP, on dispose du théorème:

Théorème: Si $\underline{\underline{A}}$ est SDP alors la méthode de G-S et la méthode S.O.R (pour $\omega \in]0, 2[$) convergent.

Démo: (Admise)

Rq: Il n'y a aucun résultat de convergence pour les matrices SDP avec la méthode de Jacobi. Dans certains cas, cette méthode converge (p. ex. le plus test (7)) dans d'autre, elle diverge (p. ex. la matrice de Hilbert $H_{ij} = \frac{1}{i+j-1}$ d'ordre $n = 3$, qui est SDP, vérifie $\rho_J > 1$)

Une autre classe intéressante de matrices regroupes les matrices à diagonale dominante:

Définition: Une matrice $\underline{\underline{A}}$ est dite **diagonale dominante stricte par ligne** si:

$$|A_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}|, \quad \forall i = 1, \dots, n$$

Exemple: Considérons la matrice d'ordre 3:

$$\underline{\underline{A}} = \begin{bmatrix} -3 & 2 & 0 \\ 1 & 4 & -2 \\ 1 & -1 & 3 \end{bmatrix}$$

pour la ligne 1: $|-3| > 2$, ligne 2: $|4| > |-2| + |1|$, ligne 3: $|3| > |1| + |$
donc cette matrice est strictement diagonale dominante par ligne.

Théorème: Si $\underline{\underline{A}}$ est diagonale dominante stricte par ligne, alors les méthode de Jacobi et Gauss-Seidel convergent, et la méthode S.O.R converge si $\omega \in]0, 1]$.

La méthode de Richardson

Dans les sections précédentes, on a établi que les méthodes itératives de base (Jacobi, G-S, S.O.R) s'écrivaient sous la forme (4) :

$$\underline{\underline{P}}\,\underline{X}^{k+1} = \underline{\underline{N}}\,\underline{X}^{k} + \underline{F}, \qquad (17)$$

avec $\underline{\underline{N}} = \underline{\underline{P}} - \underline{\underline{A}}$, et la matrice d'itération s'écrit :

$$\underline{\underline{B}}_1 = \underline{\underline{P}}^{-1}\underline{\underline{N}} = \underline{\underline{I}} - \underline{\underline{P}}^{-1}\underline{\underline{A}}.$$

Il est intéressant d'écrire (17) sous la forme :

$$\underline{\underline{P}}\,\underline{X}^{k+1} = \underline{\underline{P}}\,\underline{X}^{k} + \underline{R}^{k}, \qquad (18)$$

avec R^k le résidu de l'équation à la $k^{ième}$ itération, défini par :

$$\underline{R}^{k} = \underline{F} - \underline{\underline{A}}\,\underline{X}^{k}. \qquad (19)$$

Une généralisation de (18) consiste à introduire un paramètre d'accélération $\alpha \neq 0$, tel que :

$$\underline{\underline{P}}\,\underline{X}^{k+1} = \underline{\underline{P}}\,\underline{X}^{k} + \alpha\,\underline{R}^{k}. \qquad (20)$$

Cette itération est appellée méthode de Richardson.
Sa matrice d'itération a la forme :

$$\underline{\underline{B}}_\alpha = \underline{\underline{I}} - \alpha\,\underline{\underline{P}}^{-1}\underline{\underline{A}} \qquad (21).$$

Le paramètre $\alpha \neq 0$ doit être choisi de manière à accélérer la convergence de l'itération (18), soit :

$$\rho(\underline{\underline{B}}_\alpha) < \rho(\underline{\underline{B}}_1).$$

Comme on l'a vu en TD, il est possible de calculer la valeur optimale de α à partir des val. propres de $\underline{\underline{P}}^{-1}\underline{\underline{A}}$:

Théorème : Si $P^{-1}A$ possède des val. propres positives strictement, notées $\lambda_1 > \lambda_2 > \dots > \lambda_n > 0$, alors le paramètre α_{opt} qui minimise $\rho(B_\alpha)$ est :

$$\alpha_{opt} = \frac{2}{\lambda_1 + \lambda_n}$$

et la méthode de Richardson a pour facteur de réduction

$$\rho_{opt} = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} < 1.$$

La méthode (21), avec $\alpha = \alpha_{opt}$, appellée méthode de Richardson à pas fixe, est un premier exemple d'accélérateurs de convergence.

Dans les années suivantes, vous verrez que la plupart des méthodes itératives modernes (ie, utilisées dans les logiciels commerciaux de calcul scientifique sera pour la mécanique), sont la combinaison de :

* Un préconditionneur $\underline{\underline{P}}$: (ex: Jacobi, S.O.R., ou multi-grilles)
* Un accélérateur de convergence : basé sur des améliorations de la méthode de Richardson (méthode du Gradient Conjugué, G.M.R.E.S, etc...)

Pb 1 bis On va utiliser ce théorème pour accélérer la méthode S.O.R., définie par $P_{SOR}(\omega_0) = \begin{bmatrix} \frac{2}{\omega_0} & 0 \\ -1 & \frac{2}{\omega_0} \end{bmatrix}$

et $\omega_0 = 8 - 4\sqrt{3}$ la valeur optimale du chap. III.

Pour A pris calculé, on trouve que :

$$P^{-1}A = \begin{bmatrix} \omega_0 & -\frac{1}{2}\omega_0 \\ \frac{1}{2}\omega_0(\omega_0 - 1) & \omega_0\left(1 - \frac{\omega_0}{4}\right) \end{bmatrix}$$

et les val. propres de cette matrice sont doubles :

$$\lambda_1 = \lambda_2 = \left(1 - \frac{1}{8}\omega_0 + \sqrt{16 - 16\omega_0 + \omega_0^2}\right)\omega_0 .$$

Pour la valeur optimale $\omega_0 = 8 - 4\sqrt{3}$, on trouve :

$$\lambda_1 = \lambda_2 = 4\sqrt{3} - 6 > 0 .$$

Les v.p. de $P^{-1}A$ sont positives, on peut appliquer le théorème pour trouver le paramètre d'accélération optimal :

$$\alpha = \frac{2}{\lambda_1 + \lambda_2} = \frac{2\sqrt{3} + 3}{6} \simeq 1.077 .$$

Le facteur de réduction est alors :

$$\rho_\alpha = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} = 0$$

ce qui est encore meilleur que $\rho_1 = \omega_0 - 1 \simeq 0.97$ de la méthode S.O.R.

R.Q. : L'application de ~~la méthode de Richardson~~ cet accélération de CV est moins rapidement ses limites : il faut que les v.p. de $P^{-1}A$ soient positives, ce qui n'est pas le cas pour le problème test avec $n \geq 2$. Dans ces années ultérieures, vous verrez des accélérations de CV plus sophistiquées (méthodes du gradient).

V-1 Critère d'arrêt

Théoriquement, les méthodes itératives donnent la solution exacte d'un système linéaire lorsque $k \to \infty$. En pratique, on se satisfait d'arrêter l'algorithme lorsque ~~l'erreur relative est~~ nous jugeons que l'erreur relative est suffisamment petite, par exemple lorsque :

$$\frac{\|X^k - X\|}{\|X\|} < \delta \qquad (22)$$

avec $\delta = 10^{-2}$ (i.e., 1% d'erreur relative).

Puisque l'on ne connaît pas la solution exacte, on effectue généralement ce test d'arrêt sur le résidu :

$$\frac{\|R^k\|}{\|F\|} < \varepsilon, \qquad (23)$$

où $\varepsilon > 0$ est à choisir. Afin de bien choisir la tolérance nous allons relier ce test d'arrêt à (22). D'après la définition du résidu (19), on obtient :

$$A^{-1} R^k = A^{-1} F - X^k = X - X^k,$$

d'où on déduit l'inégalité :

$$\frac{\|X - X^k\|}{\|X\|} \leq \|\|A^{-1}\|\| \, \frac{\|R^k\|}{\|X\|} \qquad (24)$$

D'autre part, on tire de $F = AX$ l'inégalité

$$\|F\| \leq \|\|A\|\| \, \|X\|,$$

et l'inégalité (24) devient finalement

$$\frac{\|X - X^k\|}{\|X\|} \leq \chi(A) \, \frac{\|R^k\|}{\|F\|} \qquad (25)$$

où $\chi(A)$ est le nombre de conditionnement de A.

Ainsi, pour obtenir une erreur relative sur la solution de l'ordre de $\delta = 1\%$, il suffit d'utiliser le critère d'arrêt

avec :

$$\varepsilon = \frac{\delta}{\chi(A)}. \qquad (26)$$

Contrairement aux méthodes directes, une méthode itérative permet de contrôler les erreurs d'arrondis ~~effectuées~~ de la solution numérique (Cf Fig. 1 cours sur méthodes directes).

V-2 Algorithme pratique

On souhaite maintenant programmer l'itération de Richardson (20) sur un ordinateur. Pour cela, il est commode de la réécrire sous la forme :

$$P \, \delta X = \alpha \, R^k, \quad \text{avec} \quad \delta X = X^{k+1} - X^k$$

On l'implémente par l'algorithme suivant :

- **Initialisation** : calculer le résidu $R^0 = F - AX^0$
  - On choisit le nombre d'itérations maximales, par exemple : $k_{max} = 100$.
  - On choisit la tolérance ε, par exemple avec le critère (26).
  - On choisit une cond. initiale $X^0 \in \mathbb{R}^n$.
  - On calcule le résidu initial $R^0 = F - AX^0$.

- **Itérations** : Pour $k = 0, \ldots, k_{MAX} - 1$:
  - Résoudre $P \, \delta X = \alpha R^k$
  - Calculer $X^{k+1} = X^k + \delta X$
  - Calculer $R^{k+1} = F - A X^{k+1}$
  - Test d'arrêt : on stoppe l'algo si $\|R^{k+1}\| < \varepsilon$

Analyse Numérique - Licence de Mécanique

TD 1 - Interpolation - Intégration numérique

Exercice 1. ***(Interpolation de Lagrange - Phénomène de Runge - Applications à l'intégration et différentiation numérique)***

Nous cherchons à interpoler une fonction $f(x)$ par un polynôme de degré $n = 2$ aux points $x_0 = -1$, $x_1 = 0$ et $x_2 = 1$.

1) Calculez les éléments $\{L_j(x)\}$ de la base de Lagrange. Représentez-les sur un graphe.
2) On veut interpoler une fonction quelconque $f(x)$. Ecrivez la forme générale du polynôme d'interpolation de Lagrange, et ordonnez-le en puissances de x.
3) Interpolez les fonctions :

$$f(x) = 1, \qquad f(x) = x, \qquad f(x) = x^2, \qquad f(x) = x^3.$$

Que remarquez-vous. Pouviez-vous prévoir théoriquement ces résultats ?

4) Interpolez la fonction $f(x) = \mathrm{e}^x$ dans $\Omega =]-1, 1[$ et représentez sur le même graphe cette fonction et son interpolant $\pi_2 f(x)$. Estimez l'erreur relative d'interpolation commise au point $x = 0.5$.
5) Même question pour la fonction de Runge définie dans $\Omega =]-1, 1[$ par :

$$f(x) = \frac{1}{1 + 100x^2}.$$

(Les interpolants de degré supérieur à 2 sont représentés sur la figure 1.)

6) Utilisez la question 2) pour bâtir une formule approchée permettant de calculer l'intégrale :

$$I(f) = \int_{-1}^{1} f(t)\ \mathrm{d}t.$$

Donnez une signification géométrique à cette formule. Utilisez-la pour calculer une valeur approchée de $I(\mathrm{e}^x)$ et comparez avec la solution exacte.

7) A l'aide du changement de variables

$$x = \varphi(t) \equiv a + (b - a)\,\frac{t + 1}{2},$$

utilisez la question précédente pour établir une approximation de l'intégrale

$$\int_a^b g(x)\ \mathrm{d}x.$$

(Cette formule de quadrature, utilisant une interpolation parabolique, est connue sous le nom de *formule de Simpson*).

8) On souhaite maintenant développer une approximation discrète de $f''(x)$ en un point $x = x_i$ qui utilise les valeurs de f aux points équidistants :

$$x_{i-1} = x_i - h, \quad x_i, \quad x_{i+1} = x_i + h, \qquad \text{avec } h > 0.$$

Pour cela, construisez le polynôme d'interpolation $\pi_2 f(x)$ passant par ces trois points, dérivez-le deux fois, et montrez que vous obtenez la *formule aux différences* :

$$f''(x_i) \cong \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2}.$$

Pourquoi est-elle communément appellée formule centrée ?

Exercice 2. ***(Estimation de l'ordre des erreurs numériques)***

Soit I_h une formule de quadrature composite de pas h, appliquée à l'intégration de $I(f)$. On suppose que I_h est une formule d'ordre q, et donc l'erreur commise avec un pas d'intégration h est telle que :

$$E_h = |I_h - I| \sim Ch^q, \tag{1}$$

lorsque h est suffisamment petit. Dans de nombreuse situations, on est amené à vérifier *a posteriori* que l'ordre de précision q est atteint, par exemple pour vérifier qu'il n'y a pas d'erreur dans un programme informatique. Cet exercice a pour but d'évaluer l'ordre de précision d'une méthode à partir des résultats numériques obtenus sur un problème test.

1) Le rapport entre l'erreur commise avec un pas h et un pas réduit de moitié est appelé facteur de réduction (FdR). Montrez que pour une méthode d'ordre q, le FdR s'écrit :

$$\text{FdR} = 2^q.$$

En déduire l'intérêt que présente une méthode d'ordre élevé (p.ex. $q = 4$) par rapport à une méthode basique (p.ex. $q = 2$).

2) Application : pour calculer l'intégrale

$$I(f) = \int_0^{2\pi} x\mathrm{e}^x \cos 2x \ \mathrm{d}x,$$

on a utilisé deux formules de quadratures pour des valeurs décroissantes de h. Les résultats sont reportés dans le tableau 1. Complétez ce tableau en calculant le FdR de chacune des 2 méthodes, et déduisez-en leur ordre de précision q. L'une de ces méthodes est la quadrature du rectangle, l'autre la quadrature de Simpson : pouvez-vous les identifier ?

h	Erreur, I_h^A	FdR	Erreur, I_h^B	FdR
$2\pi/10$	1.83×10^{-2}	–	4.66×10^{-4}	–
$2\pi/20$	4.26×10^{-3}		3.05×10^{-5}	
$2\pi/40$	1.05×10^{-3}		1.92×10^{-6}	
$2\pi/80$	2.60×10^{-4}		1.20×10^{-7}	

TAB. 1 – Erreur E_h commises par les formules de quadrature I_h^A et I_h^B.

Exercice 3. ***(Extrapolation de Richardson - Quadrature de Romberg)***

Pour le calcul approché de l'intégrale

$$I = \int_a^b f(x) \ \mathrm{d}x,$$

on va développer des solutions numériques de haute précision en se basant sur le principe de l'extrapolation de Richardson.

L'extrapolation de Richardson est une technique générale permettant d'obtenir une estimation numérique précise à partir de deux (ou plusieurs) solutions données par une méthode basique, de faible précision. La méthode de base que nous utiliserons est la quadrature du trapèze, notée $I_{1,h}$. L'ingrédient essentiel de cette extrapolation repose sur la connaissance de la forme de l'erreur de troncature de la méthode de base, ici :

$$I = I_{1,h} + c_1 h^2 + c_2 h^4 + c_3 h^6 + \cdots . \tag{1}$$

Il est alors possible de combiner deux calculs obtenus avec la quadrature du trapèze pour éliminer le terme en $\mathcal{O}(h^2)$ et obtenir une estimation d'ordre supérieur.

1) Ecrivez l'erreur de troncature de la formule du trapèze pour le pas $h/2$. Effectuez une combinaison linéaire de cette équation avec (1) pour éliminer le terme d'erreur du second-ordre. Montrez qu'on définit alors une nouvelle formule de quadrature, notée $I_{2,h}$, de la forme :

$$I_{2,h} = \frac{4I_{1,h/2} - I_{1,h}}{3}, \tag{2}$$

dont l'erreur de troncature s'écrit

$$I = I_{2,h} - \frac{c_2}{4}h^4 - \frac{5}{16}c_3h^6 + \cdots . \tag{3}$$

Calculez l'expression analytique de cette nouvelle formule de quadrature, et montrez qu'on retrouve la quadrature de Simpson. En déduire que la formule de quadrature de Simpson est précise au quatrième ordre.

2) Appliquez de nouveau le principe de l'itération de Richardson à $I_{2,h}$ pour obtenir la formule du sixième ordre :

$$I_{3,h} = \frac{16I_{2,h/2} - I_{2,h}}{15}. \tag{4}$$

Le principe d'extrapolation peut être répété de nouveau pour éliminer le terme en $\mathcal{O}(h^6)$, etc .... Cet algorithme est alors connu sous le nom d'intégration de Romberg.

3) Application : utilisez la quadrature de Romberg pour calculer l'intégrale

$$I = \int_1^\pi \frac{\sin x}{2x^3}\,\mathrm{d}x = 0.198557\ldots \tag{5}$$

La première colonne du tableau 2 donne les résultats de l'intégrale du trapèze sur un nombre croissants d'intervalles n. Complétez ce tableau en utilisant les extrapolations développées dans les questions précédentes. Vérifiez la convergence des approximations vers le résultat exact.

n			
2	$I_{1,h} = 0.278173$		
4	$I_{1,h/2} = 0.220713$	$I_{2,h}$	
8	$I_{1,h/4} = 0.204304$	$I_{2,h/2}$	$I_{3,h}$
16	$I_{1,h/8} = 0.200009$	$I_{2,h/4}$	$I_{3,h/2}$

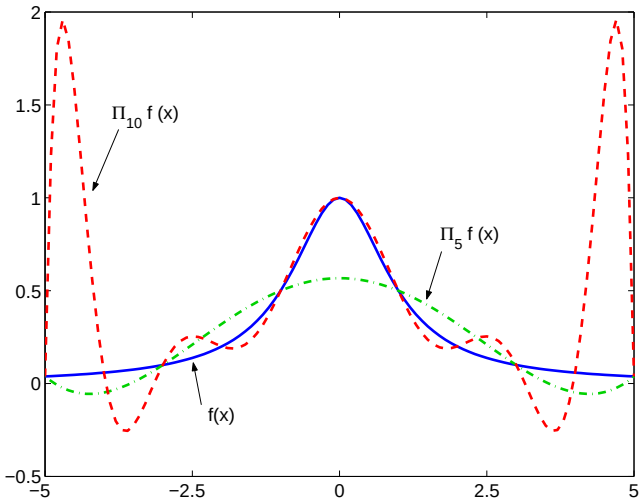
TAB. 2 – Résultats de l'approximation de l'intégrale (5).

4) Ecrivez la procédure *Maple* `trapeze=proc(a,b,n,f)` qui calcule l'intégrale $\int_a^b f(x)\mathrm{d}x$ par la formule du trapèze composite avec n sous intervalles de taille uniforme[1]. Utilisez-la avec Maple pour retrouver les résultats de la question précédente.

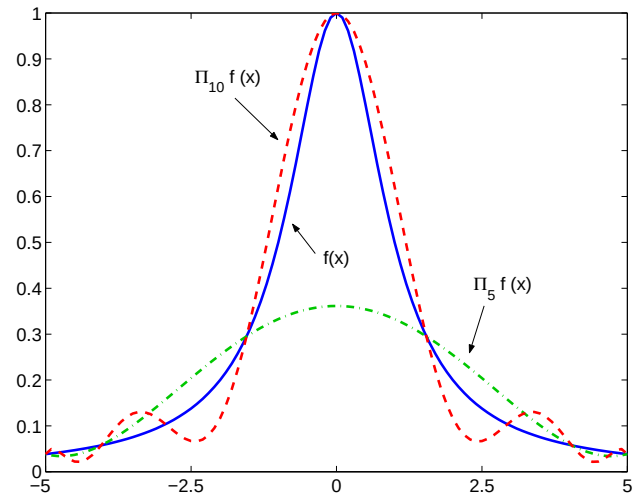
[1] On rappelle que, d'après le cours, cette formule s'écrit :

$$I_h = \frac{h}{2}\sum_{k=0}^{n-1}\left[\,f(x_k) + f(x_{k+1})\,\right],$$

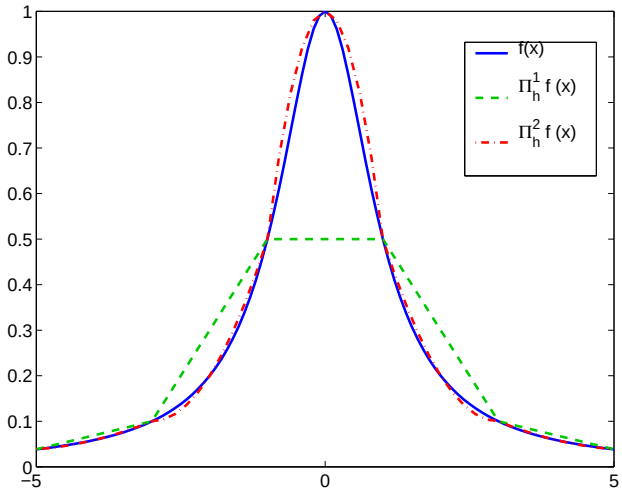
avec $h = (b-a)/n$ et $x_k = a + kh$.



(a)



(b)



(c)

FIG. 1 – Interpolation de la function de Runge $f(x) = \frac{1}{1+20x^2}$ dans $\Omega =]-5, 5[$. Dans *(a)*, l'interpolant global $\pi_n f$ qui utilise $n+1$ nœuds équidistants présente des oscillations qui augmentent avec le degré n du polynôme. Dans *(b)*, les oscillations sont fortement atténuées lorsque les points de Tchebychev ($x_i = -5\cos i\pi/n$, $i = 0, \ldots, n$) sont utilisés. Dans *(c)*, les oscillations ont disparues en utilisant l'interpolant $\pi_n^h f$ sur 5 sous-intervalles de longueur $h = 2$.

ANALYSE NUMERIQUE EX1 du TD1

Les Graphes

```
>
```

```
> L0:=x->x^2/2-x/2;
```

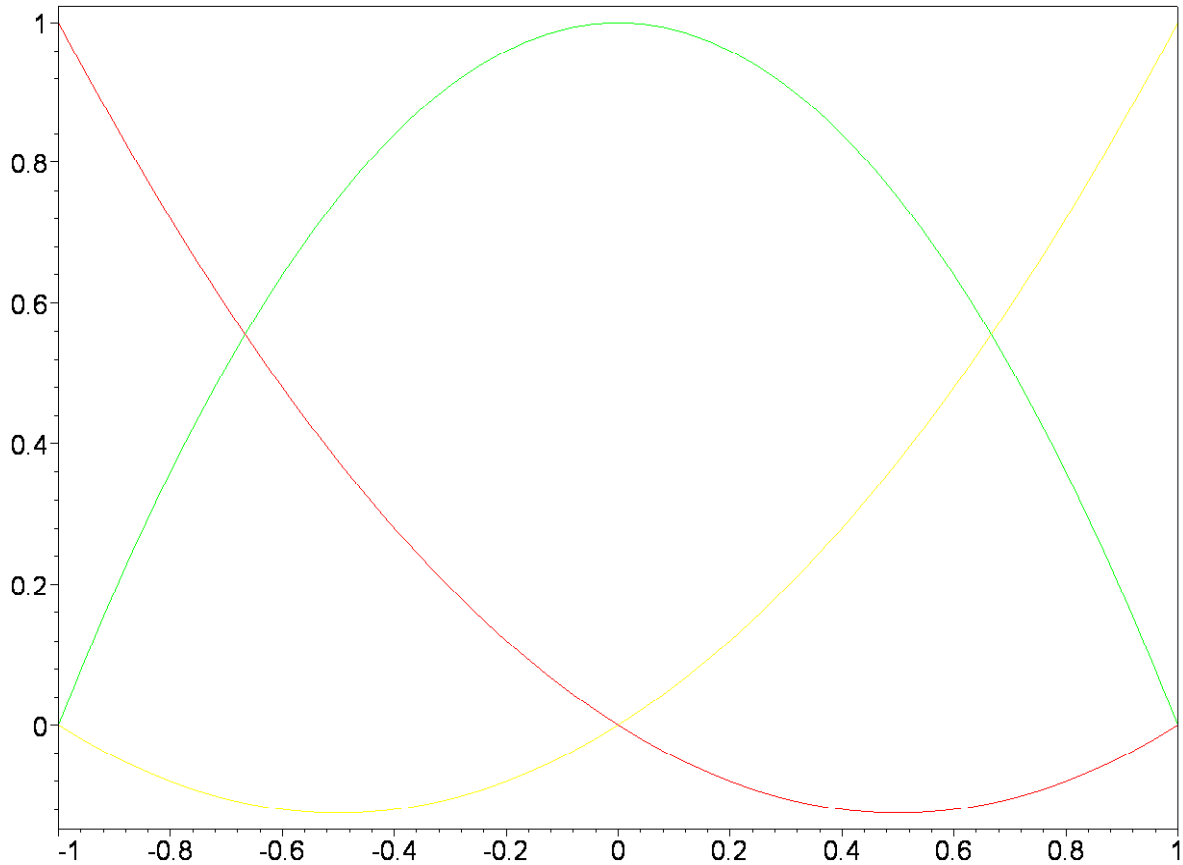
```
> L1:=x->1-x**2;
```

$$L1 := x \to 1 - x^2$$

```
> L2:=x->x^2/2+x/2;
```

$$L2 := x \to \frac{1}{2}x^2 + \frac{1}{2}x$$

```
> plot({L0,L1,L2},-1..1);
```



Elements de la base de Lagrange (question 1)

```
> a0:=exp(-1.);a1:=exp(0.);a2:=exp(1.);
```

$$a0 := .3678794412$$

$$a1 := 1.$$

$$a2 := 2.718281828$$

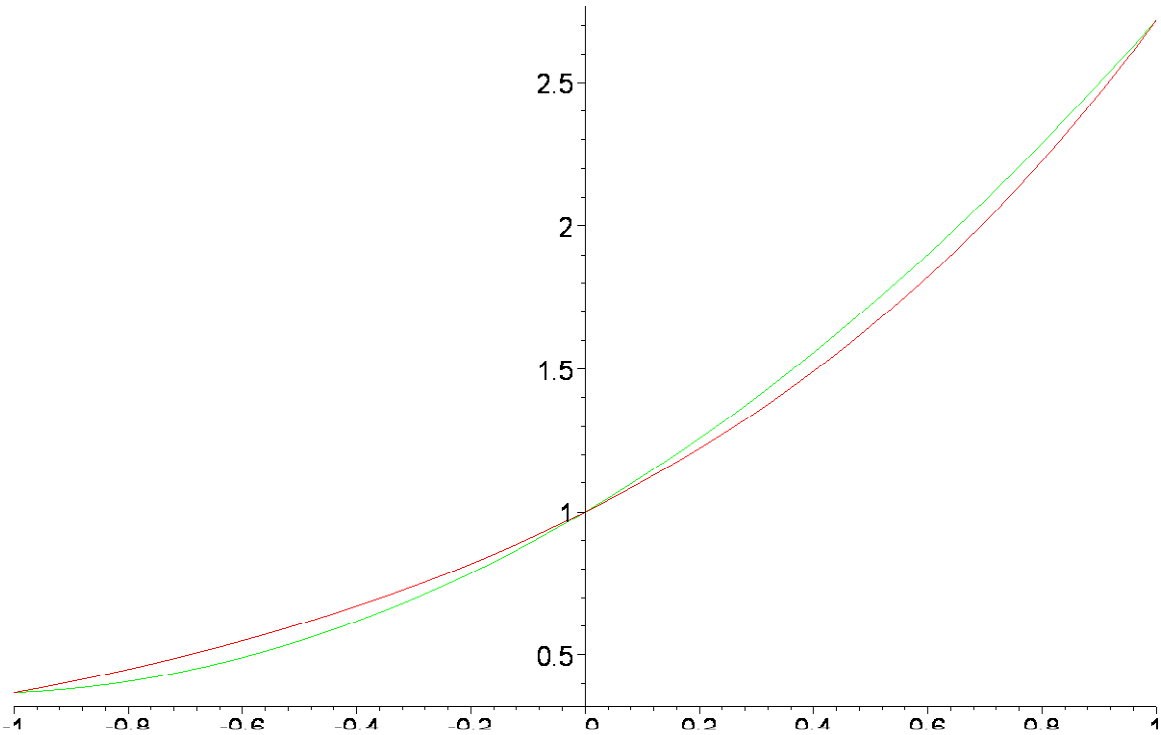
```
> pol:=x->a0*L0(x)+a1*L1(x)+a2*L2(x);
```

$$pol := x \to a0\ \mathrm{L0}(x) + a1\ \mathrm{L1}(x) + a2\ \mathrm{L2}(x)$$

```
> (pol(0.5)-exp(0.5))/exp(0.5);
```

$$.04527720138$$

```
> plot({exp,pol},-1..1);
```



function exp(x) et son interpolant de Lagrange (Q. 4)

```
> runge:=x->1/(1+100*(x-0.)^2);a0:=runge(-1.);a1:=runge(0.);a2:=ru
  nge(1.);
```

$$runge := x \to \frac{1}{1 + 100\,x^2}$$

$$a0 := .009900990099$$

$$a1 := 1.$$

$$a2 := .009900990099$$

```
> pol:=x->a0*L0(x)+a1*L1(x)+a2*L2(x);
> pol(x);
```

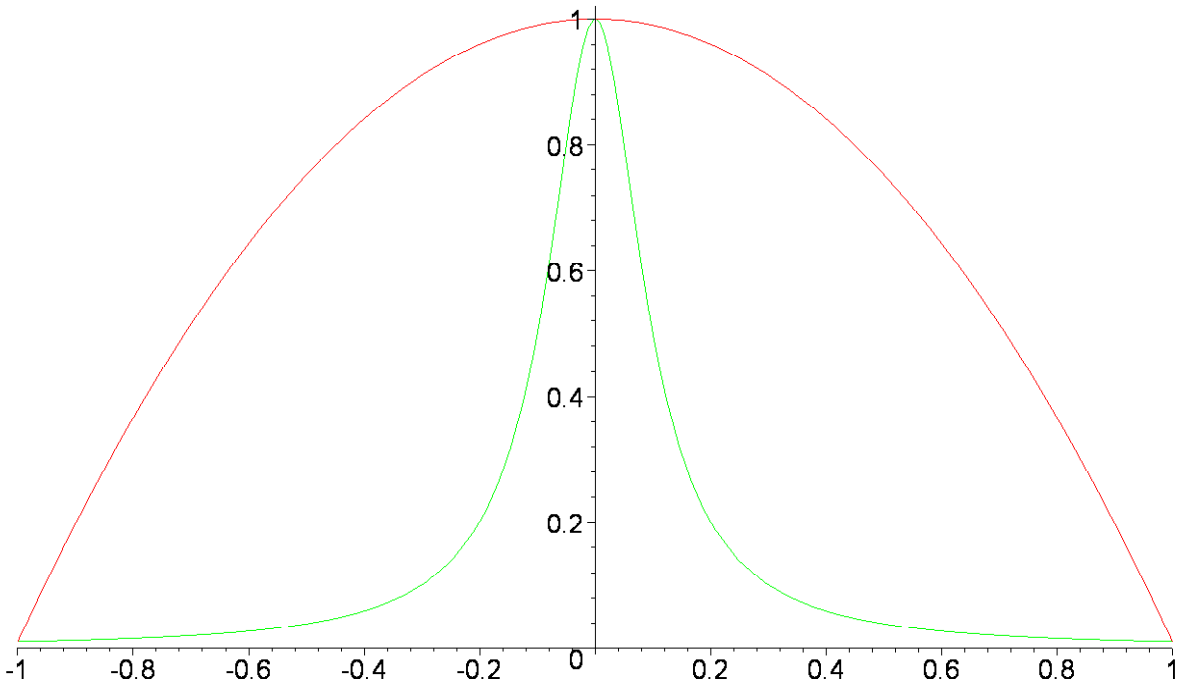
$$pol := x \to a0\,\mathrm{L0}(x) + a1\,\mathrm{L1}(x) + a2\,\mathrm{L2}(x)$$

$$-.9900990099\,x^2 + 1.$$

```
> (pol(0.5)-runge(0.5))/runge(0.5);
```

$$18.56435643$$

```
> plot({runge,pol},-1..1);
```



Function de Runge et son interpolant de Lagrange (Q. 5)

>

Analyse Numérique - Licence de Mécanique

TD 2 - Equations Non-Linéaires

Exercice 1. ***(Quelques itérations du point fixe)***

On veut résoudre le problème

$$\text{Trouver la racine } x_\star \text{ de la fonction} \quad f(x) = \sin 2x - 1 + x, \tag{1}$$

par diverses méthodes itératives. Le graphe de f est représenté sur la figure 1 : la racine $x_\star \simeq 0.35$ est unique.

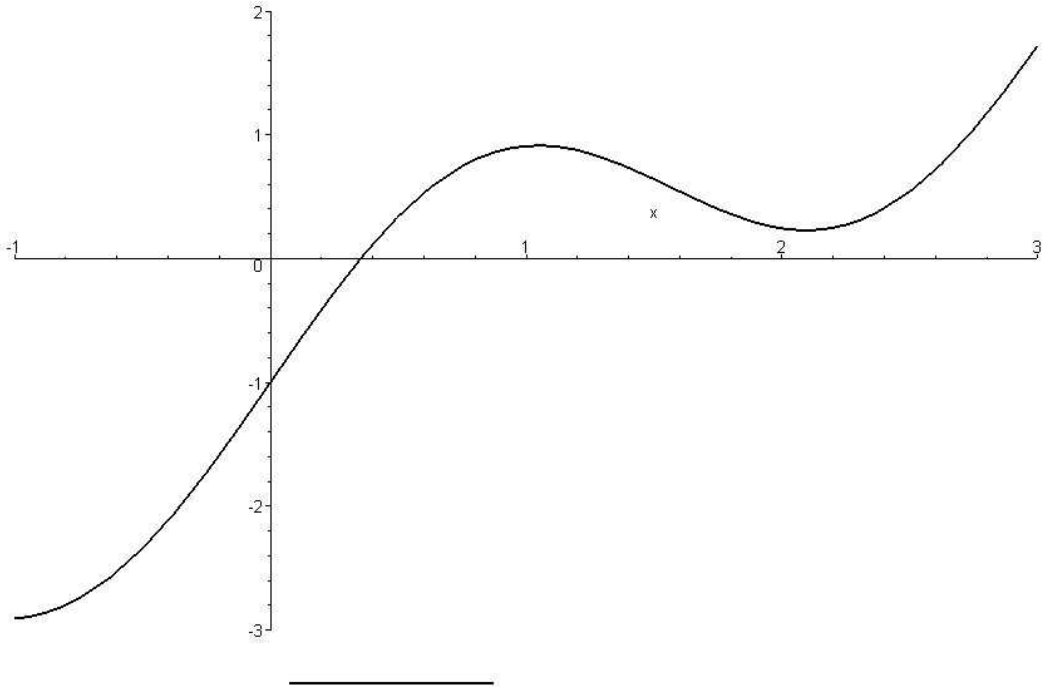


Figure 1: Graphe de la fonction $f(x) = \sin 2x - 1 + x$.

1) La méthode de bissection va-t-elle converger ? Proposez un intervalle I_0 pour l'initialiser.

2) Pour $i = 1$ et 2, on considère les méthodes itératives

$$\begin{cases} x_0 \in \mathbb{R}, \\ x_{k+1} = \phi_i(x_k), \end{cases}$$

avec $\phi_1(x) = 1 - \sin(2x)$, et $\phi_2(x) = \frac{1}{2}\arcsin(1-x)$ pour $x \in]0, 2[$.

   a) A quelle classe de méthodes appartiennent-elles ? Sont-elles consistantes avec le problème (1) ?

   b) En utilisant la figure 2, construisez géométriquement les 4 premiers itérés de chacune des méthodes à partir de $x_0 = 0$. Que remarquez-vous ?

   c) Etudiez la stabilité de ces méthodes (conseil : utilisez la figure 3). Pouvez-vous prédire les comportements observés dans la question précédente ?

3) Ecrivez la méthode de Newton pour résoudre (1). Pour ce problème, quel est l'ordre de convergence de cette méthode ? En partant de $x_0 = 0$, construisez géométriquement les premiers itérés de la méthode de Newton. Même question pour $x_0 = 0.9$.

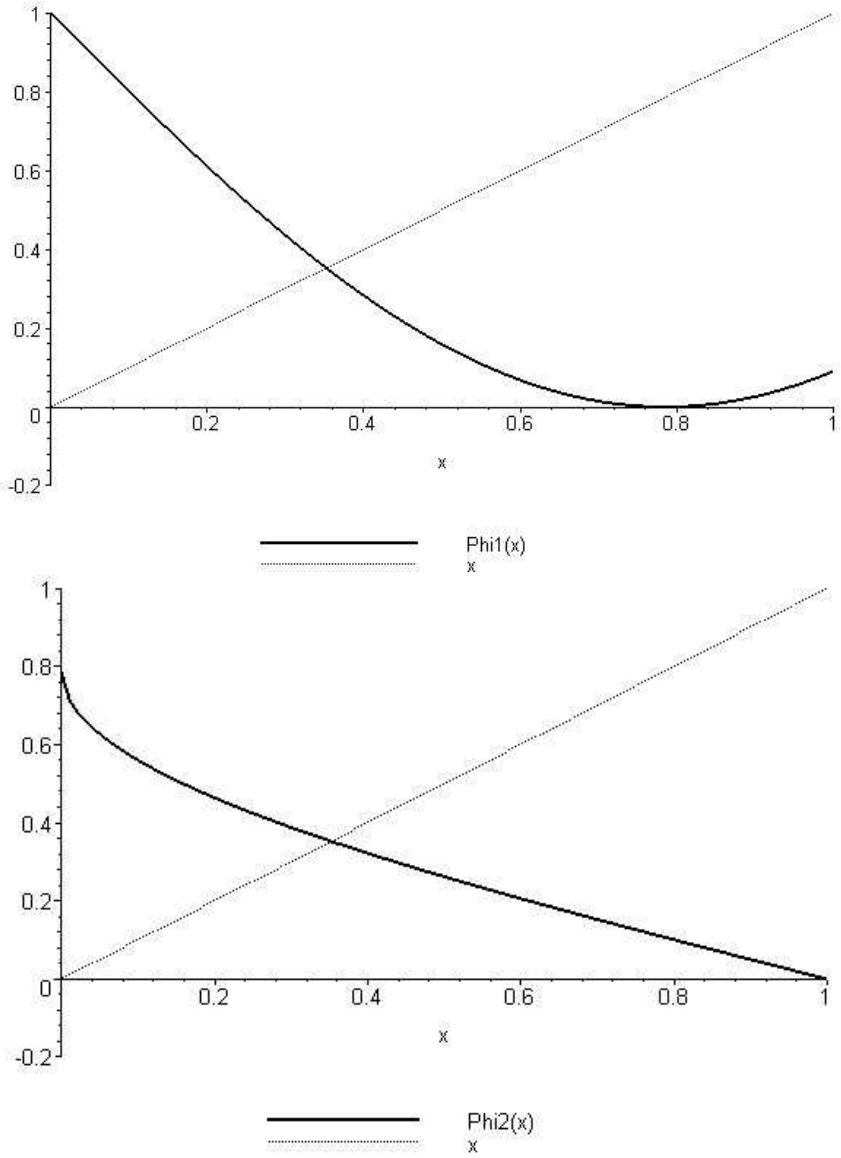


Figure 2: Graphe des fonctions d'itération $\phi_1(x)$ et $\phi_2(x)$.

4) Dans l'algorithme de Newton, la partie la plus coûteuse en temps de calcul est l'évaluation de $f'(x_k)$ à chaque itération[1]. Pour cela, il peut se révéler judicieux de remplacer $f'(x_k)$ par une approximation bien choisie : on obtient alors des méthodes de Newton modifiées ou quasi-Newton. Le choix

$$\forall\, k \geq 1, \qquad f'(x_k) \cong f'(x_0),$$

correspond à la *méthode de la corde*, ainsi appellée car elle utilise une approximation linéaire de $f(x_{k+1})$ avec toujours la même pente $f'(x_0)$.

   a) Ecrivez l'algorithme de la corde. A quelle classe de méthodes appartient-elle ? Cet algorithme est-il consistant avec le problème (1) ?

   b) Etudiez la stabilité de cet algorithme. En déduire que sa convergence est au plus d'ordre 1. Que se passe-t-il pour $x_0 = 0$ et $x_0 = 0.9$? Construisez les premiers itérés pour ces conditions initiales.

[1] C'est encore plus vrai pour des systèmes d'équations non-linéaires dans $\mathbb{R}^n$.

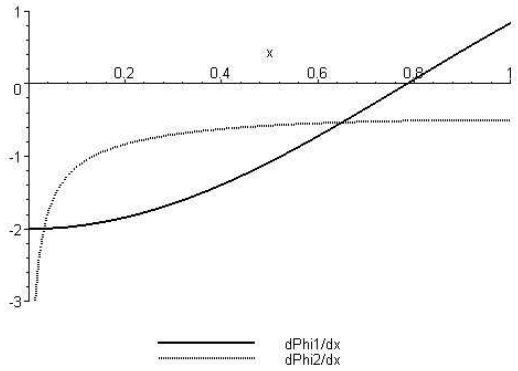


Figure 3: Graphe de la dérivée première des fonctions d'itération.

Exercice 2. ***(Performance des méthode de bissection et de Newton)***

On cherche le(s) zéro(s) de $f(x) = \ln(x+1) - 2$ sur l'intervalle $I_0 = [4, 12]$.

1) Montrer que cette fonction admet un zéro $x_\star$ dans I_0, et qu'il est unique.

2) En utilisant la méthode de la bissection dans I_0, estimer le nombre minimal d'itérations nécessaire pour calculer $x_\star$ avec la tolérance $\epsilon = 10^{-1}$ et 10^{-10}.

3) On considère la méthode du point fixe définie par sa fonction d'itération :

$$\phi(x) = x - \big(x+1\big)\big(\ln(x+1) - 2\big). \tag{1}$$

   a) Vérifiez la consistance et la stabilité de cette méthode. La reconnaissez-vous ?

   b) Trouvez une constante $C > 0$ telle que

$$|x_{k+1} - x_\star| \leq C|x_k - x_\star|^2.$$

   c) Supposons que la méthode (1) soit initiée par $x_0 \in I_0$ tel que $|x_0 - x_\star| \leq 10^{-1}$. En utilisant la question précédente, trouvez le nombre minimal d'itérations nécessaire pour obtenir une erreur inférieure à $\epsilon = 10^{-10}$.

4) On applique maintenant la méthode de la bissection pour trouver une approximation $\widetilde{x}$ de la racine, telle que $|\widetilde{x} - x_\star| \leq 10^{-1}$, et ensuite la méthode du point fixe (1) à partir de la condition initiale $x_0 = \widetilde{x}$. Estimer, dans ce cas, le nombre minimal d'itérations nécessaire pour obtenir une erreur inférieure à $\epsilon = 10^{-10}$.

Exercice 3. ***(Méthode de Newton modifiée pour les racines doubles)***

Soit $f(x)$ une fonction admettant une racine double $x_\star$.

1) En écrivant f sous la forme

$$f(x) = (x - x_\star)^2 g(x), \qquad g(x_\star) \neq 0,$$

   vérifiez que la convergence de la méthode de Newton est seulement linéaire.

2) On considère maintenant la méthode de Newton modifiée :

$$x_{k+1} = x_k - 2\frac{f(x_k)}{f'(x_k)}.$$

   Etudiez la consistance et la stabilité de cet algorithme. Montrez qu'il converge au second ordre.

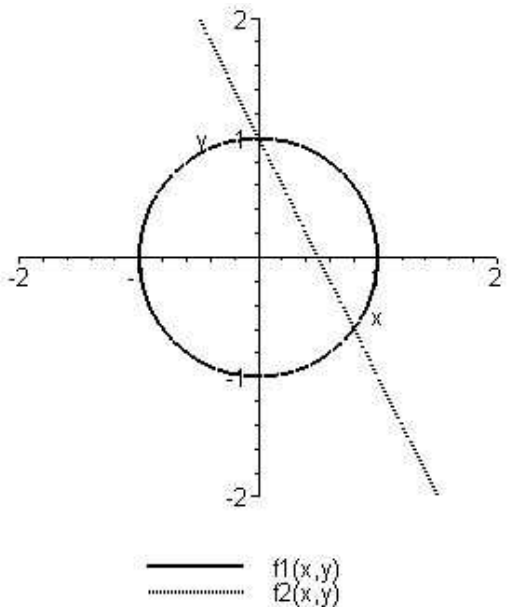


Figure 4: Courbes $f_1(x,y) = 0$ et $f_2(x,y) = 0$. Leurs intersections sont les solutions du système (1).

Exercice 4. ***(Système d'équations non linéaires)***

On pose $\boldsymbol{x} = (x, y)$, et on considère le système non-linéaire dans $\mathbb{R}^2$:

$$\boldsymbol{f}(\boldsymbol{x}) = 0, \qquad \text{avec} \quad \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(x,y) = x^2 + y^2 - 1 \\ f_2(x,y) = 2x + y - 1 \end{pmatrix}, \tag{1}$$

dont les solutions sont $\boldsymbol{x}_1^\star = (0, 1)$ et $\boldsymbol{x}_2^\star = (4/5, -3/5)$, cf. Fig. 4.

1) Pour résoudre (1), on considère la méthode du point fixe définie par sa fonction d'itération :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} \phi_1(x,y) = \dfrac{1-y}{2} \\ \phi_2(x,y) = \sqrt{1-x^2} \end{pmatrix}. \tag{2}$$

   Vérifiez la consistance et stabilité de cet algorithme lorsqu'on l'applique au calcul de $\boldsymbol{x}_1^\star$. Pouvez-vous l'utiliser pour calculer $\boldsymbol{x}_2^\star$?

2) Proposez une modification de $\boldsymbol{\phi}(\boldsymbol{x})$ permettant de calculer $\boldsymbol{x}_2^\star$. Montrez qu'elle est convergente pour $\boldsymbol{x}_0$ suffisamment proche de cette racine.

3) Décrivez la méthode de Newton-corde pour résoudre le système (1) en partant de $\boldsymbol{x}_0 = (1/2, 0)$.

Analyse Numérique - Licence de Mécanique

TD 3 - Equations Différentielles Ordinaires

Exercice 1. ***(Stabilité des schémas d'Euler)***

On veut résoudre numériquement le problème de Cauchy

$$\begin{cases} \dot{y}(t) + 0.5y(t) = 0, & t \in [0, 20], \\ y(0) = 1, \end{cases}$$

avec les méthodes d'Euler.

1) Enoncez la condition de stabilité du schéma d'Euler progressif (EP1). Pour les pas de temps $h = 4.2$, 3, et 1, établissez la formule de récurrence donnant la solution numérique à l'instant t_n en fonction de la condition initiale. Pour ces valeurs de h, représentez la solution numérique obtenue (complétez la figure 1). Commentez ces résultats.

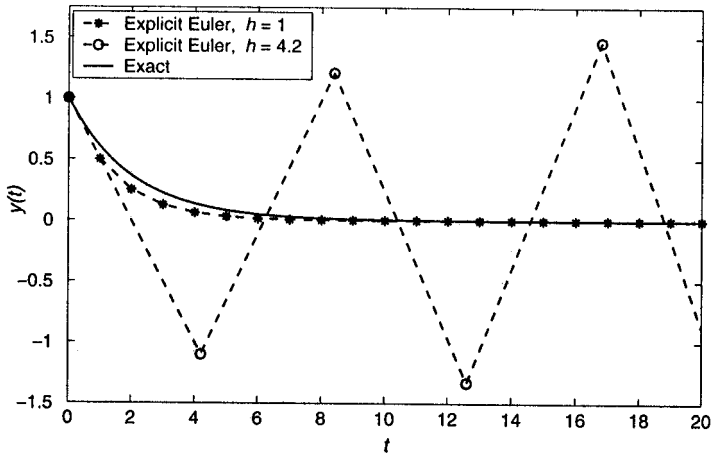


Figure 1: Solution numérique obtenue avec le schéma EP1.

2) Même question pour le schéma d'Euler rétrograde (ER1).

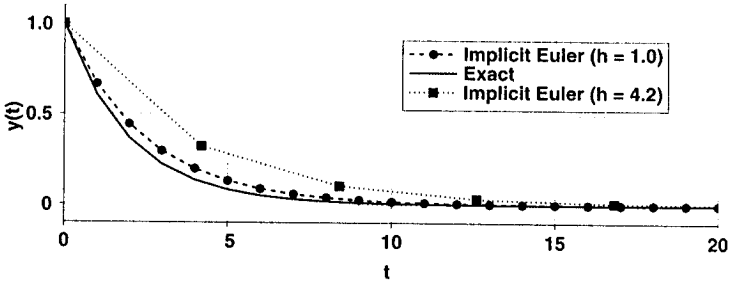


Figure 2: Solution numérique obtenue avec le schéma ER1.

Exercice 2. ***(Résolution d'une E.D.O. non-linéaire)***

On considère le problème

$$\begin{cases} \dot{y}(t) = -t\, y^2(t), & t \in [0,1], \\ y(0) = 2, \end{cases}$$

dont la solution exacte est $y(t) = 2/(1+t^2)$.

1) Cette *E.D.O.* est elle linéaire ? Pour le schéma EP1, établir la relation de récurrence donnant y^{n+1} en fonction de y^n. Pour $h = 1/10$, calculez la solution aux temps t_1 et t_2.

2) Déterminez une condition nécessaire de stabilité pour EP1. Vérifiez que la méthode est stable pour le pas de temps utilisé dans la question précédente.

3) Appliquez ER1 à la résolution de cette EDO. Montrez que ce schéma se met sous la forme de récurrence :

$$G(t_{n+1}, y^{n+1}) = H(t_n, y^n),$$

et explicitez les fonctions G et H. Pouvez-vous facilement calculer y^{n+1} ? Pourquoi ? Décrivez une méthode itérative permettant de calculer y^{n+1}.

Exercice 3. ***(Schéma de Crank-Nicolson)***

On considère le problème de Cauchy :

$$\dot{y}(t) = f(t, y(t)), \qquad t \in [0,T] \tag{1a}$$

$$y(0) = y_0. \tag{1b}$$

1) En appliquant la quadrature du trapèze entre les instants t_n et $t_{n+1} = t_n + h$, montrez qu'on définit le schéma :

$$\frac{y^{n+1} - y^n}{h} = \frac{1}{2}\left[\, f(t_n, y^n) + f(t_{n+1}, y^{n+1}) \,\right]. \tag{2}$$

qui est connu sous le nom de *schéma de Crank-Nicolson (C-N)*. Cette méthode est-elle explicite ou implicite ?

2) En utilisant des développements de Taylor, montrez que le schéma ER1 est précis au premier ordre, tandis que C-N est au second-ordre.

3) On applique le schéma C-N au problème test

$$\dot{y}(t) = \lambda y(t), \qquad \lambda < 0.$$

Calculez son facteur d'amplification. Vérifiez que la méthode est bien au second-ordre. Etudiez la stabilité de la solution numérique.

4) On a utilisé deux schémas X et Y pour intégrer numériquement une E.D.O. de la forme (1). Les résultats sont reportés dans le tableau 1. L'une de ces méthodes est ER1, l'autre C-N : pouvez-vous les identifier ?

h	Schéma X	Schéma Y
1	3.05×10^{-2}	5.22×10^{-3}
0.5	1.46×10^{-2}	1.03×10^{-3}
0.25	7.16×10^{-3}	2.50×10^{-4}
0.125	3.55×10^{-3}	6.19×10^{-5}
0.0625	1.77×10^{-3}	1.54×10^{-5}

Table 1: Erreurs commises à l'instant $t = t^\star$ pour différentes valeurs du pas de temps.

Exercice 4. ***(Systéme d'EDO linéaires : l'oscillateur mécanique)***

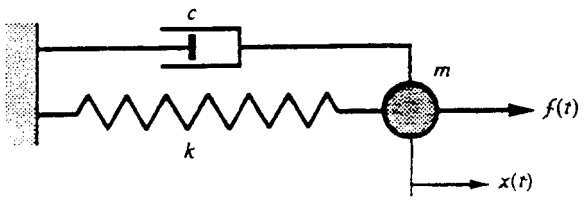


Figure 3: Schéma de l'oscillateur mécanique linéaire.

On considère l'oscillateur mécanique représenté sur la figure 3, formé de :

- Masse m indéformable telle que $x(t)$ représente son déplacement horizontal.
- Ressort sans masse de raideur $k > 0$, fournissant une force élastique proportionnelle et opposée au sens du déplacement, soit : $-kx(t)$.
- Amortisseur fournissant une force de freinage proportionnelle et opposée à la vitesse, soit : $-C\dot{x}(t)$, avec $C \geq 0$ la constante d'amortissement visqueux.
- Une force extérieure $f(t)$ agissant sur la masse m.

Au temps $t = 0$, la masse placée en x_0 est lachée à la vitesse v_0.

1) Par application du principe fondamental de la dynamique, montrez que le déplacement de la masse est régi par l'EDO du second-ordre :

$$\ddot{x}(t) + 2\gamma\,\dot{x}(t) + \omega_\star^2\,x(t) = \frac{f(t)}{m}, \tag{1}$$

avec $\gamma = C/(2m)$ le coefficient d'amortissement et $\omega_\star = \sqrt{k/m}$ la pulsation propre de l'oscillateur.

2) Calculez l'énergie mécanique $E(t)$ du système. Dans le cas où le système n'est pas forcé (*i.e.* $f(t) \equiv 0$), utilisez l'EDO (1) pour montrer l'identité :

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -C\,[\,\dot{x}(t)\,]^2\,.$$

En déduire le comportement du système lorsqu'il est freiné ou non freiné.

Dans la suite de l'exercice, les coefficients de raideur et d'amortissement sont choisis tels que $\gamma > \omega_\star > 0$.

3) Formulez l'EDO (1) sous la forme du problème de Cauchy :

$$\begin{cases} \dot{X}(t) = \mathcal{A}X(t) + F(t), \\ X(0) = X_0. \end{cases} \tag{2}$$

Appliquez le schéma EP1 à la résolution de ce système.

4) Cette question concerne la stabilité du schéma de la question précédente.

   a) A partir de la diagonalisation $\mathcal{A} = \mathcal{S}\mathcal{D}\mathcal{S}^{-1}$, rappelez les étapes principales de l'étude de stabilité du schéma.

   b) Calculez les valeurs propres λ_1, λ_2 de $\mathcal{A}$ en fonction de γ et $\omega_\star$, et déterminez leur signe.

   c) En déduire la condition de stabilité du schéma. Application numérique : déterminez cette condition dans le cas $\gamma = 5$, $\omega_\star = 4$.

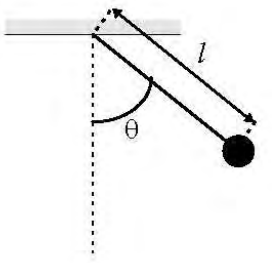


Figure 4: Le pendule oscillant. La position de la masse est repérée par l'angle θ. L'origine $\theta = 0$ est fixée à la position de repos de la masse.

5) Appliquez les schémas ER1 et CN à la résolution de (2) et énoncez leur condition de stabilité.

Exercice 5. ***(Pendule oscillant - Système d'équations non-linéaires - Diagrammes de stabilité - Schéma RK4)***

Considérons le pendule constitué d'une masse m attachée à une tige rigide de longueur l, qui se meut dans un plan vertical (cf. Fig. 4).

1) A partir de la loi de conservation de l'énergie mécanique, montrez que le mouvement de la masse est régi par l'EDO :
$$\ddot{\theta} = -\omega_\star^2 \sin\theta, \tag{1}$$
avec $\omega_\star = \sqrt{g/l}$, où g est la constante de gravitation. Mettez cette équation sous la forme du problème de Cauchy
$$\dot{X}(t) = F(t, X(t)), \tag{2}$$
avec
$$X(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}, \qquad F = \begin{pmatrix} f_1(t, x_1, x_2) \\ f_2(t, x_1, x_2) \end{pmatrix}.$$
Par consistance avec la suite de l'énoncé, on aura défini x_1 comme l'angle du pendule et x_2 sa vitesse angulaire.

2) On veut résoudre le système non-linéaire (2) par une méthode numérique. D'après l'exercice précédent, rappelez les étapes principales de l'analyse de stabilité appliquée à (2).

3) Calculez la matrice jacobienne de F, soit :
$$\mathcal{A} = \frac{\partial F}{\partial X} = \left(\frac{\partial f_i}{\partial x_j}\right)_{i,j}.$$
Montrez que ses valeurs propres s'écrivent :
$$\lambda = \pm\sqrt{-\cos x_1}\ \omega_\star.$$

4) Déduire de la question précédente que pour des oscillations de faible amplitude ($\theta \simeq 0$), l'équation du pendule s'écrit
$$\ddot{\theta} = -\omega_\star^2 \theta, \tag{3}$$
et calculez ses valeurs propres. A quel système physique correspond cette EDO ?

5) On souhaite utiliser les schémas classiques (EP1, ...) pour résoudre (1) ou (3). Pouvez-vous appliquer les conditions de stabilité usuelles ?

Les systèmes d'EDO à valeurs propres complexes sont souvent qualifiés de systèmes oscillants. Leur approximation numérique motive l'étude de la stabilité des schémas pour l'EDO (scalaire) modèle

$$\dot{y}(t) = \lambda y(t), \qquad \lambda = \lambda_{\mathrm{R}} + \underline{i}\,\lambda_{\mathrm{I}} \in \mathbb{C}, \qquad \lambda_{\mathrm{R}} \leq 0. \tag{4}$$

6) Montrez que le schéma EP1 appliqué à (4) est stable si le pas de temps h vérifie

$$\sqrt{\left(1 + \lambda_{\mathrm{R}} h\right)^2 + \left(\lambda_{\mathrm{I}}\, h\right)^2} < 1. \tag{5}$$

Vérifiez que pour $\lambda \in \mathbb{R}^{-,\star}$, on retrouve la condition de stabilité usuelle. Qu'en est-il dans le cas purement complexe ? Conseillerez-vous d'utiliser EP1 pour résoudre les systèmes oscillants ?

7) Effectuez l'analyse de stabilité des schémas ER1 et C-N.

Peut-on utiliser des schémas explicites pour résoudre des systèmes oscillants ? Tournons-nous maintenant vers les schémas RK2-α (cf. cours), ainsi que le schéma RK4 définit par

$$K_1 = hf(t_n, y^n), \tag{6a}$$
$$K_2 = hf\big(t_n + \tfrac{h}{2},\, y^n + \tfrac{1}{2}K_1\big), \tag{6b}$$
$$K_3 = hf\big(t_n + \tfrac{h}{2},\, y^n + \tfrac{1}{2}K_2\big), \tag{6c}$$
$$K_4 = hf\big(t_n + h,\, y^n + K_3\big), \tag{6d}$$
$$y^{n+1} = y^n + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right). \tag{6e}$$

8) A quelle classe de méthodes ce schéma appartient-il ? Est-il explicite ou implicite ?

9) Calculez son facteur d'amplification σ et montrez que le schéma RK4 est précis au quatrième ordre.

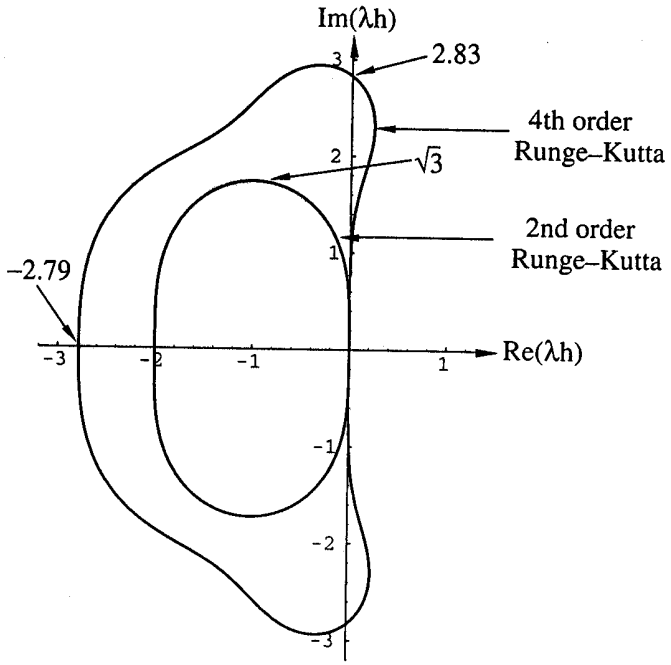


Figure 5: Diagramme de stabilité des schémas de Runge-Kutta. Le schéma est stable pour les valeurs de λh situées à l'intérieur de la courbe fermée. On peut remarquer que la stabilité des méthodes R-K croît avec leur ordre de précision.

Afin d'évaluer et comparer la stabilité des schémas numériques, il est très courant de représenter leur region de stabilité dans le plan complexe $(\lambda_{\mathrm{R}} h, \lambda_{\mathrm{I}} h)$, c'est-à-dire l'ensemble des complexes λh qui vérifient

$$|\sigma(\lambda h)| < 1.$$

Les zones de stabilité de RK2-α et de RK4 sont représentées[1] sur la figure 5.

10) Complétez la figure 5 en traçant la région de stabilité de EP1 (aidez-vous de la question 6).

11) A, l'aide des diagrammes de la figure 5, déterminez graphiquement les conditions de stabilité de RK2-α et RK4 dans les cas purement réel et complexe.

12) Confirmez que RK2-α est instable pour les problèmes oscillants en calculant $|\sigma(\underline{i}\,\lambda_{\mathrm{I}})|$.

13) Quel schéma conseillez vous pour résoudre les systèmes (1) et (3) ? Pour chacun des systèmes, déterminez ses conditions de stabilité lorsque $g = 9.81\,m/\mathrm{sec}^2$ et $l = 0.6\,m$.

Schémas	Type	Ordre q	$\sigma(\lambda h)$	Stabilité pour $\dot{y} = \lambda y$	
				$\lambda \in \mathbb{R}^{-,\star}$	$\lambda \in \underline{i}\,\mathbb{R}^{\star}$
EP1	Expl.	1	$1 + \lambda h$	$h < \dfrac{2}{-\lambda}$	
ER1	Impl.	1	$\dfrac{1}{1 - \lambda h}$	Oui	Oui
CN					
RK2-α ($\alpha = \frac{1}{2}$: Préd./Corr., $\alpha = 1$: Heun)	Expl.	2	$1 + \lambda h + \dfrac{(\lambda h)^2}{2!}$	$h < \dfrac{2}{-\lambda}$	
RK4					

Table 2: Principales propriétés des schémas numériques pour EDO.

Exercice 6. Utilisez les exercices précédents pour compléter le tableau 2.

[1] Ces diagrammes sont réalisés en traçant la frontière de ces courbes, correspondant aux points vérifiant

$$\sigma(\lambda h) - \mathrm{e}^{\underline{i}\varphi} = 0,$$

en faisant varier φ dans $[0, 2\pi[$. Les diagrammes de stabilité compliqués, comme celui de RK4, nécessitent l'utilisation de méthodes numériques pour extraire les racines d'une fonction à valeur complexe.

Analyse Numérique - Licence de Mécanique

TD 4 - Système d'Equations Linéaires

Exercice 1. ***(Algorithme de Thomas pour systèmes tridiagonaux)***

On considère le système linéaire $\mathcal{A}X = F$ d'ordre n, où $\mathcal{A}$ est la matrice tridiagonale définie par

$$\mathcal{A} = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix}.$$

Le but de cet exercice est de construire la méthode de résolution directe connue sous le nom d'*algorithme de Thomas*, qui est un cas particulier de la méthode d'élimination de Gauss pour systèmes tridiagonaux.

1) On cherche la décomposition $\mathcal{A} = \mathcal{L}\mathcal{U}$ avec $\mathcal{L}$ et $\mathcal{U}$ des matrices bidiagonales de la forme :

$$\mathcal{L} = \begin{bmatrix} 1 & 0 & & & \\ \alpha_2 & 1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{n-1} & 1 & 0 \\ & & & \alpha_n & 1 \end{bmatrix}, \qquad \mathcal{U} = \begin{bmatrix} \beta_1 & c_1 & & & \\ 0 & \beta_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \beta_{n-1} & c_{n-1} \\ & & & 0 & \beta_n \end{bmatrix},$$

où les coefficients $\{\alpha_i,\, i = 2, \ldots, n\}$ et $\{\beta_i,\, i = 1, \ldots, n\}$ sont à déterminer.

a) En identifiant les éléments de la matrice $\mathcal{A}$ avec ceux du produit $\mathcal{L}\mathcal{U}$, montrez que les α_i et β_i s'écrivent : $\beta_1 = b_1$ et

$$\alpha_i = \frac{a_i}{\beta_{i-1}}, \quad \beta_i = b_i - \alpha_i\, c_{i-1}, \qquad \text{pour } i = 2, \ldots, n.$$

b) En posant $X = (x_i,\, i = 1, \ldots, n)^{\mathrm{T}}$ et $F = (f_i,\, i = 1, \ldots, n)^{\mathrm{T}}$, résolvez le système par la méthode LU. Plus précisément, on vous demande d'établir la formule de récurrence progressive pour résoudre le système bidiagonal inférieur, puis la formule de récurrence rétrograde pour résoudre le système bidiagonal supérieur.

2) On appelle *algorithme de Thomas* les formules de récurrence de la question précédente : elles sont destinées à être programmées sur un ordinateur. Vérifiez qu'elles sont correctes en les appliquant au système d'ordre 3 défini par

$$\mathcal{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}, \qquad F = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

dont la solution est $X = (1,\, 1,\, 1)^{\mathrm{T}}$.

3) Donnez le nombre d'opérations algébriques élémentaires (flops) de l'algorithme de Thomas. Comparez sa complexité algorithmique avec celle d'un algorithme de résolution directe pour système quelconque (e.g., la méthode de Gauss).

Exercice 2. On considère le système linéaire $\mathcal{A}X = B$ avec

$$\mathcal{A} = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}, \qquad B = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

1) Montrez que le système admet une solution unique, et qu'il est symétrique défini positif (SPD). Que pouvez-vous dire sur la convergence des méthodes de Jacobi et de Gauss-Seidel (GS) ?

2) Construisez la matrice d'itération $\mathcal{B}_{\mathrm{J}}$ de la méthode de Jacobi, et étudiez la convergence de cette méthode.

3) Ecrivez la méthode GS pour résoudre ce système et calculez son facteur de réduction ρ_{GS}.

4) Comparez la vitesse de convergence des deux méthodes. En utilisant la relation

$$||E_k||_2 \leq \rho\, ||E_{k-1}||_2,$$

où E_k désigne l'erreur à la $k^{\text{ième}}$ itération, établissez pour chaque méthode le nombre minimum d'itérations nécessaire pour faire décroître l'erreur initiale $||E_0||_2$ d'un facteur de 1000.

Exercice 3. ***(Méthode de Richardson à pas stationnaire)***

On considère le système linéaire d'ordre n :

$$\mathcal{A}X = F. \tag{1}$$

On note $R_k = F - \mathcal{A}X_k$ le résidu de cette équation à la $k^{\text{ième}}$ itération, et on définit la méthode de Richardson

$$\mathcal{P}X_{k+1} = \mathcal{P}X_k + \alpha R_k,$$

où $\mathcal{P}$ une matrice inversible, et α un paramètre d'accélération réel à déterminer de façon judicieuse.

1) Etudiez la consistance de la méthode suivant la valeur de α.

2) Ecrivez la méthode de Richardson sous la forme

$$X_{k+1} = \mathcal{B}_\alpha X_k + F_\alpha,$$

et explicitez $\mathcal{B}_\alpha$ et F_α.

3) On note $\mathcal{A} = \mathcal{D} - \mathcal{L} - \mathcal{U}$ le "splitting" habituel de la matrice $\mathcal{A}$. Pour $\alpha = 1$ et $\mathcal{P} = \mathcal{D} - \mathcal{L}$, la méthode de Richardson correspond-elle à une méthode connue ?

4) On se place dans le cas $n = 2$, et on suppose que les valeurs propres de $\mathcal{P}^{-1}\mathcal{A}$, notées λ_i, sont strictement positives, soit : $\lambda_1 > \lambda_2 > 0$.

   a) Calculez les valeurs propres de $\mathcal{B}_\alpha$ et montrez que son rayon spectral est strictement inférieur à 1 si

   $$\rho_i(\alpha) = |1 - \alpha\lambda_i| < 1, \qquad \text{pour } i = 1, 2.$$

   b) En déduire que la méthode de Richardson converge si

   $$0 < \alpha < 2/\lambda_1.$$

   c) Tracez les $\rho_i(\alpha)$ sur un même graphe. Montrez graphiquement que le facteur de réduction est minimal lorsque le paramètre d'accélération est choisi tel que $\alpha = \alpha_{\mathrm{R,opt}}$ avec

   $$\alpha_{\mathrm{R,opt}} = \frac{2}{\lambda_1 + \lambda_2}. \tag{2}$$

   Calculez le facteur de réduction $\rho_{\mathrm{R,opt}}$ correspondant.

5) Application : pour le système (1) avec la matrice

$$\mathcal{A} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix},$$

on va comparer les performances de la méthode de Gauss-Seidel et de la méthode de Richardson avec $\mathcal{P} = \mathcal{D} - \mathcal{L}$ et le pas $\alpha_{\mathrm{R,opt}}$ donné par (2).

a) Calculez les valeurs propres de $\mathcal{P}^{-1}\mathcal{A}$. Déduisez-en le facteur de réduction de chacune des 2 méthodes.

b) Pour ce système linéaire, écrivez les deux méthodes et comparez leur coût numérique et leur vitesse de convergence.

c) Au vu des résultats de cette partie, justifiez l'assertion : *"La méthode de Richardson est un accélérateur de la convergence d'une méthode itérative de base".*

Exercice 4. ***(Performance des méthodes itératives pour les systèmes mal conditionnés)***

Soient a, b, c trois réels. On défini $\mathcal{A} = \text{Tridiag}_N[a, b, c]$ la matrice tridiagonale de taille $N \times N$ telle que :

$$\mathcal{A} = \begin{bmatrix} b & c & & & \\ a & b & c & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & c \\ & & & a & b \end{bmatrix}.$$

On notera D_N son déterminant.

1) Montrez que $\mathrm{D}_1 = b$, $\mathrm{D}_2 = b^2 - ac$, et que pour $N \geq 3$:

$$\mathrm{D}_N = b\,\mathrm{D}_{N-1} - ac\,\mathrm{D}_{N-2}.$$

2) Utiliser la question précédente pour montrer que :

$$\text{pour } N \geq 1, \quad \mathrm{D}_N = \frac{\sin(N+1)\theta}{\sin\theta}\, r^N, \tag{1}$$

avec $r = \sqrt{ac}$, et θ tel que $2r\cos\theta = b$. *(on rappelle les formules :* $\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y$, $\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y$, $\sin(x-y) + \sin(x+y) = 2\sin x \cos y$.*)*

3) Montrez que les valeurs propres de $\mathcal{A}$ sont :

$$\lambda_k = b - 2\sqrt{ac}\,\cos\frac{k\pi}{N+1}, \qquad k = 1, 2, \ldots, N.$$

(Suggestion : calculez les racines du déterminant de $\mathit{Tridiag}_N[a, b-\lambda, c]$ *en utilisant* (1)*.)*

On considère maintenant la matrice $\mathcal{K} = \text{Tridiag}_N[-1, 2, -1]$, et on pose $h = 1/(N+1)$. On souhaite comparer les performances des méthodes itératives classiques pour résoudre le système linéaire

$$\mathcal{K}X = F, \tag{2}$$

où F est un RHS donné.

4) Déduire de la question précédente que les valeurs propres de $\mathcal{K}$ s'écrivent

$$\lambda_k = 2 - 2\cos k\pi h, \qquad \text{pour} \quad k = 1, \ldots, N, \tag{3}$$

et que cette matrice est SDP. Que pouvez-vous dire sur la convergence des méthodes de Jacobi, Gauss-Seidel et *S.O.R.* appliquées au système (2) ?

5) En remarquant que $\cos N\pi h = -\cos\pi h$, montrez que le nombre de conditionnement de $\mathcal{K}$ s'écrit

$$\chi(\mathcal{K}) = \frac{1 + \cos\pi h}{1 - \cos\pi h}.$$

En déduire que le système (2) est mal conditionné lorsque N est grand.

6) On note $\mathcal{D}$ la partie diagonale de $\mathcal{K}$. En calculant les valeurs propres de la matrice d'itération $\mathcal{B}_{\rm J} = \mathcal{I} - \mathcal{D}^{-1}\mathcal{K}$ de la méthode de Jacobi, montrez que son facteur de réduction s'écrit :

$$\rho_{\rm J} = \cos \pi h, \tag{4}$$

et que $\rho_{\rm J} \simeq 1 - (\pi h)^2/2$ lorsque h est petit. Commentez les performances de la méthode de Jacobi pour les systèmes mal conditionnés.

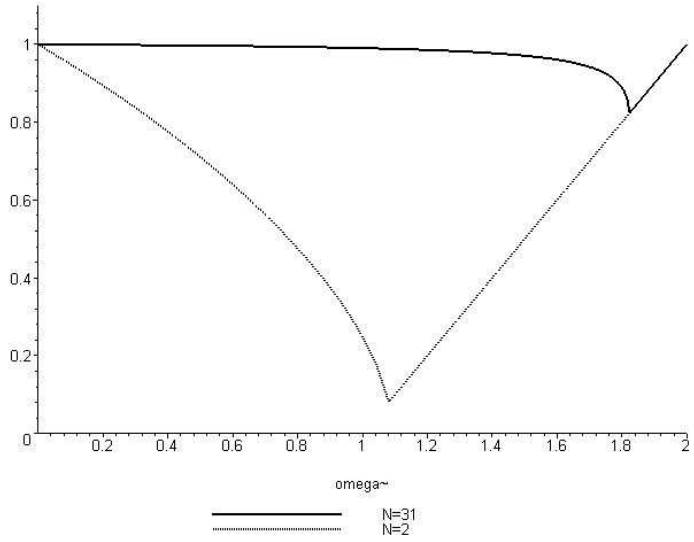


Figure 1: Facteur de réduction de la méthode *S.O.R.* appliqué à la résolution du système (2) pour $N = 2$ ($\rho_{\rm J} = 1/2$) et $N = 31$ ($\rho_{\rm J} \simeq 0.995$).

7) Un fameux théorème dû à Young (1950) énonce que si les valeurs propres de $\mathcal{B}_{\rm J}$ sont réelles et que $\rho_{\rm J} < 1$, alors la méthode *S.O.R.* est convergente pour le paramètre de relaxation $\omega \in]0, 2[$, avec le facteur de réduction

$$\rho_{\rm S.O.R}(\omega) = \begin{cases} 1 - \omega + \dfrac{\omega^2\rho_{\rm J}^2}{2} + \omega\rho_{\rm J}\sqrt{1 - \omega + \dfrac{\omega^2\rho_{\rm J}^2}{4}} & \text{pour} \quad 0 < \omega \leq \omega_{\rm opt}, \\ \omega - 1 & \text{pour} \quad \omega_{\rm opt} \leq \omega < 2, \end{cases} \tag{5a}$$

tel que

$$\omega_{\rm opt} = \frac{2}{1 + \sqrt{1 - \rho_{\rm J}^2}} > 1. \tag{5b}$$

Le graphe de $\rho_{\rm S.O.R}(\omega)$ en fonction de ω est représenté sur la figure 1. On peut montrer que pour toute valeur de $\rho_{\rm J} < 1$, le minimum de $\rho_{\rm S.O.R}(\omega)$ est obtenu en $\omega = \omega_{\rm opt}$.

a) Montrez que le facteur de réduction de la méthode de Gauss-Seidel (G.S) s'écrit :

$$\rho_{\rm GS} = \rho_{\rm J}^2, \tag{6}$$

et que $\rho_{\rm GS} \simeq 1 - (\pi h)^2$ lorsque h est petit. Comparez les performances de G.S. avec celle de Jacobi pour la résolution du système (2).

b) Pour ce même système, montrez que le paramètre de relaxation et le facteur de réduction optimaux de la méthode *S.O.R.* sont :

$$\omega_{\rm opt} = \frac{2}{1 + \sin \pi h}, \qquad \rho_{\rm S.O.R}(\omega_{\rm opt}) = \frac{1 - \sin \pi h}{1 + \sin \pi h}, \tag{7}$$

et que $\rho_{\text{S.O.R}}(\omega_{\text{opt}}) \simeq 1 - 2\pi h$ lorsque h est petit. Comparez avec les méthodes précédentes.

8) Pour le système (2) avec $N = 31$, calculez les facteurs de réduction des trois méthodes. En utilisant la relation

$$||E_k||_2 \leq \rho\, ||E_{k-1}||_2,$$

où E_k désigne l'erreur à la $k^{\text{ième}}$ itération, établissez pour chaque méthode le nombre minimum d'itérations nécessaire pour faire décroître l'erreur initiale $||E_0||_2$ d'un facteur de 1000. Au vu de ces résultats, quelle méthode itérative recommandez-vous ?

Analyse Numérique - Licence 3 Mécanique des fluides et Energie

Enoncés des TP - Analyse numérique avec PYTHON

olivier.botella@univ-lorraine.fr

1 Présentation des TP

Les travaux pratiques d'analyse numérique s'effectuent dans le langage de programmation Python et sur l'environnement de développement Spyder[1]. Ils se déroulent sur 6 séances encadrées de 2h. Pour réaliser ces TP, vous devez télécharger sur la page ARCHE du cours[2] l'archive `tp_python_analyse_num_l3mfe.zip` et l'extraire sur votre PC. Cette archive contient les programmes Python à utiliser et compléter lors des séances.

1.1 L'environnement de développement intégré (IDE) Spyder

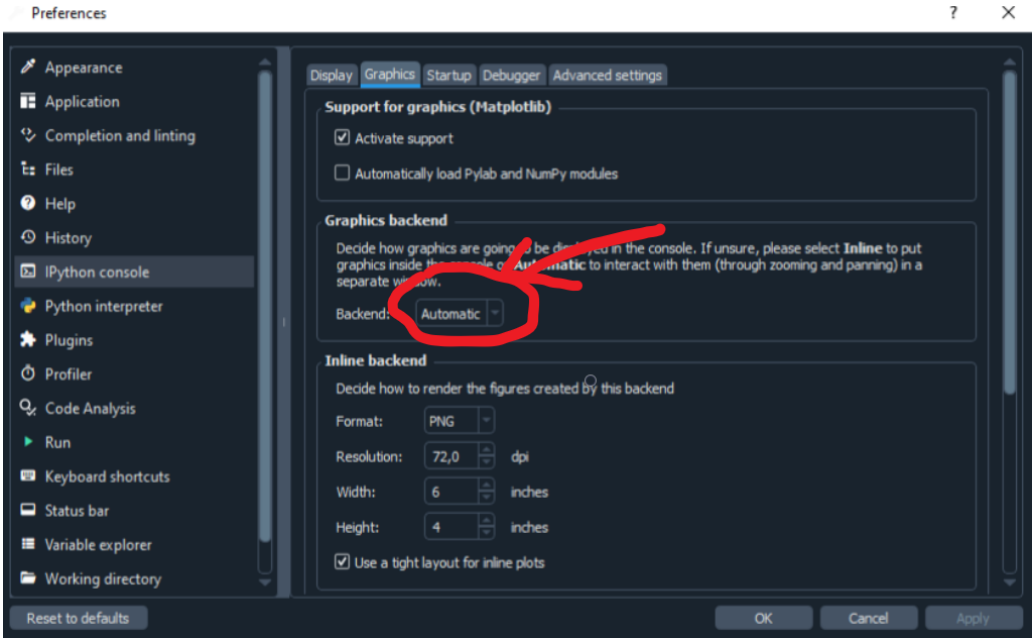


Figure 1: Sélectionner Graphics backend = Automatic dans les préférences de Spyder pour afficher les figures sur des fenêtres graphiques indépendantes. Les préférences de Spyder sont disponibles dans l'onglet Tools (ou Outils).

Le logiciel open-source Spyder est une IDE spécialement conçue pour la programmation scientifique en Python, qui inclus les bibliothèques les plus courantes du langage Python, telles que Matplotlib et Numpy. Son interface graphique et ses fonctionnalités sont très similaires à celles du logiciel Matlab, *cf.* figure 1. Le logiciel Spyder est disponible au téléchargement gratuit https://www.spyder-ide.org/. Il est aussi inclus dans la suite de logiciels Anaconda (https://www.anaconda.com/download).

[1] Spyder: Scientific PYthon Developpement EnviRonment, https://www.spyder-ide.org/.

[2] Lapageducoursest˜:https://arche.univ-lorraine.fr/course/view.php?id=6384. Alternativement, vous pouvez chercher les mots clefs *botella analyse* sur la page d'accueil de Arche: https://arche.univ-lorraine.fr/.

Suivant les salles informatiques de la FST, Spyder est directement installé ou accessible à partir d'Anaconda.

1.2 Notation des travaux pratiques

La note de TP d'analyse numérique sera basée sur l'évaluation par l'encadrant de votre travail et assiduité lors des séances, et sur les compte-rendus individuels que vous devez rendre à la fin du semestre.

Vous devez rédiger un compte-rendu individuel pour chaque TP :

- Les compte-rendu devront répondre aux questions de l'énoncé, et comporter une dizaine de pages au maximum. Les réponses doivent être rédigées de façon précise et succincte (ne recopiez pas l'énoncé !), et seront illustrées par les programmes Python que vous écrirez et les figures que vous avez réalisées.
- Les compte-rendu doivent être rédigés avec le traitement de texte Word ou convertis en fichiers Pdf. Ne rendez pas de fichiers Python. N'oubliez pas de mettre votre nom et l'intitulé du TP sur les compte-rendu.

Les 3 compte-rendu de TP doivent être déposés sur la page ARCHE du cours, avant la date limite que vous fixera l'encadrant.

2 Travaux pratiques d'analyse numérique avec Python

Vous disposez de 6 séances pour réaliser les 4 travaux pratiques ci-dessous. Le plus long TP, celui sur les EDO, peut nécessiter jusqu'à 2 séances.

2.1 Enoncés des TP

Travaux pratiques 0. ***(Les aide-mémoire Python)***

Le répertoire `AIDE_MEMOIRE_PYTHON` contient des scripts qui résument l'essentiel des fonctionnalités Python utilisées pour réaliser les TPs d'analyse numérique, et donnent de nombreux exemples :

- Script `aide_memoire_graphisme.py` : pour réaliser des figures avec la bibliothèque Matplotlib https://matplotlib.org/. Les fonctions tracées sont importées à partir du fichier `fonctions_aide_memoire_gr`
- Script `aide_memoire_python.py` : pour les bases du langage Python (boucles, tableaux Numpy https://numpy.org/, manipulation de fichiers, ...)

Ces scripts incluent des références Web pour approfondir les différentes notions abordées. les références les plus utilisées sont :

- https://courspython.com/
- https://www.w3schools.com/python/default.asp
- https://www.w3schools.com/python/numpy/default.asp

Travaux pratiques 1. ***(Intégration numérique par méthode des trapèzes)***

Le but du TP est de calculer une approximation numérique I_h de l'intégrale (vue en TD du L3 MFE) :

$$I = \int_1^\pi \frac{\sin x}{2x^3}\,\mathrm{d}x \simeq 0.198557\ldots, \tag{1}$$

à l'aide de la méthode des trapèzes, et de comparer la valeur numérique I_h avec la valeur exacte en vérifiant que l'erreur numérique :

$$E_h = |I - I_h|, \tag{2}$$

tends vers zéro lorsque $h \to 0$. Le répertoire de travail Spyder est `TP1_L3_QUADRATURE`, et le script principal est `main_TP_quadrature.py`.

1) Ecrivez la fonction Python :

```
def trapeze(a, b, f, n):
# ...
    return Ih
```

qui calcule l'intégrale $I_h = \int_a^b f(x)\mathrm{d}x$ par la formule du trapèze composite avec n sous-intervalles de taille uniforme h[3].

2) Validez votre fonction `trapeze` en vérifiant qu'elle donne l'intégrale exacte d'une fonction linéaire (polynôme de degré 1).

3) Vous allez maintenant utiliser la fonction trapeze pour calculer l'intégrale I donnée par l'équation (1).

   a) A l'aide de la bibliothèque de calcul symbolique Sympy[4], calculez une valeur précise (notée `Iex`) en tapant ces lignes dans votre programme Python :

```
import sympy as sp # Calcul symbolique avec sympy
t = sympy.symbols('t')
fex = sp.sin(t)/(2*t**3)
Iex = sp.integrate(fex, (t, 1, sp.pi)).evalf()
```

   b) Vérifiez que $E_h \to 0$, lorsque $h \to 0$, c'est-à-dire lorsque n augmente. Pour des valeurs de h suffisamment petites ($n \geq 20$), calculez le facteur de réduction FdR[5] de la méthode du trapèze. Comparez-le avec sa valeur théorique[6].

   c) Pour $n \geq 50$, représentez en échelle logarithmique l'erreur numérique E_n de la méthode du trapèze (voir les exemples données dans le script `aide_memoire_graphisme.py`) et montrez qu'elle est précise au second-ordre en traçant sur le même graphe la fonction $n \mapsto 1/n^2$ comme sur la figure 2. Pour cela, il est conseillé d'écrire dans un fichier csv plusieurs réalisations de la fonction `trapeze` sous la forme :

[3] On rappelle que, d'après le cours, cette formule s'écrit :

$$I_h = \sum_{k=0}^{n-1} h\left(\tfrac{1}{2}f(x_k) + \tfrac{1}{2}f(x_{k+1})\right),$$

avec $h = (b-a)/n$ et $x_k = a + kh$.

[4] La page web de Sympy est https://www.sympy.org/

[5] La définition du TD est $\mathrm{FdR} = E_h/E_{h/2}$ où E_h est donné par (2).

[6] D'après le TD, si h est suffisamment petit alors $FdR \simeq 2^q$ pour une méthode d'ordre de précision q.

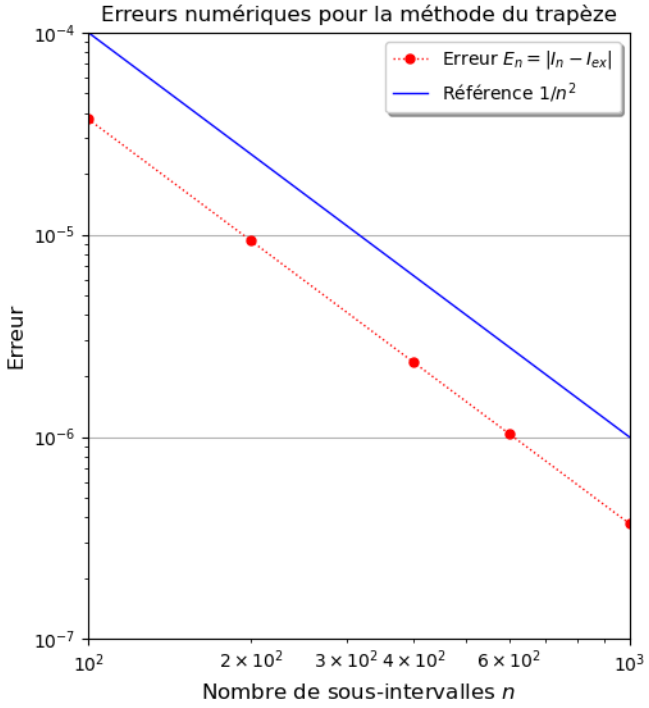


Figure 2: Résultats de la méthode du trapèze pour l'intégrale (1). Pour afficher les figures dans des fenêtres indépendantes de l'IDE, il faut sélectionner Graphics backend = Automatic dans les préférences de Spyder (*cf.* figure 1).

```
n1  E_n1
n2  E_n2
n3  E_n3
etc ...
```

puis lire les données du fichier dans un script Python, et tracer les données lues (un exemple est donné dans le script `aide_memoire_python.py`).

Travaux pratiques 2. ***(Résolution itérative d'équations non-linéaires)***

Ce TP concerne la résolution de l'équation non-linéaire :

$$f(x) = 0 \quad \text{pour x} \in [a, b], \tag{1}$$

par les méthodes itératives de la bissection et de Newton qui ont été vues en cours et TD. Le répertoire de travail Python est `'TP2_L3_NEWTON_EQT_NON_LIN'`.

1) Le script `main_TP_eqts_non_lin.py` applique l'algorithme de la bissection (donné dans le fichier `fonctions_TP_quadrature.py`) à la résolution du problème (1) pour la fonction

$$f(x) = \sin(2x) + x - 1,$$

que nous appellerons par la suite *problème test*. Le critère d'arrêt de l'algorithme est basé sur le nombre d'itérations maximal $k_{\max}$ et le résidu de l'équation $r_k = |f(x_k) - 0|$, et s'écrit :

$$k \leq k_{\max} \quad \text{et} \quad r_k < \epsilon, \qquad k_{\max} \text{ et } \epsilon \text{ définis par l'utilisateur.}$$

Le code Python appliquez l'algorithme de la bissection pour l'intervalle de départ $I_0 = [-4, 4]$ et une tolérance $\epsilon = 10^{-10}$. Un exemple de tableau solution `xB` est donné sur la figure 3. Comprenez bien ce programme, car les algorithmes itératifs que vous progammerez par la suite suivront le même principe. Exportez les graphes dans votre compte-rendu *Word* et commentez-les[7].

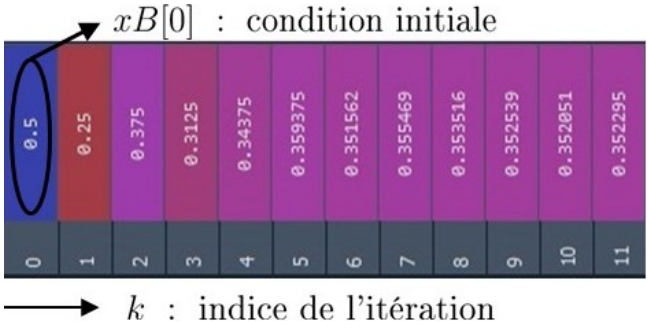


Figure 3: Tableau solution xB pour l'algorithme de la bissection, visible dans l'explorateur de variables de Spyder.

2) Ecrivez l'algorithme de Newton dans le même esprit que celui de la bissection. Plus précisément, créez la fonction Python :

```
def newton(fun, dfun, x0, eps, kmax):

    return x[0:k+1], res[0:k+1]
```

qui retourne le résidu r_k et la solution x_k à chaque itération k, avec pour variable d'entrée la fonction $f(x)$, sa dérivée $f'(x)$, la condition initiale x_0, la tolérance ϵ et le nombre maximal d'itérations $k_{\max}$.

   a) Validez votre programme en l'appliquant au problème test à partir de $x_0 = 0$.

   b) Que se passe-t-il si on initialise l'algorithme loin de la racine à trouver, par exemple $x_0 = 0.9$?

   c) Comparez la vitesse de convergence de la méthode de Newton avec celle de l'algorithme de la bissection.

3) Dans cette question vous allez combiner les algorithmes de la bissection et de Newton pour résoudre le problème des tiges (*cf.* cours) avec $a1 = 10$, $a2 = 13$, $a3 = 8$ et $a4 = 10$. Pour une valeur de l'angle ω que vous choisirez, le but de la question est de déterminer la (ou les) racines de l'équation $f_\omega(\theta) = 0$ dans l'intervalle $I = [-\pi, \pi]$.

   Afin de définir une condition initiale "suffisamment proche" pour que l'algorithme de Newton converge, vous allez appliquer la stratégie de l'exercice 2 du TD 2 :

   a) Pour chaque racine de l'intervalle I, déterminez graphiquement un encadrement initial I_0.

   b) Faites quelques itérations avec l'algorithme de la bissection, pour obtenir une approximation $\widetilde{x}$ avec la tolérance $\epsilon = 10^{-1}$.

   c) A partir de $\widetilde{x}$, appliquez ensuite l'algorithme de Newton pour calculer la racine avec la tolérance de $\epsilon = 10^{-14}$.

[7] Il est intéressant d'observer que les graphes du résidu et de l'erreur ont le même comportement : ainsi, lorsqu'on ne connaît pas la solution exacte (ce qui est le cas dans les problèmes d'ingénierie), la valeur du résidu fournit un bon

Dans le compte-rendu, expliquez votre méthodologie (valeur de l'angle ω, encadrement initial, tolérance et nombre d'itérations pour chacun des algorithmes, résultat final) et montrez le graphe des résidus.

Travaux pratiques 3. ***(Schémas numériques pour équations différentielles ordinaires)***

Ce TP concerne la résolution du problème à valeur initiale :

$$\begin{cases} \dot{y}(t) = f(t, y(t)), & t \in [t_0,\, t_{\mathrm{f}}], \\ y(t_0) = y_0, \end{cases} \tag{1}$$

par les schémas numériques vus en cours et dans le TD 3 (schémas d'Euler progressif, Runge-Kutta, ...). Le répertoire de travail Python est `TP3_L3_SCHEMAS_EDO`.

1) Le script `main_TP_schemas_EDO.py` applique le schéma EP1 à la résolution de l'EDO du premier ordre (1) pour $t \in [0, 12]$, avec $f(t, y) = -y(1/10 - \sin(t))$ et la condition initiale $y(0) = 1$, dont la solution exacte est :

$$y\,(t) = \frac{e^{-1/10\,t - \cos(t)}}{e^{-1}}.$$

Cette EDO sera appelée *exemple 1* dans la suite de l'énoncé.

   a) Tracez sur une même figure la solution numérique donnée par EP1 pour des valeurs de h de plus en plus petite : $h = 1;\ 0.5;\ 0.1;$ etc .... Qu'observez-vous lorsque $h \to 0$?

   b) Programmez le schéma de Heun, dénommé `def Heun(f, t0, tf, y0, h):`, dans le même esprit que la fonction Python EP1. Comparez sur une même figure la solution donnée par les schémas EP1 et de Heun pour un même pas de temps, par exemple $h = 0.4$. Est-ce en accord avec les propriétés théoriques de ces schémas ?

   c) Déterminez analytiquement la condition de stabilité du schéma EP1[8] pour l'exemple 1. Reportez sur une même figure les solutions données par EP1 pour des valeurs de h choisies au dessus et en dessous de la condition de stabilité[9].

2) On considère maintenant l'EDO du second-ordre :

$$\begin{cases} \ddot{\theta}(t) = f(t, \theta(t)), & t \in [t_0, t_{\mathrm{f}}], \\ \theta(t_0) = \theta_0, \quad \dot{\theta}(t_0) = \dot{\theta}_0. \end{cases} \tag{2}$$

avec $f(t, \theta) = -\omega^2\theta$, où $\omega > 0$ est la pulsation propre. Les paramètres de l'EDO sont :

$$\omega = 4, \quad t_0 = 0, \quad t_{\mathrm{f}} = 2\pi,$$

et les conditions initiales sont :

$$\theta_0 = 1, \quad \dot{\theta}_0 = 0.$$

renseignement sur la qualité de la solution numérique.

[8] Rappel de cours : le schéma EP1 est stable si

$$h < \frac{2}{\max\limits_{t \in [t_0, t_{\mathrm{f}}]} \{-\lambda\}}, \quad \text{avec } \lambda = \frac{\partial f}{\partial y}.$$

[9] Pour observer au mieux la stabilité du schéma au temps long, il est recommandé d'augmenter le temps d'intégration, par exemple `tf=50`, et de limiter l'axe des ordonnées, par exemple `plt.ylim(-2, 10)`.

Ce problème sera appelé dans la suite *exemple 2*, et sa solution exacte est :

$$\theta_{\text{exact}}(t) = \cos(\omega t). \tag{3}$$

Cette EDO est autonome (le second-membre f est indépendant du temps) et linéaire (sa solution analytique est une combinaison linéaire de sinus et cosinus), et elle représente la modélisation mathématique du pendule linéaire (tel que $\theta(t)$ représente l'angle du pendule et $\dot{\theta}(t)$ sa vitesse angulaire) ou de l'oscillateur harmonique sans frottement du TD. Cette EDO est de type conservatif car l'énergie mécanique est conservée au cours du mouvement (*cf.* TD).

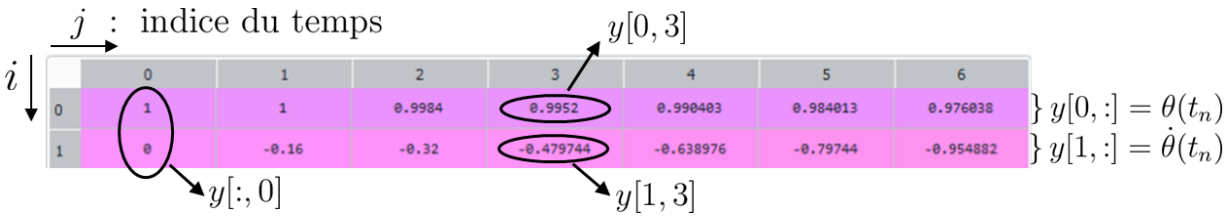
a) A partir de la fonction Python `EP1`, créez une version adaptée à la résolution d'EDO du second ordre (nommez-la par exemple `def EP1_2D(f, t0, tf, y0, h):`). Conseil : réécrivez l'EDO (2) en un système d'EDO du premier ordre (*cf.* cours et TD), de façon que le second membre s'écrive :

```
def fedo_2D(t, y):

# ... Second-membre pour l'EDO de l'exemple 2
    w = 4
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -w**2*y[0]

    return F
```

Dans le script principal, le vecteur `y0= np.array([1, 0])` correspondra à la condition initiale $(\theta_0, \dot{\theta}_0) = (1, 0)$. Dans la function `EP1_2D`, le vecteur solution sera initialisé par le tableau `y = np.zeros((2,Nt + 1))`. La condition initiale sera `y[:, 0]= y0`, et la sous-matrice colonne `y[:, n]= [y[0, n], y[1, n]]` correspondra à la solution au temps $t_n = nh$: `y[0, n]` $= \theta(t_n)$ et `y[1, n]` $= \dot{\theta}(t_n)$. Un exemple est donné sur la figure 4.



	0	1	2	3	4	5	6
0	1	1	0.9984	0.9952	0.990403	0.984013	0.976038
1	0	-0.16	-0.32	-0.479744	-0.638976	-0.79744	-0.954882

Figure 4: Exemple de tableau solution y pour la résolution d'EDO du second ordre.

b) Utilisez le schéma EP1 pour résoudre l'exemple 2 et comparez la solution numérique $\theta(t_n) =$`y[0, n]` et la solution analytique (3) sur une même figure[10] : qu'observez-vous ? Est-ce en accord avec les propriétés de stabilité théorique de EP1 pour ce type d'EDO (*cf.* TD exercice 5) ? Que penser de la stabilité du schéma de Heun pour cet exemple ?

c) Programmez le schéma RK4 (*cf.* TD). Enoncez la condition de stabilité pour ce schéma, et déterminez la valeur maximale de stabilité $h_{\max}$ du pas de temps. Utilisez RK4 avec un pas de temps h légèrement inférieur à $h_{\max}$, puis une valeur bien inférieure (*ie.*, $h_{\max}/10$). Comment qualifiez-vous le type d'erreur donné par RK4 par rapport à ce qui est représenté sur la figure 5 ?

[10] Il est à noter que la solution analytique est telle que $\theta(t) \in [-\theta_0, \theta_0]$ donc restreignez les ordonnées de la figure avec `plt.ylim(-5, 5)` par exemple.

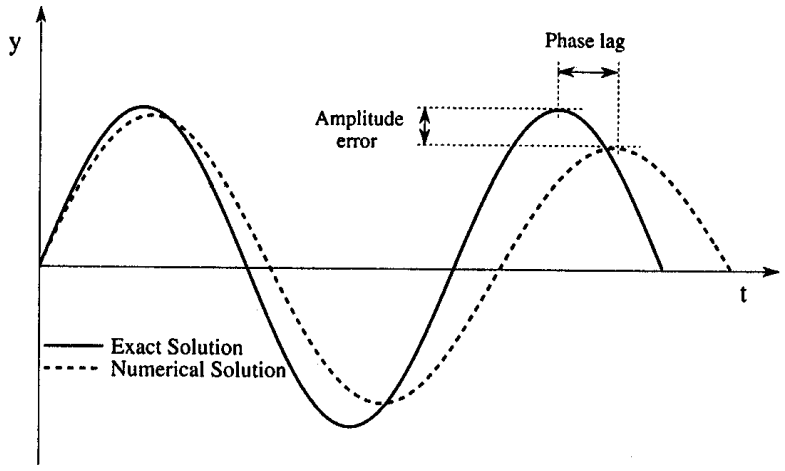


Figure 5: Décomposition de l'erreur numérique en une composante d'erreur d'amplitude (ou dissipation numérique) et une composante d'erreur de phase (ou dispersion numérique).

d) Dans le plan $(\theta, \dot{\theta})$ = `[y[0, n], y[1, n]]` appelé *espace des phases*, tracez les solutions numériques obtenues à partir de plusieurs conditions intitiales, par exemple `y0=[1, 0]`, `y0=[2, 0]` et `y0=[4, 0]`. Ce type de graphe est appelé *portrait de phases* de l'EDO, et on appelle *trajectoire* une solution partant d'une condition initiale donnée. Pourquoi les trajectoires de cette EDO correspondent-elles à des courbes fermées dans l'espace des phases ?

3) On considère maintenant l'EDO (2) avec $f(t,\theta) = -g\sin(\theta)/l$, qui correspond à l'équation du pendule non-linéaire de l'exercice 5 du TD. Les conditions initiales seront choisies telles que le pendule est lancé avec une vitesse angulaire $\dot{\theta}_0$ non-nulle à partir de la position d'équilibre stable $\theta_0 = 0$. On se placera dans les conditions de l'exercice 5, soit : $g = 9.81\,m/\mathrm{s}^2$ et $l = 0.6\,m$.

a) Enoncez la condition de stabilité du schéma RK4 pour cette EDO.

b) Utilisez le schéma RK4 pour calculer plusieurs solutions numériques de vitesse angulaire initiale $\dot{\theta}_0$ choisie au dessous et au dessus de la valeur $\dot{\theta}_0^\star = 2\sqrt{g/l}$, afin de construire un portrait de phases semblable à celui représenté sur la figure 6.

Travaux pratiques 4. ***(Méthodes itératives pour systèmes d'équations linéaires)***

Ce TP concerne la résolution de systèmes linéaires de taille $n \times n$ de la forme $\mathcal{A}X = F$ avec les méthodes itératives vues en cours : algorithme de Jacobi, Gauss-Seidel et relaxation (ou *S.O.R.*). Ces algorithmes seront appliqués à la résolution du problème test :

$$\mathcal{K}X = F, \tag{1}$$

avec $\mathcal{K}$ la matrice tridiagonale $\mathcal{K} = \mathrm{Tridiag}_n[-1, 2, -1]$ (vue en cours et dans l'exercice 4 du TD 4), et la solution exacte est choisie comme $X = (1, \ldots, 1)$. Ce TP a pour but d'illustrer les propriétés de convergence des algorithmes qui ont été vues théoriquement dans l'exercice 4, et il est donc important que cet exercice ait été entièrement fait et compris. Le répertoire de travail Python du TP est `TP4_L3_SYST_EQTS_LIN`.

1) Le script principal `main_TP_systemes_lineaires.py` applique la méthode de Jacobi à la résolution de (1). Comprenez et exécutez ce programme pour un système linéaire de petite taille, tel que $n = 2$. La suite des questions a pour but de comparer la performance des divers algorithmes pour le cas $n = 2$. Tous les tests seront effectués pour la même condition initiale $X_0 = 0$ et la tolérance $\epsilon = 10^{-10}$.

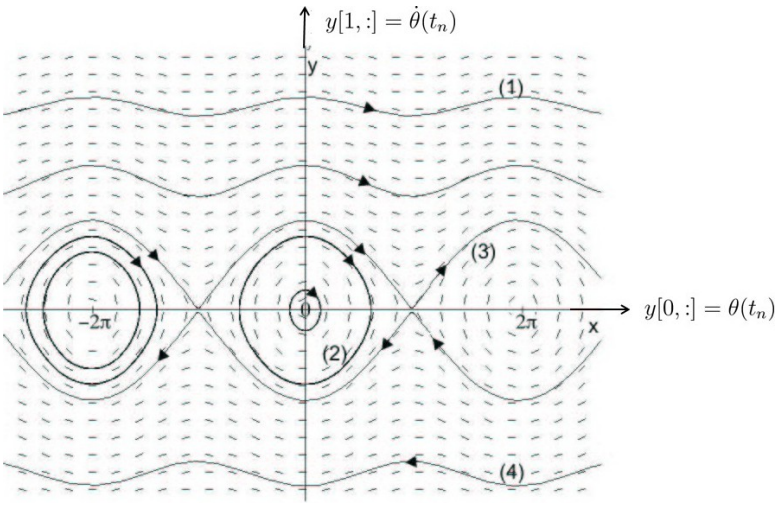


Figure 6: Portrait de phases du pendule non-linéaire pour diverses trajectoires partant de la condition initiale $\theta_0 = 0$ et $\dot{\theta}_0$. La trajectoire (3) de vitesse initiale $\dot{\theta}_0^\star = 2\sqrt{g/l}$ correspond à des solutions joignant deux positions d'équilibre instable $\theta(t) = \pi \mod (\pi)$, la trajectoire (2) de vitesse initiale faible (telle que $|\dot{\theta}_0| < |\dot{\theta}_0^\star|$) correspond à des oscillations du pendule autour de la position d'équilibre stable, les trajectoires (1) et (4) correspondent au cas où le pendule est lancé avec une vitesse initiale suffisamment grande ($|\dot{\theta}_0| > |\dot{\theta}_0^\star|$) pour lui permettre de faire des tours sur lui-même.

a) Programmez la méthode *S.O.R.* pour un paramètre de relaxation ω quelconque. Conseil : utilisez au maximum la manière utilisée pour programmer la méthode de Jacobi.

b) Programmez la fonction `def w_opt(n):` qui donne la valeur du paramètre de relaxation optimal pour le problème test (1).

c) Utilisez ces fonctions pour résoudre le problème test avec les méthodes de Gauss-Seidel et *S.O.R.* avec paramètre de relaxation optimal ω_{opt}.

d) Comparez sur le même graphe les résultats donnés par les méthodes de Jacobi, G-S et S.O.R. (résidu, erreur et facteur de réduction). Lorsque vous aurez confirmé les résultats théoriques du cours sur la valeur asymptotique du facteur de réduction, les méthodes que vous avez programmées seront considérées comme validées, et vous pourrez passer aux questions suivantes.

2) On considère dans cette question le problème test (1) avec $n = 31$.

a) Utilisez les trois méthodes itératives pour résoudre ce problème avec la tolérance de $\epsilon = 10^{-8}$.

b) Comparez les divers résultats des méthodes sur les mêmes graphes. Pourquoi observe-t-on que l'erreur relative est-elle bien supérieure à la norme du résidu ?

c) Pour chaque méthode, estimez le nombre minimal d'itérations pour que l'erreur relative soit inférieure à $\epsilon = 10^{-6}$. Est-ce en accord avec les résultats théoriques vus en TD ?