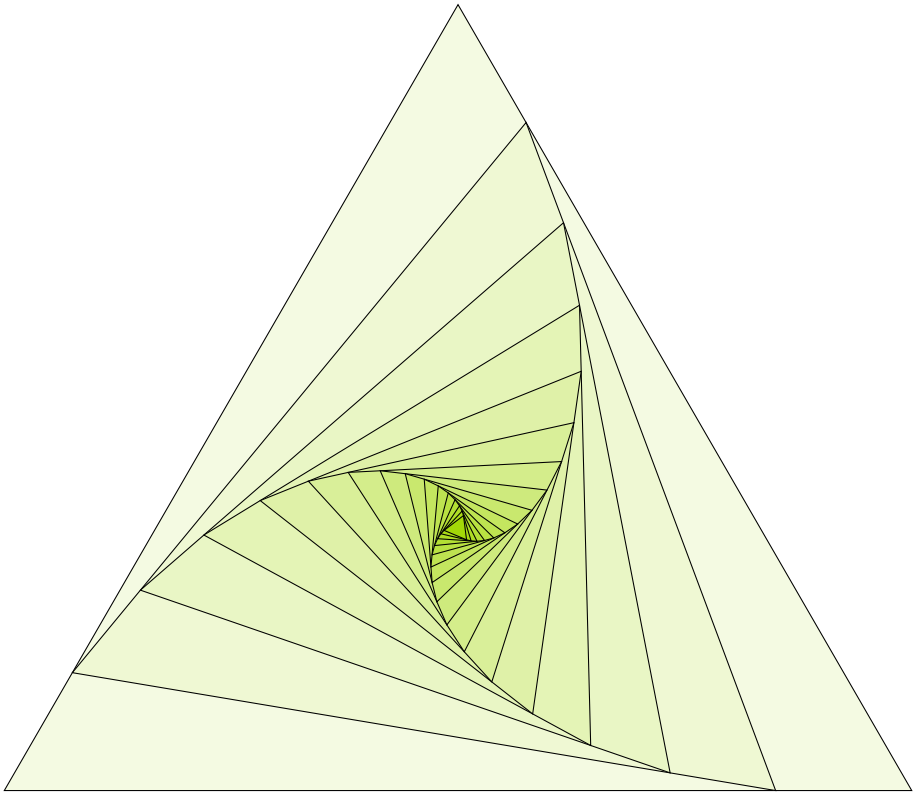
Méthodes numériques II

Notes de cours

Sup Galilée, Ingénieurs Energétique 1ère année

version du 05/06/2026 à 11:51:26



Francois Cuvelier
Université Paris XIII / Institut Galilée
L.A.G.A./Département de Mathématiques
http://www.math.univ-paris13.fr/∼cuvelier

Table des matières

Chapitre 1

Langage Algorithmique

1.1 Pseudo-langage algorithmique

Pour uniformiser l'écriture des algorithmes nous employons, un pseudo-langage contenant l'indispensable :

- variables,
- opérateurs (arithmétiques, relationnels, logiques),
- expressions,
- instructions (simples et composées),
- fonctions.

Ce pseudo-langage sera de fait très proche du langage de programmation de Matlab.

1.1.1 Données et constantes

Une donnée est une valeur introduite par l'utilisateur (par ex. une température, une vitesse, ... Une constante est un symbole ou un identificateur non modifiable (par ex. π, la constante de gravitation,...)

1.1.2 Variables

Définition 1.1.1. *Une variable est un objet dont la valeur est modifiable, qui possède un nom et un type (entier, charactère, réel, complexe, ...). Elle est rangée en mémoire à partir d'une certaine adresse.*

1.1.3 Opérateurs

Opérateurs arithmétiques

Nom	Symbole	example
addition	$+$	$a+b$
soustraction	$-$	$a-b$
opposé	$-$	$-a$
produit	$*$	$a*b$
division	$/$	a/b
puissance a^b	^	$a\hat{}b$

Table 1.1: Opérateurs arithmétiques

Opérateurs relationnels

Nom	Symbole	example	Commentaires
identique	$==$	$a == b$	vrai si a et b ont même valeur, faux sinon.
différent	$\sim=$	$a \sim= b$	faux si a et b ont même valeur, vrai sinon.
inférieur	$<$	$a < b$	vrai si a est plus petit que b, faux sinon.
supérieur	$>$	$a > b$	vrai si a est plus grand que b, faux sinon.
inférieur ou égal	$<=$	$a <= b$	vrai si a est plus petit ou égal à b, faux sinon.
supérieur ou égal	$>=$	$a >= b$	vrai si a est plus grand ou égal à b, faux sinon.

Table 1.2: Opérateurs relationnels

Opérateurs logiques

Nom	Symbole	example	Commentaires
négation	$\sim$	$\sim a$	vrai si a est faux (ou nul), faux sinon.
ou	$\vert$	$a\vert b$	vrai si a **ou** b est vrai (non nul), faux sinon.
et	$\&$	$a\&b$	vrai si a **et** b sont vrais (non nul), faux sinon.

Table 1.3: Opérateurs logiques

Opérateur d'affectation

Nom	Symbole	example	Commentaires
affectation	$\leftarrow$	$a \leftarrow b$	On affecte à la variable a le contenu de b

Table 1.4: Opérateurs d'affectation

1.1.4 Expressions

Définition 1.1.2. *Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)*

Exemple.

- *Voici un exemple classique d'expression numérique :*

$$(b*b-4*a*c)/(2*a).$$

On appelle ***opérandes*** *les identifiants a, b et c, et les nombres 4 et 2. Les symboles $*$, $-$ et $/$ sont les* ***opérateurs****.*

- *Voici un exemple classique d'expression booléenne (logique) :*

$$(x < 3.14)$$

Dans cette expression, x est une variable numérique et 3.14 est un nombre réel. Cette expression prendra la valeur vrai (i.e. 1) si x est plus petit que 3.14. Sinon, elle prendra la valeur faux (i.e. 0)

1.1.5 Instructions

Définition 1.1.3. *Une **instruction** est un ordre ou un groupe d'ordres qui déclenche l'exécution de certaines actions par l'ordinateur. Il y a deux types d'instructions : simple et structuré.*
*Les **instructions simples** sont essentiellement des ordres seuls et inconditionnels réalisant l'une des tâches suivantes :*

a. *affectation d'une valeur a une variable.*

b. *appel d'une fonction (procedure, subroutine, ... suivant les langages).*

*Les **instructions structurées** sont essentiellement :*

a. *les instructions composées, groupe de pulsieurs instructions simples,*

b. *les instructions répétitives, permettant l'exécution répétée d'instructions simples, (i.e. boucles «pour», «tant que»)*

c. *les instructions conditionnelles, lesquels ne sont exécutées que si une certaine condition est respectée (i.e. «si»)*

Les exemples qui suivent sont écrits dans un pseudo langage algorithmique mais sont facilement transposable dans la plupart des langages de programmation.

Instructions simples

Voici un exemple de l'*instruction simple d'affectation* :

1: $a \leftarrow 3.14 * R$

On évalue l'expression $3.14 * R$ et affecte le résultat à la variable a.
Un autre exemple est donné par l'*instruction simple d'affichage* :

```
affiche('bonjour')
```

Affiche la chaine de caractères `'bonjour'` à l'écran. Cette instruction fait appel à la fonction `affiche`.

Instructions composées

- **Instructions répétitives, boucle «pour»**

Algorithm 1.1 Exemple de boucle «pour»

Données : n : un entier.

1: $S \leftarrow 0$
2: **Pour** $i \leftarrow 1$ **à** n **faire**
3: $\quad S \leftarrow S + \cos(i^2)$
4: **Fin**

- **Instruction répétitive, boucle «tant que»**

Algorithm 1.2 Exemple de boucle «tant que»

1: $i \leftarrow 0,\ x \leftarrow 1$
2: **Tantque** $i < 1000$ **faire**
3: $\quad x \leftarrow x + i * i$
4: $\quad i \leftarrow i + 1$
5: **Fin**

- **Instruction répétitive, boucle «répéter ...jusqu'à»**

Algorithm 1.3 Exemple de boucle «répéter ...jusqu'à»

1: $i \leftarrow 0,\ x \leftarrow 1$
2: **Répéter**
3: $\quad x \leftarrow x + i * i$
4: $\quad i \leftarrow i + 1$
5: **jusqu'à** i>=1000

Instructions conditionnelles «si»

Algorithm 1.4 Exemple d'instructions conditionnelle «si»

Données : *age* : un entier de $\mathbb{N}$.

1: **Si** *age* >= 64 **alors**
2: affiche('retraite')
3: **Sinon Si** *age* >= 18 **alors**
4: affiche('majeur')
5: **Sinon**
6: affiche('mineur')
7: **Fin**

1.1.6 Fonctions

Les fonctions permettent

- d'automatiser certaines tâches répétitives au sein d'un même programme,
- d'ajouter à la clarté d'un programme,
- l'utilisation de portion de code dans un autre programme,
- ...

Fonctions prédéfinies

Pour faciliter leur usage, tous les langages de programmation possèdent des fonctions prédéfinies. On pourra donc supposer que dans notre langage algorithmique un grand nombre de fonctions soient prédéfinies : par exemple, les fonctions mathématiques sin, cos, exp, abs, $\cdots$ (pour ne citer quelles)

Syntaxe

On utilise la syntaxe suivante pour la définition d'une fonction

Fonction $[args_1, \ldots, args_n] \leftarrow$ NomFonction($arge_1, \ldots, arge_m$)
instructions
Fin

La fonction se nomme NomFonction. Elle admet comme paramètres d'entrée (données) les m arguments $arge_1, \ldots, arge_m$ et comme paramètres de sortie (résultats) les n arguments $args_1, \ldots, args_n$. Ces derniers doivent être déterminés dans le corps de la fonction (partie instructions).
Dans le cas ou la fonction n'admet qu'un seul paramètre de sortie, l'écriture se simplifie :

Fonction $args \leftarrow$ NomFonction($arge_1, \ldots, arge_m$)
  instructions
Fin

Ecrire ses propres fonctions

Pour écrire une fonction «propre», il faut tout d'abord déterminer exactement ce que devra faire cette fonction.
Puis, il faut pouvoir répondre à quelques questions :

a. Quelles sont les données (avec leurs limitations)?

b. Que doit-on calculer ?

Et, ensuite il faut la **commenter** : expliquer son usage, type des paramètres, ...

Exemple : résolution d'une équation du premier degré

Nous voulons écrire une fonction calculant la solution de l'équation

$$ax + b = 0,$$

où nous supposons que $a \in \mathbb{R}^*$ et $b \in \mathbb{R}$. La solution de ce problème est donc

$$x = -\frac{b}{a}.$$

Les données de cette fonction sont $a \in \mathbb{R}^*$ et $b \in \mathbb{R}$. Elle doit retourner $x = -\frac{b}{a}$ solution de $ax + b = 0$.

Algorithm 1.5 Exemple de fonction : Résolution de l'équation du premier degré $ax + b = 0$.

Données : a : nombre réel différent de 0
b : nombre réel.
Résultat : x : un réel.

1: **Fonction** $x \leftarrow$ REPD(a, b)
2: $x \leftarrow -b/a$
3: **Fin**

Remarque 1.1.1. *Cette fonction est très simple, toutefois pour ne pas «alourdir» le code nous n'avons pas vérifié la validité des données fournies.*

Exercice 1.1.1
Ecrire un algorithme permettant de valider cette fonction.

Exemple : résolution d'une équation du second degré

Nous cherchons les solutions réelles de l'équation

$$ax^2 + bx + c = 0, \tag{1.1}$$

où nous supposons que $a \in \mathbb{R}^*$, $b \in \mathbb{R}$ et $c \in \mathbb{R}$ sont donnés.
Mathématiquement, l'étude des solutions réelles de cette équation nous amène à envisager trois cas suivant les valeurs du discriminant $\Delta = b^2 - 4ac$

- si $\Delta < 0$ alors les deux solutions sont complexes,
- si $\Delta = 0$ alors la solution est $x = -\frac{b}{2a}$,
- si $\Delta > 0$ alors les deux solutions sont $x_1 = \frac{-b-\sqrt{\Delta}}{2*a}$ et $x_2 = \frac{-b+\sqrt{\Delta}}{2*a}$.

Exercice 1.1.2

a. Ecrire la fonction `discriminant` permettant de calculer le discriminant de l'équation (1.1).

b. Ecrire la fonction `RESD` permettant de résoudre l'équation (1.1) en utilisant la fonction `discriminant`.

c. Ecrire un programme permettant de valider ces deux fonctions.

Exercice 1.1.3

Même question que précédemment dans le cas complexe (solution et coefficients).

1.2 Méthodologie d'élaboration d'un algorithme

1.2.1 Description du problème

- Spécification d'un ensemble de données
  Origine : énoncé,hypothèses, sources externes, ...
- Spécification d'un ensemble de buts à atteindre
  Origine : résultats, opérations à effectuer, ...
- Spécification des contraintes

1.2.2 Recherche d'une méthode de résolution

- Clarifier l'énoncé.
- Simplifier le problème.
- Ne pas chercher à le traiter directement dans sa globalité.
- S'assurer que le problème est soluble (sinon problème d'indécidabilité!)
- Recherche d'une stratégie de construction de l'algorithme
- Décomposer le problème en sous problèmes partiels plus simples : raffinement.
- Effectuer des raffinements successifs.
- Le niveau de raffinement le plus élémentaire est celui des instructions.

1.2.3 Réalisation d'un algorithme

Il doit être conçu indépendamment du langage de programmation et du système informatique (sauf cas très particulier)

- L'algorithme doit être exécuté en un nombre fini d'opérations.
- L'algorithme doit être spécifié clairement, sans la moindre ambiguïté.
- Le type de données doit être précisé.
- L'algorithme doit fournir au moins un résultat.
- L'algorithme doit être effectif : toutes les opérations doivent pouvoir être simulées par un homme en temps fini.

Pour écrire un algorithme détaillé, il faut tout d'abord savoir répondre a quelques questions :

- Que doit-il faire ? (i.e. Quel problème est-il censé résoudre?)
- Quelles sont les données necessaires à la résolution de ce problème?
- Comment résoudre ce problème «à la main» (sur papier)?

Si l'on ne sait pas répondre à l'une de ces questions, l'écriture de l'algorithme est fortement comprise.

1.2.4 Exercices

Exercice 1.2.1 Algorithme pour une somme

Ecrire un algorithme permettant de calculer

$$S(x) = \sum_{k=1}^{n} k \sin(2 * k * x)$$

Correction 1.2.1 L'énoncé de cet exercice est imprécis. On choisi alors $x \in \mathbb{R}$ et $n \in \mathbb{N}$ pour rendre possible le calcul. Le problème est donc de calculer

$$\sum_{k=1}^{n} k \sin(2kx).$$

Toutefois, on aurait pu choisir $x \in \mathbb{C}$ ou encore un tout autre problème :

$$\text{Trouver } x \in \mathbb{R} \text{ tel que } S(x) = \sum_{k=1}^{n} k \sin(2kx)$$

où $n \in \mathbb{N}$ et S, fonction de $\mathbb{R}$ à valeurs réelles, sont les données!

Algorithme 1.6 Calcul de $S = \sum_{k=1}^{n} k \sin(2kx)$

Données : x : nombre réel,
n : nombre entier.
Résultat : S : un réel.

1: $S \leftarrow 0$
2: **Pour** $k \leftarrow 1$ **à** n **faire**
3: $S \leftarrow S + k * \sin(2 * k * x)$
4: **Fin**

Fin correction

Exercice 1.2.2 Algorithme pour un produit

Ecrire un algorithme permettant de calculer

$$P(z) = \prod_{n=1}^{k} sin(2 * k * z/n)^k$$

Correction 1.2.2 L'énoncé de cet exercice est imprécis. On choisi alors $z \in \mathbb{R}$ et $k \in \mathbb{N}$ pour rendre possible le calcul.

Algorithme 1.7 Calcul de $P = \prod_{n=1}^{k} \sin(2kz/n)^k$

Données : z : nombre réel,
k : nombre entier.
Résultat : P : un réel.

```
1: P ← 1
2: Pour n ← 1 à k faire
3:     P ← P * sin(2 * k * z/n)^k
4: Fin
```

Fin correction

Exercice 1.2.3 Série de Fourier

Soit la série de Fourier

$$x(t) = \frac{4A}{\pi}\left\{\cos\omega t - \frac{1}{3}\cos 3\omega t + \frac{1}{5}\cos 5\omega t - \frac{1}{7}\cos 7\omega t + \cdots\right\}.$$

Ecrire la fonction SFT permettant de calculer $x_n(t)$ correspondant à la série $x(t)$ tronquée au n-ième terme.

Correction 1.2.3 Nous devons écrire la fonction permettant de calculer

$$x_n(t) = \frac{4A}{\pi}\sum_{k=1}^{n}(-1)^{k+1}\frac{1}{2k-1}\cos((2k-1)\omega t)$$

Les données de la fonction sont $A \in \mathbb{R}$, $\omega \in \mathbb{R}$, $n \in \mathbb{N}^*$ et $t \in \mathbb{R}$. Grâce a ces renseignements nous pouvons déjà écrire l'entête de la fonction :

Algorithme 1.8 En-tête de la fonction SFT retournant valeur de la série de Fourier en t tronquée au n premiers termes de l'exercice 1.2.3.

Données : t : nombre réel,
n : nombre entier strictement positif
$A,\ \omega$: deux nombres réels.
Résultat : x : un réel.

```
1: Fonction x ← SFT( t, n, A, ω )
2:     ...
3: Fin
```

Maintenant nous pouvons écrire progressivement l'algorithme pour aboutir au final à une version ne contenant que des opérations élémentaires.

Algorithme 1.9 $\mathcal{R}_0$

1: $x \leftarrow \frac{4A}{\pi}\sum_{k=1}^{n}\left(\begin{array}{c}(-1)^{k+1}\frac{1}{2k-1}\times\\ \cos((2k-1)\omega t)\end{array}\right)$

Algorithme 1.9 $\mathcal{R}_1$

1: $S \leftarrow \sum_{k=1}^{n}(-1)^{k+1}\frac{1}{2k-1}\cos((2k-1)\omega t)$
2: $x \leftarrow \frac{4A}{\pi}S$

Algorithme 1.9 $\mathcal{R}_1$

1: $S \leftarrow \sum_{k=1}^{n} (-1)^{k+1} \frac{1}{2k-1} \cos((2k-1)\omega t)$
2: $x \leftarrow \frac{4A}{\pi} S$

Algorithme 1.9 $\mathcal{R}_2$

1: $S \leftarrow 0$
2: **Pour** $k = 1$ **à** n **faire**
3: $\quad S \leftarrow S + (-1)^{k+1} \frac{1}{2k-1} * \cos((2k-1)\omega t)$
4: **Fin**
5: $x \leftarrow \frac{4A}{\pi} S$

Finalement la fonction est

Algorithme 1.9 Fonction SFT retournant la valeur de la série de Fourier en t tronquée au n premiers termes de l'exercice 1.2.3.

Données : t : nombre réel,
n : nombre entier strictement positif
$A,\ \omega$: deux nombres réels.
Résultat : x : un réel.

```
1: Fonction x ← SFT( t, n, A, ω )
2:   S ← 0
3:   Pour k = 1 à n faire
4:     S ← S + ((−1)^(k + 1)) ∗ cos((2 ∗ k − 1) ∗ ω ∗ t)/(2 ∗ k − 1)
5:   Fin
6:   x ← 4 ∗ A ∗ S/π
7: Fin
```

Fin correction

Exercice 1.2.4

Reprendre les trois exercices précédents en utilisant les boucles «tant que».

1.3 Principes de «bonne» programmation pour attaquer de «gros» problèmes

Tous les exemples vus sont assez courts. Cependant, il peut arriver que l'on ait des programmes plus longs à écrire (milliers de lignes, voir des dizaines de milliers de lignes). Dans l'industrie, il arrive que des équipes produisent des codes de millions de lignes, dont certains mettent en jeux des vies humaines (contrôler un avion de ligne, une centrale nucléaire, ...). Le problème est évidemment d'écrire des programmes sûrs. Or, *un programme à* 100% *sûr, cela n'existe pas!* Cependant, plus un programme est simple, moins le risque d'erreur est grand : c'est sur cette remarque de bon sens que se basent les «bonnes» méthodes. Ainsi :

Tout problème compliqué doit être découpé en sous-problèmes plus simples

Il s'agit, lorsqu'on a un problème P à résoudre, de l'analyser et de le décomposer en un ensemble de problèmes P_1, P_2, $P_3, \ldots$ plus simples. Puis, P_1, est lui-même analysé et décomposé en $P_{11}, P_{12}, \ldots$, et P_2 en P_{21}, P_{22}, etc. On poursuit cette analyse jusqu'à ce qu'on n'ait plus que des problèmes élémentaires à résoudre. Chacun de ces problèmes élémentaires est donc traité séparément dans un module, c'est à dire un morceau de programme relativement indépendant du reste. Chaque module sera *testé et validé* séparément dans la mesure du possible et naturellement *largement documenté*. Enfin ces modules élémentaires sont assemblés en modules de plus en plus complexes, jusqu'à remonter au problème initiale. A chaque niveau, il sera important de bien réaliser les phases de test, validation et documentation des modules.

Par la suite, on s'évertue à écrire des algorithmes!
Ceux-ci ne seront pas optimisés[a]!

[a] améliorés pour minimiser le nombre d'opérations élémentaires, l'occupation mémoire, ..

1.4 Quelques simplications d'écriture

Pour simplifier l'écriture des algorithmes numériques, il sera possible d'utiliser les "raccourcis" suivant pour accéder aux éléments d'une matrice $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ (ou tableau à m lignes et n colonnes)

- $\mathbb{A}(:,j)$ correspond au j-ème vecteur colonne de $\mathbb{A}$, $j \in [\![1,n]\!]$. Si on écrit $\boldsymbol{v} \leftarrow \mathbb{A}(:,j)$ alors l'accès aux éléments de $\boldsymbol{v} \in \mathbb{R}^m$ (ou tableau à m éléments) s'effectue avec la commande $\boldsymbol{v}(i)$ avec $i \in [\![1,m]\!]$. De plus, au niveau algorithmique, si $\boldsymbol{w}$ est un vecteur colonne ou ligne de dimension m, alors $\mathbb{A}(:,j) \leftarrow \boldsymbol{w}$ est autorisé et correspond mathématiquement à $\mathbb{A}_{i,j} = \boldsymbol{w}_i$, $\forall i \in [\![1,n]\!]$.

- $\mathbb{A}(i,:)$ correspond au i-ème vecteur ligne de $\mathbb{A}$, $i \in [\![1,m]\!]$. Si on écrit $\boldsymbol{u} \leftarrow \mathbb{A}(i,:)$ alors l'accès aux éléments de $\boldsymbol{u} \in \mathbb{R}^n$ (ou tableau à n éléments) s'effectue avec la commande $\boldsymbol{u}(j)$ avec $j \in [\![1,n]\!]$. De plus, au niveau algorithmique, si $\boldsymbol{z}$ est un vecteur colonne ou ligne de dimension n, alors $\mathbb{A}(i,:) \leftarrow \boldsymbol{z}$ est autorisé et correspond mathématiquement à $\mathbb{A}_{i,j} = \boldsymbol{z}_j$, $\forall j \in [\![1,n]\!]$.

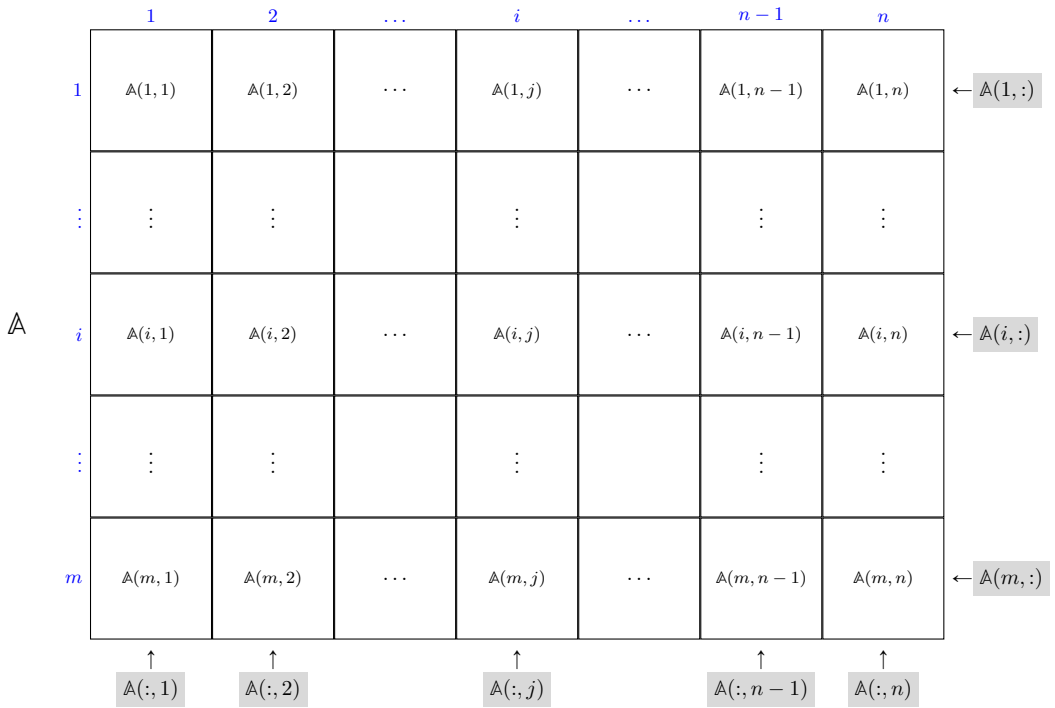


Figure 1.1: Version simplifiée d'acces aux lignes et colonnes d'une matrice $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$.

Avec les fonctions écrites dans l'Exercice 1.4.4, p.17, on a

Description	*Avec fonction*	*Instruction simplifiée*
Obtenir la j-ème colonne d'une matrice $\mathbb{A}$ avec $j \in [\![1, n]\!]$, math. noté $\mathbb{A}_{:,j}$	$\boldsymbol{u} \leftarrow \text{getMatCol}(\mathbb{A}, j)$	$\boldsymbol{u} \leftarrow \mathbb{A}(:, j)$
Obtenir la i-ème ligne d'une matrice $\mathbb{A}$ avec $i \in [\![1, m]\!]$, math. noté $\mathbb{A}_{i,j}$	$\boldsymbol{v} \leftarrow \text{getMatRow}(\mathbb{A}, i)$	$\boldsymbol{v} \leftarrow \mathbb{A}(i, :)$
Remplacer la j-ème colonne d'une matrice $\mathbb{A}$ par un vecteur $\boldsymbol{u} \in \mathbb{R}^m$ avec $j \in [\![1, n]\!]$	$\mathbb{A} \leftarrow \text{setMatCol}(\mathbb{A}, \boldsymbol{u}, j)$	$\mathbb{A}(:, j) \leftarrow \boldsymbol{u}$
Remplacer la i-ème ligne d'une matrice $\mathbb{A}$ par un vecteur $\boldsymbol{v} \in \mathbb{R}^n$ avec $i \in [\![1, m]\!]$	$\mathbb{A} \leftarrow \text{setMatRow}(\mathbb{A}, \boldsymbol{v}, j)$	$\mathbb{A}(i, :) \leftarrow \boldsymbol{v}$.

Table 1.5: Commandes algorithmiques permettant l'accès à l'intégralité d'une ligne ou d'une colonne d'un tableau à m lignes et n colonne de noté $\mathbb{A}$ que l'on identifiera à une matrice de $\mathcal{M}_{m,n}(\mathbb{R})$.

1.4.1 Exercices simples d'algèbre linéaire

Exercices sur les vecteurs

Exercice 1.4.1

Soient $\boldsymbol{u} \in \mathbb{R}^n$ et $\boldsymbol{v} \in \mathbb{R}^n$.

Q. 1 *Ecrire une fonction* dot *permettant de calculer le produit scalaire du vecteur $\boldsymbol{u}$ par $\boldsymbol{v}$, noté mathématiquement par $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$.*

Q. 2 *Ecrire une fonction* norm2 *permettant de calculer la norme euclidienne du vecteur $\boldsymbol{u}$ donnée par $\|\boldsymbol{u}\|_2 \stackrel{\text{def}}{=} \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle}$.*

Soient a et b deux réels.

Q. 3 *Ecrire une fonction* aUpbV *permettant de calculer le vecteur $\boldsymbol{w} \stackrel{\text{def}}{=} a\boldsymbol{u} + b\boldsymbol{v}$.*

Q. 4 *Ecrire un programme algorithmique permettant de calculer $\|\boldsymbol{u} - \boldsymbol{v}\|_2$ avec $\boldsymbol{u} = \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}$ et $\boldsymbol{v} = \begin{pmatrix} 2 \\ -3 \\ 2 \end{pmatrix}$*

Correction

R. 1 On a

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} \boldsymbol{u}_i \boldsymbol{v}_i \in \mathbb{R}.$$

Voici une correction possible où l'on différencie la notation mathématique $\boldsymbol{u}_i$, i-ème composante du vecteur $\boldsymbol{u}$, et la notation algorithmique $\boldsymbol{U}(i)$ permettant d'accéder à l'élément i du tableau/vecteur $\boldsymbol{U}$, avec $i \in [\![1, n]\!]$.

Algorithme 1.10 Fonction dot, produit scalaire entre deux vecteurs.

Données : $\boldsymbol{U}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1,n]\!], \boldsymbol{U}(i) = \boldsymbol{u}_i$.
$\boldsymbol{V}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1,n]\!], \boldsymbol{V}(i) = \boldsymbol{v}_i$.
Résultat : r : le réel tel que $r = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$.

```
1: Fonction r ← dot( U, V )
2:     r ← 0
3:     Pour i ← 1 à n faire
4:         r ← r + U(i) * V(i)
5:     Fin
6: Fin
```

On peut aussi utiliser la notation algorithmique $\boldsymbol{x} \in \mathbb{R}^p$ pour décrire un vecteur/tableau de $\mathbb{R}^p$ et on a alors la correspondance suivante entre notation mathématique et algorithmique:

Description	*Mathématiques*	*Algorithmique*
Composante $i \in [\![1,p]\!]$ de $\boldsymbol{x} \in \mathbb{R}^p$	$\boldsymbol{x}_i$	$\boldsymbol{x}(i)$

Cependant pour bien différencier ces deux types de notations, on continue dans cet exercice à noter $\boldsymbol{U}$ le vecteur algorithmique correpondant au vecteur mathématique $\boldsymbol{u}$.

R. 2

Voici une correction possible:

Algorithme 1.11 Fonction norm2

Données : $\boldsymbol{U}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1,n]\!], \boldsymbol{U}(i) = \boldsymbol{u}_i$.
Résultat : r : le réel tel que $r = \|\boldsymbol{u}\|_2 \stackrel{\text{def}}{=} \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle}$.

```
1: Fonction r ← norm2( U )
2:     r ← Sqrt(dot(U, U))
3: Fin
```

R. 3

On a $\boldsymbol{w} \stackrel{\text{def}}{=} a\boldsymbol{u} + b\boldsymbol{v} \in \mathbb{R}^n$ et plus précisemment

$$\forall i \in [\![1,n]\!], \quad \boldsymbol{w}_i \stackrel{\text{def}}{=} a\boldsymbol{u}_i + b\boldsymbol{v}_i.$$

Voici une correction possible:

Algorithme 1.12 Fonction aUpbV

Données : $\boldsymbol{U}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1,n]\!]$, $\boldsymbol{U}(i) = \boldsymbol{u}_i$.
$\boldsymbol{V}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1,n]\!]$, $\boldsymbol{V}(i) = \boldsymbol{v}_i$.
a : un réel,
b : un réel.
Résultat : $\boldsymbol{W}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1,n]\!]$, $\boldsymbol{W}(i) = \boldsymbol{w}_i$, avec $\boldsymbol{w}_i \overset{\text{def}}{=} a\boldsymbol{u}_i + b\boldsymbol{v}_i$.

```
1: Fonction W ← aUpbV( a, U, b, V )
2:   Pour i ← 1 à n faire
3:     W(i) ← a * U(i) + b * V(i)
4:   Fin
5: Fin
```

R. 4 Voici un premier programme possible:

$$\text{1: } \boldsymbol{U} \leftarrow \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}, \quad \boldsymbol{V} \leftarrow \begin{pmatrix} 2 \\ -3 \\ 2 \end{pmatrix}$$
$$\text{2: } \boldsymbol{x} \leftarrow \text{norm2}\big(\text{aUpbV}(1, \boldsymbol{U}, -1, \boldsymbol{V})\big) \qquad \triangleright\ x \text{ contient le réel } \|\boldsymbol{u} - \boldsymbol{v}\|_2$$

On peut aussi écrire directement

$$\text{1: } \boldsymbol{x} \leftarrow \text{norm2}\Big(\text{aUpbV}\Big(1, \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}, -1, \begin{pmatrix} 2 \\ -3 \\ 2 \end{pmatrix}\Big)\Big) \qquad \triangleright\ x \text{ contient le réel } \|\boldsymbol{u} - \boldsymbol{v}\|_2$$

Fin correction

Exercice 1.4.2

Q. 1 *Ecrire une fonction* VecZeros *retournant le vecteur nul de* $\mathbb{R}^n$.

Q. 2 *Ecrire une fonction* VecConst *retournant le vecteur de* $\mathbb{R}^n$ *dont toutes les composantes valent* $\alpha \in \mathbb{R}$.

Q. 3 *Soient* $\boldsymbol{u} \in \mathbb{R}^n$ *et* $\alpha \in \mathbb{R}$. *Ecrire une fonction* VecPlusConst *retournant le vecteur de* $\boldsymbol{v} \in \mathbb{R}^n$ *tel que*

$$\forall i \in [\![1,n]\!], \quad \boldsymbol{v}_i = \boldsymbol{u}_i + \alpha.^a$$

[a] Cette opération n'est pas une opération algébrique dans $\mathbb{R}^n$, c'est à dire mathématiquent, on ne peut pas écrire $\boldsymbol{u} + \alpha$!

On suppose que l'on dispose de la fonction algorithmique rand() retournant un réel aléatoire suivant la loi uniforme sur $[0,1]$.

Q. 4 *Ecrire une fonction* VecRand *retournant un vecteur de* $\mathbb{R}^n$ *dont toutes les composantes sont aléatoires suivant la loi uniforme sur* $[a,b]$.

Correction

R. 1

Voici une correction possible

Algorithme 1.13 Fonction VecZeros retournant le vecteur nul de $\mathbb{R}^n$.

Données : n : un entier > 0.
Résultat : $\boldsymbol{z}$: le vecteur nul de $\mathbb{R}^n$.

```
1: Fonction z ← VecZeros( n )
2:   Pour i ← 1 à n faire
3:     z(i) ← 0
4:   Fin
5: Fin
```

R. 2

Voici une correction possible

Algorithme 1.14 Fonction VecConst retournant le vecteur $\boldsymbol{v}$ de $\mathbb{R}^n$ dont toutes les composantes valent $\alpha \in \mathbb{R}$.

Données : n : un entier > 0,
α : un réel.
Résultat : $\boldsymbol{v}$: vecteur de $\mathbb{R}^n$.

```
1: Fonction v ← VecConst( n, α )
2:   Pour i ← 1 à n faire
3:     v(i) ← α
4:   Fin
5: Fin
```

R. 3

Voici une correction possible

Algorithme 1.15 Fonction VecPlusConst retournant le vecteur $\boldsymbol{v}$ de $\mathbb{R}^n$ tel que $\forall i \in [\![1, n]\!]$, $\boldsymbol{v}_i = \boldsymbol{u}_i + \alpha$ avec

Données : $\boldsymbol{u}$: vecteur de $\mathbb{R}^n$,
α : un réel.
Résultat : $\boldsymbol{v}$: vecteur de $\mathbb{R}^n$.

```
1: Fonction v ← VecPlusConst( u, α )
2:   Pour i ← 1 à n faire
3:     v(i) ← u(i) + α
4:   Fin
5: Fin
```

R. 4

On rappelle le changement de variable permettant de passer de l'intervalle $[0, 1]$ à l'intervalle $[a, b]$

$$\forall t \in [0, 1], \quad x = a + (b - a)t, \text{ avec } x \in [a, b].$$

Donc, si T est une variable aléatoire suivant la loi uniforme sur $[0, 1]$, alors $X = a + (b - a)T$ est une variable aléatoire qui suit la loi uniforme sur $[a, b]$.
Voici une correction possible

Algorithme 1.16 Fonction VecRand retournant un vecteur dont toutes les composantes sont aléatoires suivant la loi uniforme sur $[a, b]$.

Données :	n	:	un entier > 0,
	a	:	un réel,
	b	:	un réel, tel que $a < b$.
Résultat :	$\boldsymbol{x}$	:	vecteur de $\mathbb{R}^n$.

```
1: Fonction x ← VecRand( n, a, b )
2:   Pour i ← 1 à n faire
3:     x(i) ← a + (b − a) ∗ rand()
4:   Fin
5: Fin
```

Fin correction

1.4.2 Exercices sur les matrices

Exercice 1.4.3

Q. 1 *Ecrire une fonction* MatZeros *retournant la matrice nulle de* $\mathcal{M}_{m,n}(\mathbb{R})$.

Q. 2 *Ecrire une fonction* MatConst *retournant la matrice de* $\mathcal{M}_{m,n}(\mathbb{R})$ *dont toutes les composantes valent* $\alpha \in \mathbb{R}$.

Q. 3 *Soient* $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ *et* $\alpha \in \mathbb{R}$. *Ecrire une fonction* MatPlusConst *retournant la matrice* $\mathbb{B} \in \mathcal{M}_{m,n}(\mathbb{R})$ *telle que*

$$\forall (i,j) \in [\![1,m]\!] \times [\![1,n]\!], \quad \mathbb{B}_{i,j} = \mathbb{A}_{i,j} + \alpha.^a$$

[a] Cette opération n'est pas une opération algébrique dans $\mathcal{M}_{m,n}(\mathbb{R})$, c'est à dire, mathématiquement, on ne peut pas écrire $\mathbb{A} + \alpha$!

On suppose que l'on dispose de la fonction algorithmique rand() retournant un réel aléatoire suivant la loi uniforme sur $[0, 1]$.

Q. 4 *Ecrire une fonction* MatRand *retournant une matrice de* $\mathcal{M}_{m,n}(\mathbb{R})$ *dont toutes les composantes sont aléatoires suivant la loi uniforme sur* $[a, b]$.

Correction

R. 1 Voici une correction possible

Algorithme 1.17 Fonction MatZeros la matrice nulle de $\mathcal{M}_{m,n}(\mathbb{R})$.

Données : m : un entier > 0,
n : un entier > 0.
Résultat : $\mathbb{Z}$: la matrice nulle de $\mathcal{M}_{m,n}(\mathbb{R})$.

```
1: Fonction Z ← MatZeros( m, n )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à n faire
4:       Z(i, j) ← 0
5:     Fin
6:   Fin
7: Fin
```

R. 2

Voici une correction possible

Algorithme 1.18 Fonction MatConst retournant la matrice de $\mathcal{M}_{m,n}(\mathbb{R})$ dont toutes les composantes valent $\alpha \in \mathbb{R}$.

Données : m : un entier > 0,
n : un entier > 0,
α : un réel.
Résultat : $\mathbb{Z}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$.

```
1: Fonction v ← MatConst( m, n, α )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à n faire
4:       Z(i, j) ← α
5:     Fin
6:   Fin
7: Fin
```

R. 3

Voici une correction possible

Algorithme 1.19 Fonction MatPlusConst retournant la matrice $\mathbb{B} \in \mathcal{M}_{m,n}(\mathbb{R})$ telle que $\forall (i,j) \in [\![1,m]\!] \times [\![1,n]\!]$, $\mathbb{B}_{i,j} = \mathbb{A}_{i,j} + \alpha$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
α : un réel.
Résultat : $\mathbb{B}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$.

```
1: Fonction B ← MatConst( A, α )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à n faire
4:       B(i, j) ← A(i, j) + α
5:     Fin
6:   Fin
7: Fin
```

R. 4

On rappelle le changement de variable permettant de passer de l'intervalle $[0,1]$ à l'intervalle $[a,b]$

$$\forall t \in [0,1], \quad x = a + (b-a)t, \text{ avec } x \in [a,b].$$

Donc, si T est une variable aléatoire suivant la loi uniforme sur $[0,1]$, alors $X = a + (b-a)T$ est une variable aléatoire qui suit la loi uniforme sur $[a,b]$.
Voici une correction possible

Algorithme 1.20 Fonction MatRand retournant une matrice de $\mathcal{M}_{m,n}(\mathbb{R})$ dont toutes les composantes sont aléatoires suivant la loi uniforme sur $[a,b]$.

Données :	m	:	un entier > 0,
	n	:	un entier > 0,
	a	:	un réel,
	b	:	un réel, tel que $a < b$.
Résultat :	$\mathbb{X}$	:	matrice de $\mathcal{M}_{m,n}(\mathbb{R})$.

```
1: Fonction 𝕏 ← MatRand( m, n, a, b )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à n faire
4:       𝕏(i, j) ← a + (b − a) ∗ rand()
5:     Fin
6:   Fin
7: Fin
```

Fin correction

Exercice 1.4.4

Soit $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$.

Q. 1

a. *Ecrire une fonction algorithmique* setMatCol *permettant d'initialiser le j-ème vecteur colonne de $\mathbb{A}$ par un vecteur $\boldsymbol{x} \in \mathbb{R}^m$, avec $j \in [\![1,n]\!]$.*

b. *Ecrire une fonction algorithmique* getMatCol *retournant le j-ème vecteur colonne de $\mathbb{A}$ avec $j \in [\![1,n]\!]$.*

Q. 2

a. *Ecrire une fonction algorithmique* setMatRow *permettant d'initialiser le i-ème vecteur ligne de $\mathbb{A}$ par un vecteur $\boldsymbol{y} \in \mathbb{R}^n$, avec $i \in [\![1,m]\!]$.*

b. *Ecrire une fonction algorithmique* getMatRow *retournant le i-ème vecteur ligne de $\mathbb{A}$ avec $i \in [\![1,m]\!]$.*

Correction

R. 1

Mathématiquement, le j-ème vecteur colonne de $\mathbb{A}$ le vecteur (colonne) de $\mathbb{R}^m$ donné par

$$\begin{pmatrix} \mathbb{A}_{1,j} \\ \vdots \\ \mathbb{A}_{m,j} \end{pmatrix}.$$

a. Voici une correction possible

Algorithme 1.21 Fonction setMatCol

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{n,m}(\mathbb{R})$
$\boldsymbol{x}$: vecteur de $\mathbb{R}^n$,
j : entier dans $[\![1, m]\!]$.
Résultat : $\mathbb{A}$: la matrice $\mathbb{A}$ modifiée.

```
1: Fonction A ← setMatCol( A, x, j )
2:     Pour i ← 1 à n faire
3:         A(i, j) ← x(i)
4:     Fin
5: Fin
```

b. Voici une correction possible

Algorithme 1.22 Fonction getMatCol

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$
j : entier dans $[\![1, n]\!]$.
Résultat : $\boldsymbol{x}$: vecteur de $\mathbb{R}^m$, colonne j de $\mathbb{A}$.

```
1: Fonction x ← getMatCol( A, j )
2:     Pour i ← 1 à n faire
3:         x(i) ← A(i, j)
4:     Fin
5: Fin
```

R. 2

Le i-ème vecteur ligne de $\mathbb{A}$ est le vecteur ligne de $\mathbb{R}^n$ donné par

$$\begin{pmatrix} \mathbb{A}_{i,1} & \dots & \mathbb{A}_{i,n} \end{pmatrix}.$$

a. Voici une correction possible

Algorithme 1.23 Fonction setMatRow

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$
$\boldsymbol{x}$: vecteur de $\mathbb{R}^n$,
i : entier dans $[\![1, m]\!]$.
Résultat : $\mathbb{A}$: la matrice $\mathbb{A}$ modifiée.

```
1: Fonction A ← setMatRow( A, x, i )
2:     Pour j ← 1 à n faire
3:         A(i, j) ← x(j)
4:     Fin
5: Fin
```

b. Voici une correction possible

Algorithme 1.24 Fonction getMatRow

Données :	$\mathbb{A}$	:	matrice de $\mathcal{M}_{m,n}(\mathbb{R})$
	i	:	entier dans $[\![1, m]\!]$.
Résultat :	$\boldsymbol{x}$	:	vecteur de $\mathbb{R}^n$, ligne i de $\mathbb{A}$.

```
1: Fonction x ← getMatCol( A, i )
2:   Pour j ← 1 à m faire
3:     x(j) ← A(i, j)
4:   Fin
5: Fin
```

Fin correction

Exercice 1.4.5

Q. 1 *Soient $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$, $\mathbb{B} \in \mathcal{M}_{m,n}(\mathbb{R})$, $\alpha \in \mathbb{R}$ et $\beta \in \mathbb{R}$ Ecrire une fonction* aApbB *permettant de calculer la matrice $\mathbb{W} \overset{\text{def}}{=} \alpha\mathbb{A} + \beta\mathbb{B}$.*

Q. 2 *Soient $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ et $\boldsymbol{u}$ un vecteur (colonne).*

a. *A quelle(s) condition(s) le produit de la matrice $\mathbb{A}$ par le vecteur $\boldsymbol{u}$ existe-t'il?*

b. *Sous les conditions précédentes, on note $\boldsymbol{v} \overset{\text{def}}{=} \mathbb{A}\boldsymbol{u}$.*

- *A quel espace appartient $\boldsymbol{v}$?*
- *Rappeler précisement la formule permettant de calculer les composantes de $\boldsymbol{v}$.*

c. *Ecrire une fonction* ProdMatVec *permettant de calculer $\boldsymbol{v} \overset{\text{def}}{=} \mathbb{A}\boldsymbol{u}$.*

d. *Donner un exemple algorithmique d'utilisation.*

Correction

R. 1 La matrice $\mathbb{W}$ est aussi dans $\mathcal{M}_{m,n}(\mathbb{R})$ et on a

$$\forall (i,j) \in [\![1,m]\!] \times [\![1,n]\!], \quad \mathbb{W}_{i,j} = \alpha\mathbb{A}_{i,j} + \beta\mathbb{B}_{i,j}.$$

Voici une correction possible

Algorithme 1.25 Fonction aApbB retournant la matrice $\mathbb{W} \stackrel{\text{def}}{=} \alpha\mathbb{A} + \beta\mathbb{B}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\mathbb{B}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
α : un réel,
β : un réel.
Résultat : $\mathbb{W}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$.

```
1: Fonction W ← aApbB( α, A, β, B )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à n faire
4:       W(i,j) ← α * A(i,j) + β * B(i,j)
5:     Fin
6:   Fin
7: Fin
```

R. 2

a. Pour que le produit de la matrice $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ par un vecteur (colonne) $\boldsymbol{u}$ soit possible il faut et il suffit que la dimension du vecteur (nombre de lignes) soit égale au nombre de colonnes de $\mathbb{A}$, et donc $\boldsymbol{u} \in \mathbb{R}^n$.

b. Avec $\boldsymbol{u} \in \mathbb{R}^n$, on a alors $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u} \in \mathbb{R}^m$. et

$$\forall i \in [\![1, m]\!], \quad \boldsymbol{v}_i = \sum_{j=1}^{n} \mathbb{A}_{i,j}\boldsymbol{u}_j.$$

c. Voici une manière d'obtenir l'algorithme en utilisant une méthode de raffinements successifs et les notations mathématiques pour accéder aux composantes des vecteurs et des matrices

Algorithme 1.26 $\mathcal{R}_0$

```
1: Calcul de v
```

Algorithme 1.26 $\mathcal{R}_1$

```
1: Pour i = 1 à m faire
2:   v_i ← Σ_{j=1}^{n} A_{i,j} u_j
3: Fin
```

Algorithme 1.26 $\mathcal{R}_1$

```
1: Pour i ← 1 à m faire
2:   v_i ← Σ_{j=1}^{n} A_{i,j} u_j
3: Fin
```

Algorithme 1.26 $\mathcal{R}_2$

```
1: Pour i ← 1 à m faire
2:   v_i ← 0
3:   Pour j ← 1 à n faire
4:     v_i ← v_i + A_{i,j} u_j
5:   Fin
6: Fin
```

Pour écrire la fonction algorithmique, il nous faut passer des notations mathématiques aux notations algorithmiques pour les matrices et vecteurs. Les correspondances sont données en Table 1.8.

Description	*Mathématiques*	*Algorithmique*
Composante $i \in [\![1,p]\!]$ de $\boldsymbol{x} \in \mathbb{R}^p$	$\boldsymbol{x}_i$	$\boldsymbol{x}(i)$
Composante $(i,j) \in [\![1,p]\!] \times [\![1,q]\!]$ de $\mathbb{Z} \in \mathcal{M}_{p,q}(\mathbb{R})$,	$\mathbb{Z}_{i,j}$	$\mathbb{Z}(i,j)$

Table 1.6: Correspondance entre les notations mathématiques et algorithmiques pour les matrices et vecteurs

On obtient alors la fonction

Algorithme 1.26 Fonction ProdMatVec retournant le vecteur $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$, produit de la matrice $\mathbb{A}$ par le vecteur $\boldsymbol{u}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\boldsymbol{u}$: vecteur de $\mathbb{R}^n$.
Résultat : $\boldsymbol{v}$: vecteur de $\mathbb{R}^m$ tel que $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$.

```
1: Fonction v ← ProdMatVec( A, u )
2:   Pour i ← 1 à m faire
3:     v(i) ← 0
4:     Pour j ← 1 à n faire
5:       v(i) ← v(i) + A(i, j) ∗ u(j)
6:     Fin
7:   Fin
8: Fin
```

d. Par exemple,

$$\boldsymbol{x} \leftarrow \text{ProdMatVec}\left(\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix}, \begin{pmatrix}1\\2\\3\end{pmatrix}\right)$$

Fin correction

Exercice 1.4.6

Q. 1 *Soient $\boldsymbol{u}$ et $\boldsymbol{v}$ deux vecteurs de $\mathbb{R}^n$. Ecrire la fonction* `ProdSca` *permettant de retourner le produit scalaire de ces deux vecteurs noté mathématiquement $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$.*

L'objectif de la suite de l'exercice est de voir plusieurs manières de programmer le produit d'une matrice par un vecteur. Soient $\boldsymbol{u} \in \mathbb{R}^p$ et $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$.

Q. 2

a. *Rappeler précisement les hypothèses et les formules permettant le calcul de $\boldsymbol{v} = \mathbb{A}\boldsymbol{u}$.*

b. *Ecrire la fonction* ProdMatVec *permettant de retourner $\mathbb{A}\boldsymbol{u}$.*

Voici quelques notations mathématiques (non standards) que l'on va utiliser. Si $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ alors

- $\mathbb{A}_{:,j}$ correspond au j-ème vecteur colonne de $\mathbb{A}$.
- $\mathbb{A}_{i,:}$ correspond au i-ème vecteur ligne de $\mathbb{A}$.

On suppose que l'on dispose des commandes suivantes au niveau algorithmique:

Description	*Avec fonction*	*Instruction simplifiée*
Obtenir la j-ème colonne d'une matrice $\mathbb{A}$ avec $j \in [\![1, n]\!]$, math. noté $\mathbb{A}_{:,j}$	$\boldsymbol{u} \leftarrow$ getMatCol$(\mathbb{A}, j)$	$\boldsymbol{u} \leftarrow \mathbb{A}(:, j)$
Obtenir la i-ème ligne d'une matrice $\mathbb{A}$ avec $i \in [\![1, m]\!]$, math. noté $\mathbb{A}_{i,:}$	$\boldsymbol{v} \leftarrow$ getMatRow$(\mathbb{A}, i)$	$\boldsymbol{v} \leftarrow \mathbb{A}(i, :)$
Remplacer la j-ème colonne d'une matrice $\mathbb{A}$ par un vecteur $\boldsymbol{u} \in \mathbb{R}^m$ avec $j \in [\![1, n]\!]$	$\mathbb{A} \leftarrow$ setMatCol$(\mathbb{A}, \boldsymbol{u}, j)$	$\mathbb{A}(:, j) \leftarrow \boldsymbol{u}$
Remplacer la i-ème ligne d'une matrice $\mathbb{A}$ par un vecteur $\boldsymbol{v} \in \mathbb{R}^n$ avec $i \in [\![1, m]\!]$	$\mathbb{A} \leftarrow$ setMatRow$(\mathbb{A}, \boldsymbol{v}, j)$	$\mathbb{A}(i, :) \leftarrow \boldsymbol{v}$.

Table 1.7: Instructions algorithmiques pour accéder/modifier une ligne ou une colonne d'une matrice $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$.

Q. 3

a. *Sous les hypothèses de* ***Q.2*** *et avec* $\boldsymbol{v} = \mathbb{A}\boldsymbol{u}$, *écrire mathématiquement* $\boldsymbol{v}_i$ *comme un produit scalaire en utilisant les notations précisées ci-dessus.*

b. *Ecrire la fonction* ProdMatVecFun *permettant de retourner* $\mathbb{A}\boldsymbol{u}$ *en utilisant la fonction* ProdSca *et les fonctions de la Table 1.7 (i.e. sans utiliser les instructions simplifiées).*

c. *Ecrire la fonction* ProdMatVecSim *permettant de retourner* $\mathbb{A}\boldsymbol{u}$ *en utilisant la fonction* ProdSca *et les instructions simplifiées de la Table 1.7 (i.e. sans utiliser les fonctions).*

Soient $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ et $\mathbb{B} \in \mathcal{M}_{p,q}(\mathbb{R})$.

Q. 4

a. *Rappeler précisement les hypothèses et les formules permettant le calcul de* $\mathbb{G} = \mathbb{A}\mathbb{B}$.

b. *Ecrire la fonction* ProdMatMat *permettant de retourner* $\mathbb{G}$.

Q. 5

Sous les hypothèses trouvées en **Q.4**, *on note* $\mathbb{G} = \mathbb{A}\mathbb{B}$.

a. *En utilisant les notations mathèmatiques d'accès aux lignes et colonnes d'une matrice, écrire* $\mathbb{G}_{i,j}$ *comme un produit scalaire entre un vecteur ligne de* $\mathbb{A}$ *et un vecteur colonne de* $\mathbb{B}$.

b. *Ecrire la fonction* ProdMatMatFun1 *permettant de retourner* $\mathbb{G}$ *en utilisant la formule trouvée en* **Q.5** a., *des fonctions de la Table 1.7 (i.e. sans utiliser les instructions simplifiées) et la fonction* ProdSca.

c. *Ecrire la fonction* ProdMatMatSim1 *permettant de retourner* $\mathbb{G}$ *en utilisant la formule trouvée en* **Q.5** a., *des instructions simplifiées de la Table 1.7 et la fonction* ProdSca.

Q. 6

Sous les hypothèses trouvées en **Q.4**, *on note* $\mathbb{G} = \mathbb{A}\mathbb{B}$.

a. *En utilisant les notations mathèmatiques d'accès aux lignes et colonnes d'une matrice, écrire* $\mathbb{G}_{:,j}$ *(j-ème vecteur colonne de* $\mathbb{G}$*) comme le produit de la matrice* $\mathbb{A}$ *par un vecteur colonne de* $\mathbb{B}$.

b. *Ecrire la fonction* ProdMatMatFun2 *permettant de retourner* $\mathbb{G}$ *en utilisant la formule trouvée en* **Q.6** a., *des fonctions de la Table 1.7 (i.e. sans utiliser les instructions simplifiées) et la fonction* ProdMatVecFun.

c. *Ecrire la fonction* ProdMatMatSim2 *permettant de retourner* $\mathbb{G}$ *en utilisant la formule trouvée en* **Q.6** a., *des instructions simplifiées de la Table 1.7 et la fonction* ProdMatVecSim.

Correction

R. 1

On a

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} \boldsymbol{u}_i \boldsymbol{v}_i \in \mathbb{R}.$$

Voici une correction possible où l'on identifie la notation mathématique $\boldsymbol{u}_i$, i-ème composante du vecteur $\boldsymbol{u}$, et la notation algorithmique $\boldsymbol{u}(i)$ permettant d'accéder à l'élément i du tableau/vecteur $\boldsymbol{u}$, avec $i \in [\![1, n]\!]$.

Algorithme 1.27 Fonction ProSca, produit scalaire entre deux vecteurs.

Données : $\boldsymbol{u}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1, n]\!]$, $\boldsymbol{u}(i) = \boldsymbol{u}_i$.
$\boldsymbol{v}$: vecteur/tableau de $\mathbb{R}^n$, tel que $\forall i \in [\![1, n]\!]$, $\boldsymbol{v}(i) = \boldsymbol{v}_i$.
Résultat : r : le réel tel que $r = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$.

```
1: Fonction r ← ProSca( u, v ) r ← 0
2:     Pour i ← 1 à n faire
3:         r ← r + u(i) * v(i)
4:     Fin
5: Fin
```

R. 2

a. Pour que le produit de la matrice $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ par un vecteur (colonne) $\boldsymbol{u}$ soit possible il faut et il suffit que la dimension du vecteur (nombre de lignes) soit égale au nombre de colonnes de $\mathbb{A}$, et donc $\boldsymbol{u} \in \mathbb{R}^n$.

b. Avec $\boldsymbol{u} \in \mathbb{R}^n$, on a alors $\boldsymbol{v} \overset{\text{def}}{=} \mathbb{A}\boldsymbol{u} \in \mathbb{R}^m$. et

$$\forall i \in [\![1, m]\!], \quad \boldsymbol{v}_i = \sum_{j=1}^{n} \mathbb{A}_{i,j} \boldsymbol{u}_j.$$

c. Voici une manière d'obtenir l'algorithme en utilisant une méthode de raffinements succéssifs et les notations mathématiques pour accéder aux composantes des vecteurs et des matrices

Algorithme 1.28 $\mathcal{R}_0$

```
1: Calcul de v
```

Algorithme 1.28 $\mathcal{R}_1$

```
1: Pour i ← 1 à m faire
2:     v_i ← Σ_{j=1}^{n} A_{i,j} u_j
3: Fin
```

Algorithme 1.28 $\mathcal{R}_1$

```
1: Pour i ← 1 à m faire
2:
3: Fin
```

Algorithme 1.28 $\mathcal{R}_2$

```
1: Pour i ← 1 à m faire
2:     v_i ← 0
3:     Pour j ← 1 à n faire
4:         v_i ← v_i + A_{i,j} u_j
5:     Fin
6: Fin
```

Pour écrire la fonction algorithmique, il nous faut passer des notations matématiques aux notations algorithmiques pour les matrices et vecteurs. Les correspondancec sont données en Table 1.8.

Description	*Mathématiques*	*Algorithmique*
Composante $i \in [\![1,p]\!]$ de $\boldsymbol{x} \in \mathbb{R}^p$	$\boldsymbol{x}_i$	$\boldsymbol{x}(i)$
Composante $(i,j) \in [\![1,p]\!] \times [\![1,q]\!]$ de $\mathbb{Z} \in \mathcal{M}_{p,q}(\mathbb{R})$,	$\mathbb{Z}_{i,j}$	$\mathbb{Z}(i,j)$

Table 1.8: Correspondance entre les notations mathématiques et algorithmiques pour les matrices et vecteurs

On obtient alors la fonction

Algorithme 1.28 Fonction ProdMatVec1 retournant le vecteur $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$, produit de la matrice $\mathbb{A}$ par le vecteur $\boldsymbol{u}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\boldsymbol{u}$: vecteur de $\mathbb{R}^n$.
Résultat : $\boldsymbol{v}$: vecteur de $\mathbb{R}^m$ tel que $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$.

```
1: Fonction v ← ProdMatVec1( A, u )
2:   Pour i ← 1 à m faire
3:     v(i) ← 0
4:     Pour j ← 1 à n faire
5:       v(i) ← v(i) + A(i, j) ∗ u(j)
6:     Fin
7:   Fin
8: Fin
```

R. 3

a. On peut voir le calcul de $\boldsymbol{v}_i$, $i \in [\![1,m]\!]$ comme étant le produit scalaire de la i-ème ligne de $\mathbb{A}$ par le vecteur $\boldsymbol{u}$. On a donc

$$\forall i \in [\![1,m]\!], \quad \boldsymbol{v}_i = \langle \mathbb{A}(i,:), \boldsymbol{u} \rangle .$$

b. Voici une manière d'obtenir l'algorithme en utilisant une méthode de raffinements successifs et les notations mathématiques pour accéder aux composantes des vecteurs et des matrices

Algorithme 1.29 $\mathcal{R}_0$

```
1: Calcul de v
```

Algorithme 1.29 $\mathcal{R}_1$

```
1: Pour i ← 1 à m faire
2:   v_i ← ⟨A(i, :), u⟩
3: Fin
```

On obtient alors en utilisant des fonctions de la Table 1.7

Algorithme 1.29 Fonction ProdMatVecFun retournant le vecteur $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$, produit de la matrice $\mathbb{A}$ par le vecteur $\boldsymbol{u}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\boldsymbol{u}$: vecteur de $\mathbb{R}^n$.
Résultat : $\boldsymbol{v}$: vecteur de $\mathbb{R}^m$ tel que $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$.

```
1: Fonction v ← ProdMatVec1( A, u )
2:   Pour i ← 1 à m faire
3:     v(i) ← ProdSca(getMatRow(A, i), u)
4:   Fin
5: Fin
```

c. En utilisant les instructions simplifiées de la Table 1.7

Algorithme 1.30 Fonction ProdMatVecFun retournant le vecteur $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$, produit de la matrice $\mathbb{A}$ par le vecteur $\boldsymbol{u}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\boldsymbol{u}$: vecteur de $\mathbb{R}^n$.

Résultat : $\boldsymbol{v}$: vecteur de $\mathbb{R}^m$ tel que $\boldsymbol{v} \stackrel{\text{def}}{=} \mathbb{A}\boldsymbol{u}$.

```
1: Fonction v ← ProdMatVec1( A, u )
2:   Pour i ← 1 à m faire
3:     v(i) ← ProdSca(A(i,:), u)
4:   Fin
5: Fin
```

R. 4

a. Le calcul du produit matricielle $\mathbb{AB}$, avec $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$ et $\mathbb{B} \in \mathcal{M}_{p,q}(\mathbb{R})$, est possible si et seulement si le nombre de colonnes de $\mathbb{A}$ est égale au nombre de lignes de $\mathbb{B}$, c'est à dire $n = p$. Dans ce cas, la matrice résultat $\mathbb{G} \in \mathcal{M}_{m,q}(\mathbb{R})$ et on a

$$\forall (i,j) \in [\![1,m]\!] \times [\![1,q]\!], \quad \mathbb{G}_{i,j} = \sum_{k=1}^{n} \mathbb{A}_{i,k}\mathbb{B}_{k,j}.$$

b. Voici une manière d'obtenir l'algorithme en utilisant une méthode de raffinements succéssifs et les notations mathématiques usuels pour accéder aux composantes des vecteurs et des matrices

Algorithme 1.31 $\mathcal{R}_0$

```
1: Calcul de G
```

Algorithme 1.31 $\mathcal{R}_1$

```
1: Pour i ← 1 à m faire
2:   Pour j ← 1 à q faire
3:     G_{i,j} ← Σ_{k=1}^{n} A_{i,k} B_{k,j}
4:   Fin
5: Fin
```

Algorithme 1.31 $\mathcal{R}_1$

```
1: Pour i ← 1 à m faire
2:   Pour j ← 1 à q faire
3:
4:   Fin
5: Fin
```

Algorithme 1.31 $\mathcal{R}_2$

```
1: Pour i ← 1 à m faire
2:   Pour j ← 1 à q faire
3:     S ← 0
4:     Pour k ← 1 à n faire
5:       S ← S + A_{i,k} B_{k,j}
6:     Fin
7:     G_{i,j} ← S
8:   Fin
9: Fin
```

On obtient alors en utilisant les notations algorithmiques

Algorithme 1.31 Fonction ProdMatMatFun retournant la matrice $\mathbb{G} \stackrel{\text{def}}{=} \mathbb{AB}$, produit de la matrice $\mathbb{A}$ par la matrice $\mathbb{B}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\mathbb{B}$: matrice de $\mathcal{M}_{n,q}(\mathbb{R})$.
Résultat : $\mathbb{G}$: matrice de $\mathcal{M}_{m,q}(\mathbb{R})$.

```
1: Fonction Gv ← ProdMatMat( A, B )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à q faire
4:       S ← 0
5:       Pour k ← 1 à n faire
6:         S ← S + A(i, k) ∗ B(k, j)
7:       Fin
8:       G(i, j) ← S
9:     Fin
10:  Fin
11: Fin
```

R. 5

a. En effet la composante (i, j) de $\mathbb{G}$ est le produit scalaire de la ligne i de $\mathbb{A}$ et de la colonne j de $\mathbb{B}$. On a donc

$$\forall (i,j) \in [\![1,m]\!] \times [\![1,q]\!], \quad \mathbb{G}_{i,j} = \langle \mathbb{A}_{i,:}, \mathbb{B}_{:,j} \rangle .$$

b. Voici l'algorithme utilisant les notations mathématiques permettant d'accéder aux lignes et colonnes des matrices

Algorithme 1.32 $\boxed{\mathcal{R}_0}$

```
1: Pour i ← 1 à m faire
2:   Pour j ← 1 à q faire
3:     G_{i,j} ← ⟨A_{i,:}, B_{:,j}⟩
4:   Fin
5: Fin
```

On obtient alors en utilisant les fonctions algorithmiques de la Table 1.7

Algorithme 1.32 Fonction ProdMatMatFun1 retournant la matrice $\mathbb{G} \stackrel{\text{def}}{=} \mathbb{AB}$, produit de la matrice $\mathbb{A}$ par la matrice $\mathbb{B}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\mathbb{B}$: matrice de $\mathcal{M}_{n,q}(\mathbb{R})$.
Résultat : $\mathbb{G}$: matrice de $\mathcal{M}_{m,q}(\mathbb{R})$.

```
1: Fonction Gv ← ProdMatMatFun1( A, B )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à q faire
4:       G(i, j) ← ProSca(getMatRow(A, i), getMatCol(B, j))
5:     Fin
6:   Fin
7: Fin
```

c. En utilisant des instructions simplifiées de la Table 1.7 et la fonction ProdSca, on a

Algorithme 1.33 Fonction ProdMatMatSim1 retournant la matrice $\mathbb{G} \stackrel{\text{def}}{=} \mathbb{AB}$, produit de la matrice $\mathbb{A}$ par la matrice $\mathbb{B}$.

Données : $\mathbb{A}$: matrice de $\mathcal{M}_{m,n}(\mathbb{R})$,
$\mathbb{B}$: matrice de $\mathcal{M}_{n,q}(\mathbb{R})$.
Résultat : $\mathbb{G}$: matrice de $\mathcal{M}_{m,q}(\mathbb{R})$.

```
1: Fonction ℂv ← ProdMatMatFun1( 𝔸, 𝔹 )
2:   Pour i ← 1 à m faire
3:     Pour j ← 1 à q faire
4:       ℂ(i, j) ← ProSca(𝔸(i, :), 𝔹(:, j))
5:     Fin
6:   Fin
7: Fin
```

Fin correction

Chapitre 2

Dérivation numérique

Soit $y : [a, b] \longrightarrow \mathbb{R}$, de classe $\mathcal{C}^1$ et $t \in [a, b]$. La dérivée $y'(t)$ est donnée par

$$\begin{aligned} y'(t) &= \lim_{h \to 0^+} \frac{y(t+h) - y(t)}{h} \\ &= \lim_{h \to 0^+} \frac{y(t) - y(t-h)}{h} \\ &= \lim_{h \to 0} \frac{y(t+h) - y(t-h)}{2h} \end{aligned}$$

L'objectif de cette section est de déterminer des approximations de dérivées premières et secondes d'une fonction en un ensemble de points sachant que l'on connait uniquement la valeur de la fonction en ces points.
Par exemple, lorsque vous courrez avec une montre GPS, celle-ci peut afficher la vitesse *instantannée*. En fait, la montre récupère votre position GPS toutes les secondes par exemple (variable suivant les montres et les activités). Il est alors possible d'approcher votre vitesse *instantannée* en divisant la distance parcourue (en mètre) par la durée (en seconde) depuis la dernière mesure.
Qui plus est, à la fin de l'activité, la trace GPS est sauvegardée comme une succesion (en nombre fini) de positions GPS. A partir de la, on peut obtenir, en fonction du temps*, la distance parcourue et la vitesse *instantannée* pendant toute la durée de l'épreuve.
Dans ce cours, on va supposer que les mesures sont faites régulièrement, ce qui amène la notion de discrétisation régulière.

2.1 Développement de Taylor

On définit tout d'abord la notion de $\mathcal{O}(h^q)$ qui sera fortement utilisée.

Définition 2.1.1. *Soit g une fonction et $q \in \mathbb{N}$. On dit que g* ***se comporte comme un grand O de*** *h^q quand h tend vers 0 si et seulement si il existe $H > 0$ et $C > 0$ tel que*

$$\forall h \in]-H, H[, \quad |g(h)| \leqslant C|h|^q.$$

On note alors $g(h) = \mathcal{O}(h^q)$.

*Non la météo n'a rien à voir ici

On peut noter entre autres

- $\forall \alpha \in \mathbb{R}^*$,
$$g(h) = \mathcal{O}(h^q) \quad \Rightarrow \quad \alpha g(h) = \mathcal{O}(h^q),$$
c'est à dire
$$\alpha\, \mathcal{O}(h^q) = \mathcal{O}(h^q)!$$

- $\forall p \in \mathbb{Z},\ p + q \geqslant 0$,
$$g(h) = \mathcal{O}(h^q) \quad \Rightarrow \quad h^p g(h) = \mathcal{O}(h^{p+q}),$$
c'est à dire
$$h^p\, \mathcal{O}(h^q) = \mathcal{O}(h^{p+q})!$$

- $\forall (p, q) \in \mathbb{N}^2$, $\forall (\alpha_1, \alpha_2) \in \mathbb{R}^* \times \mathbb{R}^*$,
$$g_1(h) = \mathcal{O}(h^p),\ g_2(h) = \mathcal{O}(h^q) \quad \Rightarrow \quad \alpha_1 g_1(h) + \alpha_2 g_2(h) = \mathcal{O}(h^{\min(p,q)}),$$
c'est à dire
$$\alpha_1\, \mathcal{O}(h^p) + \alpha_2\, \mathcal{O}(h^q) = \mathcal{O}(h^{\min(p,q)})!$$

On rappelle maintenant plusieurs écritures du developpement de Taylor-Lagrange

Proposition 2.1.1 (Développement de Taylor-Lagrange)**.** *Soit f une application $f : [a, b] \longrightarrow \mathbb{R}$. Si $f \in \mathcal{C}^{r+1}([a, b])$ alors le* ***développement de Taylor à l'ordre*** *r s'écrit*

- $\forall (x, y) \in [a, b]^2$ *il existe un* $\xi \in]\min(x, y), \max(x, y)[$ *tel que*
$$f(x) = f(y) + \sum_{k=1}^{r} \frac{f^{(k)}(y)}{k!}(x - y)^k + \frac{f^{(r+1)}(\xi)}{(r + 1)!}(x - y)^{r+1} \tag{2.1}$$

- $\forall t \in [a, b]$, $\forall h \in \mathbb{R}^*$ *vérifiant* $(t + h) \in [a, b]$, *il existe* $\xi \in]\min(t, t + h), \max(t, t + h)[$ *tel quel*
$$f(t + h) = f(t) + \sum_{k=1}^{r} \frac{h^k}{k!} f^{(k)}(t) + \frac{h^{r+1}}{(r + 1)!} f^{(r+1)}(\xi) \tag{2.2}$$

- $\forall t \in [a, b]$, $\forall h \in \mathbb{R}^*$ *vérifiant* $(t + h) \in [a, b]$, *alors*
$$f(t + h) = f(t) + \sum_{k=1}^{r} \frac{f^{(k)}(t)}{k!} h^k + \mathcal{O}(h^{r+1}). \tag{2.3}$$

Brook Taylor 1685-1731, Mathématicien anglais

(a) *Joseph-Louis Lagrange* 1736-1813, Mathématicien italien(sarde) puis naturalisé français

2.2 Un peu de maths

Exercice 2.2.1

Soit $\varphi : [a,b] \longrightarrow \mathbb{R}$ une fonction.

Q. 1 *Montrer que si* $\varphi \in \mathcal{C}^2([a,b];\mathbb{R})$ *alors* $\forall x \in [a,b[$, $\forall h > 0$ *tel que* $(x+h) \in [a,b]$, *on a*

$$\frac{d\varphi}{dx}(x) = \frac{\varphi(x+h) - \varphi(x)}{h} + \mathcal{O}(h) \tag{2.1}$$

Q. 2 *Montrer que si* $\varphi \in \mathcal{C}^2([a,b];\mathbb{R})$ *alors* $\forall x \in]a,b]$, $\forall h > 0$ *tel que* $(x-h) \in [a,b]$, *on a*

$$\frac{d\varphi}{dx}(x) = \frac{\varphi(x) - \varphi(x-h)}{h} + \mathcal{O}(h) \tag{2.2}$$

Q. 3 *Montrer que si* $\varphi \in \mathcal{C}^3([a,b];\mathbb{R})$ *alors* $\forall x \in]a,b[$, $\forall h > 0$ *tel que* $(x+h) \in [a,b]$ *et* $(x-h) \in [a,b]$, *on a*

$$\frac{d\varphi}{dx}(x) = \frac{\varphi(x+h) - \varphi(x-h)}{2h} + \mathcal{O}(h^2) \tag{2.3}$$

Q. 4 *Montrer que si* $\varphi \in \mathcal{C}^4([a,b];\mathbb{R})$ *alors* $\forall x \in]a,b[$, $\forall h > 0$ *tel que* $(x+h) \in [a,b]$ *et* $(x-h) \in [a,b]$, *on a*

$$\frac{d^2\varphi}{dx^2}(x) = \frac{\varphi(x+h) - 2\varphi(x) + \varphi(x-h)}{h^2} + \mathcal{O}(h^2) \tag{2.4}$$

Correction 2.2.1

R. 1 On rappelle le développement de Taylor à l'ordre r d'une fonction $f \in \mathcal{C}^{r+1}([a,b];\mathbb{R})$
$\forall x \in [a,b]$, $\forall \square \in \mathbb{R}^*$ vérifiant $(x+\square) \in [a,b]$,

$$f(x+\square) = f(x) + \sum_{k=1}^{r} \frac{\square^k}{k!} f^{(k)}(x) + \mathcal{O}(\square^{r+1}) \tag{R1.1}$$

où $f^{(k)}(x) = \frac{d^k f}{dx^k}(x)$.
On peut ici utiliser un développement de Taylor à l'ordre $r = 1$ de φ car $\varphi \in \mathcal{C}^2([a,b];\mathbb{R})$ avec $\square = h$ dans (R1.1).
Soient $x \in [a,b]$ et $h \in \mathbb{R}^{*+}$, tels que $(x+h) \in [a,b]$ (donc nécessairement $x \in [a,b[$) on a

$$\varphi(x+h) = \varphi(x) + h\,\varphi^{(1)}(x) + \mathcal{O}(h^2)$$

ce qui donne

$$\begin{aligned} \varphi^{(1)}(x) &= \frac{\varphi(x+h) - \varphi(x)}{h} - \frac{1}{h}\mathcal{O}(h^2) \\ &= \frac{\varphi(x+h) - \varphi(x)}{h} + \mathcal{O}(h). \end{aligned}$$

R. 2 On peut ici utiliser un développement de Taylor à l'ordre $r = 1$ de φ car $\varphi \in \mathcal{C}^2([a,b];\mathbb{R})$ avec $\square = -h$ dans dans (R1.1).
On peut ici utiliser un développement de Taylor à l'ordre 2 car $\varphi \in \mathcal{C}^2([a,b];\mathbb{R})$.
Soient $x \in [a,b]$ et $h \in \mathbb{R}^{*+}$, tels que $(x-h) \in [a,b]$ (donc nécessairement $x \in]a,b]$) on a

$$\varphi(x-h) = \varphi(x) - h\,\varphi^{(1)}(x) + \mathcal{O}(h^2)$$

ce qui donne

$$\begin{aligned}\varphi^{(1)}(x) &= \frac{\varphi(x)-\varphi(x-h)}{h}+\frac{1}{h}\mathcal{O}(h^2)\\ &= \frac{\varphi(x)-\varphi(x-h)}{h}+\mathcal{O}(h).\end{aligned}$$

R. 3

On peut ici utiliser un développement de Taylor à l'ordre $r=2$ de φ car $\varphi\in\mathcal{C}^3([a,b];\mathbb{R})$.
Dans la formule (2.3), les termes $\varphi(x+h)$, $\varphi(x-h)$ apparaissent. Ceci suggère d'utiliser deux développement de Taylor, l'un en $x+h$ et l'autre en $x-h$.
Soient $x\in[a,b]$ et $h\in\mathbb{R}^{*+}$, tels que $(x+h)\in[a,b]$ et $(x-h)\in[a,b]$ (donc nécessairement $x\in]a,b[$).
On a alors

$$\begin{aligned}\varphi(x+h) &= \varphi(x)+h\,\varphi^{(1)}(x)+\frac{h^2}{2}\,\varphi^{(2)}(x)+\mathcal{O}(h^3),\\ \varphi(x-h) &= \varphi(x)-h\,\varphi^{(1)}(x)+\frac{h^2}{2}\,\varphi^{(2)}(x)+\mathcal{O}(h^3).\end{aligned}$$

Remarque. *Attention ici, il faut bien voir que les $\mathcal{O}$ dans les deux formules ne sont mathématiquement pas forcément identiques! En effet, en revenant aux développements de Taylor sans $\mathcal{O}$, pour la formule en $x+h$, il existe $\xi_+\in]x,x+h[$ tel que*

$$\mathcal{O}(h^3)=\frac{h^3}{3!}\,\varphi^{(3)}(\xi_+)$$

et pour la formule en $x-h$, il existe $\xi_-\in]x-h,x[$ tel que

$$\mathcal{O}(h^3)=-\frac{h^3}{3!}\,\varphi^{(3)}(\xi_-).$$

On effectuant la différence entre ces deux équations on obtient:

$$\varphi(x+h)-\varphi(x-h)=2h\,\varphi^{(1)}(x)+\mathcal{O}(h^3)$$

ce qui donne

$$\begin{aligned}\varphi^{(1)}(x) &= \frac{\varphi(x+h)-\varphi(x-h)}{2h}+\frac{1}{2h}\mathcal{O}(h^3)\\ &= \frac{\varphi(x+h)-\varphi(x-h)}{2h}+\mathcal{O}(h^2)\end{aligned}$$

R. 4

On peut ici utiliser un développement de Taylor à l'ordre $r=3$ de φ car $\varphi\in\mathcal{C}^4([a,b];\mathbb{R})$.
Dans la formule (2.4), les termes $\varphi(x+h)$, $\varphi(x-h)$ apparaissent. Ceci suggère d'utiliser deux développement de Taylor, l'un en $x+h$ et l'autre en $x-h$.
Soient $x\in[a,b]$ et $h\in\mathbb{R}^{*+}$, tels que $(x+h)\in[a,b]$ et $(x-h)\in[a,b]$ (donc nécessairement $x\in]a,b[$).
On a alors

$$\begin{aligned}\varphi(x+h) &= \varphi(x)+h\,\varphi^{(1)}(x)+\frac{h^2}{2}\,\varphi^{(2)}(x)+\frac{h^3}{3}\,\varphi^{(3)}(x)+\mathcal{O}(h^4),\\ \varphi(x-h) &= \varphi(x)-h\,\varphi^{(1)}(x)+\frac{h^2}{2}\,\varphi^{(2)}(x)-\frac{h^3}{3}\,\varphi^{(3)}(x)+\mathcal{O}(h^4).\end{aligned}$$

On effectuant la somme entre ces deux équations on obtient:

$$\varphi(x+h)+\varphi(x-h)=2\varphi(x)+h^2\,\varphi^{(2)}(x)+\mathcal{O}(h^4)$$

ce qui donne

$$\begin{aligned}\varphi^{(2)}(x) &= \frac{\varphi(x+h)-2\varphi(x)+\varphi(x-h)}{h^2}+\frac{1}{h^2}\mathcal{O}(h^4)\\ &= \frac{\varphi(x+h)-2\varphi(x)+\varphi(x-h)}{h^2}+\mathcal{O}(h^2)\end{aligned}$$

Fin correction

Définition 2.2.1. *Soit $\varphi:\mathbb{R}\longrightarrow\mathbb{R}$ une fonction suffisament régulière. Soient $x\in\mathbb{R}$, $h\in\mathbb{R}^{+*}$, $n\in\mathbb{N}^*$ et $p\in\mathbb{N}^*$.*
On dit que $\square_h(x)$ est une ***approximation d'ordre*** *p* ***de*** *$\frac{d^n\varphi}{dx^n}(x)$ si*

$$\frac{d^n\varphi}{dx^n}(x)-\square_h(x)=\mathcal{O}(h^p)$$

On a donc établi:

Lemme 2.2.1. *Soient $\varphi:\mathbb{R}\longrightarrow\mathbb{R}$ une fonction suffisament régulière, $x\in\mathbb{R}$ et $h\in\mathbb{R}^{+*}$.*

- $\frac{\varphi(x+h)-\varphi(x)}{h}$ *et* $\frac{\varphi(x)-\varphi(x-h)}{h}$ *sont des approximations d'ordre 1 de* $\frac{d\varphi}{dx}(x)$,
- $\frac{\varphi(x+h)-\varphi(x-h)}{2h}$ *est une approximation d'ordre 2 de* $\frac{d\varphi}{dx}(x)$,
- $\frac{\varphi(x+h)-2\varphi(x)+\varphi(x-h)}{h^2}$, *est une approximation d'ordre 2 de* $\frac{d^2\varphi}{dx^2}(x)$.

Exercice 2.2.2

Soit $\varphi:\mathbb{R}\longrightarrow\mathbb{R}$ une fonction suffisament régulière, $x\in\mathbb{R}$ et $h\in\mathbb{R}^{+*}$.

Q. 1 *Montrer que*

$$\frac{d\varphi}{dx}(x)=\frac{-3\varphi(x)+4\varphi(x+h)-\varphi(x+2h)}{2h}+\mathcal{O}(h^2) \tag{2.1}$$

Q. 2 *Montrer que*

$$\frac{d\varphi}{dx}(x)=\frac{3\varphi(x)-4\varphi(x-h)+\varphi(x-2h)}{2h}+\mathcal{O}(h^2) \tag{2.2}$$

Correction 2.2.2

R. 1 Soit φ une fonction suffisament régulière et $h>0$
Pour celà, on écrit les deux développements de Taylor en $x+h$ et $x+2h$.
Il existe $\xi_1^+\in]x,x+h[$ tel que

$$\varphi(x+h)=\varphi(x)+h\frac{d\varphi}{dx}(x)+\frac{h^2}{2!}\frac{d^2\varphi}{dx^2}(x)+\frac{h^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^+) \tag{R2.2}$$

et Il existe $\xi_2^+\in]x,x+2h[$ tel que

$$\varphi(x+2h)=\varphi(x)+(2h)\frac{d\varphi}{dx}(x)+\frac{(2h)^2}{2!}\frac{d^2\varphi}{dx^2}(x)+\frac{(2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^+) \tag{R2.3}$$

L'objectif est d'obtenir, par une combinaison linéaire entre ces deux formules, une nouvelle équation

ne comportant plus de termes en $\frac{d^2\varphi}{dx^2}$. En effectuant $4 \times$ (R2.19) $-$ (R2.20) on obtient

$$4\varphi(x+h) - \varphi(x+2h) = 3\varphi(x) + 2h\frac{d\varphi}{dx}(x) + 4\frac{h^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^+) - \frac{(2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^+)$$

On en déduit

$$\frac{d\varphi}{dx}(x) = \frac{-3\varphi(x) + 4\varphi(x+h) - \varphi(x+2h)}{2h} + \mathcal{O}(h^2)$$

R. 2

Soit φ une fonction suffisament régulière et $h > 0$.
Pour celà, on écrit les deux développements de Taylor en $x-h$ et $x-2h$.
Il existe $\xi_1^- \in]x-h, x[$ tel que

$$\varphi(x-h) = \varphi(x) - h\frac{d\varphi}{dx}(x) + \frac{(-h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(-h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^-) \tag{R2.4}$$

et Il existe $\xi_2^- \in]x-2h, x[$ tel que

$$\varphi(x-2h) = \varphi(x) + (-2h)\frac{d\varphi}{dx}(x) + \frac{(-2h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(-2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^+) \tag{R2.5}$$

L'objectif est d'obtenir, par une combinaison linéaire entre ces deux formules, une nouvelle équation ne comportant plus de termes en $\frac{d^2\varphi}{dx^2}$. En effectuant $4 \times$ (R2.21) $-$ (R2.22) on obtient

$$4\varphi(x-h) - \varphi(x-2h) = 3\varphi(x) - 2h\frac{d\varphi}{dx}(x) + 4\frac{(-h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^-) - \frac{(-2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^-)$$

On en déduit

$$\frac{d\varphi}{dx}(x) = \frac{3\varphi(x) - 4\varphi(x-h) + \varphi(x-2h)}{2h} + \mathcal{O}(h^2)$$

Fin correction

Ces résultats impliquent:

$$\frac{-3\varphi(x) + 4\varphi(x+h) - \varphi(x+2h)}{2h} \quad \text{et} \quad \frac{3\varphi(x) - 4\varphi(x-h) + \varphi(x-2h)}{2h}$$

sont des approximations d'ordre 2 de $\frac{d\varphi}{dx}(x)$.

2.3 Application numérique

Définition 2.3.1. *Soit $N \in \mathbb{N}^*$. On appelle* ***discrétisation régulière de*** $[a,b]$ ***à*** N ***pas ou*** $(N+1)$ ***points*** *l'ensemble des points $a+nh$, $n \in [\![0, N]\!]$ où le pas h est donné par $h = (b-a)/N$.*

Exercice 2.3.1

Ecrire une fonction DisReg retournant une discrétisation de l'intervalle $[a;b]$ avec N pas (constant) de discrétisation.

Correction Soient $\{t^n\}_{n\in[\![0,N]\!]}$ une discrétisation régulière de $[a,b]$, on a alors

$$\forall n \in [\![0, N]\!],\ t^n = a + nh,\ \text{avec } h = \frac{b-a}{n}.$$

Au niveau algorithmique, on stocke les $(N+1)$ réels dans un vecteur de $\mathbb{R}^{N+1}$ (i.e. tableau 1D de réels à $(N+1)$ éléments) représenté par la variable **tt**. Une représentation mémoire est donnée en Figure 2.2.

Figure 2.2: Représentation mémoire de la variable algoritmique **tt** contenant la discrétisation régulière de $[a, b]$ avec $(N + 1)$ points notée $(t^n)_{n=0}^N$.

On peut noter le lien entre la variable algorithmique **tt** et la notation mathématique t^n:

$$\mathbf{tt}(n+1) = t^n,\ \forall n \in [\![0, N]\!] \ \textbf{ou}\ \mathbf{tt}(n) = t^{n-1},\ \forall n \in [\![1, N+1]\!]$$

On a donc

$$\mathbf{tt}(n+1) = a + nh,\ \forall n \in [\![0, N]\!] \ \textbf{ou}\ \mathbf{tt}(n) = a + (n-1)h,\ \forall n \in [\![1, N+1]\!].$$

Algorithme 2.1 Fonction DisReg : discrétisation régulière de $[a, b]$ avec $(N + 1)$ points

Données : a, b : deux réels, (a<b)
N : un entier non nul (nombre de pas de discrétisation).
Résultat : **tt** : vecteur de $\mathbb{R}^{N+1}$, $\mathbf{tt}(n) = t^{n-1},\ \forall n \in [\![1, N+1]\!]$

```
1: Fonction tt ← DisReg( a, b, N )
2:    h ← (b − a)/N
3:    Pour n ← 1 à N + 1 faire
4:       tt(n) ← a + (n − 1) ∗ h
5:    Fin
6: Fin
```

D'autres variantes à cette fonction existent, en voici 2:

```
1: Fonction tt ← DisRegV1( a, b, N )
2:    h ← (b − a)/N
3:    Pour n ← 0 à N faire
4:       tt(n + 1) ← a + n ∗ h
5:    Fin
6: Fin
```

```
1: Fonction tt ← DisRegV2( a, b, N )
2:    h ← (b − a)/N
3:    tt(1) ← a
4:    Pour n ← 2 à N + 1 faire
5:       tt(n) ← tt(n − 1) + h
6:    Fin
7: Fin
```

Fin correction

Soient $\{t^n\}_{n\in[\![0,N]\!]}$ une discrétisation régulière de $[a, b]$ et $(Dy)_n$ une approximation de $y'(t^n)$. On appelle

- **différence finie progressive** l'approximation

$$(Dy)_n^P = \frac{y(t^{n+1}) - y(t^n)}{h},\ \forall n \in [\![0, N-1]\!] \tag{2.4}$$

- **différence finie rétrograde** l'approximation

$$(Dy)_n^R = \frac{y(t^n) - y(t^{n-1})}{h},\ \forall n \in [\![1, N]\!] \tag{2.5}$$

- **différence finie centrée** l'approximation

$$(Dy)_n^C = \frac{y(t^{n+1}) - y(t^{n-1})}{2h},\ \forall n \in [\![1, N-1]\!] \tag{2.6}$$

Lemme 2.3.1. *Si $y \in \mathcal{C}^2([a,b])$ alors*

$$y'(t^n) = \frac{y(t^{n+1}) - y(t^n)}{h} + \mathcal{O}(h), \quad \forall n \in [\![0, N-1]\!], \tag{2.7}$$

$$y'(t^n) = \frac{y(t^n) - y(t^{n-1})}{h} + \mathcal{O}(h), \quad \forall n \in [\![1, N]\!]. \tag{2.8}$$

Si $y \in \mathcal{C}^3([a,b])$ alors

$$y'(t^n) = \frac{y(t^{n+1}) - y(t^{n-1})}{2h} + \mathcal{O}(h^2), \quad \forall n \in [\![1, N-1]\!]. \tag{2.9}$$

Exercice 2.3.2

Soit $f \in \mathcal{C}^3([a,b];\mathbb{R})$. On note t^n, $n \in [\![0,N]\!]$, une discrétisation **régulière** de $[a,b]$ de pas h. On note $\boldsymbol{F} \in \mathbb{R}^{N+1}$ le vecteur défini par $F_{n+1} = f(t^n)$, $\forall n \in [\![0,N]\!]$.

Q. 1

a. Déterminer en fonction de h et $\boldsymbol{F}$, un vecteur $\boldsymbol{V} \in \mathbb{R}^{N+1}$ vérifiant

$$V_{n+1} = f'(t^n) + \mathcal{O}(h), \quad \forall n \in [\![0,N]\!].$$

b. Ecrire une fonction algorithmique, nommée Derive1Ordre1*, permettant, à partir du vecteur $\boldsymbol{F}$ et de la discrétisation régulière, de calculer le vecteur $\boldsymbol{V}$ précédent.*

Q. 2

a. Connaissant uniquement h et le vecteur $\boldsymbol{F}$, déterminer un vecteur $\boldsymbol{W} \in \mathbb{R}^{N+1}$ vérifiant

$$W_{n+1} = f'(t^n) + \mathcal{O}(h^2), \quad \forall n \in [\![0,N]\!]$$

b. Ecrire une fonction algorithmique, nommée Derive1Ordre2*, permettant, à partir du vecteur $\boldsymbol{F}$ et de la discrétisation régulière, de calculer le vecteur $\boldsymbol{W}$ précédent.*

Correction 2.3.2

R. 1

a. On a $h = (b-a)/N$ et $t^n = a + nh$, $\forall n \in [\![0,N]\!]$. Par la formule de Taylor, on obtient respectivement les formules progressives et régressives d'ordre 1 suivantes (voir Lemme 2.3.1)

$$\frac{f(t+h) - f(t)}{h} = f'(t) + \mathcal{O}(h) \quad \text{et} \quad \frac{f(t) - f(t-h)}{h} = f'(t) + \mathcal{O}(h).$$

On va utiliser ces formules en $t = t^n$. On note que la formule progressive n'est pas utilisable en $t = t^N$ (car $t^N + h = b + h \notin [a,b]$!) et que la formule régressive n'est pas utilisable en $t = t^0$ (car $t^0 - h = a - h \notin [a,b]$!). Plus précisement, on a

$$\frac{f(t^{n+1}) - f(t^n)}{h} = f'(t^n) + \mathcal{O}(h), \quad \forall n \in [\![0, N[\![, \tag{R2.6}$$

$$\frac{f(t^n) - f(t^{n-1})}{h} = f'(t^n) + \mathcal{O}(h), \quad \forall n \in]\!]0, N]\!] \tag{R2.7}$$

On peut alors construire le vecteur $\boldsymbol{V}$ en prenant

$$V_1 \overset{\text{def}}{=} \frac{f(t^1) - f(t^0)}{h} = f'(t^0) + \mathcal{O}(h), \quad \text{formule progressive (R2.6) avec } n = 0$$

$$V_{N+1} \overset{\text{def}}{=} \frac{f(t^N) - f(t^{N-1})}{h} = f'(t^N) + \mathcal{O}(h), \quad \text{formule régressive (R2.7) avec } n = N$$

et pour les points strictement intérieurs les deux formules sont possible :

$$V_n \overset{\text{def}}{=} \frac{f(t^{n+1}) - f(t^n)}{h} = f'(t^n) + \mathcal{O}(h), \quad \text{formule progressive (R2.6) avec } n \in]\!]0, N[\![$$
$$V_n \overset{\text{def}}{=} \frac{f(t^n) - f(t^{n-1})}{h} = f'(t^n) + \mathcal{O}(h), \quad \text{formule régressive (R2.7) avec } n \in]\!]0, N[\![$$

En choisissant par exemple la formule progressive pour les points intérieurs, on obtient en fonction de $\boldsymbol{F}$ et h

$$\begin{aligned} V_1 &\overset{\text{def}}{=} \frac{F_2 - F_1}{h} && = f'(t^0) + \mathcal{O}(h) \\ \forall n \in]\!]0, N[\![,\ V_n &\overset{\text{def}}{=} \frac{F_{n+1} - F_n}{h} && = f'(t^n) + \mathcal{O}(h) \\ V_{N+1} &\overset{\text{def}}{=} \frac{F_{N+1} - F_N}{h} && = f'(t^N) + \mathcal{O}(h). \end{aligned}$$

b. On représente tout d'abord les vecteurs $\mathbf{F}$ et $\mathbf{V}$ en Figure 2.3.

$\mathbf{F}$	$f(t^0)$	$f(t^1)$	$\cdots$	$f(t^{n-1})$	$\cdots$	$f(t^{N-1})$	$f(t^N)$
	$\mathbf{F}(1)$	$\mathbf{F}(2)$		$\mathbf{F}(n)$		$\mathbf{F}(N)$	$\mathbf{F}(N+1)$
$\mathbf{V}$	$f'(t^0)$ + $\mathcal{O}(h)$	$f'(t^1)$ + $\mathcal{O}(h)$	$\cdots$	$f'(t^{n-1})$ + $\mathcal{O}(h)$	$\cdots$	$f'(t^{N-1})$ + $\mathcal{O}(h)$	$f'(t^N)$ + $\mathcal{O}(h)$
	$\mathbf{V}(1)$	$\mathbf{V}(2)$		$\mathbf{V}(n)$		$\mathbf{V}(N)$	$\mathbf{V}(N+1)$

Figure 2.3: Représentation mémoire du vecteur $\boldsymbol{F} \in \mathbb{R}^{N+1}$ et du vecteur $\boldsymbol{V} \in \mathbb{R}^{N+1}$.

En utilisant les formules trouvées précédemment, la fonction algorithmique peut s'écrire sous la forme :

Algorithme 2.2 Calcul numérique de dérivées premières d'ordre 1 d'une fonction f définie sur un intervalle en utilisant uniquemment les valeurs de f aux points $(t^n)_{n=0}^N$ d'une discrétisation régulière de cet intervalle. Les approximations sont calculées en tous les points de la discrétisation.

Données : $\boldsymbol{F}$: vecteur de $\mathbb{R}^{N+1}$ tel que $F_{n+1} = f(t^n)$, $\forall n \in [\![0, N]\!]$.
h : nombre réel strictement positif.
Résultat : $\boldsymbol{V}$: vecteur de $\mathbb{R}^{N+1}$ tel que $V_{n+1} = f'(t^n) + \mathcal{O}(h)$, $\forall n \in [\![0, N]\!]$.

1: **Fonction** $\boldsymbol{V} \leftarrow$ Derive1Ordre1($h, \boldsymbol{F}$)
2: $V(1) \leftarrow (F(2) - F(1))/h$
3: **Pour** $n \leftarrow 2$ **à** N **faire**
4: $V(n) \leftarrow (F(n+1) - F(n))/h$
5: **Fin**
6: $V(N+1) \leftarrow (F(N+1) - F(N))/h$
7: **Fin**

D'autres façons de présenter le problème sont possibles mais les données peuvent diffèrer de celles de l'énoncé. Par exemple une autre fonction algorithmique pourrait être

Algorithme 2.3 Approximations d'ordre 1 de dérivées premières d'une fonction f définie sur un intervalle $[a,b]$ en utilisant uniquemment les valeurs de f aux points $(t^n)_{n=0}^N$ d'une discrétisation régulière de cet intervalle. Les approximations sont calculées en tous les points de la discrétisation.

Données : f : $f : [a,b] \longrightarrow \mathbb{R}$
a, b : deux réels, $a < b$,
N : nombre de pas de discrétisation
Résultat : $\boldsymbol{V}$: vecteur de $\mathbb{R}^{N+1}$ tel que $V_{n+1} = f'(t^n) + \mathcal{O}(h)$ avec $(t^n)_{n=0}^N$ discrétisation régulière de $[a,b]$.

1: **Fonction** $\boldsymbol{V} \leftarrow$ DeriveOrdre1-v1(f, a, b, N)
2: $t \leftarrow$ DisReg(a, b, N) ▷ fonction retournant la discrétisation régulière...
3: $h \leftarrow (b-a)/N$
4: $V(1) \leftarrow (f(t(2)) - f(t(1))/h$
5: **Pour** $n \leftarrow 2$ **à** N **faire**
6: $V(n) \leftarrow (f(t(n+1)) - f(t(n)))/h$
7: **Fin**
8: $V(N+1) \leftarrow (f(t(N+1)) - f(t(N))/h$
9: **Fin**

R. 2

a. Du lemme 2.3.1, on a la formule centrée d'ordre 2

$$\frac{f(t+h) - f(t-h)}{2h} = f'(t) + \mathcal{O}(h^2).$$

On va utiliser cette formule en $t = t^n$. On note que la formule n'est pas utilisable en $t = t^N = b$ et $t = t^0 = a$. On obtient

$$f'(t^n) = \frac{f(t^{n+1}) - f(t^{n-1})}{2h} + \mathcal{O}(h^2), \quad \forall n \in]\!]0, N[\![. \tag{R2.8}$$

Il reste donc à établir une formule *progressive* en t^0 d'ordre 2 (i.e. une formule utilisant les points t^0, t^1, t^2, ...) et une formule *régressive* en t^N d'ordre 2 (i.e. une formule utilisant les points t^N, t^{N-1}, t^{N-2}, ...).

De manière générique pour établir une formule *progressive* en t d'ordre 2 approchant $f'(t)$, on écrit les deux formules de Taylor aux points $t+h$ et $t+2h$.

- En $t+h$ on a
$$f(t+h) = f(t) + hf'(t) + \frac{h^2}{2!} f^{(2)}(t) + \mathcal{O}(h^3). \tag{R2.9}$$
Plus précisément, on a l'existence d'un $\xi_1 \in]t, t+h[$ tel que l'on peut remplacer le $\mathcal{O}(h^3)$ de (R2.9) par $\frac{h^3}{3!} f^{(3)}(\xi_1)$.
- En $t+2h$ on a,
$$f(t+2h) = f(t) + (2h)f'(t) + \frac{(2h)^2}{2!} f^{(2)}(t) + \mathcal{O}(h^3). \tag{R2.10}$$
Plus précisément, on a l'existence d'un $\xi_2 \in]t, t+2h[$ tel que l'on peut remplacer le $\mathcal{O}(h^3)$ de (R2.9) par $\frac{(2h)^3}{3!} f^{(3)}(\xi_2)$.

On va utiliser les formules de Taylor avec les restes explicites mais il est tout à fait possible d'utiliser celles avec les restes sous forme de $\mathcal{O}(h^3)$.
L'objectif étant d'obtenir une formule d'ordre 2 en t pour approcher $f'(t)$, on va éliminer les termes en $f^{(2)}(t)$ en effectuant la combinaison $4 \times$ (R2.9) $-$ (R2.10).
On obtient alors

$$4f(t+h) - f(t+2h) = 3f(t) + 2hf'(t) + 4\frac{h^3}{3!} f^{(3)}(\xi_1) - \frac{(2h)^3}{3!} f^{(3)}(\xi_2)$$

et donc

$$\begin{aligned} f'(t) &= \frac{-3f(t)+4f(t+h)-f(t+2h)}{2h} - 4\frac{h^2}{3!}f^{(3)}(\xi_1) + \frac{2^3h^2}{3!}f^{(3)}(\xi_2) \\ &= \frac{-3f(t)+4f(t+h)-f(t+2h)}{2h} + \mathcal{O}(h^2) \end{aligned}$$

Cette dernière formule permet alors l'obtention d'une formule d'ordre 2 en $t = t^0 = a$:

$$f'(t^0) = \frac{-3f(t^0)+4f(t^1)-f(t^2)}{2h} + \mathcal{O}(h^2) \tag{R2.11}$$

De la même manière pour établir une formule *régressive* en t d'ordre 2 approchant $f'(t)$, on écrit les deux formules de Taylor aux points $t-h$ et $t-2h$:

- En $t-h$ on a
$$f(t-h) = f(t) + (-h)f'(t) + \frac{(-h)^2}{2!}f^{(2)}(t) + \mathcal{O}(h^3), \tag{R2.12}$$
- En $t-2h$ on a
$$f(t-2h) = f(t) + (-2h)f'(t) + \frac{(-2h)^2}{2!}f^{(2)}(t) + \mathcal{O}(h^3), \tag{R2.13}$$

  L'objectif étant d'obtenir une formule d'ordre 2 en t pour approcher $f'(t)$, on va éliminer les termes en $f^{(2)}(t)$ en effectuant la combinaison $4 \times$ (R2.12) $-$ (R2.13). On obtient alors

$$4f(t-h) - f(t-2h) = 3f(t) - 2hf'(t) + \mathcal{O}(h^3)$$

  car toute combinaison de $\mathcal{O}(h^3)$ reste en $\mathcal{O}(h^3)$. On obtient donc

$$f'(t) = \frac{3f(t)-4f(t-h)+f(t-2h)}{2h} + \mathcal{O}(h^2).$$

Cette dernière formule permet alors l'obtention d'une formule d'ordre 2 en $t = t^N = b$:

$$f'(t^N) = \frac{3f(t^N)-4f(t^{N-1})+f(t^{N-2})}{2h} + \mathcal{O}(h^2) \tag{R2.14}$$

On peut alors construire le vecteur $\boldsymbol{W}$ en utilisant les formules (R2.8), (R2.11) et (R2.14) :

$$\begin{aligned} W_1 &\overset{\text{def}}{=} \frac{-3f(t^0)+4f(t^1)-f(t^2)}{2h} &&= f'(t^0) + \mathcal{O}(h^2), \quad \text{formule progressive (R2.11)} \\ W_{N+1} &\overset{\text{def}}{=} \frac{3f(t^N)-4f(t^{N-1})+f(t^{N-2})}{2h} &&= f'(t^N) + \mathcal{O}(h^2), \quad \text{formule régressive (R2.14)} \end{aligned}$$

et pour les points strictement intérieurs

$$W_{n+1} \overset{\text{def}}{=} \frac{f(t^{n+1})-f(t^{n-1})}{2h} = f'(t^n) + \mathcal{O}(h^2), \quad \text{formule centrée (R2.8) avec } n \in]\!]0, N[\![$$

On obtient alors en fonction de $\boldsymbol{F}$ et h

$$\begin{aligned} W_1 &\overset{\text{def}}{=} \frac{-3F_1+4F_2-F_3}{2h} &&= f'(t^0) + \mathcal{O}(h^2) \\ \forall n \in]\!]0, N[\![,\ W_n &\overset{\text{def}}{=} \frac{F_{n+1}-F_{n-1}}{2h} &&= f'(t^n) + \mathcal{O}(h^2) \\ W_{N+1} &\overset{\text{def}}{=} \frac{3F_{N+1}-4F_N+F_{N-1}}{2h} &&= f'(t^N) + \mathcal{O}(h). \end{aligned}$$

b. On représente tout d'abord les vecteurs $\mathbf{F}$ et $\mathbf{W}$ en Figure 2.4.

$\mathbf{F}$	$f(t^0)$	$f(t^1)$	$\cdots$	$f(t^{n-1})$	$\cdots$	$f(t^{N-1})$	$f(t^N)$
	$\mathbf{F}(1)$	$\mathbf{F}(2)$		$\mathbf{F}(n)$		$\mathbf{F}(N)$	$\mathbf{F}(N+1)$
$\mathbf{W}$	$f'(t^0)$ + $\mathcal{O}(h^2)$	$f'(t^1)$ + $\mathcal{O}(h^2)$	$\cdots$	$f'(t^{n-1})$ + $\mathcal{O}(h^2)$	$\cdots$	$f'(t^{N-1})$ + $\mathcal{O}(h^2)$	$f'(t^N)$ + $\mathcal{O}(h^2)$
	$\mathbf{W}(1)$	$\mathbf{W}(2)$		$\mathbf{W}(n)$		$\mathbf{W}(N)$	$\mathbf{W}(N+1)$

Figure 2.4: Représentation mémoire du vecteur $\boldsymbol{F} \in \mathbb{R}^{N+1}$ et du vecteur $\boldsymbol{W} \in \mathbb{R}^{N+1}$.

En utilisant les formules trouvées précédemment, la fonction algorithmique peut s'écrire sous la forme :

Algorithme 2.4 Approximations d'ordre 2 de dérivées premières d'une fonction f définie sur un intervalle en utilisant uniquemment les valeurs de f aux points $(t^n)_{n=0}^N$ d'une discrétisation régulière de cet intervalle. Les approximations sont calculées en tous les points de la discrétisation.

Données : $\boldsymbol{F}$: vecteur de $\mathbb{R}^{N+1}$ tel que $F_{n+1} = f(t^n)$, $\forall n \in [\![0, N]\!]$.
h : nombre réel strictement positif.
Résultat : $\boldsymbol{W}$: vecteur de $\mathbb{R}^{N+1}$tel que $W_{n+1} = f'(t^n) + \mathcal{O}(h^2)$, $\forall n \in [\![0, N]\!]$.

1: **Fonction** $\boldsymbol{W} \leftarrow$ Derive1Ordre2($h, \boldsymbol{F}$)
2: $W(1) \leftarrow (-3 * F(1) + 4 * F(2) - F(3))/(2 * h)$
3: **Pour** $i \leftarrow 2$ **à** N **faire**
4: $W(i) \leftarrow (F(i+1) - F(i-1))/(2 * h)$
5: **Fin**
6: $W(N+1) \leftarrow (3 * F(N+1) - 4 * F(N) + F(N-1))/(2 * h)$
7: **Fin**

D'autres façons de présenter le problème sont possibles mais les données peuvent différer de celles de l'énoncé. Par exemple une autre fonction algorithmique pourrait être

Algorithme 2.5 Approximations d'ordre 2 de dérivées premières d'une fonction f définie sur un intervalle $[a, b]$ calculées aux points de de la discrétisation régulière de $[a, b]$ avec N pas de discrétisation. Les approximations sont calculées en tous les points de la discrétisation

Données : f : $f : [a, b] \longrightarrow \mathbb{R}$
a, b : deux réels, $a < b$,
N : nombre de pas de discrétisation
Résultat : $\boldsymbol{W}$: vecteur de $\mathbb{R}^{N+1}$ tel que $W_{n+1} = f'(t^n) + \mathcal{O}(h^2)$
avec $(t^n)_{n=0}^N$ discrétisation régulière de $[a, b]$.

```
Fonction W ← Derive1Ordre2-v1( f, a, b, N )
    t ← DisReg(a, b, N)                    ▷ fonction retournant la discrétisation régulière...
    h ← (b − a)/N
    W(1) ← (−3 ∗ f(t(1)) + 4 ∗ f(t(2)) − f(t(3)))/(2 ∗ h)
    Pour i ← 2 à N faire
        W(i) ← (f(t(i + 1)) − f(t(i − 1)))/(2 ∗ h)
    Fin
    W(N + 1) ← (3 ∗ f(t(N + 1)) − 4 ∗ f(t(N)) + f(t(N − 1)))/(2 ∗ h)
Fin
```

Fin correction

Pour illustrer l'intéret de *monter* en ordre, on représente en Figure 2.5 les dérivées numériques calculées avec les fonctions DeriveOrdre1 et DeriveOrdre2. On peut noter visuellement la meilleur concordance de la dérivée numérique d'ordre 2 sur la figure de gauche. Sur celle de droite, c'est moins clair car les différentes courbes sont presque confondues.

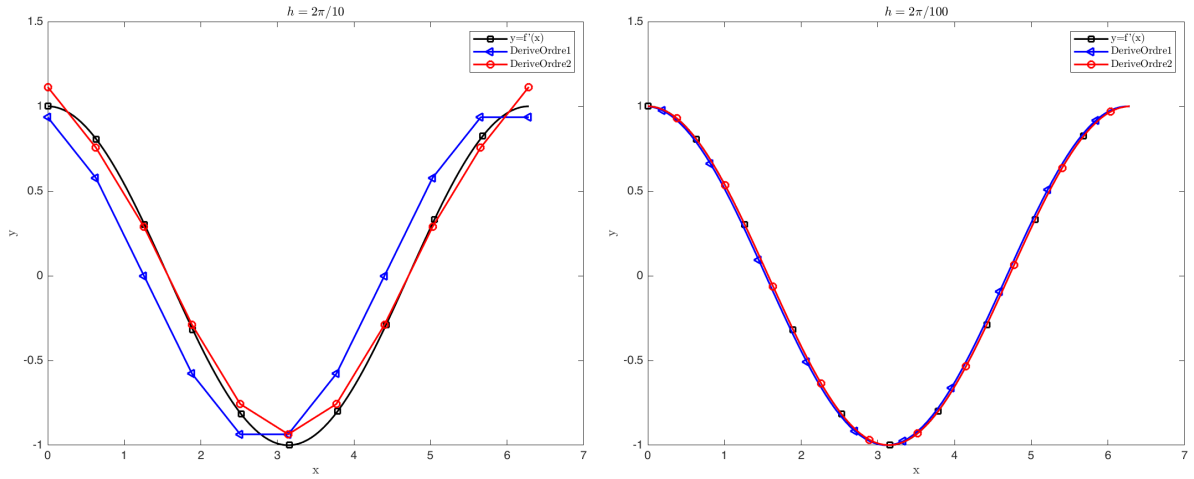


Figure 2.5: Représentation des dérivées numériques avec $f(x) := sin(x)$, $a = 0$, $b = 2\pi$. A gauche $h = \frac{2\pi}{10}$, à droite $h = \frac{2\pi}{100}$.

Pour une meilleur interprétation et connaissant la dérivée de la fonction, on représente en Figure 2.6 les erreurs entre les dérivées numériques et la dérivée exacte.

- Pour la dérivée première d'ordre 1, l'erreur maximale pour $h = \frac{2\pi}{10}$ est d'environ 0.3 et pour $h = \frac{2\pi}{100}$ d'environ 0.03 : le pas h a été divisé par 10 et comme la méthode est d'odre 1 l'erreur, qui est en $\mathcal{O}(h)$, est aussi divisée par 10.

- Pour la dérivée première d'ordre 2, l'erreur maximale pour $h = \frac{2\pi}{10}$ est d'environ 0.1 et pour $h = \frac{2\pi}{100}$ d'environ 0.001 : le pas h a été divisé par 10 et comme la méthode est d'ordre 2 l'erreur, qui est en $\mathcal{O}(h^2)$, est divisée par 100!

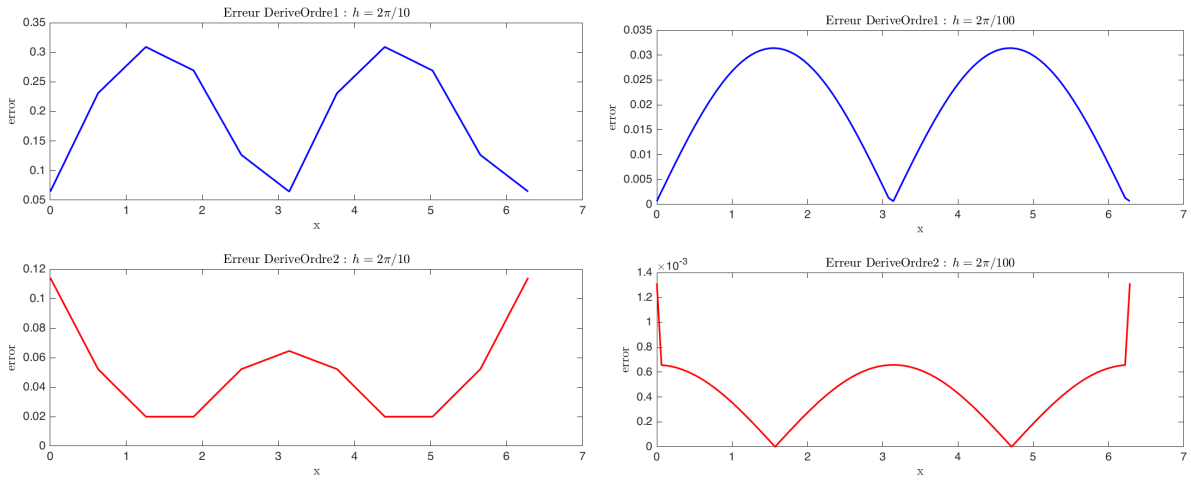


Figure 2.6: Erreur des dérivées premières numériques avec $f(x) := sin(x)$, $a = 0$, $b = 2\pi$. A gauche $h = \frac{2\pi}{10}$, à droite $h = \frac{2\pi}{100}$.

La Figure 2.7 en échelle logarithmique permet de se rendre compte graphiquement de l'intéret de *monter* en ordre.

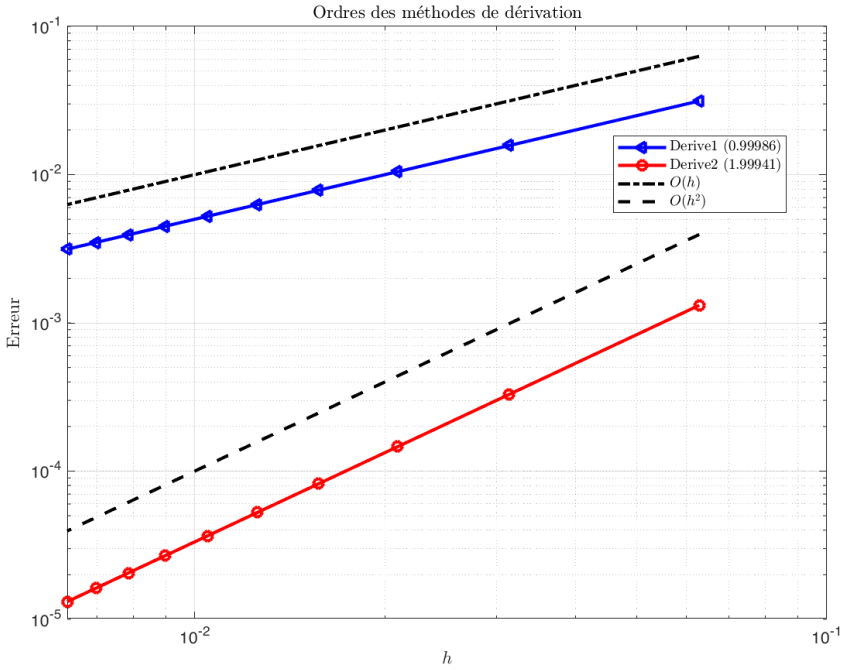


Figure 2.7: Dérivation numérique : mise en évidence de l'ordre des méthodes

Exercice 2.3.3

Q. 1

a. Ecrire un programme Matlab/Octave permettant de calculer l'ensemble des données nécessaires à la représentation graphique de l'ordre des deux méthodes (voir figure 2.7).

b. A l'aide de ces données, calculer numériquement l'ordre des deux méthodes.

Les commandes Matlab/Octave permettant de représenter des données en échelles logarithmique sont `loglog`, `semilogx` et `semilogy`. Elles s'utilisent globalement comme la fonction `plot`.

Q. 2

Ajouter au programme précédent le code permettant de reproduire la figure.

Chapitre 3

Introduction à la résolution d'E.D.O.

3.1 Introduction

Les équations différentielles ordinaires ou E.D.O.* sont utilisées pour modéliser un grand nombre de phénomènes mécaniques, physiques, chimiques, biologiques, ...

Définition 3.1.1. *On appelle* ***équation différentielle ordinaire (E.D.O.) d'ordre*** p *une équation de la forme :*

$$\mathcal{F}(t, \boldsymbol{y}(t), \boldsymbol{y}^{(1)}(t), \boldsymbol{y}^{(2)}(t), \ldots, \boldsymbol{y}^{(p)}(t)) = 0.$$

Définition 3.1.2. *On appelle* ***forme canonique d'une E.D.O.*** *une expression du type :*

$$\boldsymbol{y}^{(p)}(t) = \boldsymbol{\mathcal{G}}(t, \boldsymbol{y}(t), \boldsymbol{y}^{(1)}(t), \boldsymbol{y}^{(2)}(t), \ldots, \boldsymbol{y}^{(p-1)}(t)). \tag{3.1}$$

Proposition 3.1.1. *Toute équation différentielle d'ordre p sous forme canonique peut s'écrire comme un système de p équations différentielles d'ordre* 1.

3.1.1 Exemple en météorologie : modèle de Lorentz

Mathématiquement, le couplage de l'atmosphère avec l'océan est décrit par le système d'équations aux dérivées partielles couplées de Navier-Stokes de la mécanique des fluides. Ce système d'équations était beaucoup trop compliqué à résoudre numériquement pour les premiers ordinateurs existant au temps de Lorenz. Celui-ci eut donc l'idée de chercher un modèle très simplifié de ces équations pour étudier une situation physique particulière : le phénomène de convection de Rayleigh-Bénard. Il aboutit alors à un système dynamique différentiel possédant seulement trois degrés de liberté, beaucoup plus simple à intégrer numériquement que les équations de départ.

*En anglais, *ordinary differential equations* ou O.D.E.

(a) *Edward Norton Lorenz* 1917-2008, Mathématicien et météorologiste américain

$$\begin{cases} x'(t) &= -\sigma x(t) + \sigma y(t) \\ y'(t) &= -x(t)z(t) + \rho x(t) - y(t) \\ z'(t) &= x(t)y(t) - \beta z(t) \end{cases} \tag{3.2}$$

avec $\sigma = 10$, $\rho = 28$, $\beta = 8/3$ et les données initales $x(0) = -8$, $y(0) = 8$ et $z(0) = \rho - 1$. C'est un système de trois E.D.O. d'ordre 1.
Dans ces équations, σ,ρ et β sont trois paramètres réels.

$x(t)$ est proportionnel à l'intensité du mouvement de convection, $y(t)$ est proportionnel à la différence de température entre les courants ascendants et descendants, et $z(t)$ est proportionnel à l'écart du profil de température vertical par rapport à un profil linéaire (Lorenz 1963 p.135).

En représentant, en Figure 3.2, la courbe paramétré $(x(t), y(t), z(t))$ dans l'espace, on obtient l'*attracteur étrange de Lorenz* en forme d'aile de papillon.

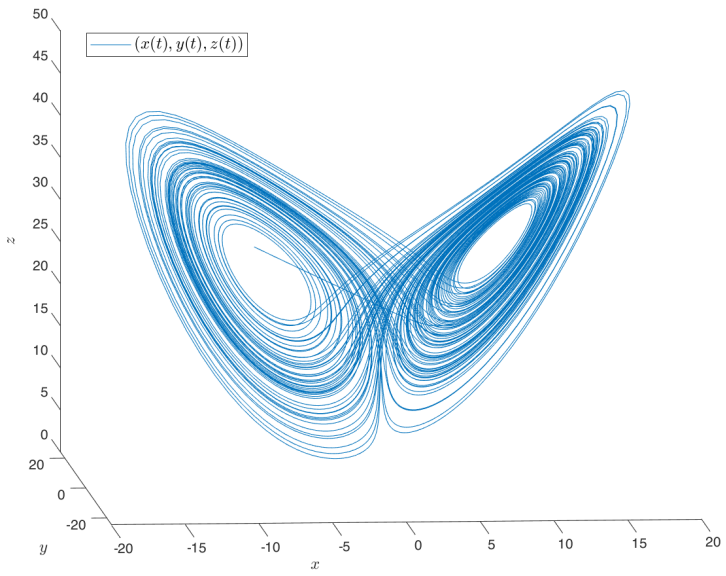


Figure 3.2: Attracteur étrange de Lorenz

Pour illustrer le **caractère chaotique** de ce système différentiel, on le résoud numériquement avec les données initiales *exactes* (courbe bleue de la figure 3.3) et avec la donnée initiale $x(0)$ *perturbée* : $x(0) = -8 + 1e - 4$ (courbe rouge de la figure 3.3). Une très légère variation des données initiales engendrent de très forte variation de la solution.

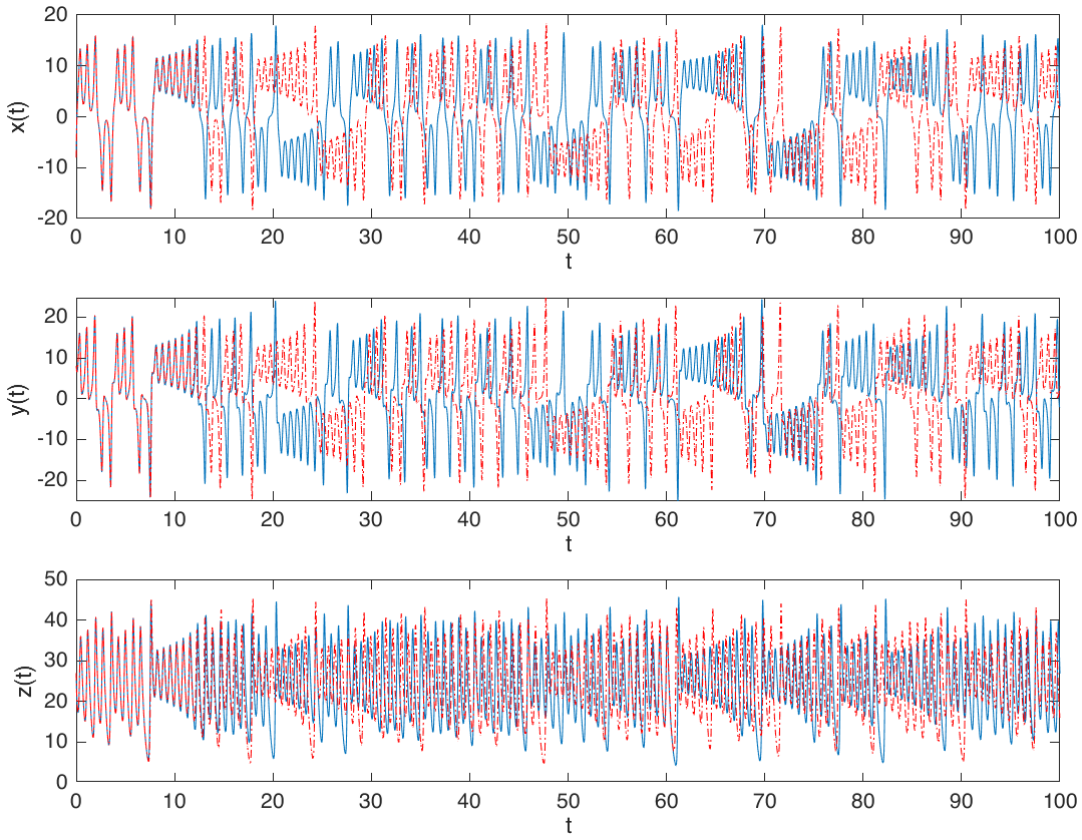


Figure 3.3: Illustration du caractère chaotique du modèle de Lorenz : en bleu, donnée initiale $x(0) = -8, y(0) = 8, z(0) = 27$, et en rouge pointillée, donnée initiale perturbée $x(0) = -8 + 1e - 4, y(0) = 8, z(0) = 27$.

3.1.2 Exemple en biologie

Considérons une population y d'animaux dans un milieu ambient où au plus B animaux peuvent coexister. On suppose que initialement la population soit $y0 << B$ et que le facteur de croissance des animaux soit égal à une constante C. Dans ce cas, l'évolution de la population au cours du temps sera proportionnelle au nombre d'animaux existants, sans toutefois que ce nombre ne dépasse la limite B. Cela peut s'exprimer à travers l'équation

$$y'(t) = Cy(t)\left(1 - \frac{y(t)}{B}\right), \ t > 0, \ y(0) = y0. \tag{3.3}$$

La résolution de cette équation permet de trouver l'évolution de la population au cours du temps.
On considère maintenant deux populations, y_1 et y_2, où y_1 sont les proies et y_2 sont les prédateurs. L'évolution des deux populations est alors décrite par le système d'équations différentielles

$$\begin{cases} y_1'(t) &= C_1 y_1(t)[1 - b_1 y_1(t) - d_2 y_2(t)], \\ y_2'(t) &= -C_2 y_2(t)[1 - d_1 y_1(t)], \end{cases} \tag{3.4}$$

où C_1 et C_2 sont les facteurs de croissance des deux populations, d_1 et d_2 tiennent compte de l'interaction entre les deux populations, tandis que b_1 est lié à la quantité de nourriture disponible pour la population des proies y_1. Ce système de deux équations différentielles d'odre 1 est connu comme modèle de *Lotka-Volterra*.

3.1.3 Exemple en chimie : La réaction de Belousov-Zhabotinsky

Sous certaines conditions, des réactions chimiques peuvent être oscillantes. Par exemple, le mélange d'une solution de bromate de potassium et d'acide sulfurique avec une solution d'acide manolique et de bromure de sodium peut entrainer une oscillation de la couleur de la solution mélange du rouge au bleue avec une période de 7 secondes.

(a) *Boris Pavlovich Belousov* 1893-1970, Chimiste et biophysicien russe

(b) *Anatol Zhabotinsky* 1938-2008, Chimiste russe

(c) *Ilya Prigogine* 1917-2003, Physicien et chimiste belge (origine russe). Prix Nobel de chimie en 1977

Un modéle dérivé est nommé **modèle du brusselator** proposé par I. Prigogine (Université libre de Bruxelle) basé sur les équations chimiques :

$$\begin{array}{lcll} A & \longrightarrow & X & (M_1) \\ 2X+Y & \longrightarrow & 3X & (M_2) \\ B+X & \longrightarrow & Y+C & (M_3) \\ X & \longrightarrow & D & (M_4) \end{array}$$

On note k_i, $i \in [\![1,4]\!]$ les vitesses de réactions des équations (M_i). On obtient alors le système différentiel suivant :

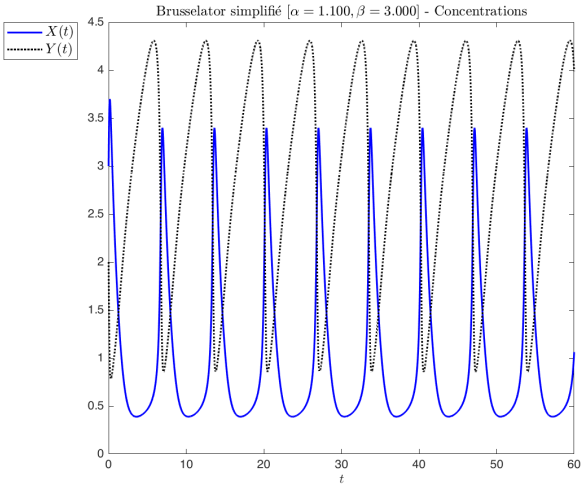
$$\left\{ \begin{array}{lcl} A'(t) & = & -k_1 A(t) \\ B'(t) & = & -k_3 B(t) X(t) \\ X'(t) & = & k_1 A(t) + k_2 X^2(t) Y(t) - k_3 B(t) X(t) - k_4 X(t) \\ Y'(t) & = & -k_2 X^2(t) Y(t) + k_3 B(t) X(t) \end{array} \right. \tag{3.5}$$

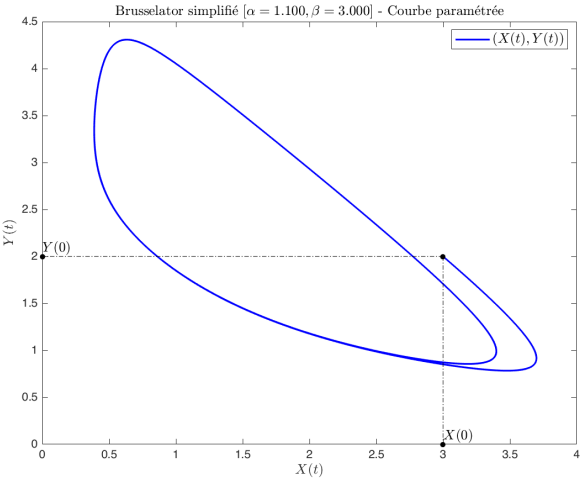
Exemple. *Dans cet exemple, on étudie le problème simplifié du Brusselator. Lorsque les réactions de (3.5) ont des constantes de réactions $k_1, \ldots, k_4$ égales respectivement à 1, $\alpha > 0$, 1 et 1, et que les concentrations de A et B sont constantes, respectivement égales à 1 et $\beta > 0$, on est alors amené à résoudre l'E.D.O. $\forall t \in]0, T]$,*

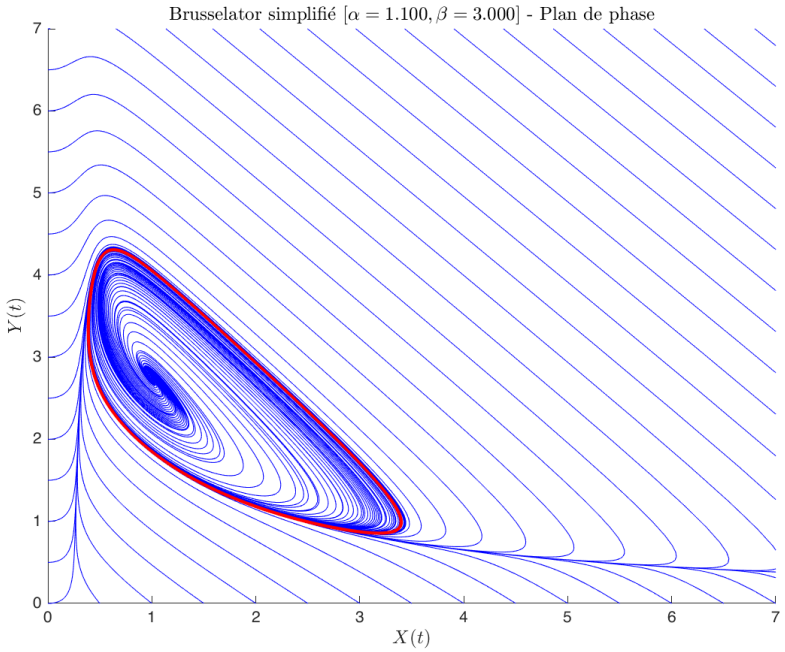
$$\left\{ \begin{array}{lcl} X'(t) & = & 1 + \alpha X^2(t) Y(t) - (\beta + 1) X(t) \\ Y'(t) & = & -\alpha X^2(t) Y(t) + \beta X(t) \end{array} \right. \tag{3.6}$$

C'est un système de deux E.D.O. d'ordre 1.

Par exemple, avec le jeu de données $\alpha = 1$, $\beta = 3.5$ et les contitions initiales $X(0) = 3$ et $Y(0) = 2$ on obtient des oscillations :







3.1.4 Exemple en mécanique

Le pendule pesant le plus simple est constitué d'un petit objet pesant accroché à une tige de masse négligeable devant celle de l'objet. L'autre extrémité de la tige est l'axe de rotation du pendule. On note $\theta(t)$ l'angle (en radian) que fait le pendule par rapport à l'axe vertical, à un instant t, L la longueur de la tige, M sa masse et g l'accélération de la pesanteur. Ce pendule est représenté en Figure 3.5.

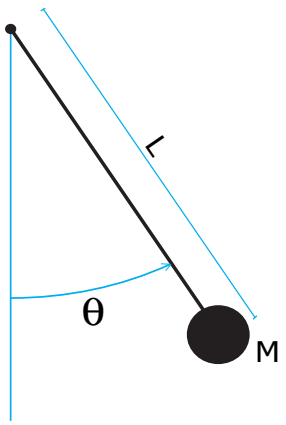


Figure 3.5: Pendule simple

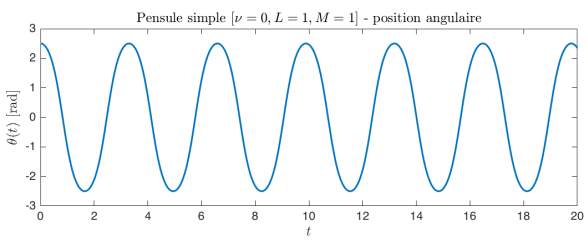
Si le pendule est soumis à un frottement visqueux de coefficient $\nu > 0$, l'équation différentielle est alors

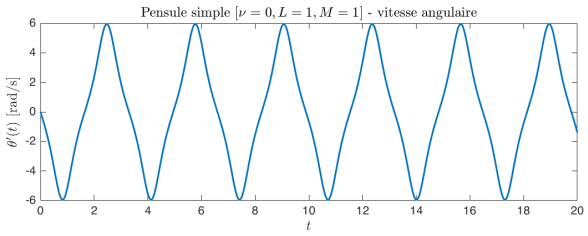
$$\theta''(t) + \frac{g}{L}\sin(\theta(t)) + \frac{\nu}{ML^2}\theta'(t) = 0, \ \forall t \in]0, T], \tag{3.7}$$

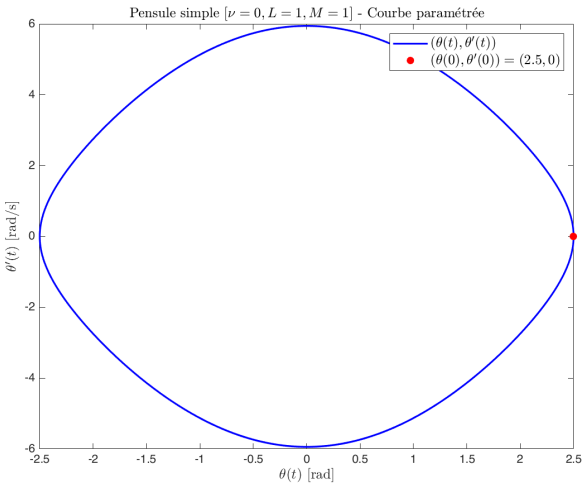
avec les conditions initiales

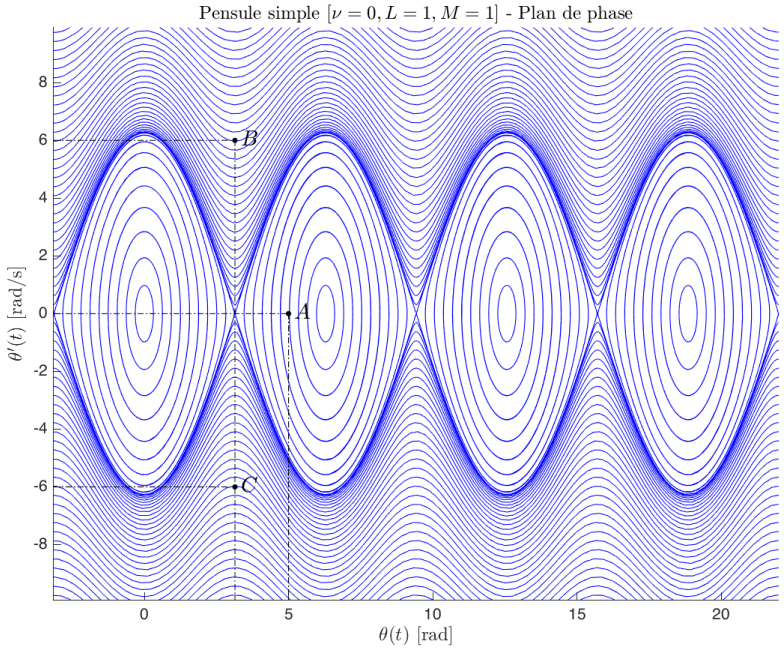
$$\begin{cases} \theta(0) &= \theta_0, \quad \text{position angulaire initiale [rad]}, \\ \theta'(0) &= \theta_0, \quad \text{vitesse angulaire initiale [rad/s]}, \end{cases} \tag{3.8}$$

C'est une E.D.O. d'ordre 2.
Par exemple, dans le cas non visqueux $\nu = 0$, avec le jeu de données $\frac{g}{L} = 3$ et les conditions initiales $\theta_0 = \frac{5\pi}{6}$ et $\theta_0' = 0$ on obtient









3.2 Problème de Cauchy

Pour résoudre numériquement une E.D.O., nous allons la réécrire sous une forme plus *générique* : le problème de Cauchy. De très nombreux résultats mathématiques existent sur les problèmes de Cauchy. Nous ne ferons que rappeler le théorème de Cauchy-Lipschitz (19ème siècle). L'ensemble des méthodes numériques que nous allons étudier auront pour but la résolution d'un problème de Cauchy quelconque. Elles pourront donc être utilisées (sans efforts) pour la résolution d'une très grande variété d'E.D.O.

(a) *Augustin Louis Cauchy* 1789-1857, mathématicien français

(b) *Rudolf Lipschitz* 1832-1903, mathématicien allemand

On commence par donner la définition d'un problème de Cauchy :

Définition 3.2.1 (problème de Cauchy ★★★★★). *Soit $\boldsymbol{f}$ l'application continue donnée par*

$$\begin{array}{rccc} \boldsymbol{f} & : & [t^0, t^0+T]\times\mathbb{R}^m & \longrightarrow & \mathbb{R}^m \\ & & (t,\boldsymbol{y}) & \longmapsto & \boldsymbol{f}(t,\boldsymbol{y}) \end{array}$$

*avec $T\in]0,+\infty[$. Le **problème de Cauchy** consiste à déterminer une fonction $\boldsymbol{y}$ définie par*

$$\begin{array}{rccc} \boldsymbol{y} & : & [t^0, t^0+T] & \longrightarrow & \mathbb{R}^m \\ & & t & \longmapsto & \boldsymbol{y}(t)=\begin{pmatrix} y_1(t) \\ \vdots \\ y_m(t)\end{pmatrix} \end{array}$$

continue et dérivable, telle que

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t,\boldsymbol{y}(t)),\quad \forall t\in[t^0,t^0+T] \tag{3.9}$$
$$\boldsymbol{y}(t^0) = \boldsymbol{y}^{[0]}\in\mathbb{R}^m. \tag{3.10}$$

Exercice 3.2.1

Quelles sont les données du problème de Cauchy (3.9)-(3.10)?

Correction Les données du problème de Cauchy (3.9)-(3.10) sont

- $t^0\in\mathbb{R}$, le temps initial,
- $T\in\mathbb{R}^{+*}$, la periode de temps,
- $\boldsymbol{y}^{[0]}\in\mathbb{R}^m$, la donnée initiale,
- $\boldsymbol{f}:[t^0,t^0+T]\times\mathbb{R}^m\longrightarrow\mathbb{R}^m$, la fonction de Cauchy.

On peut noter que $m\in\mathbb{N}^*$ est donnée implicitement par la connaissance de $\boldsymbol{y}^{[0]}$: il n'est donc pas nécessaire de l'ajouter à la liste des données.

Fin correction

Dans de nombreux cas, la variable t représente le temps et les composantes du vecteur $\boldsymbol{y}$, une famille de paramètres décrivant l'état d'un système matériel donné. L'équation différentielle (3.9) traduit physiquement la loi d'évolution du système considéré.
En général, il est impossible de trouver analytiquement des solutions à ces problèmes. Il faut alors construire des méthodes numériques pour obtenir des solutions approchées. Toutefois, il serait bon de vérifier l'existence et l'unicité d'une solution.
Par exemple, le problème suivant

$$\begin{cases} y'(t) = \sqrt[3]{y(t)}, & \text{si } t\geqslant 0 \\ y(0) = 0 \end{cases}$$

peut s'écrire sous la forme d'un problème de Cauchy avec $t^0=0$, $T=+\infty$, $m=1$, $\boldsymbol{y}^{[0]}=0$ et

$$\begin{array}{rccc} \boldsymbol{f} & : & [t^0, t^0+T]\times\mathbb{R} & \longrightarrow & \mathbb{R} \\ & & (t,v) & \longmapsto & \sqrt[3]{v} \end{array}.$$

Ce problème admet les **trois solutions** suivantes : $y(t)=0$, $y(t)=\sqrt{8t^3/27}$ et $y(t)=-\sqrt{8t^3/27}$.
Le théorème suivant assure l'existence et l'unicité sous certaines conditions.

Théorème 3.2.1 (Cauchy-Lipschitz). *Soit le problème de Cauchy donné par la définition 3.2.1. On suppose que la fonction $\boldsymbol{f}$ est continue sur un ouvert U de $\mathbb{R}\times\mathbb{R}$ et quelle est localement lipschitzienne en $\boldsymbol{y}$: $\forall(t,\boldsymbol{y})\in U$, $\exists\mathcal{W}$ voisinage $\boldsymbol{t}$, $\exists\mathcal{V}$ voisinage $\boldsymbol{y}$, $\exists L>0$ tels que*

$$\forall s\in\mathcal{W},\ \forall(\boldsymbol{u},\boldsymbol{v})\in\mathcal{V}^2,\quad \|\boldsymbol{f}(s,\boldsymbol{u})-\boldsymbol{f}(s,\boldsymbol{v})\|\leqslant L\,\|\boldsymbol{u}-\boldsymbol{v}\| \tag{3.11}$$

Sous ces hypothèses le problème de Cauchy (3.9)-(3.10) admet une unique solution.

Proposition 3.2.1. *Si* $\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t,\boldsymbol{y})$ *est continue et bornée, alors* $\boldsymbol{f}$ *satisfait la condition de Lipschitz (3.11) en* $\boldsymbol{y}$.

Exercice 3.2.2

Pour chacune des E.D.O. suivantes écrire le problème de Cauchy associé

$$(a)\ \begin{cases} x''(t)+\alpha x'(t)+\beta\cos(x(t))=\sin(t),\ t\in]0,2\pi] \\ x(0)=0,\ x'(0)=1. \end{cases}$$

$$(b)\ \begin{cases} LCv''(t)+\left(\dfrac{L}{R_2}+R_1C\right)v'(t)+\left(\dfrac{R_1}{R_2}+1\right)v(t)=e,\ t\in]0,100] \\ v(0)=0,\ v'(0)=0. \end{cases}$$

$$(c)\ \begin{cases} x''(t)=\mu(1-x^2(t))x'(t)-x(t),\ t\in]0,10] \\ x(0)=1,\ x'(0)=-1. \end{cases}$$

$$(d)\ \begin{cases} y^{(3)}(t)-\cos(t)y^{(2)}(t)+2\sin(t)y^{(1)}(t)-y(t)=0,\ t\in]0,T] \\ y(0)=u_0,\ y^{(1)}(0)=v_0,\ y^{(2)}(0)=w_0. \end{cases}$$

$$(e)\ \begin{cases} \forall t\in]0,T], & x_1''(t)-2x_2'(t)+3x_1'(t)+4x_1(t)x_2(t)=\sin(t), \\ & x_2''(t)+3x_1'(t)-2x_2'(t)-3x_1(t)x_2(t)=\cos(t), \\ & x_1(0)=0,\ x_1'(0)=-1,\ x_2(0)=1,\ x_2'(0)=-2. \end{cases}$$

Correction

(a) C'est une E.D.O. d'ordre 2. Pour écrire le problème de Cauchy assosicé, on écrit l'E.D.O. sous la forme d'un système de 2 E.D.O. d'ordre 1 (voir Proposition 3.1.1) en prenant $m=2$ et en posant

$$\boldsymbol{y}(t)\overset{\text{def}}{=}\begin{pmatrix} y_1(t)\\ y_2(t)\end{pmatrix}=\begin{pmatrix} x(t)\\ x'(t)\end{pmatrix}.$$

On a alors

$$\begin{aligned}\boldsymbol{y}'(t) &= \begin{pmatrix} x'(t)\\ x''(t)\end{pmatrix}=\begin{pmatrix} x'(t)\\ -\alpha x'(t)-\beta\cos(x(t))+\sin(t)\end{pmatrix} \\ &= \begin{pmatrix} y_2(t)\\ -\alpha y_2(t)-\beta\cos(y_1(t))+\sin(t)\end{pmatrix}=\boldsymbol{f}(t,\boldsymbol{y}(t))\end{aligned}$$

Le problème de Cauchy associé est donc

trouver la fonction $\boldsymbol{y}:[0,2\pi]\longrightarrow\mathbb{R}^2$ vérifiant

$$\begin{aligned}\boldsymbol{y}'(t) &= \boldsymbol{f}(t,\boldsymbol{y}(t)),\quad \forall t\in[0,2\pi] \\ \boldsymbol{y}(0) &= \begin{pmatrix}0\\1\end{pmatrix}\in\mathbb{R}^2\end{aligned}$$

avec

$$\begin{aligned}\boldsymbol{f}\ :\ [0,2\pi]\times\mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (t,\boldsymbol{z}) &\longmapsto \begin{pmatrix} z_2\\ -\alpha z_2-\beta\cos(z_1)+\sin(t)\end{pmatrix}\end{aligned}$$

(b) Pour cette E.D.O. on suppose les paramètres physiques L, C, R_1 et R_2 donnés. C'est une E.D.O. d'ordre 2. Pour écrire le problème de Cauchy assosicé, on écrit l'E.D.O. sous la forme d'un système de 2 E.D.O. d'ordre 1 (voir Proposition 3.1.1) en prenant $m=2$ et en posant

$$\boldsymbol{y}(t)\overset{\text{def}}{=}\begin{pmatrix} y_1(t)\\ y_2(t)\end{pmatrix}=\begin{pmatrix} v(t)\\ v'(t)\end{pmatrix}.$$

On a alors

$$\begin{aligned}\boldsymbol{y}'(t) &= \begin{pmatrix} v'(t) \\ v''(t)\end{pmatrix} = \begin{pmatrix} v'(t) \\ \frac{1}{LC}\left(e - (\frac{L}{R_2} + R_1C)v'(t) - (\frac{R_1}{R_2}+1)v(t)\right)\end{pmatrix} \\ &= \begin{pmatrix} y_2(t) \\ \frac{e}{LC} - (\frac{1}{CR_2} + \frac{R_1}{L})y_2(t) - \frac{1}{LC}(\frac{R_1}{R_2}+1)y_1(t)\end{pmatrix} = \boldsymbol{f}(t,\boldsymbol{y}(t))\end{aligned}$$

Le problème de Cauchy associé est donc

trouver la fonction $\boldsymbol{y} : [0,100] \longrightarrow \mathbb{R}^2$ vérifiant

$$\begin{aligned}\boldsymbol{y}'(t) &= \boldsymbol{f}(t,\boldsymbol{y}(t)), \quad \forall t \in [0,100] \\ \boldsymbol{y}(0) &= \begin{pmatrix}0\\0\end{pmatrix} \in \mathbb{R}^2\end{aligned}$$

avec

$$\begin{array}{rccl} \boldsymbol{f} & : & [0,100]\times\mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & & (t,\boldsymbol{z}) & \longmapsto & \begin{pmatrix} z_2 \\ \frac{e}{LC} - (\frac{1}{CR_2} + \frac{R_1}{L})z_2 - \frac{1}{LC}(\frac{R_1}{R_2}+1)z_1\end{pmatrix}\end{array}$$

(*c*) Pour cette E.D.O. on suppose le paramètre μ donné. C'est une E.D.O. d'ordre 2. Pour écrire le problème de Cauchy assosicé, on écrit l'E.D.O. sous la forme d'un système de 2 E.D.O. d'ordre 1 (voir Proposition 3.1.1) en prenant $m = 2$ et en posant

$$\boldsymbol{y}(t) \stackrel{\text{def}}{=} \begin{pmatrix} y_1(t) \\ y_2(t)\end{pmatrix} = \begin{pmatrix} x(t) \\ x'(t)\end{pmatrix}.$$

On a alors

$$\begin{aligned}\boldsymbol{y}'(t) &= \begin{pmatrix} x'(t) \\ x''(t)\end{pmatrix} = \begin{pmatrix} x'(t) \\ \mu(1-x^2(t))x'(t) - x(t)\end{pmatrix} \\ &= \begin{pmatrix} y_2(t) \\ \mu(1-y_1^2(t))y_2(t) - y_1(t)\end{pmatrix} = \boldsymbol{f}(t,\boldsymbol{y}(t))\end{aligned}$$

Le problème de Cauchy associé est donc

trouver la fonction $\boldsymbol{y} : [0,10] \longrightarrow \mathbb{R}^2$ vérifiant

$$\begin{aligned}\boldsymbol{y}'(t) &= \boldsymbol{f}(t,\boldsymbol{y}(t)), \quad \forall t \in [0,10] \\ \boldsymbol{y}(0) &= \begin{pmatrix}1\\-1\end{pmatrix} \in \mathbb{R}^2\end{aligned}$$

avec

$$\begin{array}{rccl} \boldsymbol{f} & : & [0,10]\times\mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & & (t,\boldsymbol{z}) & \longmapsto & \begin{pmatrix} z_2 \\ \mu(1-z_1^2)z_2 - z_1\end{pmatrix}\end{array}$$

(*d*) Pour cette E.D.O. on suppose les paramètres T, u_0, v_0 et w_0 donnés. C'est une E.D.O. d'ordre 3. Pour écrire le problème de Cauchy assosicé, on écrit l'E.D.O. sous la forme d'un système de 3 E.D.O. d'ordre 1 (voir Proposition 3.1.1) en prenant $m = 3$ et en posant

$$\boldsymbol{Y}(t) \stackrel{\text{def}}{=} \begin{pmatrix} Y_1(t) \\ Y_2(t) \\ Y_3(t)\end{pmatrix} = \begin{pmatrix} y(t) \\ y'(t) \\ y''(t)\end{pmatrix}.$$

On a noté ici $\boldsymbol{Y}$ au lieu de $\boldsymbol{y}$ pour éviter les confusions! On a alors

$$\begin{aligned}\boldsymbol{Y}'(t) &= \begin{pmatrix} y'(t) \\ y^{(2)}(t) \\ y^{(3)}(t)\end{pmatrix} = \begin{pmatrix} y'(t) \\ y^{(2)}(t) \\ \cos(t)y^{(2)}(t) - 2\sin(t)y^{(1)}(t) + y(t))\end{pmatrix} \\ &= \begin{pmatrix} Y_2(t) \\ Y_3(t) \\ \cos(t)Y_3(t) - 2\sin(t)Y_2(t) + Y_1(t))\end{pmatrix} = \boldsymbol{f}(t,\boldsymbol{Y}(t))\end{aligned}$$

Le problème de Cauchy associé est donc

trouver la fonction $\boldsymbol{y} : [0,T] \longrightarrow \mathbb{R}^3$ vérifiant

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}(t,\boldsymbol{y}(t)), \quad \forall t \in [0,T] \\ \boldsymbol{y}(0) &= \begin{pmatrix} u_0 \\ v_0 \\ w_0 \end{pmatrix} \in \mathbb{R}^3 \end{aligned}$$

avec

$$\begin{array}{rccc} \boldsymbol{f} & : & [0,T]\times\mathbb{R}^3 & \longrightarrow & \mathbb{R}^3 \\ & & (t,\boldsymbol{z}) & \longmapsto & \begin{pmatrix} z_2 \\ z_3 \\ \cos(t)z_3 - 2\sin(t)z_2 + z_1) \end{pmatrix} \end{array}$$

(*e*) C'est un système de deux E.D.O couplées: elles dépendent l'une de l'autre. Les deux E.D.O. oyant un terme en dérivée seconde, elles sont d'ordre 2. On va donc pouvoir *transformer* chacune des E.D.O. en deux E.D.O. d'ordre 1, pour aboutir à un système de 4 E.D.O. d'ordre 1.

On pose, par exemple,

$$\boldsymbol{y}(t) \stackrel{\text{def}}{=} \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \\ y_4(t) \end{pmatrix} = \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_1'(t) \\ x_2'(t) \end{pmatrix}.$$

Il aurait aussi été possible de prendre

$$\begin{pmatrix} x_1(t) \\ x_1'(t) \\ x_2(t) \\ x_2'(t) \end{pmatrix} \text{ ou } \begin{pmatrix} x_1'(t) \\ x_2'(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} \text{ ou } \ldots$$

Avec notre choix, on a

$$\begin{aligned} \boldsymbol{y}'(t) &= \begin{pmatrix} x_1'(t) \\ x_2'(t) \\ x_1''(t) \\ x_2''(t) \end{pmatrix} \\ &= \begin{pmatrix} y_3(t) \\ y_4(t) \\ 2x_2'(t) - 3x_1'(t) - 4x_1(t)x_2(t) + \sin(t) \\ -3x_1'(t) + 2x_2'(t) + 3x_1(t)x_2(t) + \cos(t) \end{pmatrix} \\ &= \begin{pmatrix} y_3(t) \\ y_4(t) \\ 2y_4(t) - 3y_3(t) - 4y_1(t)y_2(t) + \sin(t) \\ -3y_3(t) + 2y_4(t) + 3y_1(t)y_2(t) + \cos(t) \end{pmatrix} = \boldsymbol{f}(t,\boldsymbol{y}(t)). \end{aligned}$$

Le problème de Cauchy associé est donc

trouver la fonction $\boldsymbol{y} : [0,T] \longrightarrow \mathbb{R}^4$ vérifiant

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}(t,\boldsymbol{y}(t)), \quad \forall t \in [0,T] \\ \boldsymbol{y}(0) &= \begin{pmatrix} 0 \\ 1 \\ -1 \\ -2 \end{pmatrix} \in \mathbb{R}^4 \end{aligned}$$

avec

$$\begin{array}{rccc} \boldsymbol{f} & : & [0,T]\times\mathbb{R}^4 & \longrightarrow & \mathbb{R}^4 \\ & & (t,\boldsymbol{z}) & \longmapsto & \begin{pmatrix} z_3 \\ z_4 \\ 2z_4 - 3z_3 - 4z_1z_2 + \sin(t) \\ -3z_3 + 2z_4 + 3z_1z_2 + \cos(t) \end{pmatrix} \end{array}$$

Fin correction

3.3 Différences finies pour les E.D.O.

3.3.1 Différences finies pour le problème de Cauchy en dimension $m = 1$

On veut résoudre numériquement le problème de Cauchy (3.9)-(3.10) en utilisant les différences finies progressive, rétrograde ou centrée.
On note $t^n = t^0 + nh$, $n \in \{0, \dots, N\}$ une **discrétisation régulière de** $[t^0, t^0 + T]$ avec $h = T/N$ le **pas de temps**. On a

$$y'(t) = f(t, y(t)), \ \forall t \in [t^0, t^0 + T]$$

ce qui entraine

$$y'(t^n) = f(t^n, y(t^n)), \ \forall n \in [\![0, N]\!]. \tag{3.12}$$

En utilisant la formule des différences finies progressive, on a $\forall n \in [\![0, N-1]\!]$, $\exists \eta_n \in [t^n, t^{n+1}]$ tels que

$$\frac{y(t^{n+1}) - y(t^n)}{h} = f(t^n, y(t^n)) + \frac{h}{2} y''(\eta_n)$$

ou encore

$$y(t^{n+1}) = y(t^n) + hf(t^n, y(t^n)) + \frac{h^2}{2} y''(\eta_n).$$

On note $y^{[n]}$ une approximation de $y(t^n)$ pour $n \in [\![0, N]\!]$.
La méthode d'**Euler progressive** est donnée par le schéma

$$\left\{ \begin{array}{lcl} y^{[n+1]} & = & y^{[n]} + hf(t^n, y^{[n]}), \ \forall n \in [\![0, N-1]\!] \\ y^{[0]} & = & y(t^0) \end{array} \right. \tag{3.13}$$

Ce schéma est **explicite**, car il permet le calcul direct de $y^{[n+1]}$ en fonction de $y^{[n]}$.
De la même manière, la méthode d'**Euler régressive** est donnée par le schéma

$$\left\{ \begin{array}{lcl} y^{[n+1]} & = & y^{[n]} + hf(t^{n+1}, y^{[n+1]}), \ \forall n \in [\![0, N-1]\!] \\ y^{[0]} & = & y(t^0) \end{array} \right. \tag{3.14}$$

Ce schéma est **implicite**, car $y^{[n+1]}$ est définit implicitement en fonction de $y^{[n]}$. Il faut donc résoudre à chaque pas de temps une équation non-linéaire en utilisant des méthodes de point fixe par exemple.

Exercice 3.3.1

On veut résoudre numériquement le problème $(\mathcal{P})$ suivant : trouver y telle que

$$(\mathcal{P}) \quad \left\{ \begin{array}{lcl} y'(t) & = & \cos(t) + 1, \ \forall t \in [0, 4\pi] \\ y(0) & = & 0. \end{array} \right.$$

dont la solution exacte est $y(t) = \sin(t) + t$.
On rappelle le schéma d'Euler progressif pour la résolution d'un problème de Cauchy

$$(\mathcal{S}) \quad \left\{ \begin{array}{lcl} y^{[n+1]} & = & y^{[n]} + hf(t^n, y^{[n]}), \\ y^{[0]} & & \text{donné.} \end{array} \right.$$

avec $(t^n)_{n0}^N$ discrétisation régulière de l'intervalle $[0, 4\pi]$ avec N pas de discrétisation.

Q. 1 *Expliquer en détail comment utiliser le schéma d'Euler progressif pour résoudre le problème $(\mathcal{P})$ en précisant entre autres les données, les inconnues, les dimensions des variables, lien entre $y^{[n+1]}$ et la fonction y, ...*

Q. 2 *Soit a, b, $a < b$ deux réels. Ecrire une fonction* DisReg *retournant une discrétisation régulière de l'intervalle $[a; b]$ avec N pas de discrétisation.*

Q. 3 *Ecrire une fonction* redEUPsca *retournant l'ensemble des couples* $(t^n, y^{[n+1]})_{n=0}^{N}$ *calculés par le schéma d'Euler progressif.*

Q. 4 *Ecrire un algorithme complet de résolution de* $(\mathcal{P})$ *par le schéma d'Euler progressif.*

Correction 3.3.1

R. 1

On commence par écrire le problème de cauchy associé à $(\mathcal{P})$:

$$(\mathcal{PC}) \quad \left\{ \begin{array}{rcl} y'(t) & = & f(t, y(t)), \forall t \in [t^0, t^0 + T] \\ y(t^0) & = & y_0 \in \mathbb{R}. \end{array} \right.$$

avec $t^0 = 0$, $T = 4\pi$, $y_0 = 0$ et

$$\begin{array}{rccc} f & : & [t^0, t^0 + T] \times \mathbb{R} & \longrightarrow & \mathbb{R} \\ & & (t, z) & \longmapsto & \cos(t) + 1 \end{array}.$$

Les données du problème de Cauchy sont donc les réels t^0, T, y_0 et la fonction f. L'inconnue est la fonction $y : [t^0, t^0 + T] \longrightarrow \mathbb{R}$.
Pour résoudre numériquement le problème de Cauchy, on utilise le schéma $(\mathcal{S})$ où les données sont celles du problème de Cauchy plus le nombre de discrétisations $N \in \mathbb{N}^*$ On peux alors calculer

- t^n, $n \in [\![0, N]\!]$ qui sont les points de la discrétisation régulière à N intervalles :

$$t^n = t^0 + nh, \ \forall n \in [\![0, N]\!], \quad \text{avec } h = \frac{T}{N}.$$

- $y^{[n]}$, $n \in [\![0, N]\!]$ déterminés par le schéma $(\mathcal{S})$. On a $y^{[0]} = y^0$, puis on calcule

$$y^{[n+1]} = y^{[n]} + hf(t^n, y^{[n]}), \quad \text{pour } \ n = 0 \text{ à } N - 1$$

R. 2

Une discrétisation régulière de l'intervalle $[a, b]$ avec N pas (constant) de discrétisation est donnée par

$$t^n = a + nh, \ \forall n \in [\![0, N]\!], \quad \text{avec } h = \frac{b - a}{N}.$$

Algorithme 3.1 Fonction DisReg retournant une discrétisation régulière de l'intervalle $[a, b]$

Données : a, b : deux réels, $a < b$
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$

```
1: Fonction t ← DisReg( a, b, N )
2:   h ← (b − a)/N
3:   Pour n ← 0 à N faire
4:     t(n + 1) ← a + n ∗ h
5:   Fin
6: Fin
```

R. 3

Les données du problème de Cauchy sont

Données : f : $f : [t^0, t^0 + T] \times \mathbb{R} \longrightarrow \mathbb{R}$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
y^0 : réel, donnée initiale

auxquels, il faut ajouter le paramètre de discrétisation N

Données : N : un entier non nul (nombre de pas de discrétisation).

On choisit de retourner les couples $(t^n, y^{[n]})$, $n \in [\![0, N]\!]$ sous la forme de deux vecteurs :

Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{Y}(n) = y^{[n-1]}$, $\forall n \in [\![1, N+1]\!]$

On représente en mémoire les vecteurs $\boldsymbol{t}$ et $\boldsymbol{Y}$ en Figure 3.7.

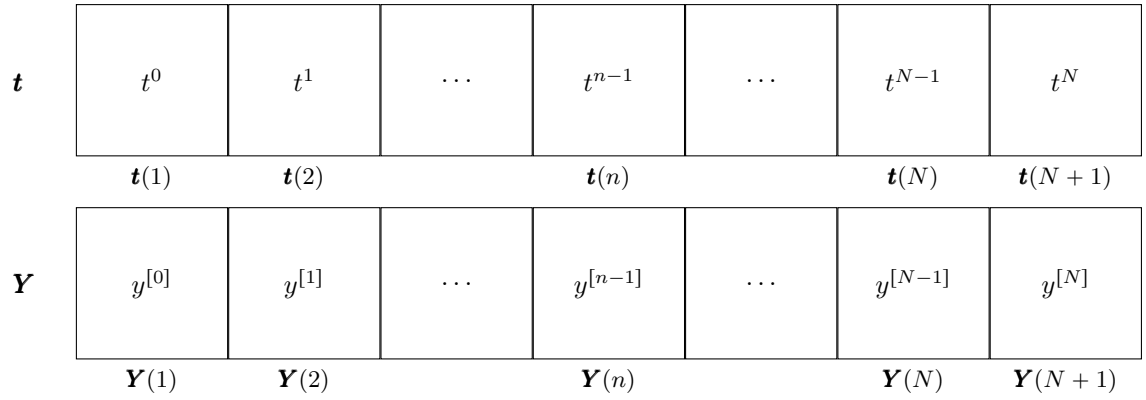


Figure 3.7: Représentation mémoire du vecteur $\boldsymbol{t} \in \mathbb{R}^{N+1}$ et du vecteur $\boldsymbol{Y} \in \mathbb{R}^{N+1}$.

Le schéma théorique est donné par

$$\begin{cases} y^{[0]} = y^0 \\ y^{[n+1]} = y^{[n]} + hf(t^n, y^{[n]}), \quad \text{pour} \quad n = 0 \text{ à } N-1. \end{cases}$$

Le schéma algorithmique s'écrit alors

$$\begin{cases} \boldsymbol{Y}(1) = y^0 & \text{(R1.15)} \\ \boldsymbol{Y}(n+2) = \boldsymbol{Y}(n+1) + hf(\boldsymbol{t}(n+1), \boldsymbol{Y}(n+1)), \quad \text{pour} \quad n = 0 \text{ à } N-1. & \text{(R1.16)} \end{cases}$$

ou de manière équivalente

$$\begin{cases} \boldsymbol{Y}(1) = y^0 & \text{(R1.17)} \\ \boldsymbol{Y}(n+1) = \boldsymbol{Y}(n) + hf(\boldsymbol{t}(n), \boldsymbol{Y}(n)), \quad \text{pour} \quad n = 1 \text{ à } N. & \text{(R1.18)} \end{cases}$$

L'algorithme de la fonction redEUPsca est :

Algorithme 3.2 Fonction redEUPsca : résolution d'un problème de Cauchy **scalaire** par le schéma d'Euler progressif

Données : f : $f : [t^0, t^0 + T] \times \mathbb{R} \longrightarrow \mathbb{R}$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
y^0 : réel, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{Y}(n) = y^{[n-1]}$, $\forall n \in [\![1, N+1]\!]$

```
1: Fonction [t,Y] ← redEUPsca( f, t^0, T, y^0, N )
2:    t ← DisReg(t^0, t^0 + T, N)
3:    h ← T/N
4:    Y(1) ← y^0
5:    Pour n ← 1 à N faire
6:       Y(n+1) ← Y(n) + h * f(t(n), Y(n))
7:    Fin
8: Fin
```

R. 4

Il faut tout d'abord écrire la fonction fCauchy correspondant à la fonction f:

Algorithme 3.3 Fonction fCauchy : fonction f du problème de Cauchy associé à $(\mathcal{P})$

Données : t : un réel
z : un réel
Résultat : w : un réel

1: **Fonction** $w \leftarrow$ fCauchy(t, y)
2: $w \leftarrow \cos(t) + 1$
3: **Fin**

L'algorithme de résolution est :

Algorithme 3.4 Résolution numérique du problème $(\mathcal{P})$

1: $t^0 \leftarrow 0$
2: $T \leftarrow 4\pi$
3: $y^0 \leftarrow 0$
4: $[\boldsymbol{t}, \boldsymbol{Y}] \leftarrow$ redEUPsca(fCauchy, $t^0, T, y^0, 500$)

Fin correction

Exemple

Soit l'E.D.O. suivante

$$\begin{cases} y'(t) &= y(t) + t^2 y^2(t), \quad \text{pour} \quad t \in [0,5], \\ y(0) &= -1 \end{cases}$$

de solution exacte

$$y(t) = 1/(e^{-t} - t^2 + 2t - 2).$$

Les résolutions numériques avec les schémas d'Euler progressif et rétrograde sont représentés en figure 3.8.

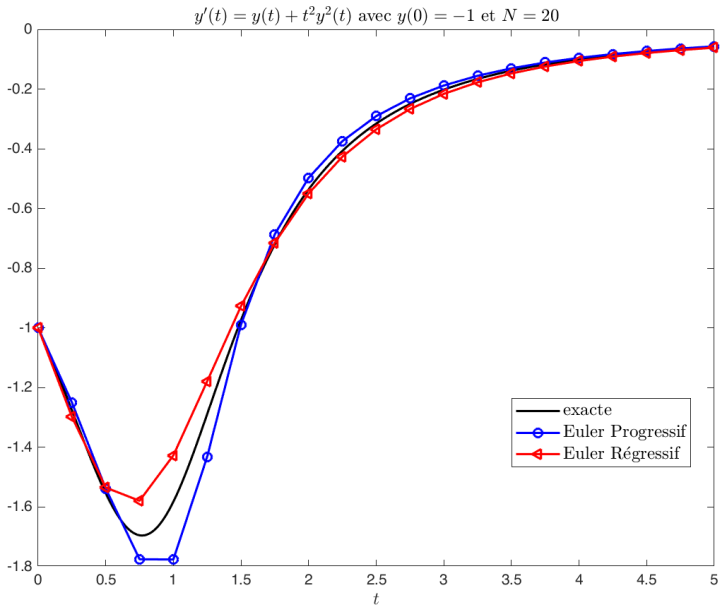


Figure 3.8: résolutions avec Euler progressif et rétrograde

3.3.2 Différences finies pour le problème de Cauchy en dimension m

On veut résoudre le problème de Cauchy :

$$(\mathcal{PC}) \quad \left\{ \begin{array}{rcl} \boldsymbol{y}'(t) & = & \boldsymbol{f}(t,\boldsymbol{y}(t)), \quad \forall t \in [t^0, t^0+T] \\ \boldsymbol{y}(t^0) & = & \boldsymbol{y}_0 \in \mathbb{R}^m. \end{array} \right.$$

La fonction $\boldsymbol{f}$ étant définie par

$$\begin{array}{rcccl} \boldsymbol{f} & : & [t^0, t^0+T] \times \mathbb{R}^m & \longrightarrow & \mathbb{R}^m \\ & & (t, \boldsymbol{z}) & \longmapsto & \boldsymbol{w} = \boldsymbol{f}(t,\boldsymbol{z}) = \begin{pmatrix} \boldsymbol{f}_1(t,\boldsymbol{z}) \\ \vdots \\ \boldsymbol{f}_d(t,\boldsymbol{z}) \end{pmatrix} \end{array}$$

En notant

$$\boldsymbol{y}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_m(t) \end{pmatrix} \in \mathbb{R}^m \quad \text{et} \quad \boldsymbol{f}(t,\boldsymbol{y}(t)) = \begin{pmatrix} f_1(t,\boldsymbol{y}(t)) \\ \vdots \\ f_m(t,\boldsymbol{y}(t)) \end{pmatrix} \in \mathbb{R}^m$$

le problème de Cauchy peut aussi s'écrire sous la forme

$$\left\{ \begin{array}{rcl} y_1'(t) & = & f_1(t,\boldsymbol{y}(t)), \\ \vdots & & \\ y_d'(t) & = & f_d(t,\boldsymbol{y}(t)), \quad \forall t \in [t^0, t^0+T] \\ \text{avec} & & \boldsymbol{y}(t^0) = \boldsymbol{y}_0 \in \mathbb{R}^m. \end{array} \right.$$

Après discrétisation et utilisation de la formule des différences finies progressive on obtient

$$\left\{ \begin{array}{rcl} y_1^{[n+1]} & = & y_1^{[n]} + h f_1(t^n, \boldsymbol{y}^{[n]}) \\ \vdots & & \\ y_d^{[n+1]} & = & y_d^{[n]} + h f_d(t^n, \boldsymbol{y}^{[n]}) \\ \text{avec} & & \boldsymbol{y}(t^0) = \boldsymbol{y}_0 \in \mathbb{R}^d. \end{array} \right.$$

où $\boldsymbol{y}^{[n]} = \begin{pmatrix} y_1^{[n]} \\ \vdots \\ y_d^{[n]} \end{pmatrix}$ et $\boldsymbol{y}^{[n]}) \approx \boldsymbol{y}(t^n)$.

On obtient alors la méthode d'**Euler progressive** sous forme vectorielle :

$$\left\{ \begin{array}{rcl} \boldsymbol{y}^{[n+1]} & = & \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}), \ \forall n \in [\![0, N-1]\!] \\ \boldsymbol{y}^{[0]} & = & \boldsymbol{y}_0 \end{array} \right. \tag{3.15}$$

Ce schéma est **explicite**, car il permet le calcul direct de $\boldsymbol{y}^{[n+1]}$ en fonction de $\boldsymbol{y}^{[n]}$.
De la même manière, la méthode d'**Euler régressive** sous forme vectorielle est donnée par le schéma

$$\left\{ \begin{array}{rcl} \boldsymbol{y}^{[n+1]} & = & \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^{n+1}, \boldsymbol{y}^{[n+1]}), \ \forall n \in [\![0, N-1]\!] \\ \boldsymbol{y}^{[0]} & = & \boldsymbol{y}_0 \end{array} \right. \tag{3.16}$$

Ce schéma est **implicite**, car $\boldsymbol{y}^{[n+1]}$ est définit implicitement en fonction de $\boldsymbol{y}^{[n+1]}$. On peut noter que cette équation implicite s'écrit aussi

$$\boldsymbol{y}^{[n+1]} = \Phi(\boldsymbol{y}^{[n+1]}), \quad \text{avec } \Phi(\boldsymbol{z}) \stackrel{\text{def}}{=} \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^{n+1}, \boldsymbol{z})$$

que l'on peut résoudre par une méthode de point fixe en prenant comme point initial $\boldsymbol{y}^{[n]}$.

Exercice 3.3.2

Soit le problème de Cauchy **vectoriel**

$$\left\{ \begin{array}{rcl} \boldsymbol{y}'(t) & = & \boldsymbol{f}(t,\boldsymbol{y}(t)), \quad \forall t \in [t^0, t^0+T], \\ \boldsymbol{y}(t^0) & = & \boldsymbol{y}_0 \in \mathbb{R}^m, \end{array} \right.$$

avec $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$.
On rappelle, entre autres, l'écriture simplifiée d'accès aux colonnes d'une matrice (voir ...)

Algorithmique		
fonction	*version simplifiée*	*Description mathématique*
$\boldsymbol{u} \leftarrow$ getMatCol$(\mathbb{A}, j)$	$\boldsymbol{u} \leftarrow \mathbb{A}(:, j)$	$\boldsymbol{u} \in \mathbb{R}^m$ est déterminé par $\boldsymbol{u}_i = \mathbb{A}_{i,j}$, $\forall i \in [\![1, n]\!]$
$\mathbb{A} \leftarrow$ setMatCol$(\mathbb{A}, \boldsymbol{u}, j)$	$\mathbb{A}(:, j) \leftarrow \boldsymbol{u}$	la colonne j de $\mathbb{A}$ est remplacée par $\boldsymbol{u} \in \mathbb{R}^m$ et on a $\mathbb{A}_{i,j} = \boldsymbol{u}_i$, $\forall i \in [\![1, n]\!]$.
$\mathbb{A} \leftarrow$ MatZeros(m, n)	$\mathbb{A} \leftarrow \mathbb{O}_{m,n}$	$\mathbb{A}$ est la matrice nulle de $\mathcal{M}_{m,n}(\mathbb{R})$.
$\boldsymbol{W} \leftarrow$ aUpbV$(a, \boldsymbol{U}, b, \boldsymbol{V})$	$\boldsymbol{W} \leftarrow a * \boldsymbol{U} + b * \boldsymbol{V}$	$(a, b) \in \mathbb{R}^2$ et $\boldsymbol{U}$, $\boldsymbol{V}$ deux vecteurs de $\mathbb{R}^n$ $\boldsymbol{W} = a\boldsymbol{U} + b\boldsymbol{V}$, $\boldsymbol{W} \in \mathbb{R}^n$.

Table 3.1: Ecriture algorithmique avec quelques fonctions usuelles et leur version simplifiée

On souhaite écrire une fonction algorithmique redEUPVec permettant de résoudre ce problème de Cauchy (vectoriel) par le schéma **vectoriel** explicite d'Euler progressif

$$\begin{cases} \boldsymbol{y}^{[n+1]} &= \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}), \ \forall n \in [\![0, N-1]\!] \\ \boldsymbol{y}^{[0]} &= \boldsymbol{y}_0 \end{cases}$$

avec $(t^n)_{n=0}^N$ la discrétisation régulière de $[t^0, t^0+T]$ avec N pas de discrétisation et $\boldsymbol{y}^{[n]} = \begin{pmatrix} y_1^{[n]} \\ \vdots \\ y_m^{[n]} \end{pmatrix}$ et $\boldsymbol{y}^{[n]}) \approx \boldsymbol{y}(t^n)$. Cette fonction devra retourner l'ensemble des t^n et des $\boldsymbol{y}^{[n]}$ pour $n[\![0, N]\!]$.

Q. 1

a. *Rappeler précisement les données du problème de Cauchy* ***vectoriel****.*

b. *Quelles sont les données de la fonction algorithmique* redEUPVec *en précisant le type et la dimension pour chacune?*

c. *Quelles sont les sorties/résultats de la fonction algorithmique* redEUPVec *en précisant le type et la dimension pour chacun?*

Q. 2

Ecrire la fonction algorithmique redEUPVec *permettant de résoudre ce problème de Cauchy (vectoriel) par le schéma explicite d'Euler progressif. On utilisera l'écriture algorithmique simplifiée d'accès aux éléments d'une matrice (voir Table* **??***).*

Q. 3

Ecrire la fonction algorithmique redEUPVecfun *permettant de résoudre ce problème de Cauchy (vectoriel) par le schéma explicite d'Euler progressif. On utilisera l'écriture algorithmique avec fonctions pour l'accès aux éléments d'une matrice (voir Table* **??***).*

Correction

R. 1

a. Les données du problème de Cauchy **vectoriel** sont

- $t^0 \in \mathbb{R}$, le temps initial,
- $T \in \mathbb{R}^{+*}$, la periode de temps,
- $\boldsymbol{y}^{[0]} \in \mathbb{R}^m$, la donnée initiale,
- $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$, la fonction de Cauchy.

b. Les données de la fonction algorithmique redEUPVec sont celles du problème de Cauchy **vectoriel** auxquels, il faut ajouter le paramètre de discrétisation N

- N, un entier non nul (nombre de pas de discrétisation).

c. On choisit de retourner $(t^n)_{n=0}^N$ sous la forme d'un vecteur $\boldsymbol{t} \in \mathbb{R}^{N+1}$ et $(\boldsymbol{y}^{[n]})_{n=0}^N$ $n \in [\![0, N]\!]$ sous la forme d'une matrice de $\mathbb{Y} \in \mathcal{M}_{m,N+1}(\mathbb{R})$ (ou tableau à m lignes et $N+1$ colonnes). Plus précisement, on a

Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\mathbb{Y}$: matrice de $\mathcal{M}_{m,N+1}(\mathbb{R})$, $\mathbb{Y}(i,n) = y_i^{[n-1]}$, $\forall (i,n) \in [\![1,m]\!] \times [\![1, N+1]\!]$.

On représente *en mémoire* le vecteur $\boldsymbol{t}$ et la matrice $\mathbb{Y}$ respectivement en Figure 3.9 et 3.10.

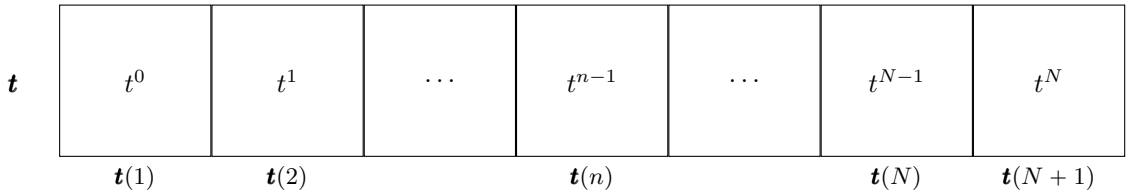


Figure 3.9: Représentation mémoire du vecteur $\boldsymbol{t} \in \mathbb{R}^{N+1}$.

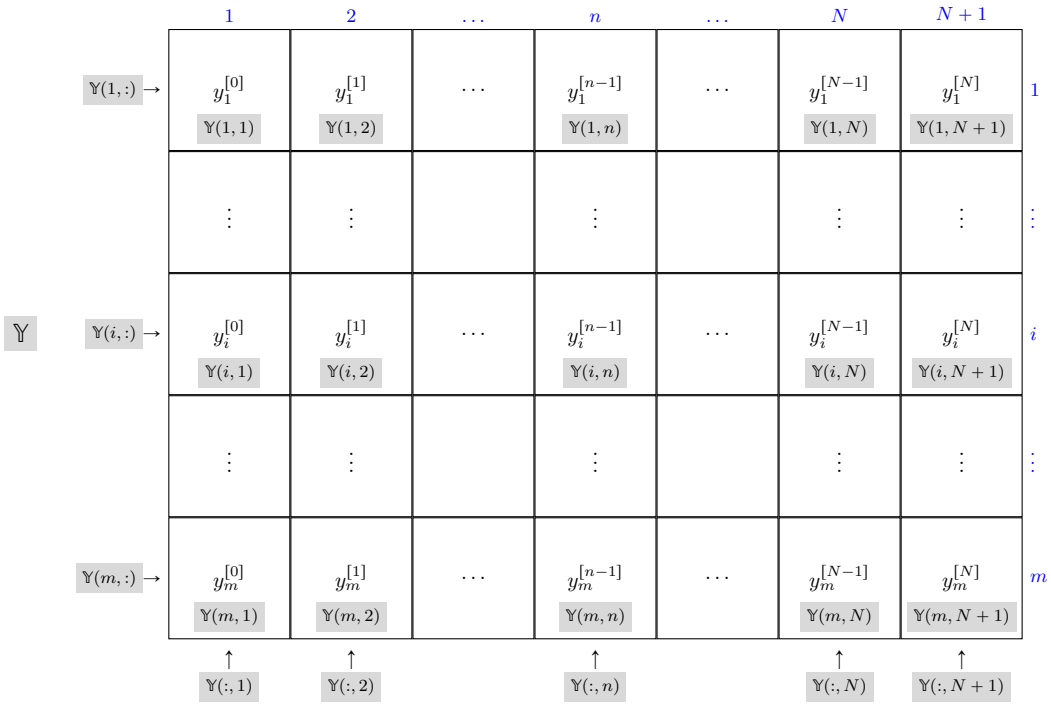


Figure 3.10: Représentation mémoire de $\mathbb{Y} \in \mathcal{M}_{m,N+1}(\mathbb{R})$.

Une autre façon de représenter *vectoriellement* la matrice $\mathbb{Y} \in \mathcal{M}_{m,N+1}(\mathbb{R})$.

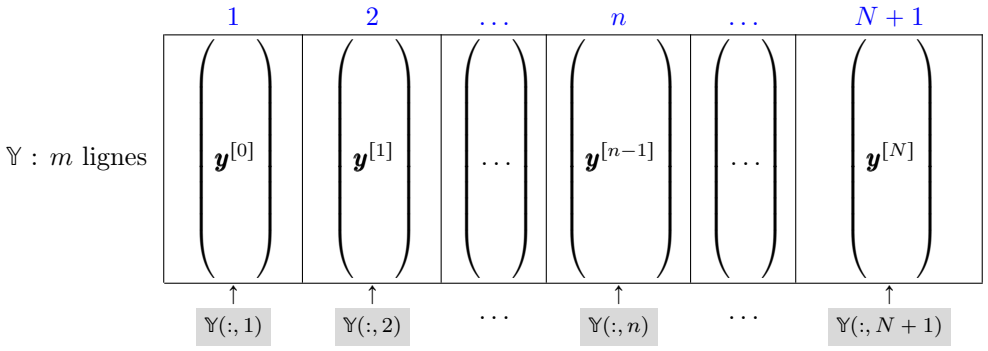


Table 3.2: Représentation mémoire de $\mathbb{Y} \in \mathcal{M}_{m,N+1}(\mathbb{R})$ à l'aide des vecteurs colonnes.

R. 2

Voici la description de la fonction redEUPVec (sans son implémentation)

Algorithme 3.5 Fonction redEUPVec : résolution d'un problème de Cauchy **vectoriel** par le schéma d'Euler progressif (entête de la fonction)

Données :	$\boldsymbol{f}$	:	$\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$ fonction d'un problème de Cauchy (vectoriel)
	t^0	:	réel, temps initial
	T	:	réel > 0
	$\boldsymbol{y}^0$	:	$\mathbb{R}^m$, donnée initiale
	N	:	un entier non nul (nombre de pas de discrétisation).
Résultat :	$\boldsymbol{t}$	:	vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1},\ \forall n \in [\![1, N+1]\!]$
	$\mathbb{Y}$	:	matrice de $\mathcal{M}_{m,N+1}(\mathbb{R})$, $\mathbb{Y}(i,n) = y_i^{[n-1]},\ \forall (i,n) \in [\![1,m]\!] \times [\![1, N+1]\!]$.

```
1: Fonction [t, Y] ← redEUPVec( f, t^0, T, y^0, N )
2:     ...
3: Fin
```

Nous allons maintenant utiliser une méthode de *raffinement* pour obtenir au final l'algorithme permettant de *remplir* le vecteur $\boldsymbol{t}$ et la matrice $\mathbb{Y}$ en utilisant le langage algorithmique simplifié.

Algorithme 3.5 $\mathcal{R}_0$

```
1: Calculer t
2: Calculer Y
```

Algorithme 3.5 $\mathcal{R}_1$

```
1: t ← DisReg(t^0, t^0 + T, N)
2: Mettre y^0 en colonne 1 de Y
3: Pour n ← 0 à N − 1 faire
4:     Calculer y^[n+1] par formule et
       le mettre en colonne (n + 2) de Y.
5: Fin
```

Algorithme 3.5 $\mathcal{R}_1$

1: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
2: Mettre $\boldsymbol{y}^0$ en colonne 1 de $\mathbb{Y}$
3: **Pour** $n \leftarrow 0$ **à** $N-1$ **faire**
Calculer $\boldsymbol{y}^{[n+1]}$ par formule et le mettre en colonne $(n+2)$ de $\mathbb{Y}$.
4:
5: **Fin**

Algorithme 3.5 $\mathcal{R}_2$

1: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
2: $\mathbb{Y}(:,1) \leftarrow \boldsymbol{y}^0$
3: $h \leftarrow T/N$ ▷ Nécessaire dans formule
4: **Pour** $n \leftarrow 0$ **à** $N-1$ **faire**
5: $\boldsymbol{Ytmp} \leftarrow \boldsymbol{y}^{[n]} + hf(t^n, \boldsymbol{y}^{[n]})$
6: $\mathbb{Y}(:,n+2) \leftarrow \boldsymbol{Ytmp}$
7: **Fin**

Or $\boldsymbol{t}(n)$ est stocké en $\boldsymbol{t}(n+1)$ et $\boldsymbol{y}^{[n]}$ est stocké en colonne $n+1$ de la matrice $\mathbb{Y}$, c'est à dire en $\mathbb{Y}(:,n)$. On obtient alors

Algorithme 3.5 $\mathcal{R}_2$

1: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
2: $\mathbb{Y}(:,1) \leftarrow \boldsymbol{y}^0$
3: $h \leftarrow T/N$ ▷ Nécessaire dans formule
4: **Pour** $n \leftarrow 0$ **à** $N-1$ **faire**
5: $\boldsymbol{Ytmp} \leftarrow \boldsymbol{y}^{[n]} + hf(t^n, \boldsymbol{y}^{[n]})$
6: $\mathbb{Y}(:,n+2) \leftarrow \boldsymbol{Ytmp}$
7: **Fin**

Algorithme 3.5 $\mathcal{R}_3$

1: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
2: $\mathbb{Y}(:,1) \leftarrow \boldsymbol{y}^0$
3: $h \leftarrow T/N$
4: **Pour** $n \leftarrow 0$ **à** $N-1$ **faire**
5: $\boldsymbol{Ytmp} \leftarrow \mathbb{Y}(:,n+1) + h * f(\boldsymbol{t}(n+1), \mathbb{Y}(:,n+1))$
6: $\mathbb{Y}(:,n+2) \leftarrow \boldsymbol{Ytmp}$
7: **Fin**

Dans ce dernier raffinement, purement algorithmique (les précédents raffinements contenaient encore des notations mathématiques), on peut changer la boucle pour alléger l'écriture:

Algorithme 3.5 Fonction redEUPVec : résolution d'un problème de Cauchy **vectoriel** par le schéma d'Euler progressif (langage algorithmique simplifié)

Données : $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$ fonction d'un problème de Cauchy (vectoriel)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: $\mathbb{R}^m$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\mathbb{Y}$: matrice de $\mathcal{M}_{m,N+1}(\mathbb{R})$, $\mathbb{Y}(i,n) = y_i^{[n-1]}$, $\forall (i,n) \in [\![1,m]\!] \times [\![1,N+1]\!]$.

1: **Fonction** $[\boldsymbol{t}, \mathbb{Y}] \leftarrow$ redEUPVec$(\ f, t^0, T, \boldsymbol{y}^0, N\ )$
2: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
3: $\mathbb{Y}(:,1) \leftarrow \boldsymbol{y}^0$
4: $h \leftarrow T/N$
5: **Pour** $n \leftarrow 1$ **à** N **faire** ▷ Décalage d'indice
6: $\boldsymbol{Ytmp} \leftarrow \mathbb{Y}(:,n) + h * f(\boldsymbol{t}(n), \mathbb{Y}(:,n))$
7: $\mathbb{Y}(:,n+1) \leftarrow \boldsymbol{Ytmp}$
8: **Fin**
9: **Fin**

Bien sûr les lignes 6 et 7 de cet algorithme peuvent être condensée en

$$\mathbb{Y}(:,n+1) \leftarrow \mathbb{Y}(:,n) + h * f(\boldsymbol{t}(n), \mathbb{Y}(:,n)).$$

R. 3

Dans l'Algorithme 3.5, seules les lignes 3, 6 et 7 sont à remplacer par des fonctions effectuant les mêmes opérations. Pour les lignes 3 et 7, on utilisera la fonction setMatCol. La ligne 6 correspond en fait à une combinaison linéaire entre les deux vecteurs de $\mathbb{R}^m$, $\mathbb{Y}(:,n)$ et $f(\boldsymbol{t}(n),\mathbb{Y}(:,n))$

Algorithme 3.6 $\mathcal{R}_3$

1: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
2: $\mathbb{Y}(:,1) \leftarrow \boldsymbol{y}^0$
3: $h \leftarrow T/N$
4: **Pour** $n \leftarrow 1$ **à** N **faire**
5: $\quad \boldsymbol{Ytmp} \leftarrow \mathbb{Y}(:,n) + h * f(\boldsymbol{t}(n), \mathbb{Y}(:,n))$
6: $\quad \mathbb{Y}(:,n+1) \leftarrow \boldsymbol{Ytmp}$
7: **Fin**

Algorithme 3.6 $\mathcal{R}_4$

1: $\boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
2: $\mathbb{Y} \leftarrow \text{MatZeros}(m, N+1)$ ▷ ou implicite
3: $\mathbb{Y} \leftarrow \text{setMatCol}(\mathbb{Y}, \boldsymbol{y}^0, 1)$
4: $h \leftarrow T/N$
5: **Pour** $n \leftarrow 1$ **à** N **faire**
6: $\quad \boldsymbol{Ytmp} \leftarrow \text{getMatCol}(\mathbb{Y}, n)$
7: $\quad \boldsymbol{Ytmp} \leftarrow \text{aUpbV}\big(1, \boldsymbol{Ytmp}, h, f(\boldsymbol{t}(n), \boldsymbol{Ytmp})\big)$
8: $\quad \mathbb{Y} \leftarrow \text{setMatCol}(\mathbb{Y}, \boldsymbol{Ytmp}, n+1)$
9: **Fin**

Algorithme 3.6 Fonction redEUPVecfun : résolution d'un problème de Cauchy **vectoriel** par le schéma d'Euler progressif (langage algorithmique non simplifié)

Données :
- $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$ fonction d'un problème de Cauchy (vectoriel)
- t^0 : réel, temps initial
- T : réel > 0
- $\boldsymbol{y}^0$: $\mathbb{R}^m$, donnée initiale
- N : un entier non nul (nombre de pas de discrétisation).

Résultat :
- $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
- $\mathbb{Y}$: matrice de $\mathcal{M}_{m,N+1}(\mathbb{R})$, $\mathbb{Y}(i,n) = y_i^{[n-1]}$, $\forall (i,n) \in [\![1,m]\!] \times [\![1, N+1]\!]$.

1: **Fonction** $[\boldsymbol{t}, \mathbb{Y}] \leftarrow$ redEUPVecfun$(\ f, t^0, T, \boldsymbol{y}^0, N\ )$
2: $\quad \boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
3: $\quad \mathbb{Y} \leftarrow \text{MatZeros}(m, N+1)$
4: $\quad \mathbb{Y} \leftarrow \text{setMatCol}(\mathbb{Y}, \boldsymbol{y}^0, 1)$
5: $\quad h \leftarrow T/N$
6: $\quad$ **Pour** $n \leftarrow 1$ **à** N **faire**
7: $\quad\quad \boldsymbol{Ytmp} \leftarrow \text{getMatCol}(\mathbb{Y}, n)$
8: $\quad\quad \boldsymbol{Ytmp} \leftarrow \text{aUpbV}\big(1, \boldsymbol{Ytmp}, h, f(\boldsymbol{t}(n), \boldsymbol{Ytmp})\big)$
9: $\quad\quad \mathbb{Y} \leftarrow \text{setMatCol}(\mathbb{Y}, \boldsymbol{Ytmp}, n+1)$
10: $\quad$ **Fin**
11: **Fin**

Une autre possibilité d'écriture est:

1: **Fonction** $[\boldsymbol{t}, \mathbb{Y}] \leftarrow$ redEUPVecfunv1$(\ f, t^0, T, \boldsymbol{y}^0, N\ )$
2: $\quad \boldsymbol{t} \leftarrow \text{DisReg}(t^0, t^0+T, N)$
3: $\quad \mathbb{Y} \leftarrow \text{MatZeros}(m, N+1)$
4: $\quad \mathbb{Y} \leftarrow \text{setMatCol}(\mathbb{Y}, \boldsymbol{y}^0, 1)$
5: $\quad h \leftarrow T/N$
6: $\quad \boldsymbol{Ytmp} \leftarrow \boldsymbol{y}^0$
7: $\quad$ **Pour** $n \leftarrow 1$ **à** N **faire**
8: $\quad\quad \boldsymbol{Ytmp} \leftarrow \text{aUpbV}\big(1, \boldsymbol{Ytmp}, h, f(\boldsymbol{t}(n), \boldsymbol{Ytmp})\big)$
9: $\quad\quad \mathbb{Y} \leftarrow \text{setMatCol}(\mathbb{Y}, \boldsymbol{Ytmp}, n+1)$
10: $\quad$ **Fin**
11: **Fin**

Fin correction

Exercice 3.3.3

Soit l'équation différentielle ordinaire linéaire du troisième ordre avec conditions initiales données par

$$(1+t+t^2)\,y^{(3)}(t)+(3+6t)\,y^{(2)}(t)+6\,y^{(1)}(t)=6t,\qquad \forall t\in[0,T],$$
$$y(0)=\alpha,\quad y^{(1)}(0)=\beta,\quad y^{(2)}(0)=\gamma.$$

Ici $y^{(k)}$ note la dérivée k-ième de y.
Pour cette EDO, il existe une unique solution donnée par

$$y(t)=\frac{t^4+2\,At^2+4\,Bt+4\,C}{4\,(t^2+t+1)}$$

avec $(A,B,C)\in\mathbb{R}^3$ vérifiant

$$C=\alpha,\quad B-C=\beta\ \text{ et }\ A-2\,B=\gamma.$$

On a aussi

$$y^{(1)}(t) = \frac{t^3+At+B}{t^2+t+1}-\frac{\big(t^4+2\,At^2+4\,Bt+4\,C\big)(2\,t+1)}{4\,(t^2+t+1)^2}$$
$$y^{(2)}(t) = \frac{3\,t^2+A}{t^2+t+1}-\frac{2\,\big(t^3+At+B\big)(2\,t+1)}{(t^2+t+1)^2}+\frac{\big(t^4+2\,At^2+4\,Bt+4\,C\big)(2\,t+1)^2}{2\,(t^2+t+1)^3}-\frac{t^4+2\,At^2+4\,Bt+4\,C}{2\,(t^2+t+1)^2}.$$

Q. 1 *Déterminer le problème de Cauchy vectoriel associé à cette EDO*

Dans la suite, on prendra $T=10$, $\alpha=6$, $\beta=-5$ et $\gamma=-2$.

Q. 2 *Ecrire un programme permettant de résoudre numériquement le problème de Cauchy associé à cette EDO à l'aide de la fonction algorithmique* $[\boldsymbol{t},\mathbb{Y}]\leftarrow$ redEUPvec$(f,t0,T,Y0,N)$ *(voir Exercice précédent).*

On suppose que notre language algorithmique dispose d'une fonction graphique plot(X,Y) reliant par des segments les points successifs

$$\big(X(1),Y(1)\big),\big(X(2),Y(2)\big),\ldots,\big(X(end),Y(end)\big)$$

les tableaux X et Y ayant même longeurs et correspondent respectivement aux tableaux des abscisses et des ordonnées.
On pourra utiliser la version simplifiée du langage algorithmique.

Q. 3 *Donner les commandes permettant, après avoir utilisé le programme algorithmique précédent, de représenter graphiquement les approximations obtenues par le schéma, de*

$$\big(y(t^n)\big)_{n=0}^N,\quad \big(y^{(1)}(t^n)\big)_{n=0}^N\quad et\quad \big(y^{(2)}(t^n)\big)_{n=0}^N$$

Q. 4 *Ecrire un programme algorithmique permettant de représenter graphiquement les solutions exactes aux points de discrétisation, c'est à dire*

$$\big(y(t^n)\big)_{n=0}^N,\quad \big(y^{(1)}(t^n)\big)_{n=0}^N\quad et\quad \big(y^{(2)}(t^n)\big)_{n=0}^N$$

Q. 5 *Ecrire un programme algorithmique permettant de représenter graphiquement les erreurs numériques commises en valeurs absolues par le schéma pour les approximations de*

$$\big(y(t^n)\big)_{n=0}^N,\quad \big(y^{(1)}(t^n)\big)_{n=0}^N\quad et\quad \big(y^{(2)}(t^n)\big)_{n=0}^N$$

Correction

R. 1

C'est une EDO d'ordre 3, nous allons donc les transformer en 3 EDO d'ordre 1 en posant

$$\begin{array}{rccl} \boldsymbol{Y} & : & [0,T] & \longrightarrow & \mathbb{R}^3 \\ & & t & \longmapsto & \boldsymbol{Y}(t) = \begin{pmatrix} \boldsymbol{Y}_1(t) \\ \boldsymbol{Y}_2(t) \\ \boldsymbol{Y}_3(t) \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} y(t) \\ y^{(1)}(t) \\ y^{(2)}(t) \end{pmatrix}. \end{array}$$

On cherche tout d'abord à établir la fonction de Cauchy $\boldsymbol{f} : [0,T] \times \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ associée à l'EDO et vérifiant

$$\forall t \in [0,T], \quad \boldsymbol{Y}'(t) = \boldsymbol{f}(t, \boldsymbol{Y}(t)).$$

Soit $t \in [0,T]$, on a

$$\boldsymbol{Y}'(t) = \begin{pmatrix} \boldsymbol{Y}_1'(t) \\ \boldsymbol{Y}_2'(t) \\ \boldsymbol{Y}_3'(t) \end{pmatrix} = \begin{pmatrix} y^{(1)}(t) \\ y^{(2)}(t) \\ y^{(3)}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{Y}_2(t) \\ \boldsymbol{Y}_3(t) \\ y^{(3)}(t). \end{pmatrix}$$

De plus, comme $1 + t + t^2 \neq 0$, on a

$$y^{(3)}(t) = \frac{6t - (3+6t)\, y^{(2)}(t) - 6\, y^{(1)}(t)}{1 + t + t^2}.$$

On en déduit donc

$$\boldsymbol{Y}(t) = \begin{pmatrix} \boldsymbol{Y}_2(t) \\ \boldsymbol{Y}_3(t) \\ \dfrac{6t - (3+6t)\, y^{(2)}(t) - 6\, y^{(1)}(t)}{1 + t + t^2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{Y}_2(t) \\ \boldsymbol{Y}_3(t) \\ \dfrac{6t - (3+6t)\boldsymbol{Y}_3(t) - 6\boldsymbol{Y}_2(t)}{1 + t + t^2} \end{pmatrix}$$

La fonction de Cauchy s'écrit alors

$$\begin{array}{rccl} \boldsymbol{f} & : & [0,T] \times \mathbb{R}^3 & \longrightarrow & \mathbb{R}^3 \\ & & (t, \boldsymbol{Z}) & \longmapsto & \begin{pmatrix} \boldsymbol{Z}_2 \\ \boldsymbol{Z}_3 \\ \dfrac{6t - (3+6t)\boldsymbol{Z}_3 - 6\boldsymbol{Z}_2}{1 + t + t^2} \end{pmatrix} \end{array}$$

Le problème de Cauchy associé à l'EDO s'écrit alors:

Trouver $\boldsymbol{Y} : [0,T] \longrightarrow \mathbb{R}^3$ telle que

$$\begin{cases} \boldsymbol{Y}'(t) = \boldsymbol{f}(t, \boldsymbol{Y}(t)), & \forall t \in [0,T], \\ \boldsymbol{Y}(0) = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \in \mathbb{R}^3. \end{cases} \tag{$\mathcal{P}_3$}$$

R. 2

On écrit tout d'abord la fonction algorithmique, nommée fC3 par exemple, correspondant à la fonction de Cauchy $\boldsymbol{f}$

Algorithme 3.7 Fonction fCauchy : fonction $\boldsymbol{f}$ du problème de Cauchy

Données : t : réel positif
$\boldsymbol{Z}$: un vecteur de $\mathbb{R}^3$
Résultat : $\boldsymbol{W}$: un vecteur de $\mathbb{R}^3$

1: **Fonction** $\boldsymbol{W} \leftarrow$ fCauchy($t, \boldsymbol{Z}$)
2: $\boldsymbol{W} \leftarrow \begin{pmatrix} \boldsymbol{Z}(2) \\ \boldsymbol{Z}(3) \\ \big(6 * t - (3 + 6 * t) * \boldsymbol{Z}(3) - 6 * \boldsymbol{Z}(2)\big)/(1 + t + t\hat{}2) \end{pmatrix}$
3: **Fin**

Algorithme 3.8 Programme permettant de résoudre numériquement le problème de Cauchy avec $T = 10$, $\alpha = 6$, $\beta = -5$ et $\gamma = -2$.

1: $\boldsymbol{Y0} \leftarrow \begin{pmatrix} 6 \\ -5 \\ -2 \end{pmatrix}$ ▷ Données initiales
2: $[\boldsymbol{t}, \mathbb{Y}] \leftarrow$ redEUPvec(fCauchy, $0, 10, \boldsymbol{Y0}, 1000$)

R. 3

Dans le programme algorithmique précédent, on a utilisé

$$[\boldsymbol{t}, \mathbb{Y}] \leftarrow \text{redEUPvec}(\text{fCauchy}, 0, 10, \boldsymbol{Y0}, 1000)$$

pour résoudre le problème de Cauchy vectoriel $(\mathcal{P}_3)$ obtenu en **Q.1**.
La fonction redEUPvec permet de résoudre numériquement ce problème de Cauchy en utilisant le schéma **vectoriel** explicite d'Euler progressif générique suivant:

$$\begin{cases} \boldsymbol{y}^{[n+1]} &= \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}), \ \forall n \in [\![0, N-1]\!] \\ \boldsymbol{y}^{[0]} &= \boldsymbol{y}_0 \end{cases}$$

avec $(t^n)_{n=0}^N$ la discrétisation régulière de $[t^0, t^0 + T]$ avec N pas de discrétisation et $\boldsymbol{y}^{[n]} = \begin{pmatrix} y_1^{[n]} \\ \vdots \\ y_m^{[n]} \end{pmatrix}$
et $\boldsymbol{y}^{[n]} \approx \boldsymbol{Y}(t^n)$, où $\boldsymbol{Y}$ est la solution exacte du problème de Cauchy $(\mathcal{P}_3)$.
Dans le programme algorithmique, on a $m = 3$ et on a choisi $N = 1000$. On a donc pour tout n dans $[\![0, N]\!]$:

$$\boldsymbol{y}^{[n]} = \begin{pmatrix} \boldsymbol{y}_1^{[n]} \\ \boldsymbol{y}_2^{[n]} \\ \boldsymbol{y}_3^{[n]} \end{pmatrix} \approx \boldsymbol{Y}(t^n) = \begin{pmatrix} \boldsymbol{Y}_1(t^n) \\ \boldsymbol{Y}_2(t^n) \\ \boldsymbol{Y}_3(t^n) \end{pmatrix}.$$

De plus, par définition de $\boldsymbol{Y}$ (voir **Q.1**), on a

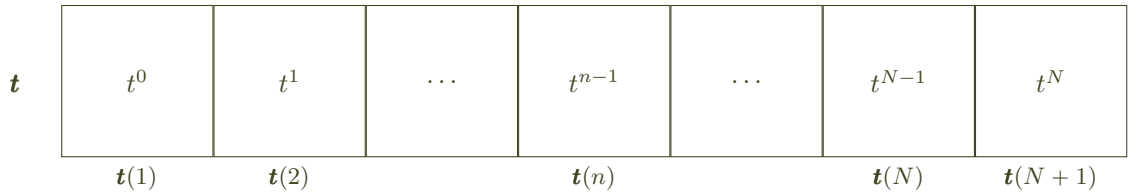
$$\begin{pmatrix} \boldsymbol{Y}_1(t^n) \\ \boldsymbol{Y}_2(t^n) \\ \boldsymbol{Y}_3(t^n) \end{pmatrix} \stackrel{\text{def}}{=} \begin{pmatrix} y(t^n) \\ y^{(1)}(t^n) \\ y^{(2)}(t^n) \end{pmatrix}.$$

où y est la solution de l'EDO initiale du 3ème ordre. On obtient donc

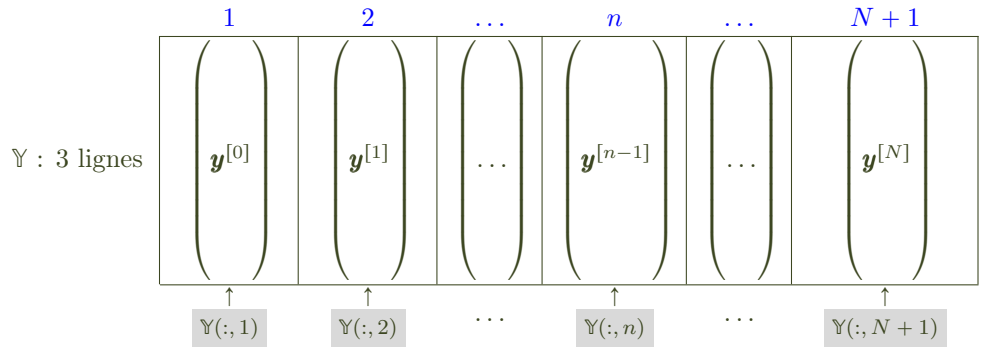
$$\boldsymbol{y}^{[n]} = \begin{pmatrix} \boldsymbol{y}_1^{[n]} \\ \boldsymbol{y}_2^{[n]} \\ \boldsymbol{y}_3^{[n]} \end{pmatrix} \approx \begin{pmatrix} y(t^n) \\ y^{(1)}(t^n) \\ y^{(2)}(t^n) \end{pmatrix}.$$

La fonction redEUPvec retourne donc

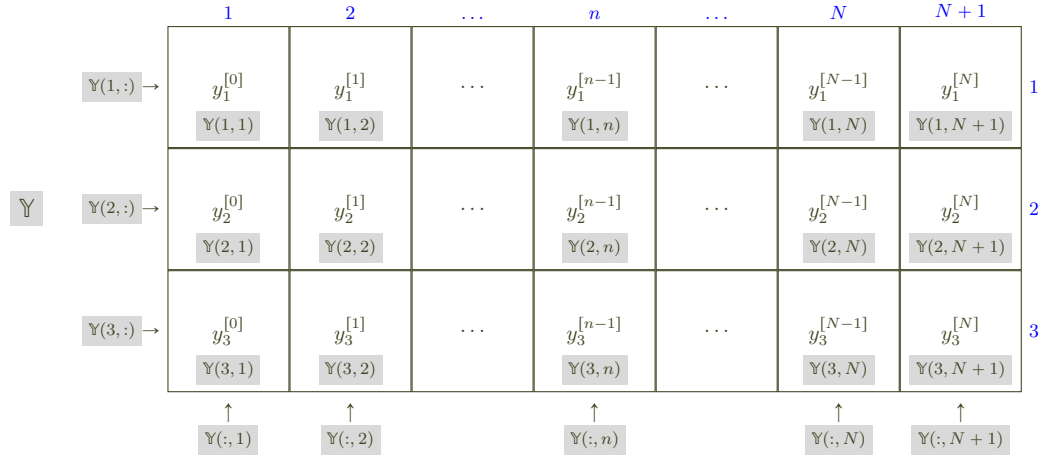
- $\boldsymbol{t} \in \mathbb{R}^{N+1}$ contenant l'ensemble de la discrétisation régulière de l'intervalle $[0, 10]$:



- $\mathbb{Y} \in \mathcal{M}_{3,N+1}(\mathbb{R})$ contenant l'ensemble des $\boldsymbol{y}^{[n]}$, $n \in [\![0, N]\!]$:



Voici une autre façon de représenter $\mathbb{Y}$ sous la forme d'un tableau de 3 lignes et $(N+1)$ colonnes:



Tout le blabla précédent a été rédigé pour aider à la compréhension, mais, avec un peu d'habitude, on a immédiatement

- les approximations de $\big(y(t^n)\big)_{n=0}^N$ correspondent à la première ligne de $\mathbb{Y}$ avec
$$\mathbb{Y}(1, n+1) \approx y(t^n), \quad \forall n \in [\![0, n]\!]$$
et la commande (utilisant le langage algorithmique simplifié) qui permet de représenter ces approximations est:
$$\text{plot}(\boldsymbol{t}, \mathbb{Y}(1,:))$$
- les approximations de $\big(y^{(1)}(t^n)\big)_{n=0}^N$ correspondent à la deuxième ligne de $\mathbb{Y}$ avec
$$\mathbb{Y}(2, n+1) \approx y^{(1)}(t^n), \quad \forall n \in [\![0, n]\!]$$
et la commande (utilisant le langage algorithmique simplifié) qui permet de représenter ces approximations est:
$$\text{plot}(\boldsymbol{t}, \mathbb{Y}(2,:))$$
- les approximations de $\big(y^{(2)}(t^n)\big)_{n=0}^N$ correspondent à la troisième ligne de $\mathbb{Y}$ avec
$$\mathbb{Y}(3, n+1) \approx y^{(2)}(t^n), \quad \forall n \in [\![0, n]\!]$$
et la commande (utilisant le langage algorithmique simplifié) qui permet de représenter ces approximations est:
$$\text{plot}(\boldsymbol{t}, \mathbb{Y}(3,:))$$

R. 4

Il faut tout d'abord écrire les fonctions algoritmiques correspondant à la solution exacte $y(t)$, sa dérivée $y'(t)$ et sa dérivée seconde $y'(t)$ que l'on nommera respectivement yex, dyex, et d2yex.

Algorithme 3.9 Fonction yex : retourne la solution exacte en t

Données : t : un réel,
A, B, C : trois réels
Résultat : w : un réel tel que $w = y(t)$

1: **Fonction** $w \leftarrow$ yex(t, A, B, C)
2: $w \leftarrow (2 * A * t\hat{}2 + 4 * B * t + 4 * C + t\hat{}4)/(4 * (t\hat{}2 + t + 1))$
3: **Fin**

Algorithme 3.10 Fonction dyex : retourne la dérivée de la solution exacte en t

Données : t : un réel,
A, B, C : trois réels
Résultat : w : un réel tel que $w = y'(t)$

1: **Fonction** $w \leftarrow$ dyex(t, A, B, C)
2: $w \leftarrow \ldots$ ▷ à faire
3: **Fin**

Algorithme 3.11 Fonction d2yex : retourne la dérivée seconde de la solution exacte en t

Données : t : un réel,
A, B, C : trois réels
Résultat : w : un réel tel que $w = y''(t)$

1: **Fonction** $w \leftarrow$ d2yex(t, A, B, C)
2: $w \leftarrow \ldots$ ▷ à faire
3: **Fin**

Algorithme 3.12 Programme permettant de représenter graphiquement la solution exacte, sa dérivée première et sa dérivée seconde à partir des données N, α, β et γ.

1: $N \leftarrow 1000,\ \alpha \leftarrow 6,\ \beta \leftarrow -5,\ \gamma \leftarrow -2$
2: $C \leftarrow \alpha,\ B \leftarrow \beta + C,\ A \leftarrow \gamma + 2 * B$
3: $\boldsymbol{t} \leftarrow \text{DisReg}(0, 10, N)$
4: **Pour** $n \leftarrow 1$ **à** $N + 1$ **faire**
5: $\boldsymbol{Yex}(n) \leftarrow \text{yex}(\boldsymbol{t}(n), A, B, C)$
6: $\boldsymbol{dYex}(n) \leftarrow \text{dyex}(\boldsymbol{t}(n), A, B, C)$
7: $\boldsymbol{d2Yex}(n) \leftarrow \text{d2yex}(\boldsymbol{t}(n), A, B, C)$
8: **Fin**
9: plot($\boldsymbol{t}, \boldsymbol{Yex}$) ▷ Représentation de la solution exacte
10: plot($\boldsymbol{t}, \boldsymbol{dYex}$) ▷ Représentation de sa dérivée
11: plot($\boldsymbol{t}, \boldsymbol{d2Yex}$) ▷ Représentation de sa dérivée seconde

R. 5

L'erreur est la différence entre la solution exacte et la solution numérique au même point de discrétisation.
Par exemple les erreurs numériques commises en valeurs absolues par le schéma pour les approximations de $\big(y(t^n)\big)_{n=0}^{N}$, sont les $(N+1)$ réels

$$|y(t^n) - \mathbb{Y}(1, n+1)|, \quad \forall n \in [\![0, N]\!].$$

On en déduit alors le programme suivant:

Algorithme 3.13 Programme permettant de représenter graphiquement les erreurs numériques commises par le schéma d'Euler progressif

1: $N \leftarrow 1000,\ \alpha \leftarrow 6,\ \beta \leftarrow -5,\ \gamma \leftarrow -2$
2: $\boldsymbol{Y0} \leftarrow \begin{pmatrix} 6 \\ -5 \\ -2 \end{pmatrix}$ ▷ Données initiales
3: $[\boldsymbol{t}, \mathbb{Y}] \leftarrow \text{redEUPvec}(\text{fCauchy}, 0, 10, \boldsymbol{Y0}, 1000)$
4: $C \leftarrow \alpha,\ B \leftarrow \beta + C,\ A \leftarrow \gamma + 2 * B$
5: **Pour** $n \leftarrow 1$ **à** $N + 1$ **faire**
6: $\quad \boldsymbol{E}(n) \leftarrow \text{abs}\Big(\text{yex}\big(\boldsymbol{t}(n), A, B, C\big) - \mathbb{Y}(1, n)\Big)$
7: $\quad \boldsymbol{dE}(n) \leftarrow \text{abs}\Big(\text{dyex}\big(\boldsymbol{t}(n), A, B, C\big) - \mathbb{Y}(2, n)\Big)$
8: $\quad \boldsymbol{d2E}(n) \leftarrow \text{abs}\Big(\text{d2yex}\big(\boldsymbol{t}(n), A, B, C\big) - \mathbb{Y}(3, n)\Big)$
9: **Fin**
10: $\text{plot}(\boldsymbol{t}, \boldsymbol{E})$
11: $\text{plot}(\boldsymbol{t}, \boldsymbol{dE})$
12: $\text{plot}(\boldsymbol{t}, \boldsymbol{d2E})$

Fin correction

3.4 Méthodes à un pas ou à pas séparés

Les méthodes proposées dans cette section ont objectif la résolution du problème de Cauchy (voir Definition 3.2.1, page 49).
On note $(t^n)_{n=0}^N$ la discrétisation régulière de l'intervalle $[t^0, t^0 + T]$. Il est toutefois possible de prendre une discrétisation où le pas n'est pas constant et vérifiant $t^0 = a < t^1 < \ldots < t^N = b$ mais cette possibilité ne sera pas étudiée dans ce cours.
Les méthodes sont dites **à un pas** car le calcul d'une approximation $\boldsymbol{y}^{[n+1]}$ de $\boldsymbol{y}(t^{n+1})$ ne nécessite que la connaissance de la valeur $\boldsymbol{y}^{[n]} \approx \boldsymbol{y}(t^n)$.
De manière générique, ces schémas sont donnés par

Définition 3.4.1 (Méthodes à un pas)**.** *Les méthodes à un pas utilisent la formule générale:*

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + h\boldsymbol{\Phi}(t^n, \boldsymbol{y}^{[n]}, h) \tag{3.17}$$

Le schéma (3.17) *converge sur l'intervalle* $[t^0, t^0 + T]$ *si, pour la suite des* $\boldsymbol{y}^{[n]}$ *calculés, l'écart maximum avec la solution exacte diminue quand le pas* h *diminue:*

$$\lim_{h=\frac{T}{N}\to 0} \max_{n\in\{0,\ldots,N\}} \left\| \boldsymbol{y}^{[n]} - \boldsymbol{y}(t^n) \right\|_\infty = 0$$

Pour la méthode d'Euler progressif, la fonction $\boldsymbol{\Phi}(t, \boldsymbol{y}, h)$ est $\boldsymbol{f}(t, \boldsymbol{y})$.

Définition 3.4.2 (consistance)**.** *Le schéma de calcul* (3.17) *est consistant avec le problème de Cauchy* (3.9)-(3.10) *si*

$$\lim_{h=\frac{T}{N}\to 0} \max_{n} \left\| \frac{\boldsymbol{y}(t^{n+1}) - \boldsymbol{y}(t^n)}{h} - \boldsymbol{\Phi}(t^n, \boldsymbol{y}(t^n), h) \right\|_\infty = 0$$

Cela signifie que le schéma doit être une approximation vraisemblable, bien construite.

Théorème 3.4.1 (consistance (admis)). *Le schéma* (3.17) *est consistant avec le problème de Cauchy* (3.9)-(3.10) *si* $\Phi(t, \boldsymbol{y}, 0) = f(t, \boldsymbol{y})$.

Définition 3.4.3 (stabilité). *La méthode est stable si une petite perturbation sur* $\boldsymbol{y}^{[0]}$ *ou* $\boldsymbol{\Phi}$ *n'entraîne qu'une petite perturbation sur la solution approchée, et cela quel que soit le pas* h.

Souvent , lorsque la méthode n'est pas stable, l'amplification des erreurs est exponentielle et, au bout de quelques calculs, les résultats entraînent des dépassements de capacité.

Théorème 3.4.2 (stabilité (admis)). *Si* $\boldsymbol{\Phi}(t, \boldsymbol{y}, h)$ *vérifie la condition de Lipschitz en* $\boldsymbol{y}$ *alors la méthode est stable.*

Théorème 3.4.3 (convergence (admis)). *Si la méthode est* ***stable et consistante****, alors elle* ***converge*** *pour n'importe quelle valeur initiale.*

Définition 3.4.4 (Ordre d'un schéma). *Le schéma* (3.17) *est d'ordre* p *si la solution* $\boldsymbol{y}$ *du problème de Cauchy* (3.9)-(3.10) *vérifie*

$$\max_n \left\| \frac{\boldsymbol{y}(t^{n+1}) - \boldsymbol{y}(t^n)}{h} - \boldsymbol{\Phi}(t^n, \boldsymbol{y}(t^n), h) \right\|_\infty = \mathcal{O}(h^p)$$

Lemme 3.4.1 ((admis)). *Soient* $\boldsymbol{y}$ *la solution du problème de Cauchy* (3.9)-(3.10). *et* $(\boldsymbol{y}^{[n]})_{n \in [\![0,N]\!]}$ *donnés par un schéma à un pas* (3.17) *d'ordre* p *avec* $\boldsymbol{y}^{[0]} = \boldsymbol{y}(t^0)$. *On a alors*

$$\max_{n \in [\![0,N]\!]} \left\| \boldsymbol{y}(t^n) - \boldsymbol{y}^{[n]} \right\|_\infty = \mathcal{O}(h^p) \tag{3.18}$$

Proposition 3.4.1 ((admis)). *Le schéma d'Euler progressif est une méthode à un pas d'ordre* 1.

Il est possible de vérifier/retrouver numériquement l'ordre du schéma d'Euler progressif. Pour celà on choisit un problème de Cauchy dont la solution exacte est connue et on calcule pour différentes valeurs de h (et donc différentes valeurs de N) le maximum de l'erreur commise entre la solution exacte et la solution numérique donnée par le schéma d'Euler progressif :

$$E(h) = \max_{n \in [\![0,N]\!]} \left\| \boldsymbol{y}(t^n) - \boldsymbol{y}^{[n]} \right\|_\infty$$

On représente ensuite la fonction $h \mapsto E(h)$. La méthode d'Euler progressive étant d'ordre 1, on a alors $E(h) = \mathcal{O}(h) \approx Ch$ quand h est suffisament petit : la courbe obtenue doit donc être une droite.

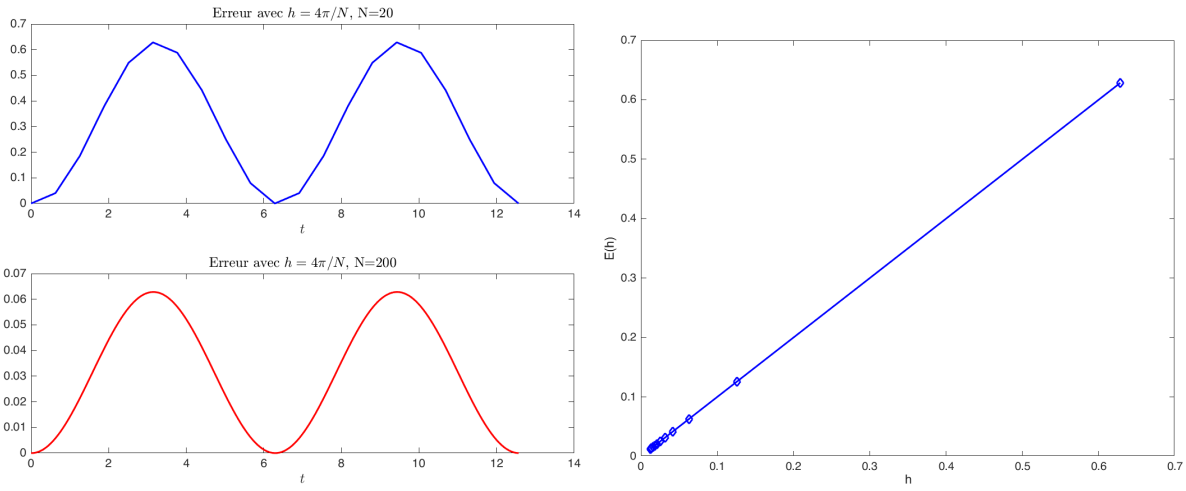


Figure 3.11: Méthode d'Euler progressive : vérification numérique de l'ordre pour l'E.D.O. de l'exercice 3.3.1

Dans la section suivante on étudie une classe de méthodes à un pas très usitées.

3.5 Méthodes de Runge-Kutta

Les méthodes de Runge-Kutta sont des méthodes à un pas dédiées à la résolution de problèmes de Cauchy (3.9)-(3.10).

(a) *Carle Runge* 1856-1927, mathématicien et physicien allemand

(b) *Martin Wilhelm Kutta* 1867-1944, Mathématicien allemand

(c) *John C. Butcher* 1933, Mathématicien appliqué néozélandais

3.5.1 Principe

Pour simplifier, on suppose $h_n = h$. L'idée fondamentale des méthodes de Runge-Kutta est d'intégrer l'équation (3.9)) sur $[t^n, t^{n+1}]$ et de calculer:

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

en utilisant une formule d'intégration numérique à q points intermédiaires pour évaluer l'intégrale.
La fonction $\boldsymbol{\Phi}$ associée à une méthode de Runge-Kutta à q évaluations de $\boldsymbol{f}$ peut s'écrire sous la forme :

$$\boldsymbol{\Phi}(t, \boldsymbol{y}, h) = \sum_{i=1}^{q} c_i \boldsymbol{k}^{[i]}(t, \boldsymbol{y}, h)$$

avec

$$\boldsymbol{k}^{[i]}(t, \boldsymbol{y}, h) = \boldsymbol{f}\left(t + ha_i, y + h\sum_{j=1}^{q} b_{i,j}\boldsymbol{k}^{[j]}(t, \boldsymbol{y}, h)\right),\ 1 \leqslant i \leqslant q$$

que l'on peut représenter sous la forme d'un tableau dit **tableau de Butcher** :

$$\begin{array}{c|c} \boldsymbol{a} & \mathbb{B} \\ \hline & \boldsymbol{c}^t \end{array} \tag{3.19}$$

avec $\mathbb{B} = (b_{i,j})_{i,j\in[\![1,q]\!]} \in \mathcal{M}q, q(\mathbb{R})$, $\boldsymbol{a} = (a_i)_{i\in[\![1,q]\!]} \in \mathbb{R}^q$ et $\boldsymbol{c} = (c_i)_{i\in[\![1,q]\!]} \in \mathbb{R}^q$

Proposition 3.5.1 ((admis)). *a. Les méthodes de Runge-Kutta explicites sont stables si $\boldsymbol{f}$ est contractante en $\boldsymbol{y}$.*

b. Une méthode de Runge-Kutta est d'ordre 0 si

$$a_i = \sum_{j=1}^{q} b_{ij}.$$

c. Une méthode de Runge-Kutta est d'ordre 1 (et donc consistante) si elle est d'ordre 0 et si

$$\sum_{i=1}^{q} c_i = 1.$$

d. Une méthode de Runge-Kutta est d'ordre 2 si elle est d'ordre 1 et si

$$\sum_{i=1}^{q} c_i a_i = 1/2.$$

e. Une méthode de Runge-Kutta est explicite si la matrice $\mathbb{B}$ est triangulaire inférieure à diagonale nulle :

$$\forall (i,j) \in [\![1,q]\!],\ j \geqslant i, \quad b_{ij} = 0.$$

3.5.2 Formules explicites de Runge-Kutta d'ordre 2

Le tableau de Butcher associé aux méthodes de Runge-Kutta d'ordre 2 s'écrit sous la forme

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2\alpha} & \frac{1}{2\alpha} & 0 \\ \hline & 1-\alpha & \alpha \end{array} \tag{3.20}$$

Pour la méthode de Runge-Kutta d'ordre 2, la fonction $\boldsymbol{\Phi}$ associée au schéma général (3.17) est donnée par

$$\boldsymbol{\Phi}(t,\boldsymbol{y},h) = (1-\alpha)\boldsymbol{f}(t,\boldsymbol{y}) + \alpha \boldsymbol{f}(t+\frac{h}{2\alpha}, \boldsymbol{y} + \frac{h}{2\alpha}\boldsymbol{f}(t,\boldsymbol{y})).$$

- Avec $\alpha = \frac{1}{2}$, on obtient la **méthode de Heun** :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) + \frac{h}{2}\boldsymbol{f}\left(t^{n+1}, \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})\right).$$

- Avec $\alpha = 1$, on obtient la **méthode d'Euler modifiée** ou **méthode du point milieu**:

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + h\boldsymbol{f}\left(t^n + \frac{h}{2}, \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})\right).$$

Exercice 3.5.1

la **méthode de Heun** est donnée par

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) + \frac{h}{2}\boldsymbol{f}\left(t^{n+1}, \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})\right).$$

Q. 1 *Ecrire la fonction algorithmique* REDHeunVec *permettant de résoudre un problème de Cauchy vectoriel par la méthode de Heun en utilisant au plus $2N$ évaluation de $\boldsymbol{f}$.*

Q. 2 *Ecrire un programme algorithmique permettant de retrouver numériquement l'ordre de cette méthode.*

Correction 3.5.1

R. 1

Le schéma de Heun peut s'écrire sous la forme

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{2}(\boldsymbol{k}_1^n + \boldsymbol{k}_2^n)$$

avec

$$\begin{aligned} \boldsymbol{k}_1^n &= \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) \\ \boldsymbol{k}_2^n &= \boldsymbol{f}(t^{n+1}, \boldsymbol{y}^{[n]} + h\boldsymbol{k}_1) \end{aligned}$$

L'algorithme de la fonction REDHeunVec s'écrit alors :

Algorithme 3.14 Fonction REDHeunVec : résolution d'un problème de Cauchy par le schéma de Heun

Données : $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: un vecteur de $\mathbb{R}^d$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).

Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1},\ \forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: matrice réelle de dimension $d \times (N+1)$, $\boldsymbol{Y}(:,n) = \boldsymbol{y}^{(n-1)},\ \forall n \in [\![1, N+1]\!]$

```
Fonction [t,Y] ← REDHeunVec( f, t^0, T, y^0, N )
    t ← DisReg(t^0, t^0 + T, N)
    h ← T/N
    Y(:,1) ← y^0
    Pour n ← 1 à N faire
        k_1 ← f(t(n), Y(:,n))
        k_2 ← f(t(n+1), Y(:,n) + h k_1)
        Y(:,n+1) ← Y(:,n) + (h/2) * (k_1 + k_2)
    Fin
Fin
```

R. 2

Il est possible de vérifier/retrouver numériquement l'ordre du schéma de Heun. Pour celà on choisi un problème de Cauchy dont la solution exacte est connue et on calcule pour différentes valeurs de h (et donc différentes valeurs de N) le maximum de l'erreur commise entre la solution exacte et la solution numérique donnée par le schéma de Heun :

$$E(h) = \max_{n \in [\![0,N]\!]} \left\| \boldsymbol{y}(t^n) - \boldsymbol{y}^{[n]} \right\|_\infty$$

On représente ensuite la fonction $h \mapsto E(h)$. La méthode de Heun étant d'ordre 2, on a alors $E(h) = \mathcal{O}(h^2) \approx Ch^2$ quand h est suffisament petit. On utilise alors une échelle logarithmique pour représenter la courbe. En effet, on a

$$\log E(h) \approx \log(Ch^2) = \log(C) + 2\log(h)$$

En posant $X = \log(h)$ et $Y = \log E(h)$, coordonnées en échelle logarithmique, on a

$$Y \approx \log(C) + 2X$$

qui est l'équation d'une droite. La pente de cette droite est donc l'ordre de la méthode.

```
1: t^0 ← 0, T ← 4π,
2: f : t, z ⟶ cos(t) + 1, y_ex : t ⟶ sin(t) + t
3: y^0 ← y_ex(t^0)
4: LN ← 100 : 50 : 1000
5: nLN ← length(LN)
6: H ← 0_{nLN},                                   ▷ pour stocker les h
7: E ← 0_{nLN},                                   ▷ pour stocker les erreurs
8: Pour k ← 1 à nLN faire
9:    N ← LN(k)
10:   [t, y] ← REDHeunVec(f, t^0, T, y^0, N)
11:   E(k) ← max(abs(y − y_ex(t)))
12:   H(k) ← T/N
13: Fin
14: ...                                           ▷ Representation graphique
```

Fin correction

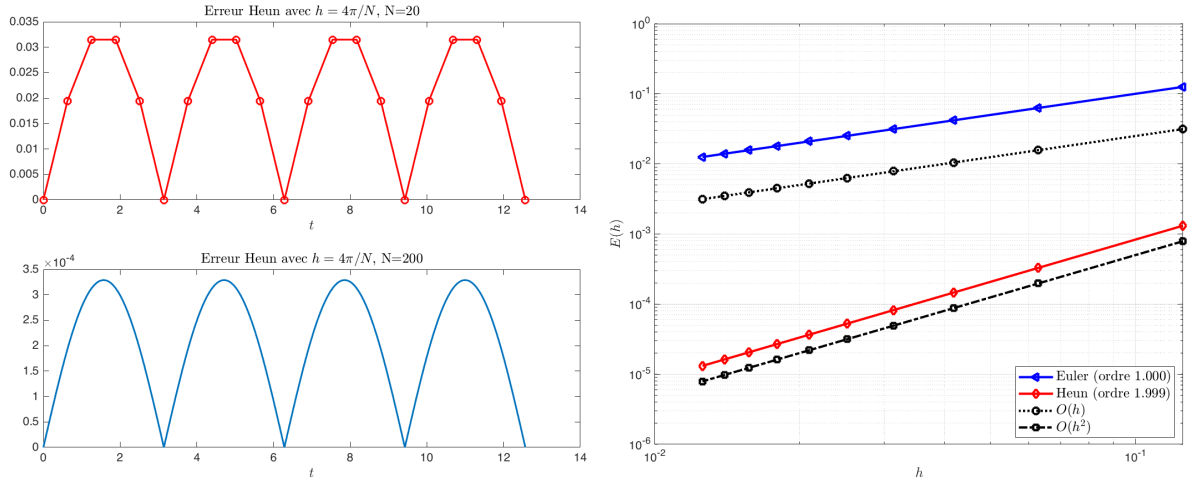


Figure 3.13: Méthode de Heun : vérification numérique de l'ordre et comparaison avec la méthode d'Euler progressive

Les méthodes d'ordre $p > 1$ sont plus précises que la méthode d'Euler, et d'autant plus que p augmente. On se limite dans la pratique à $p = 4$. Cette méthode est connue sous le nom de *Runge-Kutta 4*. On peut noter que pour ces méthodes le pas peut être adapté à chaque itération.

3.5.3 Méthodes de Runge-Kutta d'ordre 4

La méthode explicite la plus utilisée est donnée par le tableau de Buchler suivant

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array} \tag{3.21}$$

Ce qui donne le schéma explicite de Runge-Kutta d'ordre 4 :

$$\begin{array}{rcl} \boldsymbol{k}_1^{[n]} & = & \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) \\ \boldsymbol{k}_2^{[n]} & = & \boldsymbol{f}(t^n + \frac{h}{2}, \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{k}_1^{[n]}) \\ \boldsymbol{k}_3^{[n]} & = & \boldsymbol{f}(t^n + \frac{h}{2}, \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{k}_2^{[n]}) \\ \boldsymbol{k}_4^{[n]} & = & \boldsymbol{f}(t^n + h, \boldsymbol{y}^{[n]} + h\boldsymbol{k}_3^{[n]}) \\ \hline \boldsymbol{y}^{[n+1]} & = & \boldsymbol{y}^{[n]} + \frac{h}{6}(\boldsymbol{k}_1^{[n]} + 2\boldsymbol{k}_2^{[n]} + 2\boldsymbol{k}_3^{[n]} + \boldsymbol{k}_4^{[n]}). \end{array} \tag{3.22}$$

Exercice 3.5.2

la **méthode de Runge-Kutta d'ordre 4** est donnée par

$$\begin{array}{lcl} \boldsymbol{k}_1^{[n]} & = & \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) \\ \boldsymbol{k}_2^{[n]} & = & \boldsymbol{f}(t^n + \frac{h}{2}, \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{k}_1^{[n]}) \\ \boldsymbol{k}_3^{[n]} & = & \boldsymbol{f}(t^n + \frac{h}{2}, \boldsymbol{y}^{[n]} + \frac{h}{2}\boldsymbol{k}_2^{[n]}) \\ \boldsymbol{k}_4^{[n]} & = & \boldsymbol{f}(t^n + h, \boldsymbol{y}^{[n]} + h\boldsymbol{k}_3^{[n]}) \\ \hline \boldsymbol{y}^{[n+1]} & = & \boldsymbol{y}^{[n]} + \frac{h}{6}(\boldsymbol{k}_1^{[n]} + 2\boldsymbol{k}_2^{[n]} + 2\boldsymbol{k}_3^{[n]} + \boldsymbol{k}_4^{[n]}). \end{array}$$

Q. 1 *Ecrire une fonction algorithmique* REDRK4Vec *permettant de résoudre un problème de Cauchy vectoriel par la méthode de Runge-Kutta d'ordre 4.*

Q. 2 *Ecrire un programme algorithmique permettant de retrouver numériquement l'ordre de cette méthode.*

Correction 3.5.2

R. 1

Il faut noter qu'il n'est pas nécessaire de stocker l'ensemble des vecteurs $\boldsymbol{k}_1^{[n]}, \ldots, \boldsymbol{k}_4^{[n]}$, $n \in [\![1, N]\!]$. Pour minimiser l'occupation mémoire de l'ordinateur, à chaque itération, on calcule $\boldsymbol{k}_1 \leftarrow \boldsymbol{k}_1^{[n]}(t^n, \boldsymbol{y}^{[n]})$, ... $\boldsymbol{k}_4 \leftarrow \boldsymbol{k}_4^{[n]}(t^n, \boldsymbol{y}^{[n]})$. L'algorithme de la fonction REDRK4Vec s'écrit alors :

Algorithme 3.15 Fonction REDRK4Vec : résolution d'un problème de Cauchy par le schéma de RK4

Données : $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0 + T] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: un vecteur de $\mathbb{R}^d$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: matrice réelle de dimension $d \times (N+1)$, $\boldsymbol{Y}(:, n) = \boldsymbol{y}^{(n-1)}$, $\forall n \in [\![1, N+1]\!]$

Fonction $[\boldsymbol{t}, \boldsymbol{Y}] \leftarrow$ REDRK4Vec$(\ f, t^0, T, \boldsymbol{y}^0, N\ )$
  $\boldsymbol{t} \leftarrow DisReg(t^0, t^0 + T, N)$
  $h \leftarrow T/N$
  $\boldsymbol{Y}(:, 1) \leftarrow \boldsymbol{y}^0$
  Pour $n \leftarrow 1$ **à** N **faire**
    $\boldsymbol{k}_1 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n), \boldsymbol{Y}(:, n))$
    $\boldsymbol{k}_2 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n) + h/2, \boldsymbol{Y}(:, n) + (h/2)\boldsymbol{k}_1)$
    $\boldsymbol{k}_3 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n) + h/2, \boldsymbol{Y}(:, n) + (h/2)\boldsymbol{k}_2)$
    $\boldsymbol{k}_4 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n) + h, \boldsymbol{Y}(:, n) + h\boldsymbol{k}_3)$
    $\boldsymbol{Y}(:, n+1) \leftarrow \boldsymbol{Y}(:, n) + (h/6) * (\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$
  Fin
Fin

R. 2

voir aussi correction Exercice 3.5.1-Q2 : l'ordre 2 *étant remplacé par* 4 *ici!* Il est possible de vérifier/retrouver numériquement l'ordre du schéma de Runge-Kutta 4. Pour celà on choisi un problème de Cauchy dont la solution exacte est connue et on calcule pour différentes valeurs de h (et donc différentes valeurs de N) le maximum de l'erreur commise entre la solution exacte et la solution numérique donnée par le schéma de Runge-Kutta 4 :

$$E(h) = \max_{n \in [\![0, N]\!]} \left\| \boldsymbol{y}(t^n) - \boldsymbol{y}^{[n]} \right\|_\infty$$

On représente ensuite la fonction $h \mapsto E(h)$. La méthode de Runge-Kutta 4 étant d'ordre 4, on a

alors théoriquement $E(h) = \mathcal{O}(h^4) \approx Ch^4$ quand h est suffisament petit. On utilise alors une échelle logarithmique pour représenter la courbe. En effet, on a

$$\log E(h) \approx \log(Ch^4) = \log(C) + 4\log(h)$$

En posant $X = \log(h)$ et $Y = \log E(h)$, coordonnées en échelle logarithmique, on a

$$Y \approx \log(C) + 4X$$

qui est l'équation d'une droite. La pente de cette droite est donc l'ordre de la méthode.

1: $t^0 \leftarrow 0$, $T \leftarrow 4\pi$,
2: $f : t, z \longrightarrow \cos(t) + 1$, $y_{ex} : t \longrightarrow \sin(t) + t$
3: $y^0 \leftarrow y_{ex}(t^0)$
4: $LN \leftarrow 100 : 50 : 1000$
5: $nLN \leftarrow \text{length}(LN)$
6: $H \leftarrow \mathbb{0}_{nLN}$, ▷ pour stocker les h
7: $E \leftarrow \mathbb{0}_{nLN}$, ▷ pour stocker les erreurs
8: **Pour** $k \leftarrow 1$ **à** nLN **faire**
9: $N \leftarrow LN(k)$
10: $[t, y] \leftarrow \text{REDRK4Vec}(f, t^0, T, \boldsymbol{y}^0, N)$
11: $E(k) \leftarrow \max(\text{abs}(y - y_{ex}(t)))$
12: $H(k) \leftarrow T/N$
13: **Fin**
14: ... ▷ Representation graphique

Fin correction

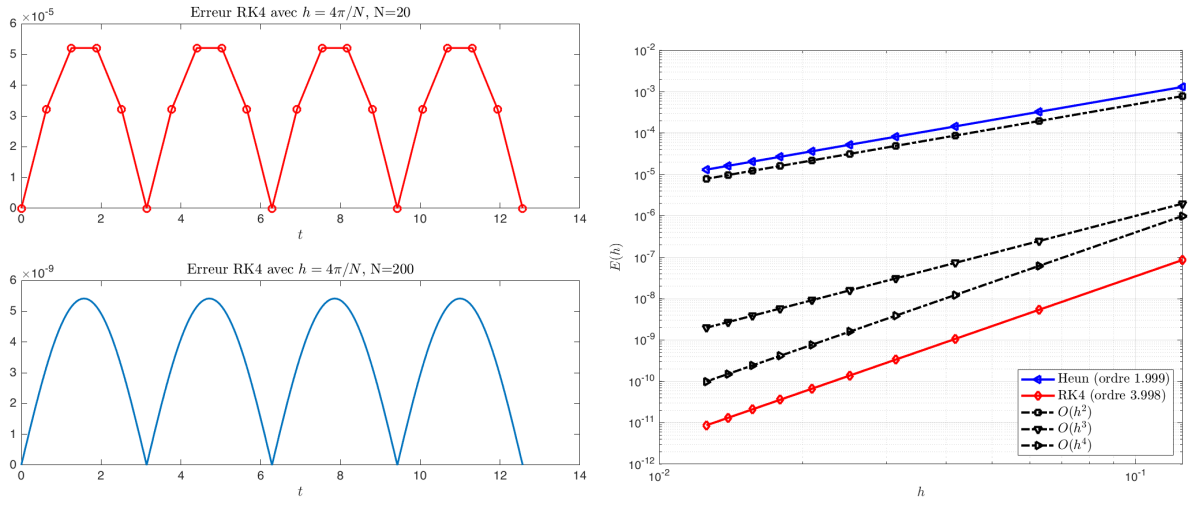


Figure 3.14: Méthode de Runge-Kutta 4 : vérification numérique de l'ordre et comparaison avec la méthode de Heun

Lorsque l'on diminue encore le pas on obtient la Figure 3.15. L'erreur engendrée par la méthode de Runge-Kutta 4 se *stabilise* autour de la valeur $1e-14$: la précision machine est atteinte!

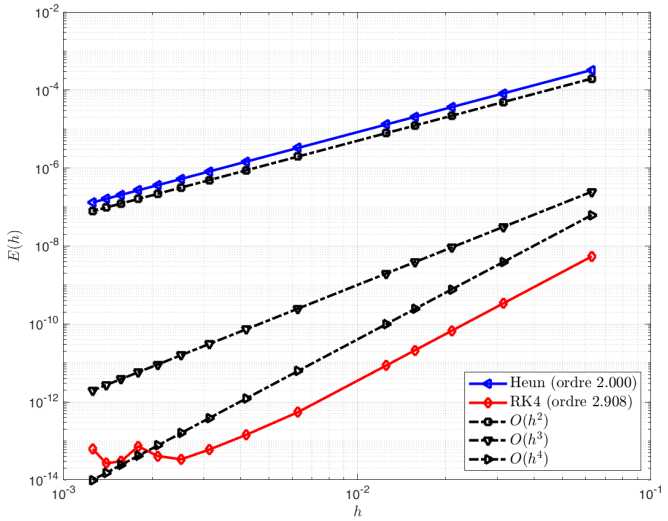


Figure 3.15: Méthode de Runge-Kutta 4 : vérification numérique de l'ordre avec précision machine atteinte

3.6 Méthodes à pas multiples

3.6.1 Exemple : schéma de point milieu

Considérons la méthode à deux pas définie par la récurrence:

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n-1]} + 2h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}). \tag{3.23}$$

Cette méthode est d'ordre 2.
De manière évidente, ces méthodes à pas multiples posent des problèmes de démarrage: ici, on ne peut calculer directement $\boldsymbol{y}^{[1]}$. Les premières valeurs doivent être calculées par un autre schéma.

3.6.2 Le principe

Dans tous les schémas présentés, la valeur de $\boldsymbol{y}^{[n+1]}$ était déterminée à chaque pas de façon explicite en fonction uniquement du point $\boldsymbol{y}^{[n]}$. On peut tenir compte de plusieurs valeurs $\boldsymbol{y}^{[i]}$ précédemment calculées et ainsi travailler sur un nombre de pas multiples.

Définition 3.6.1 (Méthodes à pas multiples)**.** *Les méthodes à pas multiples s'écrivent sous la forme générale:*

$$\sum_{i=0}^{k} \alpha_i \boldsymbol{y}^{[n+i]} = h \sum_{i=0}^{k} \beta_i \boldsymbol{f}(t^{n+i}, \boldsymbol{y}^{[n+i]}) \tag{3.24}$$

où k est le nombre de pas, $\alpha_k \neq 0$ et $|\alpha_0| + |\beta_0| > 0$.

Remarque 3.6.1. *Si $\beta_k = 0$ le schéma est explicite, sinon il est implicite.*

Définition 3.6.2 (ordre)**.** *Soit $\boldsymbol{y}$ la solution d'un problème de Cauchy (3.9)-(3.10) et $\boldsymbol{y}^{[n+k]}$ le terme obtenu par le schéma (3.24) en prenant $\boldsymbol{y}^{[n+i]} = \boldsymbol{y}(t^{n+i})$, $\forall i \in [\![0, k-1]\!]$. Alors, l'erreur locale est*

$$\tau(n+k) = \left\| \boldsymbol{y}(t^{n+k}) - \boldsymbol{y}^{[n+k]} \right\|_\infty .$$

*Le schéma (3.24) est alors d'**ordre** p si*

$$\tau(n+k) = \mathcal{O}(h^{p+1}).$$

Théorème 3.6.1 (ordre schémas à pas multiples (admis)). *Un schéma à pas multiples de type* (3.24) *est d'ordre p si et seulement si*

$$\begin{aligned}\sum_{i=0}^{k}\alpha_i &= 0,\\ \sum_{i=0}^{k}\alpha_i i^q &= q\sum_{i=0}^{k}\beta_i i^{q-1}, \quad \forall q\in[\![1,p]\!].\end{aligned}$$

Propriété 3.6.1 (stabilité schémas à pas multiples (admis)). *Soit une méthode à pas multiples donnée par* (3.24). *On note P le polynôme défini par*

$$P(\lambda)=\sum_{i=0}^{k}\alpha_i\lambda^i.$$

La méthode à pas multiples est ***stable****, si*

a. *toutes les racines de P sont de module inférieur ou égal à 1,*

b. *une racine de module égal à 1 est une racine simple de P.*

Pour le schéma (3.23), on a $k=2$ et

$$\begin{cases}\alpha_0 &= -1\\ \alpha_1 &= 0\\ \alpha_2 &= 1\end{cases}\qquad\begin{cases}\beta_0 &= 0\\ \beta_1 &= 2\\ \beta_2 &= 0\end{cases}$$

On obtient donc $P(\lambda)=-1+\lambda^2=(\lambda+1)(\lambda-1)$: le schéma (3.23) est stable.

Théorème 3.6.2 (convergence (admis)). *On suppose que les k valeurs initiales vérifient,*

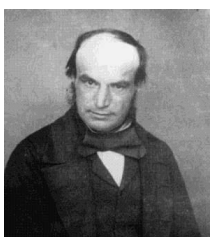
$$\left\|\boldsymbol{y}(t^i)-\boldsymbol{y}^{[i]}\right\|\leqslant C_0h^p,\ \forall i\in[\![0,k-1]\!].$$

Si le schéma (3.24) *est* ***stable et d'ordre*** *p, alors il est* ***convergent*** *d'ordre p :*

$$\left\|\boldsymbol{y}(t^n)-\boldsymbol{y}^{[n]}\right\|\leqslant Ch^p,\ \forall n\in[\![0,N]\!].$$

Remarque 3.6.2. *Pour obtenir, à partir d'un schéma à k pas, un schéma d'ordre p il faut obligatoirement initialiser les k premièrs termes $(\boldsymbol{y}^{[n]})_{n=0}^{k-1}$ à l'aide d'un schéma d'ordre p au moins pour conserver l'ordre.*

Nous allons maintenant donner quelques schémas explicites et implicites à pas multiples développer par :



John Couch Adams 1819-1892, mathématicien et astronome britannique

Francis Bashforth 1819-1912, mathématicien appliqué britannique

(a) *Forest Ray Moulton* 1872-1952, mathématicien et astronome américain

3.6.3 Méthodes explicites d'Adams-Bashforth

On note en abrégé $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$. Voici trois schémas :

- schéma explicite d'Adams-Bashforth d'ordre 2 à 2 pas :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{2}\left(3\boldsymbol{f}^{[n]} - \boldsymbol{f}^{[n-1]}\right). \tag{3.25}$$

- schéma explicite d'Adams-Bashforth d'ordre 3 à 3 pas :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{12}\left(23\boldsymbol{f}^{[n]} - 16\boldsymbol{f}^{[n-1]} + 5\boldsymbol{f}^{[n-2]}\right). \tag{3.26}$$

- schéma explicite d'Adams-Bashforth d'ordre 4 à 4 pas :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(55\boldsymbol{f}^{[n]} - 59\boldsymbol{f}^{[n-1]} + 37\boldsymbol{f}^{[n-2]} - 9\boldsymbol{f}^{[n-3]}\right). \tag{3.27}$$

Ces schémas sont **explicites** et leur ordre correspond au nombre de pas.

Exercice 3.6.1

La **méthode de Adam-Bashforth d'ordre 4** explicite est donnée par

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \tfrac{h}{24}\left(55\boldsymbol{f}^{[n]} - 59\boldsymbol{f}^{[n-1]} + 37\boldsymbol{f}^{[n-2]} - 9\boldsymbol{f}^{[n-3]}\right). \tag{3.1}$$

avec $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$.

Q. 1 *Ecrire la fonction algorithmique REDAB4Vec permettant de résoudre un problème de Cauchy vectoriel par cette méthode.*

Correction 3.6.1

R. 1

Soit $(t^{[n]})_{n=0}^{N}$ la discrétisation régulière de $[t^0, t^0 + T]$ avec $h = T/N$. On a donc $t^{[n]} = t^0 + nh$, $\forall n \in [\![0, N]\!]$.
On ne peut *utiliser* le schéma à pas multiples (3.1) que pour $n \geqslant 3$. On va alors utiliser un schéma à un pas d'ordre (au moins) 4 pour calculer les 4 premiers termes $\boldsymbol{y}^{[0]}$, $\boldsymbol{y}^{[1]}$, $\boldsymbol{y}^{[2]}$ et $\boldsymbol{y}^{[3]}$ nécessaire pour *démarrer* le schéma (3.1). On choisi par exemple la méthode de Runge-Kutta d'ordre 4 pour initialiser ces 4 termes. Deux possibilités :

- on réécrit le schéma de Runge-Kutta d'ordre 4 pour ces 4 premiers termes

```
1: Y(:,1) ← y^0
2: Pour n ← 1 à 3 faire
3:    k_1 ← f(t(n), Y(:,n))
4:    k_2 ← f(t(n) + h/2, Y(:,n) + (h/2)k_1)
5:    k_3 ← f(t(n) + h/2, Y(:,n) + (h/2)k_2)
6:    k_4 ← f(t(n) + h, Y(:,n) + hk_3)
7:    Y(:,n+1) ← Y(:,n) + (h/6) * (k_1 + 2k_2 + 2k_3 + k_4)
8: Fin
```

- on utilise la fonction REDRK4Vec avec ses paramètres d'entrées judicieusement choisis :

```
1: [t_ini, Y_ini] ← REDRK4Vec(f, t^0, 3 * h, y^0, 3)
2: Pour n ← 1 à 4 faire
3:    Y(:,n) ← Y_ini(:,n)
4: Fin
```

En choisissant cette dernière solution, l'algorithme de la fonction REDAB4Vec s'écrit alors :

Algorithme 3.16 Fonction REDAB4Vec : résolution d'un problème de Cauchy par le schéma explicite d'Adams-Bashforth d'ordre 4

Données : f : $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: un vecteur de $\mathbb{R}^d$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).

Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1},\ \forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: matrice réelle de dimension $d \times (N+1)$, $\boldsymbol{Y}(:,n) = \boldsymbol{y}^{(n-1)},\ \forall n \in [\![1, N+1]\!]$

Fonction $[\boldsymbol{t}, \boldsymbol{Y}] \leftarrow$ REDAB4Vec($f, t^0, T, \boldsymbol{y}^0, N$)
  $\boldsymbol{t} \leftarrow$ DisReg(t^0, t^0+T, N)
  $h \leftarrow T/N$
  $[\boldsymbol{t}_{ini}, \boldsymbol{Y}_{ini}] \leftarrow$ REDRK4Vec$(f, t^0, 3*h, \boldsymbol{y}^0, 3)$
  Pour $n \leftarrow 1$ **à** 4 **faire**
    $\boldsymbol{Y}(:,n) \leftarrow \boldsymbol{Y}_{ini}(:,n)$
  Fin
  $\boldsymbol{k}_1 \leftarrow \boldsymbol{f}(\boldsymbol{t}(3), \boldsymbol{Y}(:,3))$
  $\boldsymbol{k}_2 \leftarrow \boldsymbol{f}(\boldsymbol{t}(2), \boldsymbol{Y}(:,2))$
  $\boldsymbol{k}_3 \leftarrow \boldsymbol{f}(\boldsymbol{t}(1), \boldsymbol{Y}(:,1))$
  Pour $n \leftarrow 4$ **à** N **faire**
    $\boldsymbol{k}_0 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n), \boldsymbol{Y}(:,n))$
    $\boldsymbol{Y}(:,n+1) \leftarrow \boldsymbol{Y}(:,n) + (h/24) * (55 * \boldsymbol{k}_0 - 59 * \boldsymbol{k}_1 + 37 * \boldsymbol{k}_2 - 9 * \boldsymbol{k}_3)$
    $\boldsymbol{k}_3 \leftarrow \boldsymbol{k}_2,\ \boldsymbol{k}_2 \leftarrow \boldsymbol{k}_1,\ \boldsymbol{k}_1 \leftarrow \boldsymbol{k}_0$
  Fin
Fin

Fin correction

3.6.4 Méthodes implicites d'Adams-Moulton

On note en abrégé $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$. Voici trois schémas :

- schéma d'Adams-Moulton d'ordre 2 à 1 pas :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{2}\left(\boldsymbol{f}^{[n+1]} + \boldsymbol{f}^{[n]}\right). \tag{3.28}$$

- schéma d'Adams-Moulton d'ordre 3 à 2 pas :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{12}\left(5\boldsymbol{f}^{[n+1]} + 8\boldsymbol{f}^{[n]} - \boldsymbol{f}^{[n-1]}\right) \tag{3.29}$$

- schéma d'Adams-Moulton d'ordre 4 à 3 pas :

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(9\boldsymbol{f}^{[n+1]} + 19\boldsymbol{f}^{[n]} - 5\boldsymbol{f}^{[n-1]} + \boldsymbol{f}^{[n-2]}\right) \tag{3.30}$$

Ces schémas sont **implicites** et leur ordre correspond au nombre de pas plus un.

3.6.5 Schéma prédicteur-correcteur

Principe

Il s'agit là d'une des méthodes les plus employées. Une méthode de prédiction-correction procède en deux étapes à chacune des itérations :

- **Prédiction** : on calcule une approximation de $\boldsymbol{y}(t_{n+1})$ notée $\overline{\boldsymbol{y}}^{[n+1]}$ à l'aide du **schéma explicite**
- **Correction** : on utilise le schéma implicite dans lequel les fonctions $\boldsymbol{f}$ utilisant $\boldsymbol{y}^{[n+1]}$ sont remplacées par les fonctions $\boldsymbol{f}$ utilisant $\overline{\boldsymbol{y}}^{[n+1]}$.

1: **Pour** $n \leftarrow 0$ **à** N **faire**
2: $\quad \overline{\boldsymbol{y}}^{[n+1]} \leftarrow$ donné par un **schéma explicite**
3: $\quad \boldsymbol{y}^{[n+1]} \leftarrow$ donné par **un schéma implicite**, inconnue $\boldsymbol{y}^{[n+1]}$ remplacée par $\overline{\boldsymbol{y}}^{[n+1]}$
4: **Fin**

Exemple

Choisissons la méthode d'Euler explicite pour prédicteur et la méthode implicite des trapèzes comme correcteur.

Euler explicite : $\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$

Trapèze implicite : $\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{2}(\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) + \boldsymbol{f}(t^{n+1}, \boldsymbol{y}^{[n+1]}))$

On obtient :

$$\begin{cases} \overline{\boldsymbol{y}}^{[n+1]} &= \boldsymbol{y}^{[n]} + h\boldsymbol{f}(t^n, \boldsymbol{y}^{[n]}) & \textbf{Prédiction} \\ \boldsymbol{y}^{[n+1]} &= \boldsymbol{y}^{[n]} + \frac{h}{2}(\boldsymbol{f}(t^{n+1}, \overline{\boldsymbol{y}}^{[n+1]}) + \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})) & \textbf{Correction} \end{cases}$$

Remarque 3.6.3. *On retrouve ici, pour ce cas simple, une formule de Runge-Kutta 2.*

En pratique, on peut utiliser un schéma d'Adams explicite (voir section 3.6.3) pour la prédiction et un autre implicite (voir section 3.6.4) pour la correction.

Exercice 3.6.2

On pose $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$. La **méthode de Adams-Bashforth d'ordre 4** explicite est donnée par

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(55\boldsymbol{f}^{[n]} - 59\boldsymbol{f}^{[n-1]} + 37\boldsymbol{f}^{[n-2]} - 9\boldsymbol{f}^{[n-3]}\right)$$

et la **méthode de Adams-Moulton d'ordre 4** implicite par

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(9\boldsymbol{f}^{[n+1]} + 19\boldsymbol{f}^{[n]} - 5\boldsymbol{f}^{[n-1]} + \boldsymbol{f}^{[n-2]}\right)$$

avec $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$.

Q. 1 *Ecrire la fonction algorithmique* REDPC4Vec *permettant de résoudre un problème de Cauchy (vectoriel) par une méthode de prédiction-correction utilisant ces deux schémas. On minimisera le nombre d'appel à la fonction $\boldsymbol{f}$ dans la boucle principale.*

Correction 3.6.2

R. 1 On utilise le schéma de Runge-Kutta d'ordre 4 pour initialiser les 4 premières valeurs.
Ensuite on utilise comme prédicteur le schéma explicite et comme correcteur le schéma implicite. Le principe est donc

- Calcul à l'aide du prédicteur :

$$\hat{\boldsymbol{y}}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(55\boldsymbol{f}^{[n]} - 59\boldsymbol{f}^{[n-1]} + 37\boldsymbol{f}^{[n-2]} - 9\boldsymbol{f}^{[n-3]}\right)$$

- Calcul à l'aide du correcteur :

$$\begin{aligned} \hat{\boldsymbol{f}}^{[n+1]} &= \boldsymbol{f}(t^{n+1}, \hat{\boldsymbol{y}}^{[n+1]}) \\ \boldsymbol{y}^{[n+1]} &= \boldsymbol{y}^{[n]} + \tfrac{h}{24}\left(9\hat{\boldsymbol{f}}^{[n+1]} + 19\boldsymbol{f}^{[n]} - 5\boldsymbol{f}^{[n-1]} + \boldsymbol{f}^{[n-2]}\right) \end{aligned}$$

L'algorithme de la fonction REDPC4Vec s'écrit alors :

Algorithme 3.17 Fonction REDPC4Vec : résolution d'un problème de Cauchy par prédiction-correction (Adams-Bashforth/Adams-Moulton) d'ordre 4

Données : $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: un vecteur de $\mathbb{R}^d$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1},\ \forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: matrice réelle de dimension $d \times (N+1)$, $\boldsymbol{Y}(:,n) = \boldsymbol{y}^{(n-1)},\ \forall n \in [\![1, N+1]\!]$

Fonction $[\boldsymbol{t}, \boldsymbol{Y}] \leftarrow$ REDPC4Vec$(\ f, t^0, T, \boldsymbol{y}^0, N\ )$
  $\boldsymbol{t} \leftarrow DisReg(t^0, t^0+T, N)$
  $h \leftarrow T/N$
  $[\boldsymbol{t}_{ini}, \boldsymbol{Y}_{ini}] \leftarrow$ REDRK4Vec$(f, t^0, 3*h, \boldsymbol{y}^0, 3)$
  Pour $n \leftarrow 1$ **à** 4 **faire**
    $\boldsymbol{Y}(:,n) \leftarrow \boldsymbol{Y}_{ini}(:,n)$
  Fin
  $\boldsymbol{k}_1 \leftarrow \boldsymbol{f}(\boldsymbol{t}(3), \boldsymbol{Y}(:,3))$
  $\boldsymbol{k}_2 \leftarrow \boldsymbol{f}(\boldsymbol{t}(2), \boldsymbol{Y}(:,2))$
  $\boldsymbol{k}_3 \leftarrow \boldsymbol{f}(\boldsymbol{t}(1), \boldsymbol{Y}(:,1))$
  Pour $n \leftarrow 4$ **à** N **faire**
    $\boldsymbol{k}_0 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n), \boldsymbol{Y}(:,n))$
    $\hat{\boldsymbol{Y}} \leftarrow \boldsymbol{Y}(:,n) + (h/24) * (55 * \boldsymbol{k}_0 - 59 * \boldsymbol{k}_1 + 37 * \boldsymbol{k}_2 - 9 * \boldsymbol{k}_3)$
    $\hat{\boldsymbol{F}} \leftarrow \boldsymbol{f}(\boldsymbol{t}(n+1), \hat{\boldsymbol{Y}})$
    $\boldsymbol{Y}(:,n+1) \leftarrow \boldsymbol{Y}(:,n) + (h/24) * (9 * \hat{\boldsymbol{F}} + 19 * \boldsymbol{k}_0 - 5 * \boldsymbol{k}_1 + \boldsymbol{k}_2)$
    $\boldsymbol{k}_3 \leftarrow \boldsymbol{k}_2$
    $\boldsymbol{k}_2 \leftarrow \boldsymbol{k}_1$
    $\boldsymbol{k}_1 \leftarrow \boldsymbol{k}_0$
  Fin
Fin

Fin correction

Comparaison avec Runge-Kutta

L'intérêt d'une méthode de résolution numérique d'équations différentielles se mesure principalement suivant deux critères:

- son coût pour obtenir une précision donnée (c'est à dire le nombre d'évaluations de fonctions par étapes).
- sa stabilité.

La caractéristique des méthodes de Runge-Kutta est que le pas est assez facile à adapter, la mise en oeuvre informatique plus aisée. Mais pour des méthodes du même ordre de consistance, les méthodes de Runge-Kutta exigent plus d'évaluations de fonctions. Quand on sait que la solution de l'équation n'est pas sujette à de brusques variations de la dérivée, on prendra une méthode de type prédicteur-correcteur mais, si le pas doit être adapté plus précisément, on préfèrera une méthode de Runge-Kutta.

3.7 Applications

3.7.1 Modèle de Lorentz

On rappelle le modèle de Lorentz (3.7.1) (voir section 3.1.1)

$$\left\{\begin{array}{lcl} x'(t) &=& -\sigma x(t)+\sigma y(t)\\ y'(t) &=& -x(t)z(t)+\rho x(t)-y(t)\\ z'(t) &=& x(t)y(t)-\beta z(t)\end{array}\right.$$

avec $\sigma = 10$, $\rho = 28$, $\beta = 8/3$ et les données initales $x(0) = -8$, $y(0) = 8$ et $z(0) = \rho - 1$. On souhaite représenter l'attracteur étrange de Lorentz pour $t \in [0, 100]$ (*Papillon*, Figure 3.2) où la résolution du système d'EDO se fera à l'aide d'une des fonctions que nous avons écrite et résolvant un problème de Cauchy vectoriel (par exemple la fonction REDRK4Vec donnée par l'Algorithme 3.15, page 74)
La première chose est d'écrire (sur le papier) le problème de Cauchy correspondant à notre problème. Ici on a un système de 3 EDO d'ordre 1 et donc $m = 3$ dans la Définition 3.2.1. On prend $t^0 = 0$ et $T = 100$. Soit $\boldsymbol{y}$ la fonction vectorielle

$$\begin{array}{rcccl}\boldsymbol{y} & : & [0,100] & \longrightarrow & \mathbb{R}^3\\ & & t & \longmapsto & \boldsymbol{y}(t)=\begin{pmatrix}y_1(t)\\ y_2(t)\\ y_3(t)\end{pmatrix}\end{array}$$

donnée par (lien avec le système d'EDO de Lorentz)

$$\boldsymbol{y}(t)=\begin{pmatrix}x(t)\\ y(t)\\ z(t)\end{pmatrix}$$

La fonction de Cauchy associé au modèle de Lorentz est donc

$$\begin{array}{rcccl}\boldsymbol{f} & : & [0,100]\times\mathbb{R}^3 & \longrightarrow & \mathbb{R}^3\\ & & (t,\square) & \longmapsto & \boldsymbol{f}(t,\square)=\begin{pmatrix}-\sigma\square_1+\sigma\square_2\\ -\square_1\square_3+\rho\square_1-\square_2\\ \square_1\square_2-\beta\square_3\end{pmatrix}\end{array}$$

ou de manière plus classique

$$\begin{array}{rcccl}\boldsymbol{f} & : & [0,100]\times\mathbb{R}^3 & \longrightarrow & \mathbb{R}^3\\ & & (t,\boldsymbol{z}) & \longmapsto & \boldsymbol{f}(t,\boldsymbol{z})=\begin{pmatrix}-\sigma z_1+\sigma z_2\\ -z_1z_3+\rho z_1-z_2\\ z_1z_2-\beta z_3\end{pmatrix}\end{array}$$

Le problème de Cauchy est donc de chercher la fonction vectorielle $\boldsymbol{y}$ solution de

$$\begin{array}{rcl}\boldsymbol{y}'(t) &=& \boldsymbol{f}(t,\boldsymbol{y}(t)),\quad \forall t\in[t^0,t^0+T]\\ \boldsymbol{y}(t^0) &=& \boldsymbol{y}^{[0]}=\begin{pmatrix}-8\\ 8\\ \rho-1\end{pmatrix}\in\mathbb{R}^m.\end{array}$$

```
1 function w=fLorentz(t,z,beta,rho,sigma)
2   % Function de Cauchy associee au modele de Lorentz
3   %   beta, rho et sigma parametres physiques
4   w=[-sigma*z(1)+sigma*z(2); -z(1)*z(3)+rho*z(1)-z(2); z(1)*z(2)-beta*z(3)];
5 end
```

Listing 3.1: fonction de Cauchy : fichier `fLorentz.m`

```
1 sigma=10;rho=28;beta=8/3;
2 fCauchy=@(t,z) fLorentz(t,z,beta,rho,sigma);
3
```

```
y0=[-8;8;rho-1];
[t,Y]=redRK4Vec(fCauchy,0,100,y0,10000);

opts={'interpreter','latex','Fontsize',12};
figure(1)
plot3(Y(1,:),Y(2,:),Y(3,:))
legend('$(x(t),y(t),z(t))$','Location','northeast',opts{:})
xlabel('$x$',opts{:})
ylabel('$y$',opts{:})
zlabel('$z$',opts{:})
view(-5,10)

figure(2)
subplot(3,1,1)
plot(t,Y(1,:));
xlabel('$t$',opts{:}),ylabel('$x(t)$',opts{:})
subplot(3,1,2)
plot(t,Y(2,:));
xlabel('$t$',opts{:}),ylabel('$y(t)$',opts{:})
subplot(3,1,3)
plot(t,Y(3,:));
xlabel('$t$',opts{:}),ylabel('$z(t)$',opts{:})
```

Listing 3.2: Résolution du modèle de Lorentz : script `prgLorentz.m`

On représente en Figure 3.17 les deux graphiques générés par le programme `prgLorentz.m` sous Matlab

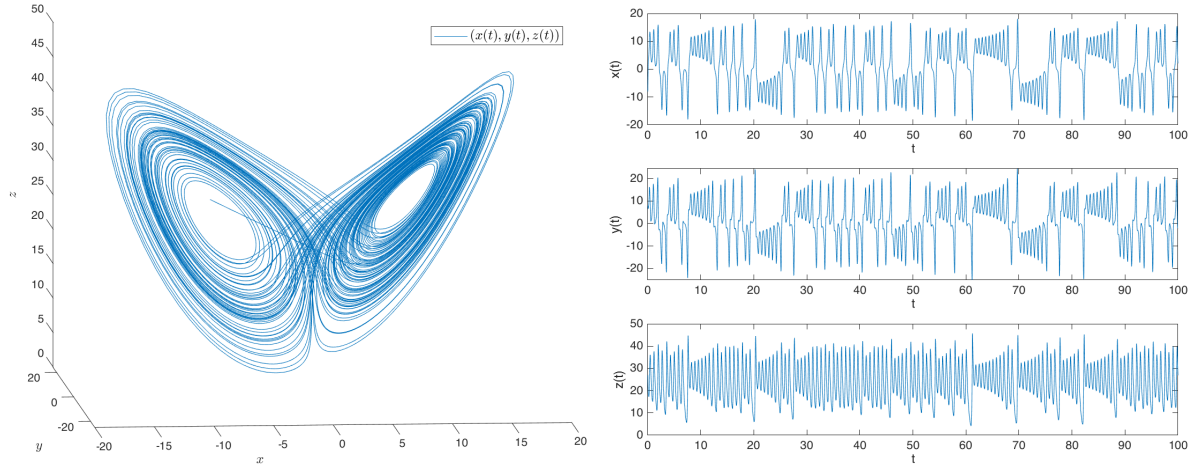
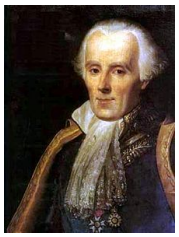


Figure 3.17: Attracteur étrange de Lorentz (gauche) et les trois courbes $x(t)$, $y(t)$ et $(z(t)$ (droite)

Chapitre 4

Introduction à la résolution d'E.D.P.



Pierre-Simon Laplace 1749-1827, mathématicien, astronome, physicien et homme politique français

Siméon Denis Poisson 1781-1840, mathématicien, géomètre et physicien français

Jean-Baptiste Joseph Fourier 1768-1830, mathématicien et physicien français

Jean-Baptiste le Rond d'Alembert 1717-1783, mathématicien, mécanicien, physicien et phylosophe français

Leonhard Euler 1707-1783, mathématicien, physicien, astronome et ingénieur suisse

Daniel Bernoulli 1700-1783, mathématicien et physicien suisse

4.1 Exemples d'EDP

4.1.1 Equation de Laplace et équation de Poisson

L'équation aux dérivées partielles du second ordre suivante :

$$-\Delta u = f, \tag{4.1}$$

où Δ est l'opérateur laplacien*, est appellée **équation de Laplace** si $f \equiv 0$ et **équation de Poisson** sinon. Introduite pour les besoins de la mécanique newtonienne, ces équations apparaissent aussi en astronomie, électrostatique, mécanique quantique, mécanique des fluides, propagation de la chaleur, ...

Sur un domaine borné $\Omega \subset \mathbb{R}^n$ et de frontière suffisament régulière, en imposant des **conditions aux limites** sur le bord du domaine noté $\partial\Omega$, on peut aboutir à un **problème bien posé** : la solution existe et est unique.

Les condition aux limites peuvent être de type

- **Dirichlet** si on impose, sur une partie de $\partial\Omega$,

$$u = g, \quad \text{sur } \Gamma_D \subset \partial\Omega. \tag{4.2}$$

- **Neumann** si on impose sur une partie de $\partial\Omega$,

$$\frac{\partial u}{\partial n} = g, \quad \text{sur } \Gamma_N \subset \partial\Omega. \tag{4.3}$$

  où $\frac{\partial u}{\partial n} = \langle \mathbf{grad}\, u, \boldsymbol{n} \rangle$ avec $\boldsymbol{n}$ normale exterieure unitaire à Ω

- **Robin** si on impose sur une partie de $\partial\Omega$

$$\frac{\partial u}{\partial n} + \alpha u = g, \quad \text{sur } \Gamma_R \subset \partial\Omega. \tag{4.4}$$

Comme première application, on cherche à déterminer le potentiel u (exprimé en volt) dans un "condensateur" 2D décrit par le problème suivant :

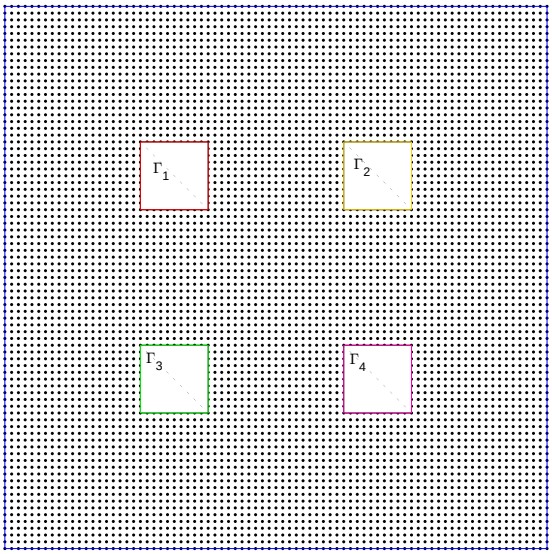
Problème de condensateur en 2D

Find $u : \Omega \longrightarrow \mathbb{R}$ such that

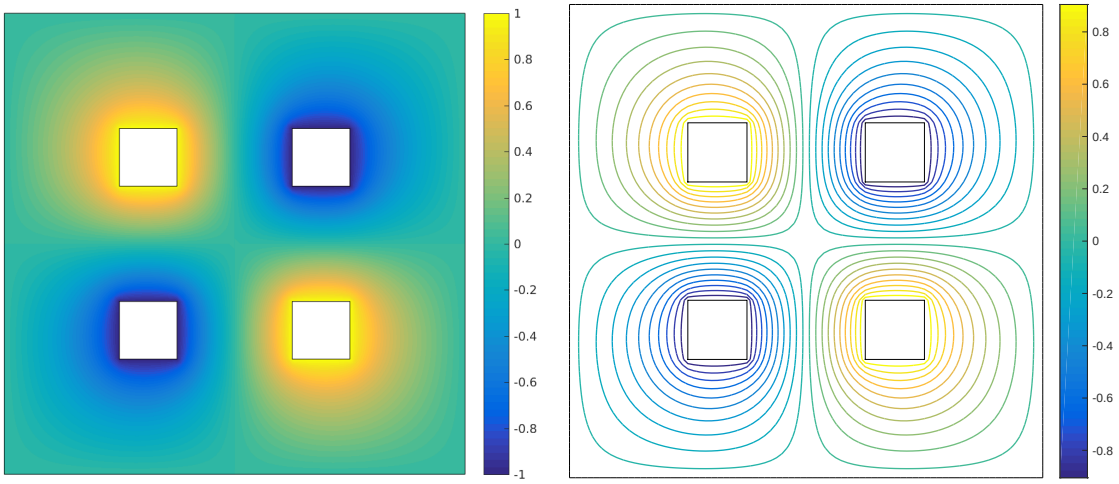
$$\begin{aligned} -\Delta u &= 0 \ \text{ in } \Omega \subset \mathbb{R}^2, \\ u &= 0 \ \text{ on } \Gamma_{10}, \\ u &= -1 \ \text{ on } \Gamma_2 \cup \Gamma 3, \\ u &= 1 \ \text{ on } \Gamma_1 \cup \Gamma 4, \end{aligned}$$

où Ω et ses frontières sont représentés en Figure 4.2 (haut). Une solution numérique de ce problème utilisant un schéma aux différences finies d'ordre 2 est représentée en Figure 4.2 (bas).

*Dans $\mathbb{R}^n$, on a $\Delta u = \frac{\partial^2 u}{\partial x_1^2} + \ldots + \frac{\partial^2 u}{\partial x_n^2}$.



Frontières du domaine Ω



Potentiel u Lignes de niveaux du potentiel u

Figure 4.2: Problème de condensateur en 2D

Comme second problème, on cherche à construire un champ de vitesses $\boldsymbol{V}$ obtenu à partir d'un potentiel u, c'est à dire tel que $\boldsymbol{V} = \boldsymbol{\nabla} u$[†] avec u solution du problème suivant

Champ de vitesses en 2D

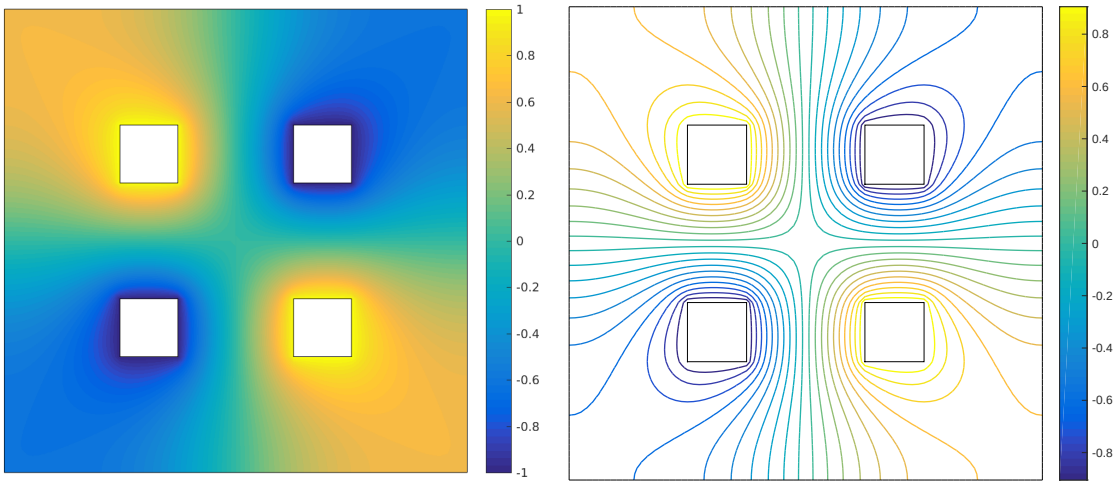
Trouver $u : \Omega \longrightarrow \mathbb{R}$ tel que

$$\begin{aligned} -\Delta u &= 0 \text{ dans } \Omega \subset \mathbb{R}^2, \\ u &= -1 \text{ sur } \Gamma_2 \cup \Gamma 3, \\ u &= 1 \text{ sur } \Gamma_1 \cup \Gamma 4, \\ \frac{\partial u}{\partial n} &= 0 \text{ sur } \Gamma_{10}. \end{aligned}$$

Le champ de vitesses $\boldsymbol{V}$ associé est donné par $\boldsymbol{V} = \boldsymbol{\nabla} u$

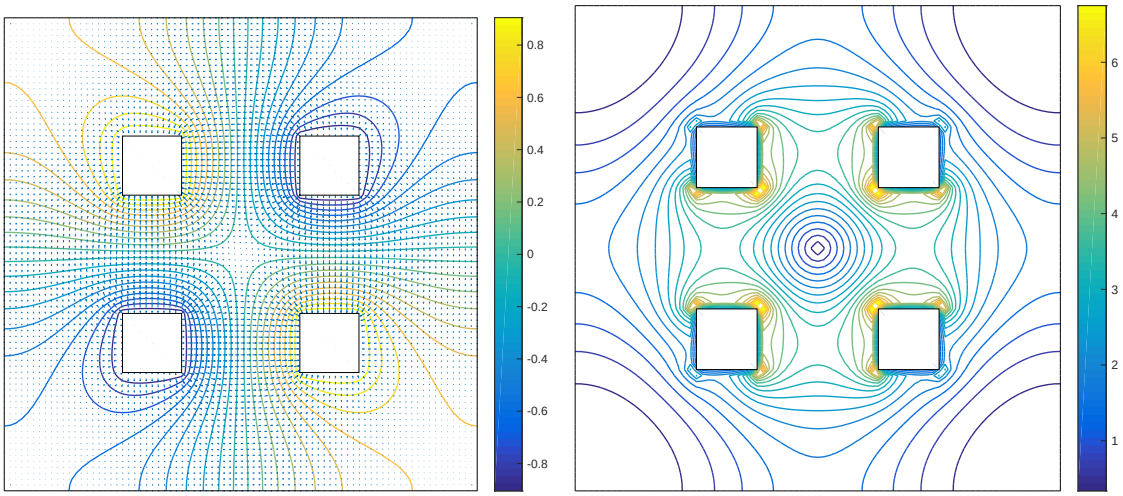
où Ω et ses frontières sont identiques à l'exemple précédant (voir Figure 4.2 haut). Une solution numérique de ce problème utilisant un schéma aux différences finies d'ordre 2 est représentée en Figure 4.3. Le champ de vecteurs qui en découle est visible en Figure 4.4.

[†]On note $\boldsymbol{\nabla} u$ le gradient de la fonction u.



Potentiel u — Lignes de niveaux du potentiel u

Figure 4.3: Potentiel u en 2D



Lignes de niveau du potentiel u et champ de vecteurs $\boldsymbol{V}$ — Lignes de niveau de $\|\boldsymbol{V}\|$

Figure 4.4: champ de vecteurs $\boldsymbol{V} = \mathbf{grad}\, u$ en 2D

Comme dernier problème, on prend

Problème en dimension n

Find $u : \Omega \longrightarrow \mathbb{R}$ such that

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \subset \mathbb{R}^n, \\ \frac{\partial u}{\partial n} &= g \quad \text{sur } \partial\Omega. \end{aligned}$$

où Ω est un domaine borné de $\mathbb{R}^n$.
Ce problème est **mal posé** car il n'y a pas unicité de la solution. En effet, si v est solution du problème alors v + constante est aussi solution.
Pour obtenir l'unicité, il faudrait imposer, sur une partie non vide du bord, une condition de Dirichlet ou une condition de Robin

4.1.2 Equation de convection-diffusion stationnaire

A faire !

4.1.3 Equation de la chaleur

‡

L'équation de la chaleur est une équation aux dérivées partielles décrivant le phénomène physique de conduction thermique. Elle a été initialement introduite par Jean Baptiste Joseph Fourier (voir son livre *Théorie analytique de la chaleur* paru en 1822).
Soit Ω un domaine de $\mathbb{R}^d$ de frontière $\partial\Omega$ et $u(t,\boldsymbol{x})$ un champ de température sur ce domaine. En présence d'une source thermique dans le domaine, et en l'absence de transport de chaleur (convection), l'équation de la chaleur s'écrit :

$$\forall \boldsymbol{x} \in \Omega,\ \forall t \in [0,T], \qquad \frac{\partial u}{\partial t}(t,\boldsymbol{x}) - D\Delta u(t,\boldsymbol{x}) = \frac{f(t,\boldsymbol{x})}{\rho c} \tag{4.5}$$

où Δu est le laplacien (en espace), $\Delta u = \frac{\partial^2 u}{\partial x_1^2} + \ldots + \frac{\partial^2 u}{\partial x_d^2}$, et

- D est le coefficient de diffusivité thermique (en m^2/s),
- f une éventuelle production volumique de chaleur (en W/m^3),
- ρ est la masse volumique du matériau (en kg/m^3),
- c la chaleur spécifique massique du matériau (en $J/kg/K$).

Pour que le problème soit mathématiquement bien posé, il faut en général spécifier :

- une **condition initiale**
$$\forall \boldsymbol{x} \in \Omega, \quad u(0,\boldsymbol{x}) = u_0(\boldsymbol{x}) \tag{4.6}$$
- des **conditions aux limites** sur le bord du domaine, par exemple de type
  - **Dirichlet** :
  $$\forall \boldsymbol{x} \in \Gamma_D \subset \partial\Omega,\ \forall t \in [0,T] \quad u(t,\boldsymbol{x}) = g_D(t,\boldsymbol{x})$$
  - **Neumann** :
  $$\forall \boldsymbol{x} \in \Gamma_N \subset \partial\Omega,\ \forall t \in [0,T] \quad D\frac{\partial u}{\partial n}(t,\boldsymbol{x}) = g_N(t,\boldsymbol{x})$$
  - **Robin** :
  $$\forall \boldsymbol{x} \in \Gamma_R \subset \partial\Omega,\ \forall t \in [0,T] \quad D\frac{\partial u}{\partial n}(t,\boldsymbol{x}) + \alpha u(t,\boldsymbol{x}) = g_N(t,\boldsymbol{x})$$

Problème de chaleur en 2D

Soit $\Omega \subset \mathbb{R}^2$ un domaine connexe borné. On veut déterminer $u : [0,T] \times \Omega \longrightarrow \mathbb{R}$ such that

$$\begin{aligned}
\frac{\partial u}{\partial t} - D\Delta u &= 0 \ \text{ dans } [0,T] \times \Omega, \\
u(0,\boldsymbol{x}) &= 20\ \forall \boldsymbol{x} \in \Omega, \\
\frac{\partial u}{\partial n} &= 0 \ \text{ sur } \Gamma_{10}, \\
u &= g_1 \ \text{ sur } \Gamma_2 \cup \Gamma 3, \\
u &= g_2 \ \text{ sur } \Gamma_1 \cup \Gamma 4,
\end{aligned}$$

‡Texte en grande partie tiré de wikipedia

où Ω (cotés de $20cm$) et ses frontières sont représentés en Figure 4.2 (haut) et

- $D = 98.8 \times 10^{-6}$ pour l'aluminium ou $D = 23.9 \times 10^{-6}$ pour le plomb,
- $\forall \boldsymbol{x} \in \Gamma_2 \cup \Gamma 3$, $g_1(t, \boldsymbol{x}) = (20 + 40t)$ si $t <= 1$ et $g_1(t, \boldsymbol{x}) = 60$ sinon,
- $\forall \boldsymbol{x} \in \Gamma_1 \cup \Gamma 4$, $g_2(t, \boldsymbol{x}) = (20 + 80t)$ si $t <= 1$ et $g_2(t, \boldsymbol{x}) = 100$ sinon.

Deux solutions numériques de ce problème utilisant un schéma aux différences finies d'ordre 2 est représentée en Figure 4.5 l'une pour l'aluminium (figures de gauche) et l'autre pour le plomb (figure de droite).

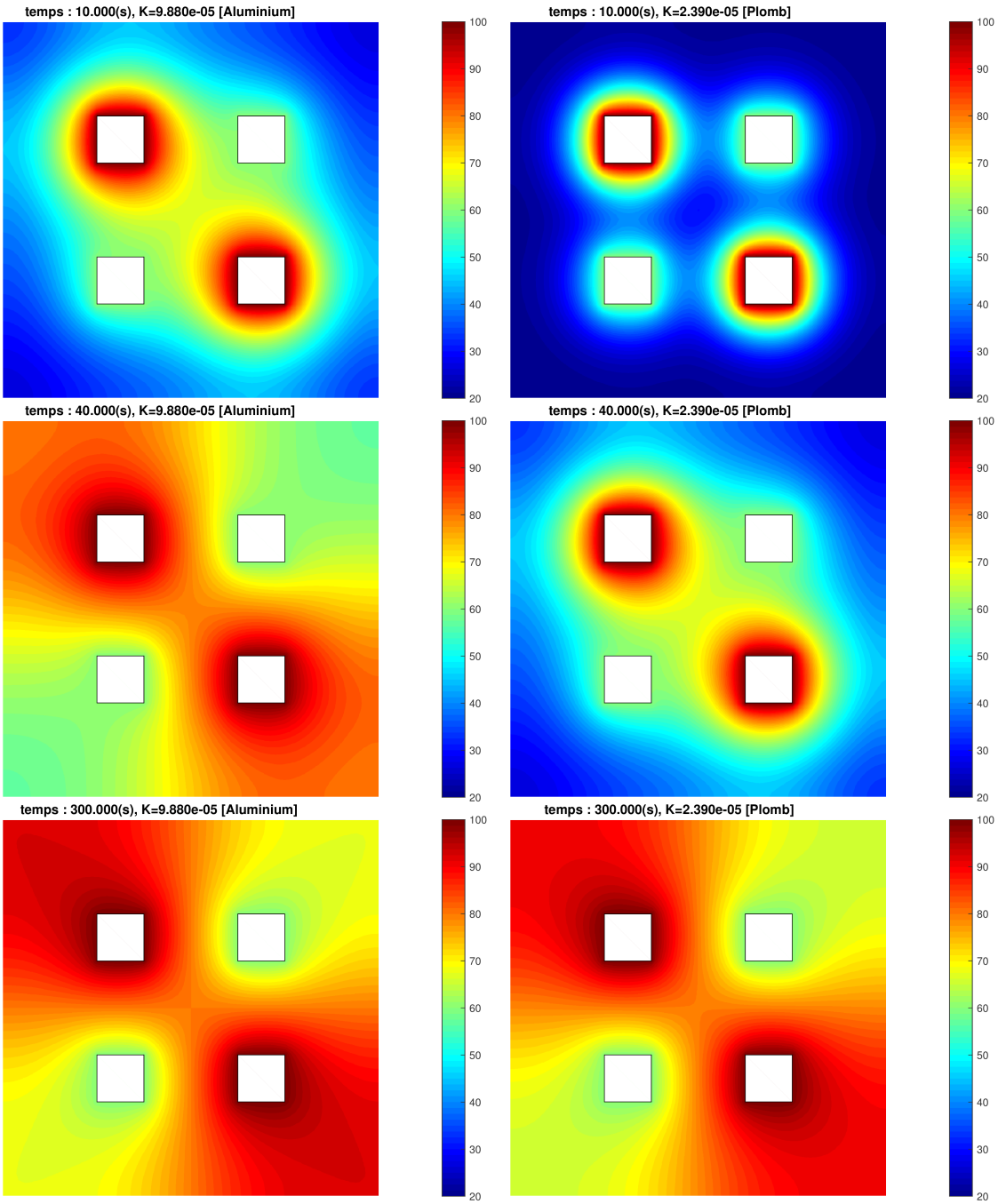


Figure 4.5: Equation de la chaleur : solutions aux temps $t = 10(s)$ (haut), $t = 40(s)$ (milieux), $t = 40(s)$ (bas), pour l'aluminium (gauche) et le plomb (droite)

4.1.4 Equation des ondes

L'équation des ondes est une équation aux dérivées partielles du second ordre décrivant les phénomènes de propagation d'ondes comme les ondes sonores, les ondes lumineuses ou les vagues... Cette équation appa-

rait dans de nombreux domaines comme par exemple l'acoustique, l'electromagnetisme ou la dynamiques des fluides.
Historiquement, le problème d'une corde vibrante comme celle d'un instrument de musique a été étudié par J. d'Alembert, L. Euler, D. Bernoulli et J. Lagrange. En 1746, d'Alembert a établi l'équation des ondes unidimensionelles. Dans les dix ans qui suivent, Euler a étendu ses résultats à l'équation des ondes tridimensionnelles.
L'équaton des ondes s'écrit Soit Ω un domaine de $\mathbb{R}^d$ de frontière $\partial\Omega$ et $u(x,t)$ une solution de l'équation des ondes suivante

$$\forall \boldsymbol{x} \in \Omega,\ \forall t \in [0,T], \qquad \frac{\partial^2 u}{\partial t^2}(t,\boldsymbol{x}) - c^2 \Delta u(t,\boldsymbol{x}) = 0 \tag{4.7}$$

où $c > 0$ correspond à la vitesse de propagation de l'onde dans le milieu considéré (par exemple, dans le cas d'une onde sonore c est la célérité du son : $343m/s$ dans l'air à 20^o)
Pour que le problème soit mathématiquement bien posé, il faut imposer des **conditions initiales** :

- **position initiale**

$$\forall \boldsymbol{x} \in \Omega, \quad u(0,\boldsymbol{x}) = u_0(\boldsymbol{x}) \tag{4.8}$$

- **vitesse initiale**

$$\forall \boldsymbol{x} \in \Omega, \quad \frac{\partial u}{\partial t}(0,\boldsymbol{x}) = v_0(\boldsymbol{x}) \tag{4.9}$$

Dans le cas d'un domaine borné $\Omega \subset \mathbb{R}^n$, il faut imposer des **conditions aux limites**, par exemple de type

- **Dirichlet** :

$$\forall \boldsymbol{x} \in \Gamma_D \subset \partial\Omega,\ \forall t \in [0,T] \quad u(t,\boldsymbol{x}) = g_D(t,\boldsymbol{x})$$

- **Neumann** :

$$\forall \boldsymbol{x} \in \Gamma_N \subset \partial\Omega,\ \forall t \in [0,T] \quad c^2 \frac{\partial u}{\partial n}(t,\boldsymbol{x}) = g_N(t,\boldsymbol{x})$$

- **Robin** :

$$\forall \boldsymbol{x} \in \Gamma_R \subset \partial\Omega,\ \forall t \in [0,T] \quad c^2 \frac{\partial u}{\partial n}(t,\boldsymbol{x}) + \alpha u(t,\boldsymbol{x}) = g_N(t,\boldsymbol{x})$$

4.2 Définitions

Une **équation eux dérivées partielles**, notée EDP, (*partial differential equation* ou PDE en anglais) fait intervenir plusieurs variables indépendantes (souvent en temps et espace), ainsi que les dérivées partielles de la fonction recherchée.
Une équation aux dérivées partielles pour la fonction $u(x_1,\dots,x_n)$ peut s'écrire sous la forme

$$\mathcal{F}(x_1,\dots,x_n,u,\frac{\partial u}{\partial x_1},\dots,\frac{\partial u}{\partial x_n},\frac{\partial^2 u}{\partial x_1^2},\dots,\frac{\partial^2 u}{\partial x_1 \partial x_n},\dots)$$

Si $\mathcal{F}$ est une fonction linéaire en u et ses dérivées, alors l'EDP est dites linéaire.
Par exemple, l'équation de convection (appelée aussi advection) est régie par

$$\frac{\partial C}{\partial t} + \alpha \frac{\partial C}{\partial x} = 0 \tag{4.10}$$

La fonction recherchée est C et les variables indépendantes sont le temps t et l'espace x. La grandeur α (homogène à une vitesse) peut être fonction de t, x et C.
L'**ordre d'une EDP** est l'ordre le plus élevé parmi toutes les dérivées partielles de l'EDP. Dans l'exemple précédant, l'EDP (4.10) est d'ordre 1. Par contre, l'EDP suivante

$$\frac{\partial C}{\partial t} + \alpha \frac{\partial C}{\partial x} - \beta \frac{\partial^2 C}{\partial x^2} = 0 \tag{4.11}$$

est une EDP d'ordre 2.
On dit qu'une EDP est **lineaire** si elle ne fait intervenir que des combinaisons linéaires des dérivées partielles de la variable dépendante. On dit qu'une EDP est **quasi-linéaire** si elle est linéaire par rapport aux dérivées d'ordre le plus élevé. Par exemple, l'EDP :

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} = d$$

est quasi-linéaire si a, b, c et d dépendent de $x, y, u, \frac{\partial u}{\partial x}$ et $\frac{\partial u}{\partial y}$.

4.3 Méthodes de résolution numérique d'EDP

A faire

Il existe trois grandes classes de méthodes déterministes pour la résolution numérique d'EDP :

- **méthode des différences finies** :
  discrétisation des opérateurs de dérivation/différentiation, en chacun des points d'une grille représentant le domaine d'étude.
- **méthode des éléments finis** :
- **méthode des volumes finis** :

4.4 Opérateurs aux différences finies

4.4.1 Dimension 1

Soit $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction suffisament régulière.
Soient $h > 0$ et $x \in \mathbb{R}$, on défini les opérateurs aux différences finies suivant

$$(D_h^+\varphi)(x) = \frac{1}{h}\left(\varphi(x+h) - \varphi(x)\right) \tag{4.12}$$

$$(D_h^-\varphi)(x) = \frac{1}{h}\left(\varphi(x) - \varphi(x-h)\right) \tag{4.13}$$

$$(D_h^0\varphi)(x) = \frac{1}{2h}\left(\varphi(x+h) - \varphi(x-h)\right) \tag{4.14}$$

Les opérateurs D_h^0, D_h^+ et D_h^- s'appellent respectivement **opérateur (aux différences finies) centré**, **opérateur (aux différences finies) progressif/décentré avancé** et **opérateur (aux différences finies) rétrograde/décentré retardé**.

Définition 4.4.1. *Soit $h > 0$. On dit qu'un opérateur aux différences finies D_h est une approximation consistante d'ordre p de $\frac{d^k\varphi}{dx^k}$ si pour tout $\varphi : [a,b] \longrightarrow \mathbb{R}$ suffisament régulière on a*

$$\max_{x\in[a,b]} \left| (D_h\varphi)(x) - \frac{d^k\varphi}{dx^k}(x) \right| \leqslant Ch^p, \tag{4.15}$$

où C est une constante indépendante de h.

Exercice 4.4.1

Soient $h >$ et les trois opérateurs aux différences finies suivant

$$(D_h^+\varphi)(x) = \frac{1}{h}\left(\varphi(x+h) - \varphi(x)\right)$$

$$(D_h^-\varphi)(x) = \frac{1}{h}\left(\varphi(x) - \varphi(x-h)\right)$$

$$(D_h^0\varphi)(x) = \frac{1}{2h}\left(\varphi(x+h) - \varphi(x-h)\right)$$

Q. 1 *Montrer que ces trois opérateurs sont linéaires (i.e. $\forall(\lambda,\mu) \in \mathbb{R}^2$, $\forall\varphi : \mathbb{R} \longrightarrow \mathbb{R}$, $\forall\psi : \mathbb{R} \longrightarrow \mathbb{R}$, $D_h(\lambda\varphi + \mu\psi) = \lambda D_h\varphi + \mu D_h\psi$.)*

Q. 2 *On suppose que $\varphi \in \mathcal{C}^k([a,b];\mathbb{R})$ avec $k \geqslant 2$. Montrer que les opérateurs D_h^+ et D_h^- sont des approximations consistantes d'ordre 1 de $\frac{d\varphi}{dx}$.*

Q. 3 *On suppose que $\varphi \in \mathcal{C}^k([a,b];\mathbb{R})$ avec $k \geqslant 3$. Montrer que l'opérateur D_h^0 est une approximation consistante d'ordre 2 de $\frac{d\varphi}{dx}$.*

Correction 4.4.1

R. 1

Soient $(\lambda,\mu)\in\mathbb{R}^2$, $\varphi:\mathbb{R}\longrightarrow\mathbb{R}$ et $\psi:\mathbb{R}\longrightarrow\mathbb{R}$. Montrer qu'un opérateur D_h est linéaire consiste à montrer que

$$D_h(\lambda\varphi+\mu\psi)=\lambda D_h\varphi+\mu D_h\psi$$

or ceci est équivalent à montrer que

$$(D_h(\lambda\varphi+\mu\psi))(x)=\lambda(D_h\varphi)(x)+\mu(D_h\psi)(x),\ \forall x\in\mathbb{R}$$

- Montrons que D_h^+ est linéaire. On a $\forall x\in\mathbb{R}$

$$\begin{aligned}(D_h^+(\lambda\varphi+\mu\psi))(x) &= \frac{(\lambda\varphi+\mu\psi)(x+h)-(\lambda\varphi+\mu\psi)(x)}{h}\\ &= \frac{(\lambda\varphi(x+h)+\mu\psi(x+h))-(\lambda\varphi(x)+\mu\psi(x))}{h}\\ &= \lambda\frac{\varphi(x+h)-\varphi(x)}{h}+\mu\frac{\psi(x+h)-\psi(x)}{h}\\ &= \lambda(D_h^+\varphi)(x)+\mu(D_h^+\psi)(x).\end{aligned}$$

- Montrons que D_h^- est linéaire. On a $\forall x\in\mathbb{R}$

$$\begin{aligned}(D_h^-(\lambda\varphi+\mu\psi))(x) &= \frac{(\lambda\varphi+\mu\psi)(x)-(\lambda\varphi+\mu\psi)(x-h)}{h}\\ &= \frac{(\lambda\varphi(x)+\mu\psi(x+h))-(\lambda\varphi(x-h)+\mu\psi(x-h))}{h}\\ &= \lambda\frac{\varphi(x)-\varphi(x-h)}{h}+\mu\frac{\psi(x)-\psi(x-h)}{h}\\ &= \lambda(D_h^-\varphi)(x)+\mu(D_h^-\psi)(x).\end{aligned}$$

- Montrons que D_h^0 est linéaire. On a $\forall x\in\mathbb{R}$

$$\begin{aligned}(D_h^0(\lambda\varphi+\mu\psi))(x) &= \frac{(\lambda\varphi+\mu\psi)(x+h)-(\lambda\varphi+\mu\psi)(x-h)}{2h}\\ &= \frac{(\lambda\varphi(x+h)+\mu\psi(x+h))-(\lambda\varphi(x-h)+\mu\psi(x-h))}{2h}\\ &= \lambda\frac{\varphi(x+h)-\varphi(x-h)}{h}+\mu\frac{\psi(x+h)-\psi(x-h)}{2h}\\ &= \lambda(D_h^0\varphi)(x)+\mu(D_h^0\psi)(x).\end{aligned}$$

R. 2

- A l'aide d'un développement de Taylor à l'ordre 2, il existe $\xi^+\in]x,x+h[$ tel que

$$\varphi(x+h)=\varphi(x)+h\frac{d\varphi}{dx}(x)+\frac{h^2}{2!}\frac{d^2\varphi}{dx^2}(\xi^+)$$

On en déduit alors

$$\frac{d\varphi}{dx}(x)=\frac{\varphi(x+h)-\varphi(x)}{h}-\frac{h}{2}\frac{d^2\varphi}{dx^2}(\xi^+)$$

Ce qui donne

$$\frac{d\varphi}{dx}(x)-(D_h^+(\varphi))(x)=-\frac{h}{2}\frac{d^2\varphi}{dx^2}(\xi^+)$$

On obtient donc

$$\begin{aligned}\max_{x\in[a,b]}\left|\frac{d\varphi}{dx}(x)-(D_h^+(\varphi))(x)\right| &\leqslant \frac{h}{2}\max_{\xi\in]x,x+h[}|\frac{d^2\varphi}{dx^2}(\xi)|\\ &\leqslant \frac{h}{2}\max_{x\in[a,b]}|\frac{d^2\varphi}{dx^2}(x)|=Ch.\end{aligned}$$

D'après la Définition 4.4.1, l'opérateur D_h^+ est une approximation consistante d'ordre 1 de φ'.

- A l'aide d'un développement de Taylor à l'ordre 2, il existe $\xi^- \in]x-h, x[$ tel que

$$\varphi(x-h) = \varphi(x) - h\frac{d\varphi}{dx}(x) + \frac{(-h)^2}{2!}\frac{d^2\varphi}{dx^2}(\xi^-)$$

On en déduit alors

$$\frac{d\varphi}{dx}(x) = \frac{\varphi(x)-\varphi(x-h)}{h} + \frac{h}{2}\frac{d^2\varphi}{dx^2}(\xi^-)$$

Ce qui donne

$$\frac{d\varphi}{dx}(x) - (D_h^-(\varphi))(x) = \frac{h}{2}\frac{d^2\varphi}{dx^2}(\xi^-)$$

On obtient donc

$$\begin{aligned}\max_{x\in[a,b]}\left|\frac{d\varphi}{dx}(x) - (D_h^-(\varphi))(x)\right| &\leqslant \frac{h}{2}\max_{\xi\in]x-h,x[}|\frac{d^2\varphi}{dx^2}(\xi)| \\ &\leqslant \frac{h}{2}\max_{x\in[a,b]}|\frac{d^2\varphi}{dx^2}(x)| = Ch.\end{aligned}$$

D'après la Définition 4.4.1, l'opérateur D_h^- est une approximation consistante d'ordre 1 de φ'.

R. 3

A l'aide de deux développements de Taylor à l'ordre 3, il existe $\xi^+ \in]x, x+h[$ tel que

$$\varphi(x+h) = \varphi(x) + h\frac{d\varphi}{dx}(x) + \frac{h^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{h^3}{3!}\frac{d^3\varphi}{dx^3}(\xi^+)$$

et il existe $\xi^- \in]x, x+h[$ tel que

$$\varphi(x-h) = \varphi(x) + (-h)\frac{d\varphi}{dx}(x) + \frac{(-h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(-h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi^-)$$

qui soutraits donnent

$$\varphi(x+h) - \varphi(x-h) = 2h\frac{d\varphi}{dx}(x) + \frac{h^3}{3!}(\frac{d^3\varphi}{dx^3}(\xi^+) + \frac{d^3\varphi}{dx^3}(\xi^-)).$$

On en déduit alors

$$\frac{d\varphi}{dx}(x) = \frac{\varphi(x+h)-\varphi(x-h)}{2h} - \frac{h^2}{12}(\frac{d^3\varphi}{dx^3}(\xi^+) + \frac{d^3\varphi}{dx^3}(\xi^-))$$

Ce qui donne

$$\frac{d\varphi}{dx}(x) - (D_h^0(\varphi))(x) = -\frac{h^2}{12}(\frac{d^3\varphi}{dx^3}(\xi^+) + \frac{d^3\varphi}{dx^3}(\xi^-)).$$

On obtient donc

$$\begin{aligned}\max_{x\in[a,b]}\left|\frac{d\varphi}{dx}(x) - (D_h^0(\varphi))(x)\right| &\leqslant \frac{h^2}{6}\max_{\xi\in]x-h,x+h[}|\frac{d^3\varphi}{dx^3}(\xi)| \\ &\leqslant \frac{h^2}{6}\max_{x\in[a,b]}|\frac{d^3\varphi}{dx^3}(x)| = Ch^2.\end{aligned}$$

D'après la Définition 4.4.1, l'opérateur D_h^0 est une approximation consistante d'ordre 2 de φ'.

Fin correction

On a donc démontré la proposition suivante

Proposition 4.4.1. *Si $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ est suffisament régulière, les opérateurs D_h^+ et D_h^- appliqués à φ sont des approximations consistantes d'ordre 1 de $\frac{d\varphi}{dx}$ et l'opérateur D_h^0 appliqué à φ est une*

approximation consistante d'ordre 2 *de* $\frac{d\varphi}{dx}$.

Proposition 4.4.2. *Soient* $\varphi \in \mathcal{C}^4([a,b];\mathbb{R})$. *On note* D_h^2 *l'opérateur défini, pour tout* $x \in]a,b[$ *et* $h>0$ *tels que* $x \pm h \in [a,b]$, *par*

$$(D_h^2\varphi)(x) \stackrel{\text{def}}{=} \frac{1}{h^2}\left[\varphi(x+h) - 2\varphi(x) + \varphi(x-h)\right]. \tag{4.16}$$

Alors $D_h^2\varphi$ *appliqué à* φ *est une approximation consistante d'ordre* 2 *de* $\frac{d^2\varphi}{dx^2}$.
De plus on a

$$D_h^2\varphi = D^0_{\frac{h}{2}}(D^0_{\frac{h}{2}}\varphi) = D_h^+(D_h^-\varphi) = D_h^-(D_h^+\varphi) \tag{4.17}$$

est une approximation consistante d'ordre 2 *de* $\frac{d^2\varphi}{dx^2}$.

Preuve. En utilisant deux développements de Taylor à l'ordre 4 en $x+h$ et $x-h$, il existe $\xi_i^+ \in]x, x+h[$ tel que

$$\varphi(x+h) = \varphi(x) + h\frac{d\varphi}{dx}(x) + \frac{h^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{h^3}{3!}\frac{d^3\varphi}{dx^3}(x) + \frac{h^4}{4!}\frac{d^4\varphi}{dx^4}(\xi^+)$$

et il existe $\xi^- \in]x, x+h[$ tel que

$$\varphi(x-h) = \varphi(x) + (-h)\frac{d\varphi}{dx}(x) + \frac{(-h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(-h)^3}{3!}\frac{d^3\varphi}{dx^3}(x) + \frac{(-h)^4}{4!}\frac{d^4\varphi}{dx^4}(\xi^-).$$

En les additionnant on a

$$\varphi(x+h) + \varphi(x-h) = 2\varphi(x) + h^2\frac{d^2\varphi}{dx^2}(x) + \frac{h^4}{4!}(\frac{d^4\varphi}{dx^4}(\xi^+) + \frac{d^4\varphi}{dx^4}(\xi^-))$$

c'est à dire

$$D_h^2\varphi(x) = \frac{d^2\varphi}{dx^2}(x) + \frac{h^2}{4!}(\frac{d^4\varphi}{dx^4}(\xi^+) + \frac{d^4\varphi}{dx^4}(\xi^-)).$$

On obtient alors

$$\left|D_h^2\varphi(x) - \frac{d^2\varphi}{dx^2}(x)\right| \leqslant \frac{h^2}{12}\sup_{\xi\in[x-h,x+h]}|\frac{d^4\varphi}{dx^4}(\xi)| \leqslant \frac{h^2}{12}\sup_{\xi\in[a,b]}|\frac{d^4\varphi}{dx^4}(\xi)|.$$

On a donc montré que $D_h^2\varphi$ est une approximation consistante d'ordre 2 de $\frac{d^2\varphi}{dx^2}$.
Pour démontrer (4.17), on calcule tout d'abord $D^0_{\frac{h}{2}}D^0_{\frac{h}{2}}\varphi$. On note $g \stackrel{\text{def}}{=} D^0_{\frac{h}{2}}\varphi$ c'est à dire

$$g(x) = D^0_{\frac{h}{2}}\varphi(x) = \frac{1}{h}\left(\varphi(x+\frac{h}{2}) - \varphi(x-\frac{h}{2})\right).$$

On a alors

$$\begin{aligned}
(D^0_{\frac{h}{2}}D^0_{\frac{h}{2}}\varphi)(x) &= (D^0_{\frac{h}{2}}g)(x) = \frac{1}{h}\left(g(x+\frac{h}{2}) - g(x-\frac{h}{2})\right)\\
&= \frac{1}{h}\left((D^0_{\frac{h}{2}}\varphi)(x+\frac{h}{2}) - (D^0_{\frac{h}{2}}\varphi)(x-\frac{h}{2})\right)\\
&= \frac{1}{h}\left(\frac{\varphi(x_+h) - \varphi(x)}{h} - \frac{\varphi(x) - \varphi(x-h)}{h}\right)\\
&= \frac{1}{h^2}(\varphi(x+h) - 2\varphi(x) + \varphi(x-h)).
\end{aligned}$$

De la même manière, en notant $g = D_h^-\varphi$ on obtient

$$\begin{aligned}
D_h^+D_h^-\varphi(x) &= D_h^+g(x) = \frac{g(x+h) - g(x)}{h} = \frac{1}{h}(D_h^-\varphi(x+h) - D_h^-\varphi(x))\\
&= \frac{1}{h}\left(\frac{\varphi(x+h) - \varphi(x)}{h} - \frac{\varphi(x) - \varphi(x-h)}{h}\right)\\
&= \frac{1}{h^2}(\varphi(x+h) - 2\varphi(x) + \varphi(x-h)).
\end{aligned}$$

Enfin avec $g = D_h^+\varphi$ on a

$$\begin{aligned} D_h^- D_h^+\varphi(x) &= D_h^- g(x) = \frac{g(x)-g(x-h)}{h} = \frac{1}{h}(D_h^+\varphi(x) - D_h^+\varphi(x-h)) \\ &= \frac{1}{h}\left(\frac{\varphi(x+h)-\varphi(x)}{h} - \frac{\varphi(x)-\varphi(x-h)}{h}\right) \\ &= \frac{1}{h^2}(\varphi(x+h) - 2\varphi(x) + \varphi(x-h)). \end{aligned}$$

□

4.4.2 Dimension $n \geqslant 1$

On commence par rappeler une extension du théorème de Taylor-Lagrange :

Proposition 4.4.3 ((admis)). *Soit U un ouvert non vide de $\mathbb{R}^n$ et f une application $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ avec $f \in \mathcal{C}^{r+1}(U)$. Soient $\boldsymbol{x} \in U$, $i \in [\![1,n]\!]$, et $h \in \mathbb{R}^*$ vérifiant $\forall t \in [0,1]$, $\boldsymbol{x}+th\boldsymbol{e}^{[i]} \in U$ où $\boldsymbol{e}^{[i]}$ est le i-ème vecteur de la base canonique de $\mathbb{R}^n$.*
Alors il existe $\theta \in]0,1[$ tel quel

$$f(\boldsymbol{x}+h\boldsymbol{e}^{[i]}) = f(\boldsymbol{x}) + \sum_{k=1}^{r} \frac{h^k}{k!}\frac{\partial^k f}{\partial x_i^k}(\boldsymbol{x}) + \frac{h^{r+1}}{(r+1)!}\frac{\partial^{r+1} f}{\partial x_i^{r+1}}(\boldsymbol{x}+\theta h\boldsymbol{e}^{[i]}) \tag{4.18}$$

où $\boldsymbol{e}^{[i]}$ est le i-ème vecteur de la base canonique de $\mathbb{R}^n$. Cette formule est le développement limité de f à l'ordre r en $\boldsymbol{x}$ dans la direction $\boldsymbol{e}^{[i]}$

Soit $\varphi : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction suffisament régulière. On note $\boldsymbol{e}^{[i]} \in \mathbb{R}^n$, $i \in [\![1,n]\!]$, le i-ème vecteur de la base canonique de $\mathbb{R}^n$ ($\boldsymbol{e}_j^{[i]} = 0$ si $i \neq j$ et $\boldsymbol{e}_i^{[i]} = 1$). Soit $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^m$ et $h > 0$ on défini les opérateurs aux différences finies suivant

$$(D_{h,i}^+\varphi)(\boldsymbol{x}) = \frac{1}{h}\left(\varphi(\boldsymbol{x}+h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x})\right) \tag{4.19}$$

$$(D_{h,i}^-\varphi)(\boldsymbol{x}) = \frac{1}{h}\left(\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x}-h\boldsymbol{e}^{[i]})\right) \tag{4.20}$$

$$(D_{h,i}^0\varphi)(\boldsymbol{x}) = \frac{1}{2h}\left(\varphi(\boldsymbol{x}+h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x}-h\boldsymbol{e}^{[i]})\right) \tag{4.21}$$

Exercice 4.4.2

Soient $\varphi : U \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ une fonction suffisament régulière, $h > 0$ et les trois opérateurs aux différences finies suivant définis pour $i \in [\![1,2]\!]$

$$\begin{aligned} (D_{h,i}^+\varphi)(\boldsymbol{x}) &= \frac{1}{h}\left(\varphi(\boldsymbol{x}+h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x})\right) \\ (D_{h,i}^-\varphi)(\boldsymbol{x}) &= \frac{1}{h}\left(\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x}-h\boldsymbol{e}^{[i]})\right) \\ (D_{h,i}^0\varphi)(\boldsymbol{x}) &= \frac{1}{2h}\left(\varphi(\boldsymbol{x}+h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x}-h\boldsymbol{e}^{[i]})\right) \end{aligned}$$

avec $\boldsymbol{e}^{[1]} = \begin{pmatrix}1\\0\end{pmatrix}$ et $\boldsymbol{e}^{[2]} = \begin{pmatrix}0\\1\end{pmatrix}$.

Q. 1 *Monter que ces trois opérateurs sont linéaires (i.e. $\forall(\lambda,\mu) \in \mathbb{R}^2$, $\forall \varphi : U \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$, $\forall \psi : U \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$, $D_{h,i}(\lambda\varphi + \mu\psi) = \lambda D_{h,i}\varphi + \mu D_{h,i}\psi$.)*

Q. 2 *On suppose que $\varphi \in \mathcal{C}^k(U \subset \mathbb{R}^2; \mathbb{R})$ avec $k \geqslant 2$. Montrer que les opérateurs $D_{h,i}^+$ et $D_{h,i}^-$ appliqués à φ sont des approximations consistantes d'ordre 1 de $\frac{\partial\varphi}{\partial x_i}$.*

Q. 3 *On suppose que $\varphi \in \mathcal{C}^k(U \subset \mathbb{R}^2; \mathbb{R})$ avec $k \geqslant 3$. Montrer que l'opérateur $D^0_{h,i}$ appliqué à φ est une approximation consistante d'ordre 2 de $\frac{\partial \varphi}{\partial x_i}$.*

Correction 4.4.2

R. 1 Soient $(\lambda, \mu) \in \mathbb{R}^2$, $\varphi : U \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ et $\psi : U \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$. Montrer qu'un opérateur $D_{h,i}$ est linéaire consiste à montrer que

$$D_{h,i}(\lambda\varphi + \mu\psi) = \lambda D_{h,i}\varphi + \mu D_{h,i}\psi$$

or ceci est équivalent à montrer que

$$(D_{h,i}(\lambda\varphi + \mu\psi))(x) = \lambda(D_{h,i}\varphi)(\boldsymbol{x}) + \mu(D_{h,i}\psi)(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{R}^2$$

Avec $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$, on note que $\boldsymbol{x} - h\boldsymbol{e}^{[1]} = (x_1 + h, x_2)$ et $\boldsymbol{x} + h\boldsymbol{e}^{[2]} = (x_1, x_2 + h)$.
Soit $i \in [\![1, 2]\!]$.

- Montrons que $D^+_{h,i}$ est linéaire. On a $\forall \boldsymbol{x} \in \mathbb{R}^2$

$$\begin{aligned}
(D^+_{h,i}(\lambda\varphi + \mu\psi))(\boldsymbol{x}) &= \frac{(\lambda\varphi + \mu\psi)(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - (\lambda\varphi + \mu\psi)(\boldsymbol{x})}{h} \\
&= \frac{(\lambda\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) + \mu\psi(\boldsymbol{x} + h\boldsymbol{e}^{[i]})) - (\lambda\varphi(x) + \mu\psi(\boldsymbol{x}))}{h} \\
&= \lambda\frac{\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x})}{h} + \mu\frac{\psi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}) - \psi(\boldsymbol{x})}{h} \\
&= \lambda(D^+_{h,i}\varphi)(\boldsymbol{x}) + \mu(D^+_{h,i}\psi)(\boldsymbol{x}).
\end{aligned}$$

- Montrons que $D^-_{h,i}$ est linéaire. On a $\forall \boldsymbol{x} \in \mathbb{R}^2$

$$\begin{aligned}
(D^-_{h,i}(\lambda\varphi + \mu\psi))(\boldsymbol{x}) &= \frac{(\lambda\varphi + \mu\psi)(\boldsymbol{x}) - (\lambda\varphi + \mu\psi)(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{h} \\
&= \frac{(\lambda\varphi(\boldsymbol{x}) + \mu\psi(\boldsymbol{x})) - (\lambda\varphi(x - h\boldsymbol{e}^{[i]}) + \mu\psi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}))}{h} \\
&= \lambda\frac{\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{h} + \mu\frac{\psi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}) - \psi(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{h} \\
&= \lambda(D^-_{h,i}\varphi)(\boldsymbol{x}) + \mu(D^-_{h,i}\psi)(\boldsymbol{x}).
\end{aligned}$$

- Montrons que $D^0_{h,i}$ est linéaire. On a $\forall x \in \mathbb{R}$

$$\begin{aligned}
(D^0_h(\lambda\varphi + \mu\psi))(\boldsymbol{x}) &= \frac{(\lambda\varphi + \mu\psi)(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - (\lambda\varphi + \mu\psi)(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{2h} \\
&= \frac{(\lambda\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) + \mu\psi(\boldsymbol{x} + h\boldsymbol{e}^{[i]})) - (\lambda\varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}) + \mu\psi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}))}{2h} \\
&= \lambda\frac{\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{h} + \mu\frac{\psi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - \psi(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{2h} \\
&= \lambda(D^0_{h,i}\varphi)(\boldsymbol{x}) + \mu(D^0_{h,i}\psi)(\boldsymbol{x}).
\end{aligned}$$

R. 2 Soit $h > 0$.

- A l'aide d'un développement de Taylor à l'ordre 2 (voir (4.18)), il existe $\theta^+ \in]0, 1[$ tel que

$$\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) = \varphi(\boldsymbol{x}) + h\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) + \frac{h^2}{2!}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]})$$

On en déduit alors

$$\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) = \frac{\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x})}{h} - \frac{h}{2}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]})$$

Ce qui donne

$$\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) - (D^+_{h,i}(\varphi))(\boldsymbol{x}) = -\frac{h}{2}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]})$$

On obtient donc

$$\max_{\boldsymbol{x}\in U}\left|\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) - (D^+_{h,i}(\varphi))(\boldsymbol{x})\right| \leqslant \frac{h}{2}\max_{x\in U}|\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x})| = Ch.$$

D'après la Définition 4.4.1, l'opérateur $D^+_{h,i}$ est une approximation consistante d'ordre 1 de $\frac{\partial \varphi}{\partial x_i}$.

- A l'aide d'un développement de Taylor à l'ordre 2 (voir (4.18)), il existe $\theta^- \in]0,1[$ tel que

$$\varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}) = \varphi(\boldsymbol{x}) - h\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) + \frac{(-h)^2}{2!}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]})$$

On en déduit alors

$$\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) = \frac{\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{h} + \frac{h}{2}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]})$$

Ce qui donne

$$\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) - (D^-_{h,i}(\varphi))(\boldsymbol{x}) = \frac{h}{2}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]})$$

On obtient donc

$$\max_{\boldsymbol{x}\in U}\left|\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) - (D^-_{h,i}(\varphi))(\boldsymbol{x})\right| \leqslant \frac{h}{2}\max_{x\in U}|\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x})| = Ch.$$

D'après la Définition 4.4.1, l'opérateur $D^-_{h,i}$ est une approximation consistante d'ordre 1 de $\frac{\partial \varphi}{\partial x_i}$.

R. 3

A l'aide de deux développements de Taylor à l'ordre 3, il existe $\theta^+ \in]0,1[$ tel que

$$\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) = \varphi(\boldsymbol{x}) + h\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) + \frac{h^2}{2!}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x}) + \frac{h^3}{3!}\frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]})$$

et $\theta^- \in]0,1[$ tel que

$$\varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}) = \varphi(\boldsymbol{x}) - h\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) + \frac{(-h)^2}{2!}\frac{\partial^2 \varphi}{\partial x_i^2}(\boldsymbol{x}) + \frac{(-h)^3}{3!}\frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]})$$

qui soutraits donnent

$$\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]}) = 2h\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) + \frac{h^3}{3!}(\frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]}) + \frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]})).$$

On en déduit alors

$$\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) = \frac{\varphi(\boldsymbol{x} + h\boldsymbol{e}^{[i]}) - \varphi(\boldsymbol{x} - h\boldsymbol{e}^{[i]})}{2h} - \frac{h^2}{24}(\frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]}) + \frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]}))$$

Ce qui donne

$$\frac{\partial \varphi}{\partial x_i}(\boldsymbol{x}) - (D^0_{h,i}(\varphi))(\boldsymbol{x}) = -\frac{h^2}{24}(\frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} + \theta^+ h\boldsymbol{e}^{[i]}) + \frac{\partial^3 \varphi}{\partial x_i^3}(\boldsymbol{x} - \theta^- h\boldsymbol{e}^{[i]})).$$

On obtient donc

$$\max_{x\in U}\left|\frac{\partial\varphi}{\partial x_i}(\boldsymbol{x})-(D^0_{h,i}(\varphi))(\boldsymbol{x})\right| \leqslant \frac{h^2}{12}\max_{x\in U}|\frac{\partial^3\varphi}{\partial x_i^3}(\boldsymbol{x})| = Ch^2.$$

D'après la Définition 4.4.1, l'opérateur $D^0_{h,i}$ est une approximation consistante d'ordre 2 de $\frac{\partial\varphi}{\partial x_i}$.

Fin correction

Proposition 4.4.4. *Si $\varphi : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ est suffisament régulière, les opérateurs $D^+_{h,i}$ et $D^-_{h,i}$ appliqués à φ sont des approximations consistantes d'ordre 1 de $\frac{\partial\varphi}{\partial x_i}$ et l'opérateur $D^0_{h,i}$ appliqué à φ est une approximation consistante d'ordre 2 de $\frac{\partial\varphi}{\partial x_i}$.*

Proposition 4.4.5. *Soient $i \in [\![1,n]\!]$, $\varphi \in \mathcal{C}^4(U\subset\mathbb{R}^n;\mathbb{R})$. On note $D^2_{h,i}$ l'opérateur défini, pour tout $\boldsymbol{x}\in U$ et $h>0$ vérifiant $\boldsymbol{x}\pm h\boldsymbol{e}^{[i]}\in U$, par*

$$(D^2_{h,i}\varphi)(\boldsymbol{x}) \overset{\text{def}}{=} \frac{1}{h^2}\left[\varphi(\boldsymbol{x}+h\boldsymbol{e}^{[i]})-2\varphi(\boldsymbol{x})+\varphi(\boldsymbol{x}-h\boldsymbol{e}^{[i]})\right] \tag{4.22}$$

Alors $D^2_{h,i}\varphi$ est approximation consistante d'ordre 2 de $\frac{\partial^2\varphi}{\partial x_i^2}$.
De plus, on a

$$D^2_{h,i}\varphi = D^0_{\frac{h}{2},i}(D^0_{\frac{h}{2},i}\varphi) = D^+_{h,i}(D^-_{h,i}\varphi) = D^-_{h,i}(D^+_{h,i}\varphi). \tag{4.23}$$

Preuve. □

4.5 Méthode des différences finies (dimension 1 en espace)

Dans cette partie, nous allons étudier des schémas aux différences finies pour la résolution d'EDP 1D en espace. La première partie sera consacrée à une EDP modèle stationnaire et la seconde à l'équation de la chaleur dans une barre.

4.5.1 EDP stationnaire avec conditions aux limites de Dirichlet

Le problème modèle que nous allons résoudre numériquement par un schéma aux différences finies est le suivant

EDP modèle stationnaire en dimension 1

Trouver $u : [a,b] \longrightarrow \mathbb{R}$ telle que

$$\begin{aligned} -u'' + cu &= f \ \text{ in }]a,b[, && (4.24)\\ u(a) &= \alpha, && (4.25)\\ u(b) &= \beta. && (4.26) \end{aligned}$$

où $a<b$, $c>0$, $\alpha\in\mathbb{R}$, $\beta\in\mathbb{R}$, et $f:[a,b]\longrightarrow\mathbb{R}$ donnés. On peut démontrer que ce problème est **bien posé**.
Dans la très grande majorité des cas, on ne peut déterminer analytiquement une solution de ce problème : on va donc chercher ici à déterminer une approximation numérique de la solution en utilisant les opérateurs aux différences finies. Pour celà on va tout d'abord récrire le problème de manière équivalente. Au lieu de chercher la fonction u, on va chercher la valeur de cette fonction en tout point, ce qui donne

EDP modèle stationnaire en dimension 1 : formulation aux points

Trouver $u(x) \in \mathbb{R}$, $\forall x \in [a,b]$ telle que

$$\begin{aligned} -u''(x) + cu(x) &= f(x) \quad \forall x \in]a,b[, && (4.27)\\ u(a) &= \alpha, && (4.28)\\ u(b) &= \beta. && (4.29) \end{aligned}$$

Le problème est identique : au lieu de chercher une fonction, on cherche une infinité de valeurs! Sur ordinateur, une infinité de valeurs celà fait *un peu* trop! On va donc se limiter à la recherche de quelques valeurs en choisissant, par exemple, de calculer la fonction uniquement aux points d'une discrétisation régulière de l'intervalle $[a,b]$.
Soit $(x_i)_{i=0}^N$ la discrétisation régulière de l'intervalle $[a,b]$ en $N+1$ points :

$$x_i = a + ih, \quad \forall i \in [\![0,N]\!], \text{ avec } h = \frac{b-a}{N}.$$

Du problème (4.27)-(4.29) on en déduit le problème suivant (la réciproque étant fausse) :

EDP modèle stationnaire en dimension 1 : formulation aux points de discrétisation

Trouver $u(x_i) \in \mathbb{R}$, $\forall i \in [\![0,N]\!]$ tels que

$$\begin{aligned} -u''(x_i) + cu(x_i) &= f(x_i) \quad \forall i \in]\!]0,N[\![, && (4.30)\\ u(x_0) &= \alpha, && (4.31)\\ u(x_N) &= \beta. && (4.32) \end{aligned}$$

Nous avons vu en Proposition 4.4.2 une approximation d'ordre 2 de la dérivée seconde :

$$u''(x_i) = (D_h^2 u)(x_i) + \mathcal{O}(h^2) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \mathcal{O}(h^2).$$

Le problème (4.30)-(4.32) peut s'écrire sans dérivées secondes sous la forme

EDP modèle stationnaire en dimension 1 : formulation aux points de discrétisation (bis)

Trouver $u(x_i) \in \mathbb{R}$, $\forall i \in [\![0,N]\!]$ tels que

$$\begin{aligned} -\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \mathcal{O}(h^2) + cu(x_i) &= f(x_i) \quad \forall i \in]\!]0,N[\![, && (4.33)\\ u(x_0) &= \alpha, && (4.34)\\ u(x_N) &= \beta. && (4.35) \end{aligned}$$

Le schéma aux différences finies associé à ce problème est obtenu en *oubliant* le $\mathcal{O}(h^2)$ et en posant $u_i \approx u(x_i)$. Il est alors donné par :

EDP modèle stationnaire en dimension 1 : schéma aux différences finies

Trouver $u_i \in \mathbb{R}$, $\forall i \in [\![0,N]\!]$ tels que

$$\begin{aligned} -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + cu_i &= f(x_i) \quad \forall i \in]\!]0,N[\![, && (4.36)\\ u_0 &= \alpha, && (4.37)\\ u_N &= \beta. && (4.38) \end{aligned}$$

Il faut noter que l'on aboutit à un système linéaire de $N+1$ équations à $N+1$ inconnues : on peut donc l'écrire sous forme matricielle chacune des équations correspondant à une ligne du système. Pour alléger les écritures on note que (4.36) peut se récrire

$$-u_{i+1} + \mu u_i - u_{i-1} = h^2 f(x_i)$$

avec $\mu = 2 + ch^2$. Les équations discrétisées peuvent s'écrire

$$\left\{\begin{array}{lcll} u_0 & = & \alpha & \leftarrow \text{eq. en } x_0 \\ -u_2 + \mu u_1 - u_0 & = & h^2 f(x_1) & \leftarrow \text{eq. en } x_1 \\ -u_3 + \mu u_2 - u_1 & = & h^2 f(x_2) & \leftarrow \text{eq. en } x_2 \\ & \vdots & & \\ -u_{N-1} + \mu u_{N-2} - u_{N-3} & = & h^2 f(x_{N-2}) & \leftarrow \text{eq. en } x_{N-2} \\ -u_N + \mu u_{N-1} - u_{N-2} & = & h^2 f(x_{N-1}) & \leftarrow \text{eq. en } x_{N-1} \\ u_N & = & \beta & \leftarrow \text{eq. en } x_N \end{array}\right.$$

En multipliant les équations en x_0 et en x_N par h^2, le système linéaire équivalent à (4.36)-(4.38) peut alors s'écrire sous la forme matricielle suivante

$$\mathbb{A}\boldsymbol{U} \overset{\text{def}}{=} \left(\begin{array}{c|ccccccc} h^2 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ \hline -1 & \mu & -1 & 0 & \dots & \dots & 0 & 0 \\ 0 & -1 & \mu & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \dots & 0 & 1 & \mu & -1 & 0 \\ 0 & 0 & \dots & \dots & 0 & -1 & \mu & -1 \\ \hline 0 & \dots & \dots & \dots & \dots & \dots & 0 & h^2 \end{array}\right) \left(\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ \hline u_N \end{array}\right) = h^2 \left(\begin{array}{c} \alpha \\ \hline f(x_1) \\ f(x_2) \\ \vdots \\ \vdots \\ f(x_{N-2}) \\ f(x_{N-1}) \\ \hline \beta \end{array}\right) \overset{\text{def}}{=} \boldsymbol{B} \tag{4.39}$$

Proposition 4.5.1 (admis). *Le schéma aux différences finies* (4.36)-(4.38) *est consistant à l'ordre* 2 *avec l'EDP* (4.24)-(4.26) *et on a*

$$\max_{i \in [\![0,N]\!]} |u(x_i) - u_i| = \mathcal{O}(h^2). \tag{4.40}$$

Exercice 4.5.1

Q. 1 *Ecrire la fonction* AssembleMat1D *retournant la matrice* $\mathbb{M} \in \mathcal{M}_d(\mathbb{R})$ *définie par*

$$\mathbb{M} = \begin{pmatrix} \gamma & 0 & 0 & \dots & \dots & \dots & 0 \\ \beta & \alpha & \beta & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \beta & \alpha & \beta \\ 0 & \dots & \dots & \dots & 0 & 0 & \gamma \end{pmatrix} \tag{4.1}$$

où α, β *et* γ *sont des réels donnés.*

On souhaite résoudre par un schéma aux différences finies l'EDP suivante

$$\begin{array}{rcl} -u'' + cu & = & f \ \text{ in }]a,b[, \\ u(a) & = & \alpha, \\ u(b) & = & \beta. \end{array}$$

Q. 2 *En prenant le jeu de données $a = 0$, $b = 2\pi$, $c = 1$, $\alpha = 1$, $\beta = -1$ et $f : x \mapsto \cos(x^2)$, écrire une fonction permettant de résoudre l'EDP précédente. On pourra utiliser la fonction $\boldsymbol{X} \leftarrow$ Solve$(\mathbb{A}, \boldsymbol{B})$ retournant la solution du système linéaire $\mathbb{A}\boldsymbol{X} = \boldsymbol{B}$.*

Q. 3 *En choisissant judicieusement un jeu de données écrire un programme permettant de vérifier l'ordre du schéma utilisé à l'aide de la formule* (4.40).

Correction 4.5.1

R. 1

Algorithme 4.1 Fonction AssembleMat1D

Données : d : dimension de la matrice,
α, β, γ : trois réels
Résultat : $\mathbb{M}$: matrice $d \times d$

Fonction $x \leftarrow$ AssembleMat1D(d, α, β, γ)
  $\mathbb{M} \leftarrow \mathbb{0}_{d,d}$
  $\mathbb{M}(1,1) \leftarrow \gamma$, $\mathbb{M}(d,d) \leftarrow \gamma$,
  Pour $i = 2$ **à** $d-1$ **faire** ▷ Initialisation de la ligne i
    $\mathbb{M}(i,i) \leftarrow \alpha$, $\mathbb{M}(i,i-1) \leftarrow \beta$, $\mathbb{M}(i,i+1) \leftarrow \beta$
  Fin
Fin

Le code Matlab/Octave correspondant est donné en Listing 4.1.

R. 2

Il nous faut résoudre le système (4.5.2) avec le jeu de données. On peut par exemple écrire la fonction SolveEDP1 qui a partir du jeu de données et du nombre de discrétisation N va nous retourner le vecteur solution de (4.5.2).

Algorithme 4.2 Fonction SolveEDP1 : résolution de l'EDP (4.24)-(4.26) par le schéma aux différences finies (4.36)-(4.38)

Données : a, b : $a < b$,
c : $c > 0$,
α, β : deux réels,
f : fonction de $[a,b]$ à valeurs réelles,
N : nombre de discrétisation
Résultat : $\boldsymbol{x}$: vecteur de $\mathbb{R}^{N+1}$, discrétisation régulière de l'intervalle $[a,b]$, avec $N+1$ points. $\boldsymbol{x}(i+1) = x_i$.
$\boldsymbol{U}$: vecteur de $\mathbb{R}^{N+1}$ tel que $\boldsymbol{U}(i+1) \approx u(x_i)$

Fonction $[\boldsymbol{x}, \boldsymbol{U}] \leftarrow$ SolveEDP1($a, b, c, \alpha, \beta, f, N$)
  $\boldsymbol{x} \leftarrow$ DisReg(a, b, N)
  $h \leftarrow (b-a)/N$
  $\mathbb{A} \leftarrow$ AssembleMat1D$(N+1, 2+ch^2, -1, h^2)$
  $\boldsymbol{B}(1) \leftarrow h^2\alpha$, $\boldsymbol{B}(N+1) \leftarrow h^2\beta$
  Pour $i = 2$ **à** N **faire** ▷ Initialisation de la ligne i de $\boldsymbol{B}$
    $\boldsymbol{B}(i) \leftarrow h^2 f(\boldsymbol{x}(i))$
  Fin
  $\boldsymbol{U} \leftarrow$ Solve$(\mathbb{A}, \boldsymbol{B})$
Fin

Le code Matlab/Octave correspondant est donné en Listing 4.2. Ensuite on peut utiliser directement

cette fonction avec le jeu de données :

$$\boldsymbol{U} \leftarrow \text{SolveEDP1}(0, 2\pi, 1, 1, -1, , x \mapsto \cos(x^2), 1000)$$

```
function M=AssembleMat1D(d,alp,bet,gam)
  M=sparse(d,d);
  M(1,1)=gam;M(d,d)=gam;
  for i=2:d-1
    M(i,i)=alp;
    M(i,i-1)=bet;M(i,i+1)=bet;
  end
end
```

Listing 4.1: fonction Matlab/Octave `AssembleMat1D`

```
function [x,U]=solveEDP1(a,b,c,alp,bet,f,N)
  h=(b-a)/N;
  x=a:h:b;
  A=AssembleMat1D(N+1,2+c*h*h,-1,h*h);
  B=zeros(N+1,1);
  B(1)=alp;B(N+1)=bet;
  for i=2:N
    B(i)=f(x(i));
  end
  B=h*h*B;
  U=A\B;
end
```

Listing 4.2: fonction Matlab/Octave `solveEDP1`

R. 3

Nous allons calculer pour différentes valeurs de N (nombre de discrétisations avec $h = (b-a)/N$) l'erreur $E(h)$ commise

$$E(h) = \max_{i \in [\![0,N]\!]} |u(x_i) - u_i|.$$

Il nous faut donc choisir les données pour avoir une solution analytique. En fait, pour cela, on choisit une fonction u suffisament régulière, par exemple $u(x) = \sin(x^2)$, puis on détermine la fonction f par $f(x) = -u''(x) + cu(x)$. Avec l'exemple pour u, on obtient ainsi $f(x) = 4x^2 \sin(x^2) - 2\cos(x^2) + \sin(x^2)$. Les conditions aux limites sont alors $u(a) = \sin(a^2)$ et $u(b) = \sin(b^2)$. Avec $c = 1$, l'EDP admettant une unique solution celle-ci est nécessairement la fonction $\sin(x^2)$!
Comme pour les EDO, la Figure 4.6 permet grâce à une représentation en echelle logarithmique de retrouver l'ordre 2 du schéma.

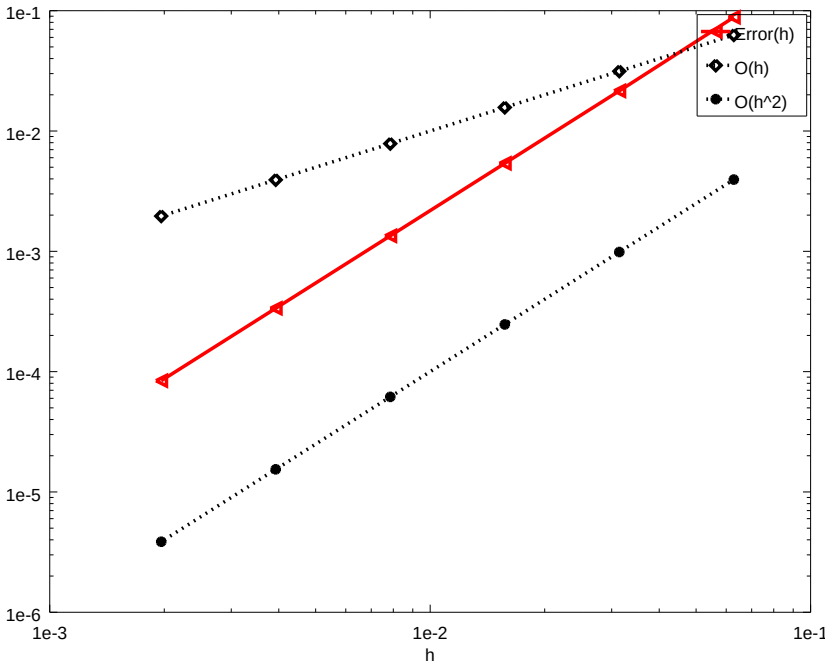


Figure 4.6: Représentation en échelle logarithmique

On donne le code Matlab/Octave en Listing 4.3 permettant de générer cette figure.

```
clear all
close all
% Initialisation des donnees
uex=@(x) sin(x.^2);
```

```
c=1;
f=@(x) 4*x^2*sin(x^2) - 2*cos(x^2) + c*sin(x^2);
a=0;b=2*pi;
% Calcul des erreurs
LN=[100,200,400,800,1600,3200];
k=1;
for N=LN
  [x,U]=solveEDP1(a,b,c,uex(a),uex(b),f,N);
  H(k)=(b-a)/N;
  E(k)=max(abs(uex(x)'-U));
  k=k+1;
end
% Representation graphique
loglog(H,E,'r<-','LineWidth',2)
hold on
loglog(H,H,'kd:','LineWidth',2)
loglog(H,H.^2,'k*:','LineWidth',2)
legend('Error(h)','O(h)','O(h^2)')
xlabel('h')
```

Listing 4.3: Script Matlab/Octave pour la représentation de l'ordre

Fin correction

4.5.2 EDP stationnaire avec conditions aux limites mixtes

Le problème modèle que nous allons résoudre numériquement par un schéma aux différences finies est le suivant

EDP modèle stationnaire en dimension 1 avec condition de Dirichlet à droite et Neumann à gauche

Trouver $u : [a,b] \longrightarrow \mathbb{R}$ telle que

$$-u'' + cu = f \text{ in }]a,b[, \tag{4.41}$$
$$u(a) = \alpha, \tag{4.42}$$
$$u'(b) = \beta. \tag{4.43}$$

où $a < b$, $c > 0$, $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}$, et $f : [a,b] \longrightarrow \mathbb{R}$ donnés. On peut démontrer que ce problème est **bien posé**.
Le démarche à suivre est identique à ce qui a été fait en section précédente. Seul changement, la condition de Neumann au point $x_N = b$ et donc il nous faut juste modifier la dernière ligne du système linéaire pour prendre en compte cette condition au limite.
On ne peut intégrer directement l'équation $u'(x_N) = \beta$ dans le système linéaire. La première idée consiste à approcher u' en x_N à l'aide de l'opérateur D_h^- definit en (4.1) (On ne peut utiliser l'opérateur D_h^+ car $x_N + h$ est en dehors de l'intervalle d'étude). On a alors

$$u'(x_N) = (D_h^- u)(x_N) + \mathcal{O}(h) = \frac{u(x_N) - u(x_{N-1})}{h} + \mathcal{O}(h).$$

On en déduit alors l'équation dicrétisée approchant (4.43) à l'ordre 1:

$$\frac{u_N - u_{N-1}}{h} = \beta.$$

Le schéma complet est alors le suivant:

EDP modèle stationnaire en dimension 1 avec condition de Dirichlet à droite et Neumann à gauche: schéma aux différences finies (ordre 1)

Trouver $u_i \in \mathbb{R}$, $\forall i \in [\![0, N]\!]$ tels que

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + cu_i = f(x_i) \quad \forall i \in]\!]0, N[\![, \tag{4.44}$$

$$u_0 = \alpha, \tag{4.45}$$

$$\frac{u_N - u_{N-1}}{h} = \beta. \tag{4.46}$$

Il existe de nombreuses manières d'écrire le système linéaire associé à la discrétisation par différences finies de l'E.D.P. (4.41)-(4.42) , mais les **plus simples à écrire** sont celles pour lesquelles la matrice du système contient le moins possible de fractions. Par exemple, les équations discrétisées peuvent s'écrire

$$\left\{\begin{array}{lcll} u_0 & = & \alpha & \leftarrow \text{eq. en } x_0 \\ -u_2 + \mu u_1 - u_0 & = & h^2 f(x_1) & \leftarrow \text{eq. en } x_1 \\ -u_3 + \mu u_2 - u_1 & = & h^2 f(x_2) & \leftarrow \text{eq. en } x_2 \\ & \vdots & & \\ -u_{N-1} + \mu u_{N-2} - u_{N-3} & = & h^2 f(x_{N-2}) & \leftarrow \text{eq. en } x_{N-2} \\ -u_N + \mu u_{N-1} - u_{N-2} & = & h^2 f(x_{N-1}) & \leftarrow \text{eq. en } x_{N-1} \\ u_N - u_{N-1} & = & h\beta & \leftarrow \text{eq. en } x_N \end{array}\right.$$

avec $\mu = 2 + ch^2$.

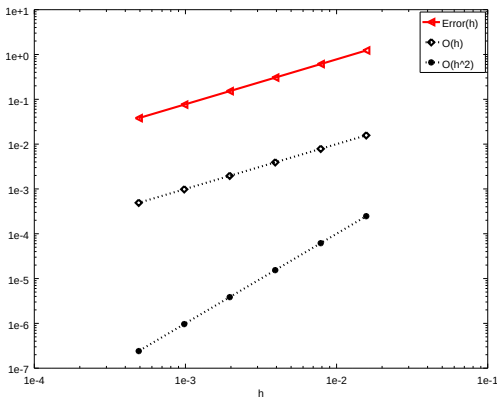
On obtient alors le système linéaire

$$\left(\begin{array}{c|ccccccc} 1 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ \hline -1 & \mu & -1 & 0 & \dots & \dots & 0 & 0 \\ 0 & -1 & \mu & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \dots & 0 & -1 & \mu & -1 & 0 \\ 0 & 0 & \dots & \dots & 0 & -1 & \mu & -1 \\ \hline 0 & \dots & \dots & \dots & \dots & 0 & -1 & 1 \end{array}\right) \left(\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ \hline u_N \end{array}\right) = \left(\begin{array}{c} \alpha \\ \hline h^2 f(x_1) \\ h^2 f(x_2) \\ \vdots \\ \vdots \\ h^2 f(x_{N-2}) \\ h^2 f(x_{N-1}) \\ \hline h\beta \end{array}\right) \tag{4.47}$$

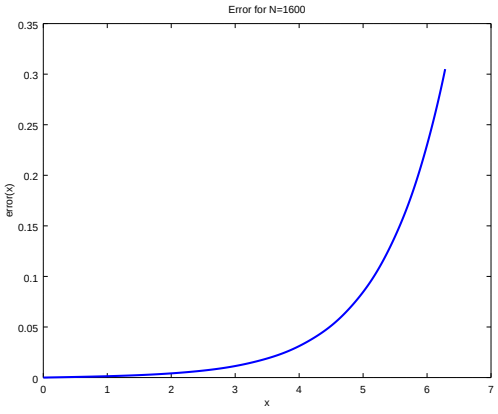
En multipliant par h^2 la première ligne du système précédent et par h la dernière, on obtient le système linéaire équivalent

$$\left(\begin{array}{c|ccccccc} h^2 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ \hline -1 & \mu & -1 & 0 & \dots & \dots & 0 & 0 \\ 0 & -1 & \mu & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \dots & 0 & -1 & \mu & -1 & 0 \\ 0 & 0 & \dots & \dots & 0 & -1 & \mu & -1 \\ \hline 0 & \dots & \dots & \dots & \dots & 0 & -h & h \end{array}\right) \left(\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ \hline u_N \end{array}\right) = h^2 \left(\begin{array}{c} \alpha \\ \hline f(x_1) \\ f(x_2) \\ \vdots \\ \vdots \\ f(x_{N-2}) \\ f(x_{N-1}) \\ \hline \beta \end{array}\right)$$

On represente l'ordre du schéma ainsi obtenu en Figure 4.7a et on trouve l'ordre 1!



(a) Représentation en échelle logarithmique de l'ordre du schéma



(b) Représentation de l'erreur en fonction de x pour $N = 1600$

Figure 4.7: Problème modèle stationnaire avec Neumann approché à l'ordre 1

On peut toujours espérer être en ordre 1 au point $x_N = b$ (puisque l'on a approché à l'ordre 1) et être d'ordre 2 sur les autres points mais ce n'est pas le cas. Il est clair d'après la Figure 4.7b que l'erreur sur le schéma en x_N s'est propagée aux autres points.
Pour retrouver un schéma globale d'ordre 2, il faut aussi déterminer une nouvelle équation approchant la condition de Neumann en $x_N = b$ à l'order 2. Pour celà deux méthodes sont proposées pour approcher à l'ordre 2 la dérivée première de u en b: la première utilisera une formule générique tandis que la seconde utilisera l'équation (4.41).

Condition de Neumann: méthode 1

Pour écrire un schéma d'ordre 2 pour la condition de Neumann en $x_N = b$, nous allons utiliser un des résultats de l'exercice suibant:

Exercice 4.5.2

Soit φ une fonction suffisament régulière et $h > 0$

Q. 1 *Montrer que*

$$\frac{d\varphi}{dx}(x) = \frac{-3\varphi(x) + 4\varphi(x+h) - \varphi(x+2h)}{2h} + \mathcal{O}(h^2) \tag{4.1}$$

Q. 2 *Montrer que*

$$\frac{d\varphi}{dx}(x) = \frac{3\varphi(x) - 4\varphi(x-h) + \varphi(x-2h)}{2h} + \mathcal{O}(h^2) \tag{4.2}$$

Q. 3 *Déterminer une formule permettant de calculer une approximation à l'ordre 2 de $\frac{d^2\varphi}{dx^2}(x)$ en utilisant uniquement des valeurs de la fonction φ aux points $x + ih$ avec $i \in \mathbb{N}$.*

Q. 4 *Déterminer une formule permettant de calculer une approximation à l'ordre 2 de $\frac{d^2\varphi}{dx^2}(x)$ en utilisant uniquement des valeurs de la fonction φ aux points $x - ih$ avec $i \in \mathbb{N}$.*

Correction 4.5.2

R. 1 Soit φ une fonction suffisament régulière et $h > 0$
Pour celà, on écrit les deux développements de Taylor en $x + h$ et $x + 2h$.
Il existe $\xi_1^+ \in]x, x+h[$ tel que

$$\varphi(x+h) = \varphi(x) + h\frac{d\varphi}{dx}(x) + \frac{h^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{h^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^+) \tag{R2.19}$$

et Il existe $\xi_2^+ \in]x, x+2h[$ tel que

$$\varphi(x+2h) = \varphi(x) + (2h)\frac{d\varphi}{dx}(x) + \frac{(2h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^+) \tag{R2.20}$$

L'objectif est d'obtenir, par une combinaison linéaire entre ces deux formules, une nouvelle équation ne comportant plus de termes en $\frac{d^2\varphi}{dx^2}$. En effectuant $4 \times$ (R2.19) $-$ (R2.20) on obtient

$$4\varphi(x+h) - \varphi(x+2h) = 3\varphi(x) + 2h\frac{d\varphi}{dx}(x) + 4\frac{h^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^+) - \frac{(2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^+)$$

On en déduit

$$\frac{d\varphi}{dx}(x) = \frac{-3\varphi(x) + 4\varphi(x+h) - \varphi(x+2h)}{2h} + \mathcal{O}(h^2)$$

R. 2

Soit φ une fonction suffisament régulière et $h > 0$.
Pour celà, on écrit les deux développements de Taylor en $x-h$ et $x-2h$.
Il existe $\xi_1^- \in]x-h, x[$ tel que

$$\varphi(x-h) = \varphi(x) - h\frac{d\varphi}{dx}(x) + \frac{(-h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(-h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^-) \tag{R2.21}$$

et Il existe $\xi_2^- \in]x-2h, x[$ tel que

$$\varphi(x-2h) = \varphi(x) + (-2h)\frac{d\varphi}{dx}(x) + \frac{(-2h)^2}{2!}\frac{d^2\varphi}{dx^2}(x) + \frac{(-2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^+) \tag{R2.22}$$

L'objectif est d'obtenir, par une combinaison linéaire entre ces deux formules, une nouvelle équation ne comportant plus de termes en $\frac{d^2\varphi}{dx^2}$. En effectuant $4 \times$ (R2.21) $-$ (R2.22) on obtient

$$4\varphi(x-h) - \varphi(x-2h) = 3\varphi(x) - 2h\frac{d\varphi}{dx}(x) + 4\frac{(-h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_1^-) - \frac{(-2h)^3}{3!}\frac{d^3\varphi}{dx^3}(\xi_2^-)$$

On en déduit

$$\frac{d\varphi}{dx}(x) = \frac{3\varphi(x) - 4\varphi(x-h) + \varphi(x-2h)}{2h} + \mathcal{O}(h^2)$$

Fin correction

On utilise alors (4.2) pour réécrire la condition de Neumann $u'(x_N) = \beta$:

$$\frac{3u(b) - 4u(b-h) + u(b-2h)}{2h} + \mathcal{O}(h^2) = \beta$$

Comme $x_N = b$ et $x_{N-1} = b-h$, on en déduit une équation approchant la condition de Neumann à l'ordre 2:

$$\frac{3u_N - 4u_{N-1} + u_{N-2}}{2h} = \beta. \tag{4.48}$$

Condition de Neumann: méthode 2

En écrivant le développement de Taylor à l'ordre 3 de $u(b-h)$ on a

$$u(b-h) = u(b) - hu'(b) + \frac{h^2}{2}u''(b) + \mathcal{O}(h^3)$$

et en écrivant (4.41) en $x = b$ on obtient

$$-u''(b) + cu(b) = f(b).$$

En remplaçant dans le développement de Taylor $u''(b)$ par $cu(b) - f(b)$ on abouti à

$$u(b-h) = u(b) - hu'(b) + \frac{h^2}{2}\big(cu(b) - f(b)\big) + \mathcal{O}(h^3)$$

et donc

$$u'(b) = \frac{u(b) - u(b-h)}{h} + \frac{h}{2}(cu(b) - f(b)) + \mathcal{O}(h^2). \tag{4.49}$$

Comme $x_N = b$ et $x_{N-1} = b - h$, on en déduit l'équation approchant la condition de Neumann à l'ordre 2:

$$\frac{u_N - u_{N-1}}{h} + \frac{h}{2}\big(cu_N - f(x_N)\big) = \beta$$

c'est à dire

$$(1 + c\frac{h^2}{2})u_N - u_{N-1} = h\beta + \frac{h^2}{2}f(x_N). \tag{4.50}$$

Système linéaire associé

Dans le schéma aux différences finies (4.36)-(4.38), on peut remplacer (4.38) par (4.48) ou (4.50) Par exemple, en utilisant (4.50), Le schéma complet est alors le suivant:

EDP modèle stationnaire en dimension 1 avec condition de Dirichlet à droite et Neumann à gauche: schéma aux différences finies (ordre 2, méthode 1)

Trouver $u_i \in \mathbb{R}$, $\forall i \in [\![0, N]\!]$ tels que

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + cu_i = f(x_i) \quad \forall i \in]\!]0, N[\![, \tag{4.51}$$

$$u_0 = \alpha, \tag{4.52}$$

$$\frac{3u_N - 4u_{N-1} + u_{N-2}}{2h} = \beta. \tag{4.53}$$

Les équations discrétisées peuvent alors s'écrire sous la forme

$$\left\{\begin{array}{lcll}
u_0 & = & \alpha & \text{(eq. en } x_0\text{)} \quad \text{ligne } 1 \\
-u_2 + \mu u_1 - u_0 & = & h^2 f(x_1) & \text{(eq. en } x_1\text{)} \quad \text{ligne } 2 \\
-u_3 + \mu u_2 - u_1 & = & h^2 f(x_2) & \text{(eq. en } x_2\text{)} \quad \text{ligne } 3 \\
 & \vdots & & \\
-u_{N-1} + \mu u_{N-2} - u_{N-3} & = & h^2 f(x_{N-2}) & \text{(eq. en } x_{N-2}\text{)} \quad \text{ligne } N-1 \\
-u_N + \mu u_{N-1} - u_{N-2} & = & h^2 f(x_{N-1}) & \text{(eq. en } x_{N-1}\text{)} \quad \text{ligne } N \\
3u_N - 4u_{N-1} + u_{N-2} & = & 2h\beta & \text{(eq. en } x_N\text{)} \quad \text{ligne } N+1
\end{array}\right.$$

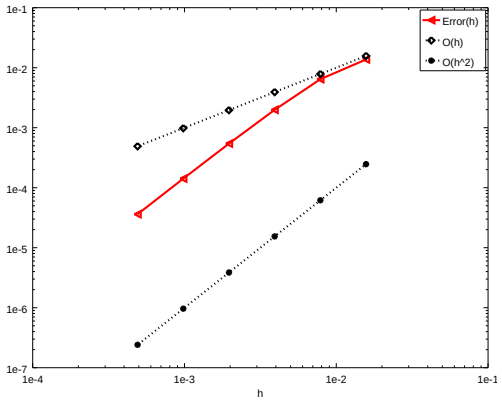
avec $\mu = 2 + ch^2$. Dans ce cas le système linéaire s'écrit

$$\left(\begin{array}{c|ccccccc}
1 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ \hline
-1 & \mu & -1 & 0 & \dots & \dots & 0 & 0 \\
0 & -1 & \mu & -1 & 0 & \dots & 0 & 0 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\
\vdots & 0 & \dots & 0 & 1 & \mu & -1 & 0 \\
0 & 0 & \dots & \dots & 0 & -1 & \mu & -1 \\ \hline
0 & \dots & \dots & 0 & 0 & 1 & -4 & 3
\end{array}\right)
\left(\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ \hline u_N \end{array}\right)
=
\left(\begin{array}{c} \alpha \\ \hline h^2 f(x_1) \\ h^2 f(x_2) \\ \vdots \\ \vdots \\ h^2 f(x_{N-2}) \\ h^2 f(x_{N-1}) \\ \hline 2h\beta \end{array}\right) \tag{4.54}$$

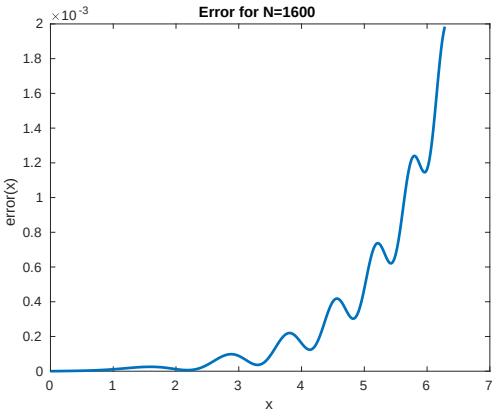
On peut aussi l'écrire sous la forme

$$\left(\begin{array}{c|ccccccc}
h^2 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ \hline
-1 & \mu & -1 & 0 & \dots & \dots & 0 & 0 \\
0 & -1 & \mu & -1 & 0 & \dots & 0 & 0 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\
\vdots & 0 & \dots & 0 & 1 & \mu & -1 & 0 \\
0 & 0 & \dots & \dots & 0 & -1 & \mu & -1 \\ \hline
0 & \dots & \dots & \dots & 0 & h & -4h & 3h
\end{array}\right)
\left(\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ \hline u_N \end{array}\right)
= h^2
\left(\begin{array}{c} \alpha \\ \hline f(x_1) \\ f(x_2) \\ \vdots \\ \vdots \\ f(x_{N-2}) \\ f(x_{N-1}) \\ \hline 2\beta \end{array}\right) \tag{4.55}$$

On represente l'ordre du schéma en Figure 4.8a et cette fois ci on trouve l'ordre 2!



(a) Représentation en échelle logarithmique de l'ordre du schéma



(b) Représentation de l'erreur en fonction de x pour $N = 1600$

Figure 4.8: Problème modèle stationnaire avec Neumann approché à l'ordre 2

Exercice 4.5.3 (type examen)

Soit le problème suivant

$$-u''(x) + c(x)u(x) = f(x), \ \forall x \in]a;b[, \tag{4.1}$$
$$u'(a) = \alpha, \tag{4.2}$$
$$u(b) = \beta. \tag{4.3}$$

où c est une fonction positive.

Q. 1

a. *Quelles sont les données du problème* (4.1)-(4.3)*? (préciser le type de chaque donnée : réel, entier, fonction, vecteur, ...)*

b. *Quelles sont les inconnues du problème* (4.1)-(4.3)*? (préciser le type)*

c. *Quelles sont les conditions initiales?*

d. *Quelles sont les conditions aux limites?*

Q. 2

Construire une discrétisation régulière de $[a;b]$ *avec* N *pas de discrétisation en espace.*

On note x_i, $i \in [\![0, N]\!]$ cette discrétisation. On souhaite résoudre (4.1) à l'aide du schéma numérique

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + c_i u_i = f_i. \tag{4.4}$$

Q. 3

a. *Expliquer comment le schéma* (4.4) *a été obtenu à partir de* (4.1) *et préciser ce que représente les termes* u_i, f_i, c_i *et* Δx*?*

b. *Donner l'ensemble* $\mathcal{E}$ *des valeurs que peut prendre* i *dans le schéma* (4.1)*.*

c. *Construire une discrétisation des conditions aux limites d'ordre 2 au moins.*

d. *Le schéma global est de quel ordre? Justifiez.*

On note $\boldsymbol{V}$ le vecteur de dimension $N+1$, de composantes $\boldsymbol{V}_i = u_{i-1}$, $\forall i \in [\![1, N+1]\!]$.

Q. 4

Montrer que le vecteur $\boldsymbol{V}$ *est solution du système linéaire*

$$\mathbb{A}\boldsymbol{V} = \boldsymbol{F} \tag{4.5}$$

en explicitant la matrice $\mathbb{A}$ *et le vecteur* $\boldsymbol{F}$ *(préciser les dimensions).*

Q. 5 *Ecrire un algorithme complet de résolution du problème* (4.1) *à* (4.3) *basé sur* (4.5). *(Utiliser au maximum les fonctions). On pourra utiliser la fonction* $\boldsymbol{X} \leftarrow$ Solve$(\mathbb{A}, \boldsymbol{B})$ *retournant la solution du système linéaire* $\mathbb{A}\boldsymbol{X} = \boldsymbol{B}$.

Correction 4.5.3

R. 1

a. Les données du problème (4.1)-(4.3) sont :

Données :
a, b : deux réels, $a < b$
α, β : deux réels,
c : une fonction $x : [a, b] \longrightarrow \mathbb{R}$ telle que $c(x) \geqslant 0$,
f : une fonction $f : [a, b] \longrightarrow \mathbb{R}$

b. L'inconnue du problème (4.1)-(4.3) est la fonction u, $u : [a, b] \longrightarrow \mathbb{R}$.

c. Il n'y a pas de condition initiale, le problème étant stationnaire!

d. Les conditions aux limites sont (4.2) (appelée condition de Neumann) et (4.3) (appelée condition de Dirichlet).

R. 2

Soit $\Delta_x = (b-a)/N$. La discrétisation régulière de l'intervalle $[a; b]$ avec N pas de discrétisation est l'ensemble des points x_i, $i \in [\![0, N]\!]$ tel que

$$x_i = a + i\Delta_x, \quad \forall i \in [\![0, N]\!]$$

R. 3

a. De (4.1), on déduit

$$-u''(x_i) + c(x_i)u(x_i) = f(x_i), \quad \forall i \in [\![0, N]\!] \tag{R3.23}$$

En utilisant deux développements de Taylor à l'ordre 4 en $x + h \in [a, b]$ et $x - h \in [a, b]$ on obtient

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2!}u''(x) + \frac{h^3}{3!}u^{(3)}(x) + \mathcal{O}(h^4) \tag{R3.24}$$

$$u(x-h) = u(x) - hu'(x) + \frac{(-h)^2}{2!}u''(x) + \frac{(-h)^3}{3!}u^{(3)}(x) + \mathcal{O}(h^4). \tag{R3.25}$$

En sommant ces deux équations, on abouti a

$$u(x+h) + u(x-h) = 2u(x) + h^2u''(x) + \mathcal{O}(h^4)$$

et donc

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + \mathcal{O}(h^2).$$

Cette dernière équation peut s'appliquer en $x = x_i$ avec $h = \Delta x$ pour tout $i \in]\!]0, N[\!]$ et on a alors

$$u''(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{\Delta x^2} + \mathcal{O}(\Delta x^2). \tag{R3.26}$$

En remplaçant, dans (R3.27), $u''(x_i)$ par l'expression précédente on obtient

$$\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{\Delta x^2} + \mathcal{O}(\Delta x^2) + c(x_i)u(x_i) = f(x_i), \quad \forall i \in]\!]0, N[\!] \tag{R3.27}$$

Il faut noter qu'il n'est pas possible d'utiliser cette relation en $i = 0$ ou en $i = N$. Le schéma (4.4) est donc une approximation de (R3.27) où l'on a *oublié* le terme en $\mathcal{O}(\Delta x^2)$ et noté $u_i \approx u(x_i)$, $c_i = c(x_i)$ et $f_i = f(x_i)$.

b. On a $\mathcal{E} =]\!]0, N[\!]$.

c. On note $h = \Delta_x$. La condition de Dirichlet (4.3) s'écrit de manière exacte

$$u_N = u(x_N) = \beta. \tag{R3.28}$$

Pour la condition de Neumann (4.2) il va falloir travailler un peu et déterminer une approximation de la dérivée première de u en $a = x_0$ d'ordre 2. Pour celà, on écrit les deux développements de Taylor en $a + h = x_1$ et $a + 2h = x_2$.

$$u(a+h) = u(a) + hu'(a) + \frac{h^2}{2!}u''(a) + \mathcal{O}(h^3), \tag{R3.29}$$

$$u(a+2h) = u(a) + (2h)u'(a) + \frac{(2h)^2}{2!}u''(a) + \mathcal{O}(h^3). \tag{R3.30}$$

L'objectif est d'obtenir, par une combinaison linéaire entre ces deux formules, une nouvelle équation ne comportant plus de termes en $u''(a)$. En effectuant $4 \times$ (R3.29) $-$ (R3.30) on obtient

$$4u(a+h) - u(a+2h) = 3u(a) + 2hu'(a) + \mathcal{O}(h^3)$$

On en déduit

$$u'(a) = \frac{-3u(a) + 4u(a+h) - u(a+2h)}{2h} + \mathcal{O}(h^2). \tag{R3.31}$$

De (4.2) et comme $x_0 = a$, $x_1 = a + h$, $x_2 = a + 2h$ on a

$$\frac{-3u(x_0) + 4u(x_1) - u(x_2)}{2h} + \mathcal{O}(h^2) = \alpha$$

En oubliant le terme en $\mathcal{O}(h^2)$ on abouti au schéma

$$\frac{-3u_0 + 4u_1 - u_2}{2h} = \alpha \tag{R3.32}$$

d. Le schéma global est

$$-3u_0 + 4u_1 - u_2 = 2\Delta x\alpha \tag{R3.33}$$

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + c_i u_i = f_i, \quad \forall i \in]\!]0, N[\![\tag{R3.34}$$

$$u_N = \beta \tag{R3.35}$$

Il est d'ordre 2 car toutes les dérivées partielles ont été approchées à l'ordre 2.

R. 4

Les $N + 1$ équations (R3.33)-(R3.35) sont linéaires en les $N + 1$ inconnues $(u_i)_{i\in[\![0,N]\!]}$. En notant $\boldsymbol{V} \in \mathbb{R}^{N+1}$ le vecteur tel que $\boldsymbol{V}_i = u_{i-1}$, $\forall i \in [\![1, N+1]\!]$ ces $N + 1$ équations peuvent donc s'écrire sous la forme d'un système linéaire $\mathbb{A}\boldsymbol{V} = \boldsymbol{F}$ où la matrice $\mathbb{A} \in \mathcal{M}_{N+1}(\mathbb{R})$ et $\boldsymbol{F} \in \mathbb{R}^{N+1}$.
On choisi pour

- première ligne su système l'équation (R3.33) (*portée* par le point x_0)
- dernière ligne (ligne $N + 1$) l'équation (R3.35) (*portée* par le point x_N)
- les lignes 2 à N, respectivement les équations (R3.34) de $i = 1$ à $i = N - 1$.

Pour simplifier l'écriture du système, on peut multiplier (R3.34) par Δx^2 et poser $D_i = 2 + \Delta x^2 c_i$ pour obtenir

$$D_i u_i - u_{i+1} - u_{i-1} = \Delta x^2 f_i \quad \forall i \in]\!]0, N[\![\tag{R3.36}$$

On abouti alors au système linéaire suivant

$$\mathbb{A}\boldsymbol{U} \overset{\text{def}}{=} \left(\begin{array}{c|ccccccc} -3 & 4 & -1 & 0 & \dots & \dots & 0 & 0 \\ \hline -1 & D_1 & -1 & 0 & \dots & \dots & 0 & 0 \\ 0 & -1 & D_2 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \dots & 0 & -1 & D_{N-2} & -1 & 0 \\ 0 & 0 & \dots & \dots & 0 & -1 & D_{N-1} & -1 \\ \hline 0 & 0 & \dots & \dots & \dots & \dots & 0 & 1 \end{array}\right)\left(\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ \hline u_N \end{array}\right) = \left(\begin{array}{c} 2\alpha\Delta x \\ \hline \Delta x^2 f(x_1) \\ \Delta x^2 f(x_2) \\ \vdots \\ \vdots \\ \Delta x^2 f(x_{N-2}) \\ \Delta x^2 f(x_{N-1}) \\ \hline \beta \end{array}\right) \overset{\text{def}}{=} \boldsymbol{F}$$

R. 5

L'algorithme non détaillé est simple

1: Initialisation des données
2: Calcul de la matrice $\mathbb{A}$
3: Calcul du second membre $\boldsymbol{F}$
4: Résolution du système $\mathbb{A}\boldsymbol{V} = \boldsymbol{F}$.

Pour écrire l'algorithme détaillé, nous pouvons écrire par exemple deux fonctions l'une permettant de calculer la matrice $\mathbb{A}$ et l'autre le second membre $\boldsymbol{F}$.
Pour celà, on peut écrire la fonction AssembleMat1D retournant une matrice $\mathbb{M} \in \mathcal{M}_d(\mathbb{R})$ définie par

$$\mathbb{M} = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 & \dots & \dots & 0 \\ \beta & a_1 & \beta & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \beta & a_{d-2} & \beta \\ 0 & \dots & \dots & \dots & 0 & 0 & \gamma \end{pmatrix} \tag{R3.37}$$

Dans ce cas la matrice $\mathbb{A}$ de la question précédente n'est qu'un cas particulier de la matrice $\mathbb{M}$.
Toutefois, ce choix de fonction n'est pas unique et on aurait pu choisir une fonction retournant, par exemple, la matrice

$$\begin{pmatrix} -3 & 4 & -1 & 0 & \dots & \dots & 0 \\ \beta & a_1 & \beta & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \beta & a_{d-2} & \beta \\ 0 & \dots & \dots & \dots & 0 & 0 & 1 \end{pmatrix}$$

La fonction AssembleMat1D peut s'écrire:

Algorithme 4.3 Fonction AssembleMat1D

Données : d : dimension de la matrice,
β, γ : deux réels,
$\boldsymbol{\alpha}$, : un vecteur de $\mathbb{R}^3$ tel que $\boldsymbol{\alpha}(i) = \alpha_i$,
$\boldsymbol{a}$, : un vecteur de $\mathbb{R}^{d-2}$ tel que $\boldsymbol{a}(i) = a_i$,
Résultat : $\mathbb{M}$: matrice $d \times d$

Fonction $x \leftarrow$ AssembleMat1D($d, \beta, \gamma, \boldsymbol{\alpha}, \boldsymbol{a}$)
  $\mathbb{M} \leftarrow \mathbb{0}_{d,d}$
  $\mathbb{M}(1,1) \leftarrow \boldsymbol{\alpha}(1)$, $\mathbb{M}(1,2) \leftarrow \boldsymbol{\alpha}(2)$, $\mathbb{M}(1,3) \leftarrow \boldsymbol{\alpha}(3)$
  $\mathbb{M}(d,d) \leftarrow \beta$
  Pour $i \leftarrow 2$ **à** $d-1$ **faire** ▷ Initialisation de la ligne i
    $\mathbb{M}(i,i) \leftarrow \boldsymbol{a}(i-1)$, $\mathbb{M}(i,i-1) \leftarrow \beta$, $\mathbb{M}(i,i+1) \leftarrow \beta$
  Fin
Fin

Ensuite on écrit la fonction AssembleSM retournant, par exemple, le vecteur $\boldsymbol{B} \in \mathbb{R}^d$ défini par

$$\boldsymbol{B} = \begin{pmatrix} a \\ \mu_1 \\ \vdots \\ \mu_{d-2} \\ b. \end{pmatrix} \tag{R3.38}$$

Dans ce cas, le second membre du système linéaire de la question précédente peut être obtenu à partir de cette fonction. On peut toutefois noter que de nombreuses variantes peuvent être proposer.
La fonction AssembleSM s'écrit:

Algorithme 4.4 Fonction AssembleSM

Données : d : dimension du vecteur,
a, b : deux réels,
$\boldsymbol{\mu}$, : un vecteur de $\mathbb{R}^3$ tel que $\boldsymbol{\mu}(i) = \mu_i$,
Résultat : $\boldsymbol{B}$: vecteur de $\mathbb{R}^d$

Fonction $\boldsymbol{B} \leftarrow$ AssembleSM($d, a, b, \boldsymbol{\mu}$)
  $\boldsymbol{B} \leftarrow \mathbb{0}_{d,1}$
  $\boldsymbol{B}(1) \leftarrow a$, $\boldsymbol{B}(d) \leftarrow b$
  Pour $i \leftarrow 2$ **à** $d-1$ **faire**
    $\boldsymbol{B}(i) \leftarrow \boldsymbol{\mu}(i-1)$
  Fin
Fin

Un algorithme complet utilisant ces fonctions pourrait être

$a \leftarrow 0$, $b \leftarrow 1$, $f : x \mapsto 0$, $c : x \mapsto 1 + x.^2$
$\alpha \leftarrow 0$, $\beta \leftarrow 1$
$N \leftarrow 1000$, $h \leftarrow (b-a)/N$, $\boldsymbol{x} \leftarrow a : h : b$
$\mathbb{A} \leftarrow$ AssembleMat1D$(N+1, -1, 1, [-3, 4, 1], 2 + h * h * c(\boldsymbol{x}(2:N)))$
$\boldsymbol{F} \leftarrow$ AssembleSM$(N+1, 2 * \alpha * h, \beta, h * h * f(\boldsymbol{x}(2:N)))$
$\boldsymbol{V} \leftarrow$ Solve$(\mathbb{A}, \boldsymbol{F})$

Fin correction

4.6 Problème modèle évolutif 1D : équation de la chaleur

Le problème modèle que nous allons résoudre numériquement par un schéma aux différences finies est le suivant

EDP modèle instationnaire en dimension 1 : équation de la chaleur

Trouver $u : [0,T] \times [a,b] \longrightarrow \mathbb{R}$ telle que

$$\frac{\partial u}{\partial t}(t,x) - D\frac{\partial^2 u}{\partial x^2}(t,x) = f(t,x), \qquad \forall (t,x) \in]0,T]\times]a,b[, \tag{4.56}$$

$$u(0,x) = u_0(x), \qquad \forall x \in [a,b] \tag{4.57}$$

$$-D\frac{\partial u}{\partial x}(t,a) = \alpha(t), \qquad \forall t \in [0,T] \tag{4.58}$$

$$u(t,b) = \beta(t), \qquad \forall t \in [0,T] \tag{4.59}$$

où $a < b$, $D > 0$ (coefficient de diffusivité), $\alpha : [0,T] \longrightarrow \mathbb{R}$, $\beta : [0,T] \longrightarrow \mathbb{R}$, $u_0 : [a,b] \longrightarrow \mathbb{R}$ et $f : [0,T] \times [a,b] \longrightarrow \mathbb{R}$ donnés.

Pour que le problème soit bien posé, il est nécessaire d'avoir compatibilité entre la condition initiale et la(les) condition(s) aux limites de type Dirichlet. Dans l'exemple précédent, (4.57) est la **condition initiale** et les **conditions aux limites** sont données par (4.58) (type Neumann) et (4.59) (type Dirichlet). Une **condition de compatibilité** n'apparait alors qu'au point $(t,x) = (0,b)$ et elle est donnée par

$$u_0(b) = \beta(0). \tag{4.60}$$

Comme dans le cas stationnaire, ce problème consiste en la recherche d'une fonction $u : [0,T] \times [a,b] \longrightarrow \mathbb{R}$ ou de manière équivalente en la recherche d'une infinité de valeurs $u(t,x)$, $\forall (t,x) \in [0,T] \times [a,b]$. On va donc se limiter à la recherche d'un nombre finies de valeurs. Pour celà on discrétise l'intervalle en temps et l'intervalle en espace. Soient $(x_i)_{i=0}^{N_x}$ la discrétisation régulière de l'intervalle $[a,b]$ en $N_x + 1$ points :

$$x_i = a + i\Delta_x, \quad \forall i \in [\![0, N_x]\!], \text{ avec } \Delta_x = \frac{b-a}{N_x}$$

et $(t^n)_{n=0}^{N_t}$ la discrétisation régulière de l'intervalle $[0,T]$ en $N_t + 1$ points :

$$t^n = n\Delta_t, \quad \forall n \in [\![0, N_t]\!], \text{ avec } \Delta_t = \frac{T}{N_t}.$$

Les couples de points (x_i, t^n), points bleus en Figure 4.9, forment une grille espace-temps et sont appelés les **noeuds** de la grille.

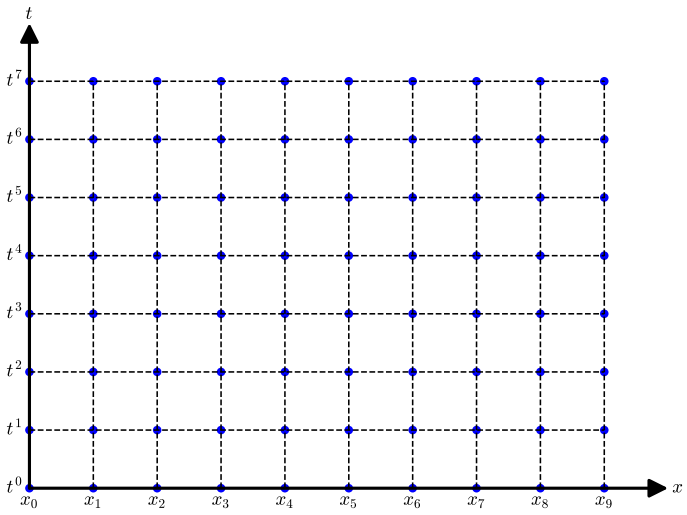


Figure 4.9: Représentation d'une grille espace-temps avec $N_t = 7$ et $N_x = 9$. Les noeuds de la grille sont les points bleus.

Du problème (4.56)-(4.59), on déduit le problème suivant écrit aux noeuds de la grille

EDP modèle d'évolution en dimension 1 : équation de la chaleur, formulation aux points de discrétisation

Trouver $u(t^n, x_i) \in \mathbb{R}$, $\forall n \in [\![0, N_t]\!]$, $\forall i \in [\![0, N_x]\!]$, tels que

$$\frac{\partial u}{\partial t}(t^n, x_i) - D\frac{\partial^2 u}{\partial x^2}(t^n, x_i) = f(t^n, x_i), \qquad \forall n \in]\!]0, N_t]\!], \ \forall i \in]\!]0, N_x[\![, \tag{4.61}$$

$$u(0, x_i) = u_0(x_i), \qquad \forall i \in [\![0, N_x]\!], \tag{4.62}$$

$$-D\frac{\partial u}{\partial x}(t^n, x_0) = \alpha(t^n), \qquad \forall n \in [\![0, N_t]\!] \tag{4.63}$$

$$u(t^n, x_{N_x}) = \beta(t^n), \qquad \forall n \in [\![0, N_t]\!] \tag{4.64}$$

Il nous faut maintenant approcher les opérateurs aux dérivées partielles. Pour celà on peut utiliser deux approches.

- Première approche : on utilise les résultats de la section 4.4.2 et pour *coller* aux notations de cette section, on note $\boldsymbol{x} \overset{\text{def}}{=} (\boldsymbol{x}_1, \boldsymbol{x}_2) = (t, x) \in \mathbb{R}^2$ (attention à ne pas confondre $\boldsymbol{x}_1$, première composante de $\boldsymbol{x} \in \mathbb{R}^2$, avec x_1 deuxième point de la discrétisation $(x_i)_{i \in [\![0, N_x]\!]}$). On a alors avec ces notations :

$$\frac{\partial u}{\partial t}(t, x) = \frac{\partial u}{\partial \boldsymbol{x}_1}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial u}{\partial \boldsymbol{x}_1}(\boldsymbol{x})$$
$$\frac{\partial^2 u}{\partial x^2}(t, x) = \frac{\partial^2 u}{\partial \boldsymbol{x}_2^2}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial^2 u}{\partial \boldsymbol{x}_2^2}(\boldsymbol{x})$$
$$\frac{\partial u}{\partial x}(t, x) = \frac{\partial u}{\partial \boldsymbol{x}_2}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial u}{\partial \boldsymbol{x}_2}(\boldsymbol{x})$$

  On déduit alors de la section 4.4.2 des approximations de ces dérivées. Par exemple, de la Proposition 4.4.5 on a, pour $h > 0$,

$$\frac{\partial^2 u}{\partial \boldsymbol{x}_2^2}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{u(\boldsymbol{x}_1, \boldsymbol{x}_2 + h) - 2u(\boldsymbol{x}_1, \boldsymbol{x}_2) + u(\boldsymbol{x}_1, \boldsymbol{x}_2 - h)}{h^2} + \mathcal{O}(h^2)$$

ce qui se *traduit* par

$$\frac{\partial^2 u}{\partial x^2}(t,x) = \frac{u(t,x+h) - 2u(t,x) + u(t,x-h)}{h^2} + \mathcal{O}(h^2) \tag{4.65}$$

- Deuxième approche : on utilise la formule de Taylor de la proposition 4.4.3 pour rechercher les *bonnes* approximations. C'est cette dernière approche que nous allons développer par la suite.

On commence par déterminer une approximation de $\frac{\partial u}{\partial t}(t,x)$. Soit $h > 0$. On peut pour celà utiliser la formule de Taylor 4.18 en $(t+h,x)$ ou en $(t-h,x)$. Il existe alors $\xi^+ \in]t, t+h[$ tel que

$$u(t+h,x) = u(t,x) + h\frac{\partial u}{\partial t}(t,x) + \frac{h^2}{2!}\frac{\partial^2 u}{\partial t^2}(\xi^+,x)$$

et il existe $\xi^- \in]t-h, t[$ tel que

$$u(t-h,x) = u(t,x) - h\frac{\partial u}{\partial t}(t,x) + \frac{(-h)^2}{2!}\frac{\partial^2 u}{\partial t^2}(\xi^-,x)$$

On peut noter que l'on peut remplacer *il existe* $\xi^+ \in]t, t+h[$ par *il existe* $\theta^+ \in]0,1[$ et poser $\xi^+ = t + h\theta^+$. (de même pour θ^-)
On obient alors les deux approximations d'ordre 1 suivantes

$$\frac{\partial u}{\partial t}(t,x) = \frac{u(t+h,x) - u(t,x)}{h} + \mathcal{O}(h) \tag{4.66}$$

$$\frac{\partial u}{\partial t}(t,x) = \frac{u(t,x) - u(t-h,x)}{h} + \mathcal{O}(h). \tag{4.67}$$

De manière similaire, on obtient les deux approximations d'ordre 1 suivantes

$$\frac{\partial u}{\partial x}(t,x) = \frac{u(t,x+h) - u(t,x)}{h} + \mathcal{O}(h) \tag{4.68}$$

$$\frac{\partial u}{\partial x}(t,x) = \frac{u(t,x) - u(t,x-h)}{h} + \mathcal{O}(h). \tag{4.69}$$

Pour déterminer une approximation de $\frac{\partial^2 u}{\partial x^2}(t,x)$ on utilise deux développements de Taylor (voir formule 4.18) en $(t+h,x)$ et en $(t-h,x)$. Il existe alors $\xi^+ \in]x, x+h[$ tel que

$$u(t,x+h) = u(t,x) + h\frac{\partial u}{\partial x}(t,x) + \frac{h^2}{2!}\frac{\partial^2 u}{\partial x^2}(t,x) + \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}(t,x) + \frac{h^4}{4!}\frac{\partial^4 u}{\partial x^4}(t,\xi^+)$$

et il existe $\xi^- \in]x-h, x[$ tel que

$$u(t,x-h) = u(t,x) - h\frac{\partial u}{\partial x}(t,x) + \frac{(-h)^2}{2!}\frac{\partial^2 u}{\partial x^2}(t,x) + \frac{(-h)^3}{3!}\frac{\partial^3 u}{\partial x^3}(t,x) + \frac{(-h)^4}{4!}\frac{\partial^4 u}{\partial x^4}(t,\xi^-)$$

En ajoutant ces deux équations on obtient

$$u(t,x+h) + u(t,x-h) = 2u(t,x) + h^2\frac{\partial^2 u}{\partial x^2}(t,x) + \frac{h^4}{4!}\left(\frac{\partial^4 u}{\partial x^4}(t,\xi^+) + \frac{\partial^4 u}{\partial x^4}(t,\xi^-)\right)$$

ce qui donne l'approximation d'ordre 2

$$\frac{\partial^2 u}{\partial x^2}(t,x) = \frac{u(t,x+h) - 2u(t,x) + u(t,x-h)}{h^2} + \mathcal{O}(h^2) \tag{4.70}$$

En choisissant $h = \Delta_x$, on déduit de (4.71) la formule suivante :

$$\frac{\partial^2 u}{\partial x^2}(t^n,x_i) = \frac{u(t^n,x_{i+1}) - 2u(t^n,x_i) + u(t^n,x_{i-1})}{\Delta_x^2} + \mathcal{O}(\Delta_x^2) \tag{4.71}$$

car $x_{i+1} = x_i + \Delta_x$ et $x_{i-1} = x_i - \Delta_x$.
De même, on note qu'en choisissant $h = \Delta_t$, on obtient de (4.66) et (4.67) les deux formules suivantes :

$$\frac{\partial u}{\partial t}(t^n,x_i) = \frac{u(t^{n+1},x_i) - u(t^n,x_i)}{\Delta_t} + \mathcal{O}(\Delta_t) \tag{4.72}$$

$$\frac{\partial u}{\partial t}(t^n,x_i) = \frac{u(t^n,x_i) - u(t^{n-1},x_i)}{\Delta_t} + \mathcal{O}(\Delta_t). \tag{4.73}$$

car $t^{n+1} = t^n + \Delta_x$ et $t^{n-1} = t^n - \Delta_x$.
En approchant, dans (4.61), la dérivée partielle en temps par la formule (4.72) on abouti à un schéma **explicite en temps** : ceci est l'objet de la prochaine section. En utilisant la formule (4.72) on obtient un schéma **implicite en temps** qui sera étudié en section 4.6.2.

4.6.1 Schéma explicite en temps

En utilisant (4.71) et (4.72), l'équation (4.61) peut donc s'écrire

$$\frac{u(t^{n+1},x_i)-u(t^n,x_i)}{\Delta_t}+\mathcal{O}(\Delta_t)-D\frac{u(t^n,x_{i+1})-2u(t^n,x_i)+u(t^n,x_{i-1})}{\Delta_x^2}+\mathcal{O}(\Delta_x^2)=f(t^n,x_i) \quad (4.74)$$

et elle n'a de sens que si $n \in [\![0,N_t[\![$ et si $i \in]\!]0,N_x[\![$. On en déduit alors le schéma numérique d'ordre 1 en temps et d'ordre 2 en espace suivant

$$\frac{\boldsymbol{u}_i^{n+1}-\boldsymbol{u}_i^n}{\Delta_t}-D\frac{\boldsymbol{u}_{i+1}^n-2\boldsymbol{u}_i^n+\boldsymbol{u}_{i-1}^n}{\Delta_x^2}=\boldsymbol{f}_i^n,\ \forall n \in [\![0,N_t[\![,\ \forall i \in]\!]0,N_x[\![\quad (4.75)$$

en posant $\boldsymbol{f}_i^n=f(t^n,x_i)$ et (en espérant) $\boldsymbol{u}_i^n \approx u(t^n,x_i)$. Ce schéma peut aussi s'écrire sous la forme

$$\boldsymbol{u}_i^{n+1}=\boldsymbol{u}_i^n+D\frac{\Delta_t}{\Delta_x^2}\left(\boldsymbol{u}_{i+1}^n-2\boldsymbol{u}_i^n+\boldsymbol{u}_{i-1}^n\right)+\Delta_t\boldsymbol{f}_i^n,\ \forall n \in [\![0,N_t[\![,\ \forall i \in]\!]0,N_x[\![\quad (4.76)$$

Ce schéma est **explicite en temps**. En fait pour déterminer $\boldsymbol{u}_i^{n+1}$ il suffit de connaître les trois valeurs $\boldsymbol{u}_{i-1}^n$, $\boldsymbol{u}_i^n$ et $\boldsymbol{u}_{i+1}^n$ comme illustré sur la Figure 4.10. Il faut toutefois noter que l'on ne peut utiliser ce schéma pour déterminer les valeurs aux limites (i.e. $\boldsymbol{u}_0^{n+1}$ et $\boldsymbol{u}_{N_x}^{n+1}$) comme illustré en Figure 4.11

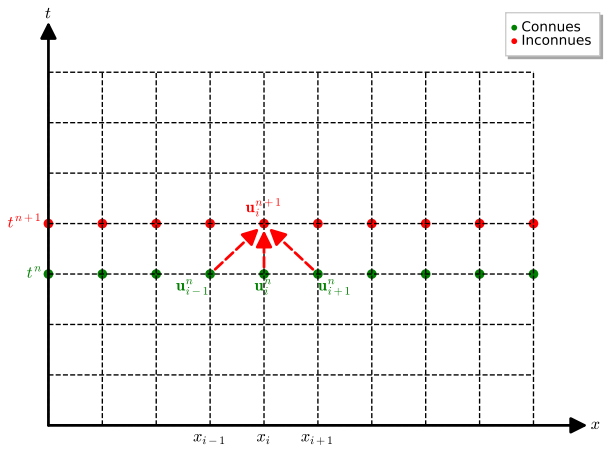


Figure 4.10: Calcul de $\boldsymbol{u}_i^{n+1}$ par le schéma explicite 4.76 : graphe de dépendance

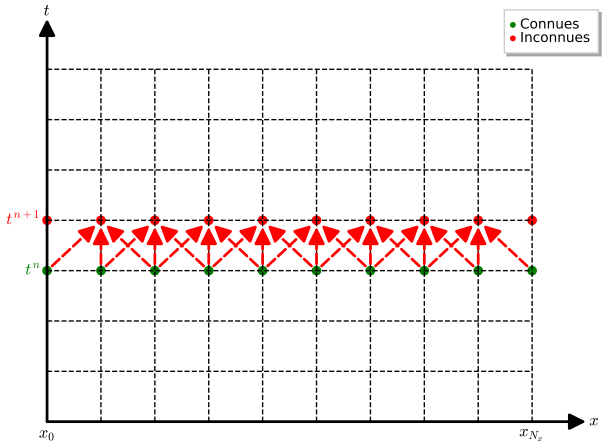


Figure 4.11: Graphes de dépendance pour les calculs de $\boldsymbol{u}_i^{n+1}$, $\forall i \in]\!]0, N_x[\![$ par le schéma explicite 4.76

Il nous reste donc à déterminer les valeurs aux limites $\boldsymbol{u}_0^{n+1}$ et $\boldsymbol{u}_{N_x}^{n+1}$ pour connaître l'intégralité des valeurs souhaitées au temps t^{n+1}. Nous allons pour se faire utiliser les conditions aux limites (4.63) et (4.64) au temps t^{n+1}.
La plus simple à traiter ici est la condition aux limites de Dirichlet (4.64) qui nous donne directement et sans approximation :

$$\boldsymbol{u}_{N_x}^{n+1} = u(t^{n+1}, x_{N_x}) = \beta(t^{n+1}). \tag{4.77}$$

Pour le calcul de la valeur $\boldsymbol{u}_0^{n+1}$, on va utiliser (4.63) au temps t^{n+1}. Il va nous falloir bien évidemment approcher $\frac{\partial u}{\partial x}(t^n, x_0)$. La première idée est d'utiliser l'une des formules (4.68) ou (4.69) en (t^{n+1}, x_0). On note que la formule (4.69) n'est pas utilisable car $x_0 - \Delta_x$ est hors intervalle. On obtient alors avec (4.68) :

$$\frac{\partial u}{\partial x}(t^{n+1}, x_0) = \frac{u(t^{n+1}, x_1) - u(t^{n+1}, x_0)}{\Delta_x} + \mathcal{O}(\Delta_x) \tag{4.78}$$

En remplaçant dans (4.63) on obtient

$$-D\frac{u(t^{n+1}, x_1) - u(t^{n+1}, x_0)}{\Delta_x} + \mathcal{O}(\Delta_x) = \alpha(t^{n+1}). \tag{4.79}$$

Ce qui donne le schéma d'ordre 1 en espace

$$-D\frac{\boldsymbol{u}_1^{n+1} - \boldsymbol{u}_0^{n+1}}{\Delta_x} = \alpha(t^{n+1})$$

ou encore

$$\boldsymbol{u}_0^{n+1} = \frac{\Delta_x}{D}\alpha(t^{n+1}) + \boldsymbol{u}_1^{n+1}. \tag{4.80}$$

Deux remarques sur cette formule :

a. il faut d'abord calculer $\boldsymbol{u}_1^{n+1}$ à l'aide du schéma (4.76) avant de l'utiliser (ce qui ne pose aucun problème).

b. elle est d'ordre 1 en espace, ce qui nous fait perdre tout l'avantage de l'ordre 2 en espace du schéma (4.76) : en utilisant (4.76), (4.80) et (4.77), on abouti à un schéma global d'ordre 1 en temps et d'ordre 1 en espace!. Avec un petit peu plus de travail, on peut obtenir à un schéma d'ordre 1 en temps et d'ordre 2 en espace.

Pour obtenir une formule d'ordre 2 approchant (4.63), il faut une approximation d'ordre 2 de $\frac{\partial u}{\partial x}(t^{n+1}, x_0)$. Comme dans le cas stationnaire, on écrit les formules de Taylor en $(t, x+h)$ et $(t, x+2h)$ (on remplacera ensuite t par t^{n+1}, x par x_0 et h par Δ_x) : Il existe alors $\xi^1 \in]t, t+h[$ tel que

$$u(t,x+h) = u(t,x) + h\frac{\partial u}{\partial x}(t,x) + \frac{h^2}{2!}\frac{\partial^2 u}{\partial x^2}(t,x) + \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}(t,\xi_1) \tag{4.81}$$

et il existe $\xi_2 \in]t, t+2h[$ tel que

$$u(t,x+2h) = u(t,x) + 2h\frac{\partial u}{\partial x}(t,x) + \frac{(2h)^2}{2!}\frac{\partial^2 u}{\partial x^2}(t,x) + \frac{(2h)^3}{3!}\frac{\partial^3 u}{\partial x^3}(t,\xi_2). \tag{4.82}$$

En effectuant la combinaison linéaire $4 \times (4.81) - (4.82)$ qui permet de supprimer les dérivées secondes, on obtient

$$4u(t,x+h) - u(t,x+2h) = 3u(t,x) + 2h\frac{\partial u}{\partial x}(t,x) + \mathcal{O}(h^3)$$

ce qui donne une approximation d'ordre 2

$$\frac{\partial u}{\partial x}(t,x) = \frac{-3u(t,x) + 4u(t,x+h) - u(t,x+2h)}{2h} + \mathcal{O}(h^2).$$

On a alors en (t^{n+1}, x_0)

$$\frac{\partial u}{\partial x}(t^{n+1},x_0) = \frac{-3u(t^{n+1},x_0) + 4u(t^{n+1},x_1) - u(t^{n+1},x_2)}{2\Delta_x} + \mathcal{O}(\Delta_x^2). \tag{4.83}$$

En remplaçant dans (4.63) on obtient

$$-D\frac{-3u(t^{n+1},x_0) + 4u(t^{n+1},x_1) - u(t^{n+1},x_2)}{2\Delta_x} + \mathcal{O}(\Delta_x^2) = \alpha(t^{n+1}). \tag{4.84}$$

Ce qui donne le schéma d'ordre 2 en espace

$$-D\frac{-3\boldsymbol{u}_0^{n+1} + 4\boldsymbol{u}_1^{n+1} - \boldsymbol{u}_2^{n+1}}{2\Delta_x} = \alpha(t^{n+1})$$

ou encore

$$\boldsymbol{u}_0^{n+1} = \frac{1}{3}\left(\frac{2\Delta_x}{D}\alpha(t^{n+1}) + 4\boldsymbol{u}_1^{n+1} - \boldsymbol{u}_2^{n+1}\right). \tag{4.85}$$

Il faut noter que pour utiliser cette formule, il faut avoir calculé au préalable les deux valeurs $\boldsymbol{u}_1^{n+1}$ et $\boldsymbol{u}_2^{n+1}$.
En résumé, soit $n \in [\![0, N_t[\![$, on vient de voir que si l'on connait $\boldsymbol{u}_i^n$, $\forall i \in [\![0, N_x]\!]$ alors le calcul des $\boldsymbol{u}_i^{n+1}$, $\forall i \in [\![0, N_x]\!]$, est possible grâce aux formules (4.76) (4.77) et (4.85) rappellées ici en notant $E = D\frac{\Delta_t}{\Delta_x^2}$ et $C = 1 - 2E$:

$$\boldsymbol{u}_i^{n+1} = C\boldsymbol{u}_i^n + E(\boldsymbol{u}_{i+1}^n + \boldsymbol{u}_{i-1}^n) + \Delta_t \boldsymbol{f}_i^n, \ \forall i \in]\!]0, N_x[\![\tag{4.76}$$

$$\boldsymbol{u}_{N_x}^{n+1} = \beta(t^{n+1}), \tag{4.77}$$

$$\boldsymbol{u}_0^{n+1} = \frac{1}{3}\left(\frac{2\Delta_x}{D}\alpha(t^{n+1}) + 4\boldsymbol{u}_1^{n+1} - \boldsymbol{u}_2^{n+1}\right). \tag{4.85}$$

Algorithmique

Pour initialiser ce schéma itératif, on utilise la condition initiale (4.62) qui donne directement

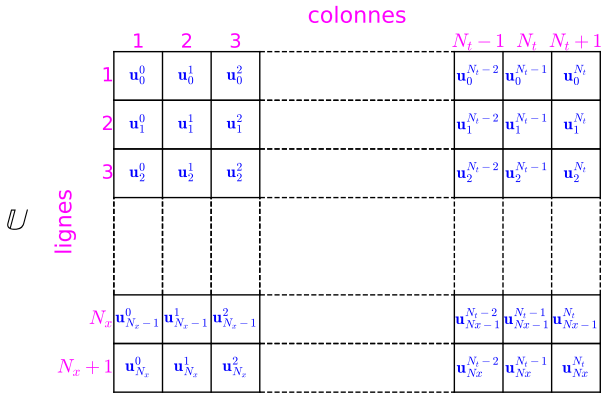
$$\boldsymbol{u}_i^0 = u_0(x_i), \ \forall i \in [\![0, N_x]\!]. \tag{4.86}$$

L'algorithme formel est donc le suivant :

1: $\boldsymbol{u}_i^0 \leftarrow u_0(x_i), \ \forall i \in [\![0, N_x]\!]$
2: **Pour** $n \leftarrow 0$ **à** $N_t - 1$ **faire**
3: Calculer $\boldsymbol{u}_i^{n+1}$, $\forall i \in]\!]0, N_x[\![$ par (4.76)
4: Calculer $\boldsymbol{u}_{N_x}^{n+1}$, par (4.77)
5: Calculer $\boldsymbol{u}_0^{n+1}$, par (4.85)
6: **Fin**

Pour obtenir l'algorithme complet, on décrit tout d'abord ce que l'on cherche :

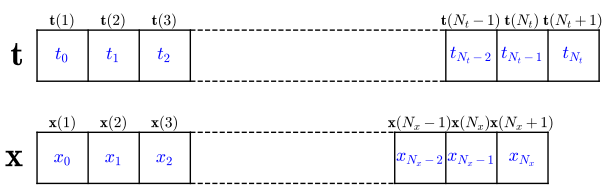
Résultat : $\mathbb{U}$: tableau 2D/matrice $(N_x+1)\times(N_t+1)$ de réels tel que $\mathbb{U}(i+1,n+1)=\boldsymbol{u}_i^n$, $\forall i\in[\![0,N_x]\!]$, $\forall n\in[\![0,N_t]\!]$.



Ensuite, on décrit l'ensemble de données nécessaire à la résolution de (4.56)-(4.59):

Données :
a,b : deux réels $a<b$,
T : $T>0$,
D : D réel strictement positif, (coefficient de diffusivité)
f : $f:[0,T]\times[a,b]\longrightarrow\mathbb{R}$,
u_0 : $u_0:[a,b]\longrightarrow\mathbb{R}$,
α : $\alpha:[0,T]\longrightarrow\mathbb{R}$,
β : $\beta:[0,T]\longrightarrow\mathbb{R}$, tel que $\beta(0)=u_0(b)$,
N_x : $N_x\in\mathbb{N}^*$, nombre de discrétisation en espace,
N_t : $N_t\in\mathbb{N}^*$, nombre de discrétisation en temps.

On choisi ici d'écrire l'algorithme sous forme d'une fonction retournant $\mathbb{U}$ ainsi que les discrétisations en espace et en temps stockées respectivement dans $\mathbf{x}\in\mathbb{R}^{N_x+1}$ et $\mathbf{t}\in\mathbb{R}^{N_t+1}$. Les vecteurs $\mathbf{x}$ et $\mathbf{t}$ sont tels que $\mathbf{x}(i+1)=x_i$, $\forall i\in[\![0,N_x]\!]$, et $\mathbf{t}(n+1)=t^n$, $\forall n\in[\![0,N_t]\!]$.



Algorithm 4.5 Fonction Heat1Dex (version non vectorisée)

Fonction $[\mathbf{t}, \mathbf{x}, \mathbb{U}] \leftarrow$ Heat1Dex($a, b, T, D, f, u_0, \alpha, \beta, N_x, N_t$)
  $\mathbf{t} \leftarrow$ DisReg$(0, T, N_t)$
  $\Delta_t \leftarrow T/N_t$
  $\mathbf{x} \leftarrow$ DisReg(a, b, N_x)
  $\Delta_x \leftarrow (b-a)/N_x$
  $E \leftarrow D\frac{\Delta_t}{\Delta_x^2}$, $C \leftarrow 1 - 2E$
  Pour $i \leftarrow 1$ **à** $N_x + 1$ **faire** ▷ Condition initiale
    $\mathbb{U}(i, 1) \leftarrow u_0(\mathbf{x}(i))$
  Fin
  Pour $n \leftarrow 1$ **à** N_t **faire** ▷ Boucle en temps
    Pour $i \leftarrow 2$ **à** N_x **faire** ▷ Schéma
      $\mathbb{U}(i, n+1) \leftarrow C * \mathbb{U}(i, n) + E * (\mathbb{U}(i+1, n) + \mathbb{U}(i-1, n)) + \Delta_t f(\mathbf{t}(n), \mathbf{x}(i))$
    Fin
    $\mathbb{U}(N_x, n+1) \leftarrow \beta(\mathbf{t}(n+1))$
    $\mathbb{U}(1, n+1) \leftarrow (\frac{\Delta_x}{D}\alpha(\mathbf{t}(n+1)) + 4\mathbb{U}(2, n+1) - \mathbb{U}(3, n+1))/3$
  Fin
Fin

En propose avec l'Algorithme 4.6, une version équivalente de cet algorithme. Celui-ci a l'avantage d'être plus concis et adapté à la programmation avec des languages dit vectorisés comme Matlab, Octave, python, scilab, R, C++ avec STL, ...

Algorithm 4.6 Fonction Heat1Dex (version vectorisée)

Fonction $[\mathbf{t}, \mathbf{x}, \mathbb{U}] \leftarrow$ Heat1Dex($a, b, T, D, f, u_0, \alpha, \beta, N_x, N_t$)
  $\mathbf{t} \leftarrow$ DisReg$(0, T, N_t)$, $\Delta_t \leftarrow T/N_t$
  $\mathbf{x} \leftarrow$ DisReg(a, b, N_x), $\Delta_x \leftarrow (b-a)/N_x$
  $E \leftarrow D\frac{\Delta_t}{\Delta_x^2}$, $C \leftarrow 1 - 2E$
  $\mathbb{U}(:, 1) \leftarrow u_0(\mathbf{x})$ ▷ Condition initiale
  $\mathbf{I} \leftarrow [2 : N_x]$ ▷ Indices des points intérieurs
  Pour $n \leftarrow 1$ **à** N_t **faire** ▷ Boucle en temps
    $\mathbb{U}(\mathbf{I}, n+1) \leftarrow C * \mathbb{U}(\mathbf{I}, n) - E * (\mathbb{U}(\mathbf{I}+1, n) + \mathbb{U}(\mathbf{I}-1, n)) + \Delta_t f(\mathbf{t}(n), \mathbf{x}(\mathbf{I}))$
    $\mathbb{U}(N_x, n+1) \leftarrow \beta(\mathbf{t}(n+1))$
    $\mathbb{U}(1, n+1) \leftarrow (\frac{\Delta_x}{D}\alpha(\mathbf{t}(n+1)) + 4\mathbb{U}(2, n+1) - \mathbb{U}(3, n+1))/3$
  Fin
Fin

Le code Matlab/Octave correspondant à l'algorithme vectorisé 4.6 est donné en Listing

```
function [t,x,U]=Heat1DexLight(a,b,T,D,f,u0,alpha,beta,Nx,Nt)
  dt=T/Nt;      t=0:dt:T;
  dx=(b-a)/Nx; x=[a:dx:b]';
  U=zeros(Nx+1,Nt+1);
  E=D*dt/dx^2;C=1-2*E;
  I=2:Nx;
  U(:,1)=u0(x);
  for n=1:Nt
    U(I,n+1)=C*U(I,n) + E*(U(I+1,n)+U(I-1,n)) + dt*f(t(n),x(I));
    U(Nx+1,n+1)=beta(t(n+1));
    U(1,n+1)=((2*dx/D)*alpha(t(n+1)) + 4*U(2,n+1) - U(3,n+1))/3;
  end
end
```

Listing 4.4: fonction pour la résolution de l'EDP de la chaleur (4.56)-(4.59) par le schéma **explicite** (4.76),(4.77),(4.85): fichier `Heat1DexLight.m`

Application avec solution exacte

Pour obtenir un problème avec solution exacte, on peut fixer la solution exacte, par exemple :

$$u(t,x) \stackrel{\text{def}}{=} cos(t)cos(x)$$

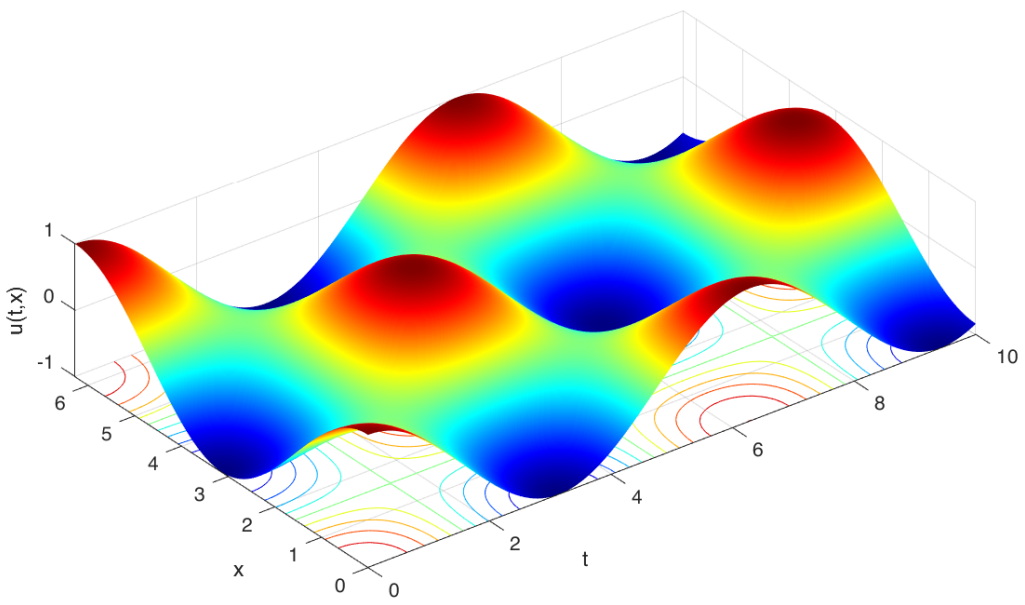
et injecter cette solution dans (4.56)-(4.59) pour déterminer les fonctions f, u_0, α et β. On obtient alors

$$f(t,x) \stackrel{\text{def}}{=} D\cos(t)\cos(x) - \sin(t)\cos(x),\ u_0(x) \stackrel{\text{def}}{=} \cos(x),\ \alpha(t) \stackrel{\text{def}}{=} D\cos(t)\sin(a) \text{ et } \beta(t) \stackrel{\text{def}}{=} \cos(t)\cos(b).$$

En Listing 14, on calcule la solution numérique de l'équation de la chaleur (4.56)-(4.59) avec les données précédentes et avec

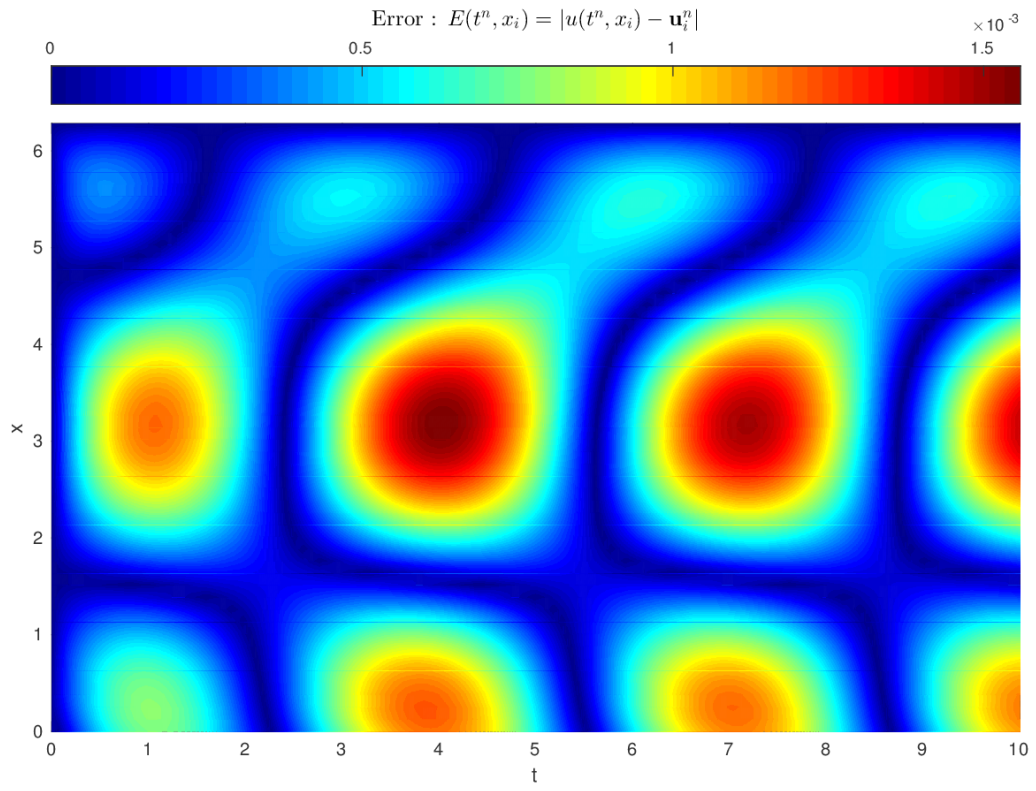
$$D = 1,\ a = 0,\ b = 2\pi,\ T = 10,\ N_x = 50 \text{ et } N_t = 5000$$

Le code proposé en Listing 11 et qui fait suite au Listing 14, permet de calculer la solution exacte aux noeuds de la grille espace-temps et de représenter l'erreur commise : $E(t^n, x_i) = |u(t^n, x_i) - \boldsymbol{u}_i^n|$.




```
a=0;b=2*pi;Nx=50;
T=10;Nt=5000
D=1;
u=@(t,x) cos(t).*cos(x); % solution exacte
f=@(t,x) D*cos(t).*cos(x) - sin(t).*cos(x);
beta=@(t) u(t,b);
alpha=@(t)  D*cos(t).*sin(a);
u0=@(x) cos(x);
[t,x,U]=Heat1DexLight(a,b,T,D,f,u0,alpha,beta,Nx,Nt);
surfc(t,x,U)
xlabel('t'),ylabel('x'),zlabel('u(t,x)')
shading interp, colormap(jet)
axis image
```

Listing 4.5: Equation de la chaleur 1D avec solution exacte. Représentation de la solution calculée. : code Matlab




```
beta=@(t) u(t,b);
alpha=@(t)  D*cos(t).*sin(a);
u0=@(x) cos(x);
[t,x,U]=Heat1DexLight(a,b,T,D,f,u0,alpha,beta,Nx,Nt);
surfc(t,x,U)
xlabel('t'),ylabel('x'),zlabel('u(t,x)')
shading interp, colormap(jet)
axis image
[T,X]=meshgrid(t,x);
Uex=u(T,X);
```

Listing 4.6: Equation de la chaleur 1D avec solution exacte. Représentation de l'erreur : code Matlab

On représente en Figure , les résultats de ces deux derniers codes mais avec cette fois $N_x = 100$ (au lieu de $N_x = 50$)

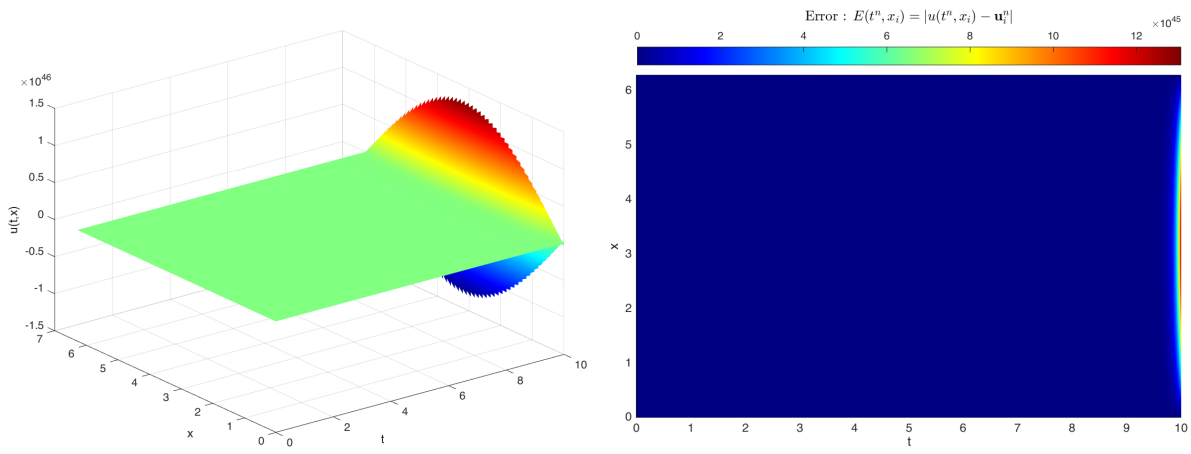


Figure 4.12: Equation de la chaleur 1D avec solution exacte. Phénomène d'instabilité.

Pour mieux visualiser ce phénomène, on représente en Figure 4.13 la solution calculée à différents pas de temps correspondant au début de nos soucis :

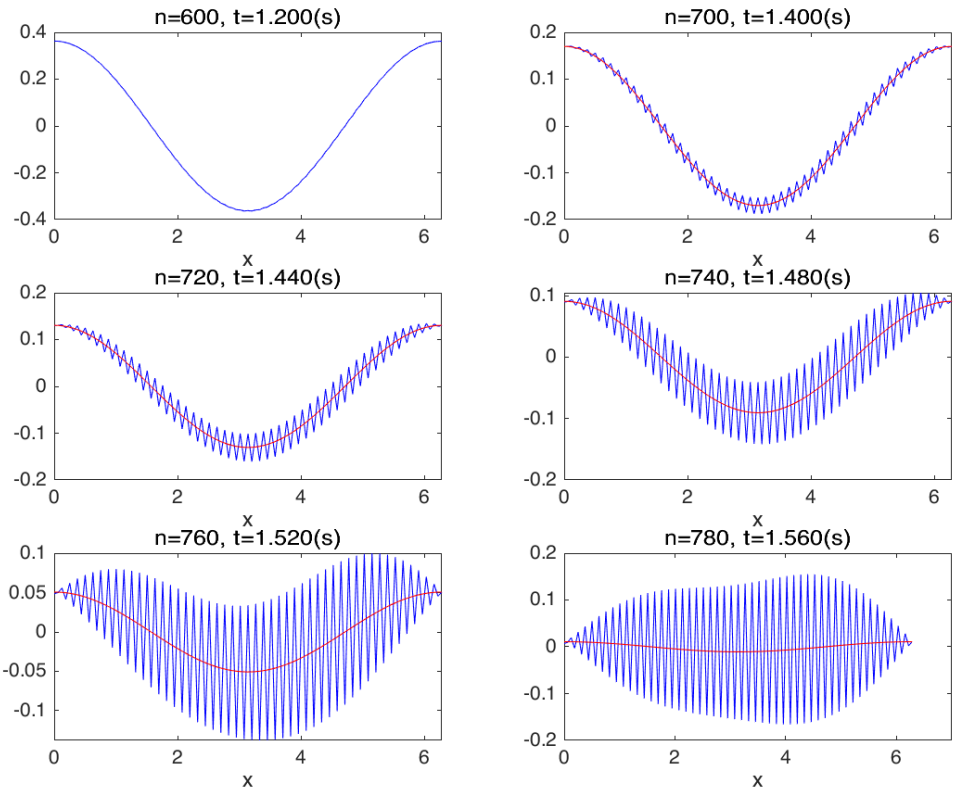


Figure 4.13: Equation de la chaleur 1D avec solution exacte (en rouge). Debut du phénomène d'instabilité.

Une étude de stabilité au sens de Von Neumann du schéma explicite pour l'équation de la chaleur (4.56) permet d'obtenir la **condition de C.F.L.** (R. Courant, K. Friedrichs, and H. Lewy en 1928) nécessaire à la stabilité du schéma :

$$D\frac{\Delta_t}{\Delta_x^2} \leqslant \frac{1}{2}. \tag{4.87}$$

On pourra se référer à [?] (section 2.2.3, page 37-42) pour plus de détails. On dit alors que le schéma est **conditionnellement stable**.

En Figure 4.14, on représente en échelle logarithmique l'erreur commise en fonction de Δ_x et Δ_t calculée par

$$E(\Delta_x, \Delta_t) = \max_{i\in[\![0,N_x]\!], n\in[\![0,N_t]\!]} |\boldsymbol{u}_i^n - u_{\text{ex}}(t^n, x_i)|, \quad \Delta_x = \frac{b-a}{N_x}, \ \Delta_t = \frac{T}{N_t}$$

où u_{ex} est la solution exacte. On peut noter qu'au dessus de la courbe $D\frac{\Delta_t}{\Delta_x^2} = \frac{1}{2}$ i.e. lorsque $D\frac{\Delta_t}{\Delta_x^2} \geqslant \frac{1}{2}$ l'erreur est supérieur à 100% (blocs gris). Par contre en dessous, i.e. lorsque la condition de CFL est vérifiée, le schéma donne de bons résultats.

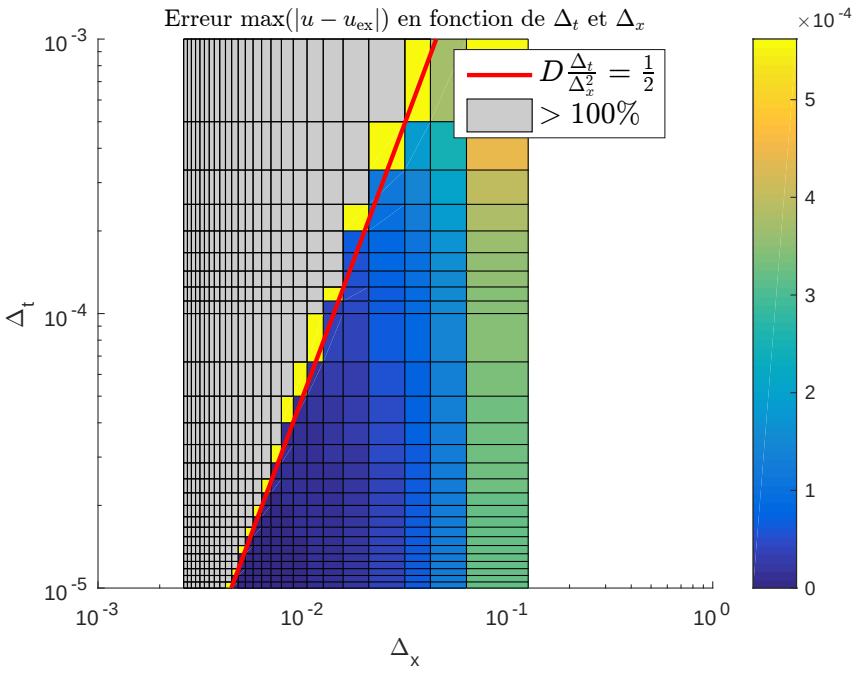


Figure 4.14: Equation de la chaleur 1D avec solution exacte. Condition de CFL.

En Figure 4.17, on représente les ordres du schéma : ordre 1 en temps (à gauche) et ordre 2 en espace (à droite). Toutefois, les phénomènes d'instabilités viennent perturber les graphes.

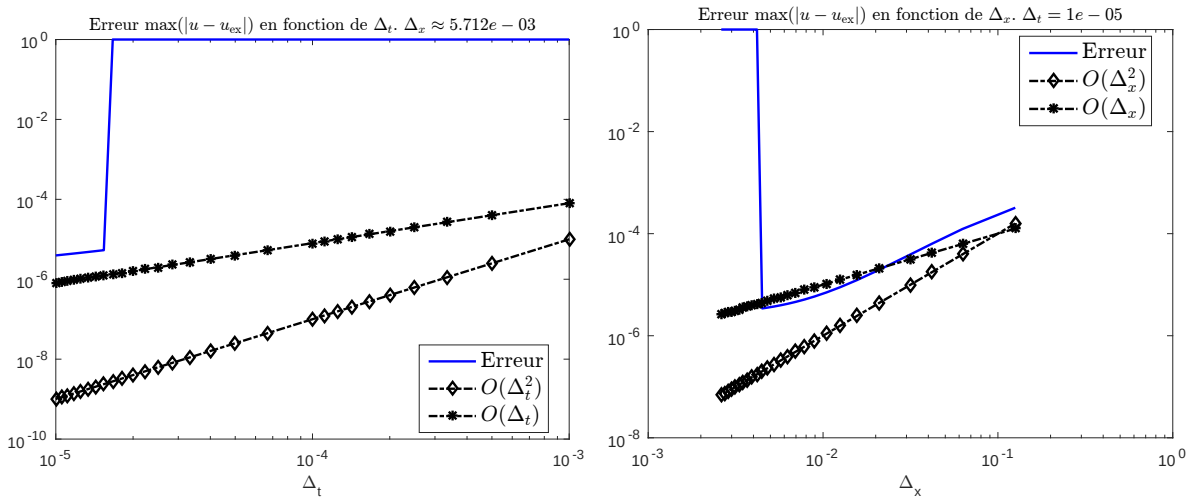


Figure 4.15: Equation de la chaleur 1D avec solution exacte. Ordres et phénomène d'instabilité.

En section 4.6.2, nous étudirons un schéma **implicite** qui sera **inconditionnellement stable**.

Application : barre chauffée en ses extremitées

On souhaite modéliser une barre isolée chauffée en ses extremitées (il n'y a donc pas d'échange thermique avec l'extérieur). La barre est en aluminium, mesure $20cm$ et elle est à 20^o au temps initial. La barre correspond à l'intervalle $[a, b]$ avec $a = 0$, $b = 0.2$.
Le problème à résoudre est alors

Barre chauffée en ses extremitées

Trouver $u : [0,T] \times [a,b] \longrightarrow \mathbb{R}$ telle que

$$\frac{\partial u}{\partial t}(t,x) - D\frac{\partial^2 u}{\partial x^2}(t,x) = 0, \qquad \forall (t,x) \in]0,T]\times]a,b[, \tag{4.88}$$

$$u(0,x) = 20, \qquad \forall x \in [a,b] \tag{4.89}$$

$$u(t,a) = \alpha(t), \qquad \forall t \in [0,T] \tag{4.90}$$

$$u(t,b) = \beta(t), \qquad \forall t \in [0,T] \tag{4.91}$$

où $D = 98.8 \times 10^{-6}$ pour l'aluminimum. Les fonctions α et β sont données par :

$$\alpha(t) = \begin{cases} 20 + 80t, & \text{si } t < 1 \\ 100 & \text{si } t \geqslant 1 \end{cases} \quad \text{et} \quad \beta(t) = \begin{cases} 20 + 40t, & \text{si } t < 1 \\ 60 & \text{si } t \geqslant 1 \end{cases}$$

On note que ces fonctions sont continues et vérifient les conditions de compatibilité :

$$\alpha(0) = u_0(a) = 20 \quad \text{et} \quad \beta(b) = u_0(b) = 20.$$

Dans l'algorithme 4.6 seul change la prise en compte de la condition aux limites en a : il faut donc remplacer la ligne 10

$$\mathbb{U}(1, n+1) \leftarrow (\frac{\Delta_x}{D}\alpha(\boldsymbol{t}(n+1)) + 4\mathbb{U}(2, n+1) - \mathbb{U}(3, n+1))/3$$

par

$$\mathbb{U}(1, n+1) \leftarrow \alpha(\boldsymbol{t}(n+1))$$

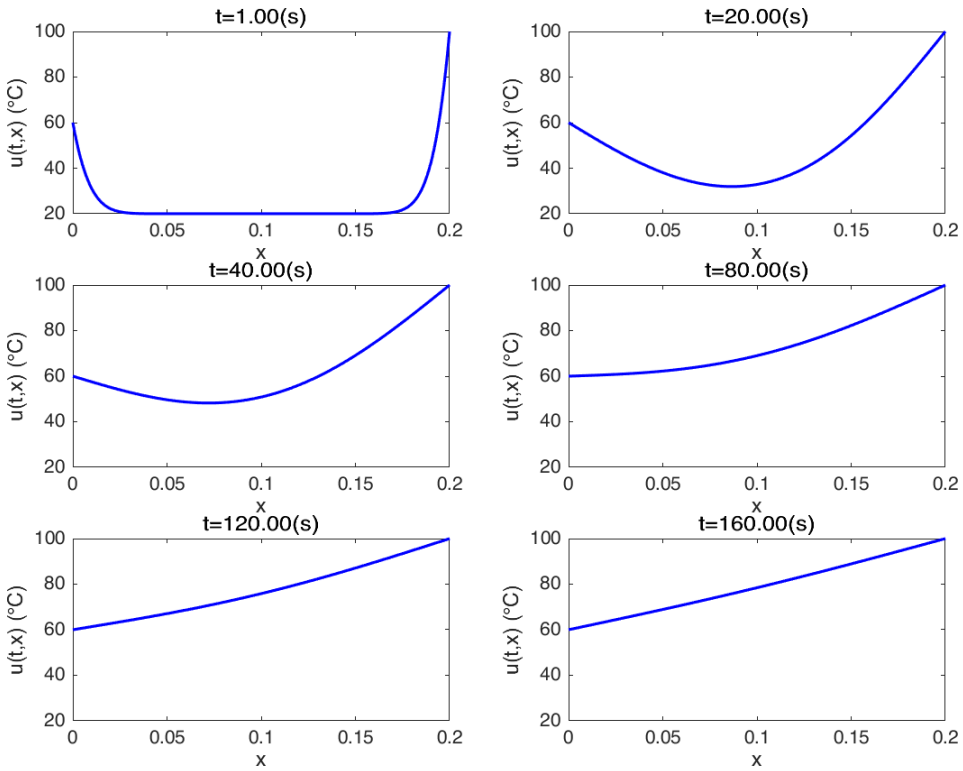


Figure 4.16: Application barre chauffée.

4.6.2 Schéma implicite en temps

Pour faciliter la lecture, on rappelle le problème modèle écrit aux noeuds de la grille

EDP modèle d'évolution en dimension 1 : équation de la chaleur, formulation aux points de discrétisation

Trouver $u(t^n, x_i) \in \mathbb{R}$, $\forall n \in [\![0, N_t]\!]$, $\forall i \in [\![0, N_x]\!]$, tels que

$$\frac{\partial u}{\partial t}(t^n, x_i) - D\frac{\partial^2 u}{\partial x^2}(t^n, x_i) = f(t^n, x_i), \qquad \forall n \in]\!]0, N_t]\!], \ \forall i \in]\!]0, N_x[\!], \tag{4.61}$$

$$u(0, x_i) = u_0(x_i), \qquad \forall i \in [\![0, N_x]\!], \tag{4.62}$$

$$-D\frac{\partial u}{\partial x}(t^n, x_0) = \alpha(t^n), \qquad \forall n \in [\![0, N_t]\!] \tag{4.63}$$

$$u(t^n, x_{N_x}) = \beta(t^n), \qquad \forall n \in [\![0, N_t]\!] \tag{4.64}$$

En utilisant (4.71) et (4.73) dans l'équation (4.61), on obtient

$$\frac{u(t^n, x_i) - u(t^{n-1}, x_i)}{\Delta_t} + \mathcal{O}(\Delta_t) - D\frac{u(t^n, x_{i+1}) - 2u(t^n, x_i) + u(t^n, x_{i-1})}{\Delta_x^2} + \mathcal{O}(\Delta_x^2) = f(t^n, x_i) \tag{4.92}$$

et elle n'a de sens que si $n \in]\!]0, N_t]\!]$ et si $i \in]\!]0, N_x[\!]$. On en déduit alors le schéma numérique d'ordre 1 en temps et d'ordre 2 en espace suivant

$$\frac{\boldsymbol{u}_i^n - \boldsymbol{u}_i^{n-1}}{\Delta_t} - D\frac{\boldsymbol{u}_{i+1}^n - 2\boldsymbol{u}_i^n + \boldsymbol{u}_{i-1}^n}{\Delta_x^2} = \boldsymbol{f}_i^n, \ \forall n \in]\!]0, N_t]\!], \ \forall i \in]\!]0, N_x[\!] \tag{4.93}$$

en posant $\boldsymbol{f}_i^n = f(t^n, x_i)$ et (en espérant) $\boldsymbol{u}_i^n \approx u(t^n, x_i)$. Ce schéma peut aussi s'écrire sous la forme

$$\boldsymbol{u}_i^n - D\frac{\Delta_t}{\Delta_x^2}\left(\boldsymbol{u}_{i+1}^n - 2\boldsymbol{u}_i^n + \boldsymbol{u}_{i-1}^n\right) = \boldsymbol{u}_i^{n-1} + \Delta_t \boldsymbol{f}_i^n, \ \forall n \in]\!]0, N_t]\!], \ \forall i \in]\!]0, N_x[\!] \tag{4.94}$$

Ce schéma est **implicite en temps** : il n'est pas possible de calculer explicitement $\boldsymbol{u}_i^n$ en fonction des $\boldsymbol{u}_i^{n-1}$ (au temps précédent). Toutefois on peut noter qu'au temps t^n, les $N_x - 1$ équations (4.94) sont linéaires en les $N_x + 1$ inconnues $(\boldsymbol{u}_i^n)_{i \in i \in [\![0, N_x]\!]}$. Ils nous manquent 2 équations (linéaires) pour obtenir un système linéaire de $N_x + 1$ équations à $N_x + 1$ inconnues. Celles-ci sont obtenues grâce aux deux contitions aux limites (4.63) et (4.64) en t^n. De (4.64) on obtient immediatement

$$\boldsymbol{u}_{N_x}^n = \beta(t^n).$$

En utilisant l'approximation (4.84) d'ordre 2, l'équation (4.63) s'écrit

$$-D\frac{-3u(t^n, x_0) + 4u(t^n, x_1) - u(t^n, x_2)}{2\Delta_x} + \mathcal{O}(\Delta_x^2) = \alpha(t^n).$$

Ce qui donne le schéma d'ordre 2 en espace

$$3\boldsymbol{u}_0^n - 4\boldsymbol{u}_1^n + \boldsymbol{u}_2^n = 2\frac{\Delta_x}{D}\alpha(t^n)$$

En résumé, soit $n \in]\!]0, N_t]\!]$, on vient de voir que si l'on connait $\boldsymbol{u}_i^{n-1}$, $\forall i \in [\![0, N_x]\!]$ alors le calcul des $\boldsymbol{u}_i^n$, $\forall i \in [\![0, N_x]\!]$, est possible en résolvant les $N_x + 1$ équations linéaires suivantes où l'on note $E = D\frac{\Delta_t}{\Delta_x^2}$ et $C = 1 + 2E$:

$$3\boldsymbol{u}_0^n - 4\boldsymbol{u}_1^n + \boldsymbol{u}_2^n = 2\frac{\Delta_x}{D}\alpha(t^n). \tag{4.95}$$

$$C\boldsymbol{u}_i^n - E(\boldsymbol{u}_{i+1}^n + \boldsymbol{u}_{i-1}^n) = \boldsymbol{u}_i^{n-1} + \Delta_t \boldsymbol{f}_i^n, \ \forall i \in]\!]0, N_x[\!] \tag{4.96}$$

$$\boldsymbol{u}_{N_x}^n = \beta(t^n), \tag{4.97}$$

On choisi l'écriture *naturelle* pour obtenir le système matricielle correspondant

$$\mathbb{A}\boldsymbol{U}^n = \boldsymbol{b}^n \tag{4.98}$$

avec $\mathbb{A} \in \mathcal{M}_{N_x+1}(\mathbb{R})$, $\boldsymbol{b}^n \in \mathbb{R}^{N_x+1}$ et

$$\boldsymbol{U}^n = \begin{pmatrix} \boldsymbol{u}_0^n \\ \boldsymbol{u}_1^n \\ \vdots \\ \boldsymbol{u}_{N_x-1}^n \\ \boldsymbol{u}_{N_x}^n \end{pmatrix} \in \mathbb{R}^{N_x+1}, \quad \text{i.e. } \boldsymbol{U}^n(i) = \boldsymbol{u}_{i-1}^n, \ \forall i \in [\![1, N_x + 1]\!]$$

Plus précisement, le système linéaire est obtenu de la manière suivante :

- la 1re ligne du système correspondra à l'équation écrite au 1er point de discrétisation x_0 (i.e. (4.95)),
- les lignes 2 à N_x du système correspondront aux équations écrites respectivement aux points intérieurs x_1 à x_{N_x-1} (i.e. (4.96)),
- la ligne N_x+1 (dernière ligne) du système correspondra à l'équation écrite au dernier point de discrétisation x_{Nx} (i.e. (4.97)),

Pour faciliter l'écriture du système linéaire, on peut se focaliser pour chaque ligne sur le terme diagonale. Par exemple, pour la ligne j du système, le coefficient diagonale correspond au coefficient de l'inconnue $\boldsymbol{U}^n(j) = \boldsymbol{u}^n_{j-1}$ dans l'équation (4.96) avec $i = j-1$: c'est à dire C. Le système matricielle (4.98) est donc

$$\left(\begin{array}{c|c|ccccc|c} 3 & -4 & 1 & 0 & \cdots & 0 & 0 \\ \hline -E & C & -E & 0 & \cdots & 0 & 0 \\ 0 & -E & C & -E & \ddots & \vdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & -E & C & -E & 0 \\ 0 & 0 & \dots & 0 & -E & C & -E \\ \hline 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{u}^n_0 \\ \hline \boldsymbol{u}^n_1 \\ \boldsymbol{u}^n_2 \\ \vdots \\ \boldsymbol{u}^n_{N_x-2} \\ \boldsymbol{u}^n_{N_x-1} \\ \hline \boldsymbol{u}^n_{N_x} \end{array}\right) = \left(\begin{array}{c} 2\frac{\Delta_x}{D}\alpha(t^n) \\ \hline \boldsymbol{u}^{n-1}_1 + \Delta_t \boldsymbol{f}^n_1 \\ \boldsymbol{u}^{n-1}_2 + \Delta_t \boldsymbol{f}^n_2 \\ \vdots \\ \boldsymbol{u}^{n-1}_{N_x-2} + \Delta_t \boldsymbol{f}^n_{N_x-2} \\ \boldsymbol{u}^{n-1}_{N_x-1} + \Delta_t \boldsymbol{f}^n_{N_x-1} \\ \hline \beta(t^n) \end{array}\right) \tag{4.99}$$

Il faut noter que dans cet exemple la matrice du système ne dépend pas du temps (i.e. de n) : elle est donc invariante au cours du temps.

Algorithmique

Comme pour le schéma explicité, l'initialisation de ce schéma itératif est obtenu par la condition initiale (4.62). On a alors

$$\boldsymbol{U}^0_i = u_0(x_i),\ \forall i \in [\![0, N_x]\!].$$

L'algorithme formel est donc le suivant :

1: $\boldsymbol{U}^0_i \leftarrow u_0(x_i),\ \forall i \in [\![0, N_x]\!]$
2: **Pour** $n \leftarrow 1$ **à** N_t **faire**
3: Calcul de $\boldsymbol{U}^n$ en résolvant le système linéaire (4.95),(4.96),(4.97)
4: **Fin**

Pour résoudre le système linéaire à l'itération n il faut avant tout être capable de le construire! Pour celà on propose dans un premier temps de construire la matrice du système puis le second membre. C'est l'objet de l'exercice 4.6.1 où le choix a été fait d'écrire ces deux opérations sous la forme de deux fonctions distinctes car la matrice est indépendante du temps et peut donc être calculée en dehors de la boucle principale contrairement au second membre.

Exercice 4.6.1

Q. 1 *Ecrire la fonction* AssembleMatGen1D *retournant la matrice* $\mathbb{M} \in \mathcal{M}_d(\mathbb{R})$ *définie par*

$$\mathbb{M} = \begin{pmatrix} a_1 & a_2 & a_3 & 0 & \dots & \dots & 0 \\ \beta & \alpha & \beta & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \beta & \alpha & \beta \\ 0 & \dots & \dots & 0 & b_3 & b_2 & b_1 \end{pmatrix} \tag{4.1}$$

où α, β, a_1, a_2, a_3, b_1, b_2 *et* b_3 *sont des réels donnés.*

Q. 2 *Ecrire la fonction* SndMbrGen1d *retournant le vecteur* $\boldsymbol{B} \in \mathbb{R}^d$ *défini par*

$$\boldsymbol{B} = \begin{pmatrix} \alpha \\ c_1 \\ \vdots \\ c_{d-2} \\ \beta \end{pmatrix} \tag{4.2}$$

où α, β, $c_1, \ldots, c_{d-2}$ *sont des réels donnés.*

Correction 4.6.1

R. 1 On a par exemple

Algorithme 4.7 Fonction AssembleMatGen1D

Données : d : dimension de la matrice,
α, β : deux réels,
$\boldsymbol{a}, \boldsymbol{b}$: deux vecteurs de $\mathbb{R}^3$ tels que $\boldsymbol{a}(i) = a_i$ et $\boldsymbol{b}(i) = b_i$.
Résultat : $\mathbb{M}$: matrice $d \times d$

Fonction $x \leftarrow$ AssembleMatGen1D($d, \boldsymbol{a}, \boldsymbol{b}, \alpha, \beta$)
  $\mathbb{M} \leftarrow \mathbb{0}_{d,d}$
  $\mathbb{M}(1,1) \leftarrow \boldsymbol{a}(1)$, $\mathbb{M}(1,2) \leftarrow \boldsymbol{a}(2)$, $\mathbb{M}(1,3) \leftarrow \boldsymbol{a}(3)$
  $\mathbb{M}(d,d) \leftarrow \boldsymbol{b}(1)$, $\mathbb{M}(d,d-1) \leftarrow \boldsymbol{b}(2)$, $\mathbb{M}(d,d-2) \leftarrow \boldsymbol{b}(3)$
  Pour $i = 2$ **à** $d-1$ **faire** ▷ Initialisation de la ligne i
    $\mathbb{M}(i,i) \leftarrow \alpha$, $\mathbb{M}(i,i-1) \leftarrow \beta$, $\mathbb{M}(i,i+1) \leftarrow \beta$
  Fin
Fin

R. 2 On a par exemple

Algorithme 4.8 Fonction SndMbrGen1d

Données : d : dimension de la matrice,
α, β : trois réels,
$\boldsymbol{c}$: un vecteur de $\mathbb{R}^{d-2}$ tel que $\boldsymbol{c}(i) = c_i$.
Résultat : $\boldsymbol{B}$: vecteur de $\mathbb{R}^d$.

Fonction $\boldsymbol{B} \leftarrow$ SndMbrGen1d($d, \boldsymbol{c}, \alpha, \beta$)
  $\boldsymbol{B}(1) \leftarrow \alpha$, $\boldsymbol{B}(2) \leftarrow \beta$
  Pour $i = 2$ **à** $d-1$ **faire** ▷ Initialisation de la ligne i
    $\boldsymbol{B}(i) \leftarrow \boldsymbol{c}(i-1)$
  Fin
Fin

Fin correction

Les données et résultats du schéma implicite sont identiques à ceux du schéma explicites. On les rappelle ici

Résultat : $\mathbb{U}$: tableau 2D/matrice $(N_x+1)\times(N_t+1)$ de réels tel que $\mathbb{U}(i+1,n+1)=\boldsymbol{u}_i^n$, $\forall i\in[\![0,N_x]\!]$, $\forall n\in[\![0,N_t]\!]$ ou encore $\mathbb{U}(:,n+1)=\boldsymbol{U}^n$, $\forall n\in[\![0,N_t]\!]$.

Données :
- a,b : deux réels $a<b$,
- T : $T>0$,
- D : D réel strictement positif, (coefficient de diffusivité)
- f : $f:[0,T]\times[a,b]\longrightarrow\mathbb{R}$,
- u_0 : $u_0:[a,b]\longrightarrow\mathbb{R}$,
- α : $\alpha:[0,T]\longrightarrow\mathbb{R}$,
- β : $\beta:[0,T]\longrightarrow\mathbb{R}$, tel que $\beta(0)=u_0(b)$,
- N_x : $N_x\in\mathbb{N}^*$, nombre de discrétisation en espace,
- N_t : $N_t\in\mathbb{N}^*$, nombre de discrétisation en temps.

L'algorithme formel un peu plus détaillé est donc le suivant :

$\mathbb{U}(i+1,1)\leftarrow u_0(x_i),\ \forall i\in[\![0,N_x]\!]$
$E\leftarrow D\frac{\Delta_t}{\Delta_x^2}$, $C\leftarrow 1+2E$
$\boldsymbol{x}\leftarrow$ DisReg(a,b,N_x)
$\boldsymbol{t}\leftarrow$ DisReg$(0,T,N_t)$
Pour $n\leftarrow 1$ **à** N_t **faire**
  $\mathbb{A}\leftarrow$ AssembleMatGen1d$(d,(3,-4,1),(1,0,0),C,-E)$
  Calcul de $\boldsymbol{v}\in\mathbb{R}^{N_x-1}$ avec $\boldsymbol{v}(i)=\mathbb{U}(i+1,n)+\Delta_t f(\boldsymbol{t}(n+1),x_i)$, $\forall i\in[\![1,N_x-1]\!]$
  $L\leftarrow 2(\Delta_x/D)\alpha(\boldsymbol{t}(n+1))$
  $\boldsymbol{B}\leftarrow$ SndMbrGen1d$(d,\boldsymbol{v},L,\beta(\boldsymbol{t}(n+1)))$
  $\mathbb{U}(:,n+1)\leftarrow$ Solve$(\mathbb{A},\boldsymbol{B})$
Fin

L'algorithme complet peut donc être écrit sous la forme d'une fonction et il est donné par l'Algorithme 4.9.

Algorithm 4.9 Fonction Heat1Dim (version non vectorisée)

Fonction $[\boldsymbol{t},\boldsymbol{x},\mathbb{U}]\leftarrow$ Heat1Dim$(\ a,b,T,D,f,u_0,\alpha,\beta,N_x,N_t\ )$
  $\boldsymbol{t}\leftarrow$ DisReg$(0,T,N_t)$
  $\Delta_t\leftarrow T/N_t$
  $\boldsymbol{x}\leftarrow$ DisReg(a,b,N_x)
  $\Delta_x\leftarrow(b-a)/N_x$
  $E\leftarrow D\frac{\Delta_t}{\Delta_x^2}$, $C\leftarrow 1+2E$
  Pour $i\leftarrow 1$ **à** N_x+1 **faire** ▷ Condition initiale
    $\mathbb{U}(i,1)\leftarrow u_0(\boldsymbol{x}(i))$
  Fin
  $\mathbb{A}\leftarrow$ AssembleMatGen1d$(d,(3,-4,1),(1,0,0),C,-E)$
  Pour $n\leftarrow 1$ **à** N_t **faire** ▷ Boucle en temps
    Pour $i\leftarrow 1$ **à** N_x-2 **faire** ▷ Schéma
      $\boldsymbol{v}(i)\leftarrow\mathbb{U}(i+1,n)+\Delta_t f(\boldsymbol{t}(n+1),\boldsymbol{x}i)$
    Fin
    $L\leftarrow 2(\Delta_x/D)\alpha(\boldsymbol{t}(n+1))$
    $\boldsymbol{B}\leftarrow$ SndMbrGen1d$(d,\boldsymbol{v},L,\beta(\boldsymbol{t}(n+1)))$
    $\mathbb{U}(:,n+1)\leftarrow$ Solve$(\mathbb{A},\boldsymbol{B})$
  Fin
Fin

Une version vectorisée est donnée en Algorithme 4.10.

Algorithm 4.10 Fonction Heat1Dim (version vectorisée)

Fonction $[\boldsymbol{t}, \boldsymbol{x}, \mathbb{U}] \leftarrow$ Heat1Dim($a, b, T, D, f, u_0, \alpha, \beta, N_x, N_t$)
  $\boldsymbol{t} \leftarrow$ DisReg$(0, T, N_t)$
  $\Delta_t \leftarrow T/N_t$
  $\boldsymbol{x} \leftarrow$ DisReg(a, b, N_x)
  $\Delta_x \leftarrow (b-a)/N_x$
  $E \leftarrow D\frac{\Delta_t}{\Delta_x^2}$, $C \leftarrow 1 + 2E$
  $\mathbb{U}(:, 1) \leftarrow u_0(\boldsymbol{x})$ ▷ Condition initiale
  $\mathbb{A} \leftarrow$ AssembleMatGen1d$(d, (3, -4, 1), (1, 0, 0), C, E)$
  Pour $n \leftarrow 1$ **à** N_t **faire** ▷ Boucle en temps
    $\boldsymbol{v} \leftarrow \mathbb{U}([2 : N_x], n) + \Delta_t f(\boldsymbol{t}(n+1), \boldsymbol{x}([2 : N_x]))$
    $L \leftarrow 2(\Delta_x/D)\alpha(\boldsymbol{t}(n+1))$
    $\boldsymbol{B} \leftarrow$ SndMbrGen1d$(d, \boldsymbol{v}, L, \beta(\boldsymbol{t}(n+1)))$
    $\mathbb{U}(:, n+1) \leftarrow$ Solve$(\mathbb{A}, \boldsymbol{B})$
  Fin
Fin

En Figure 4.17, on représente les ordres du schéma : ordre 1 en temps (à gauche) et ordre 2 en espace (à droite).

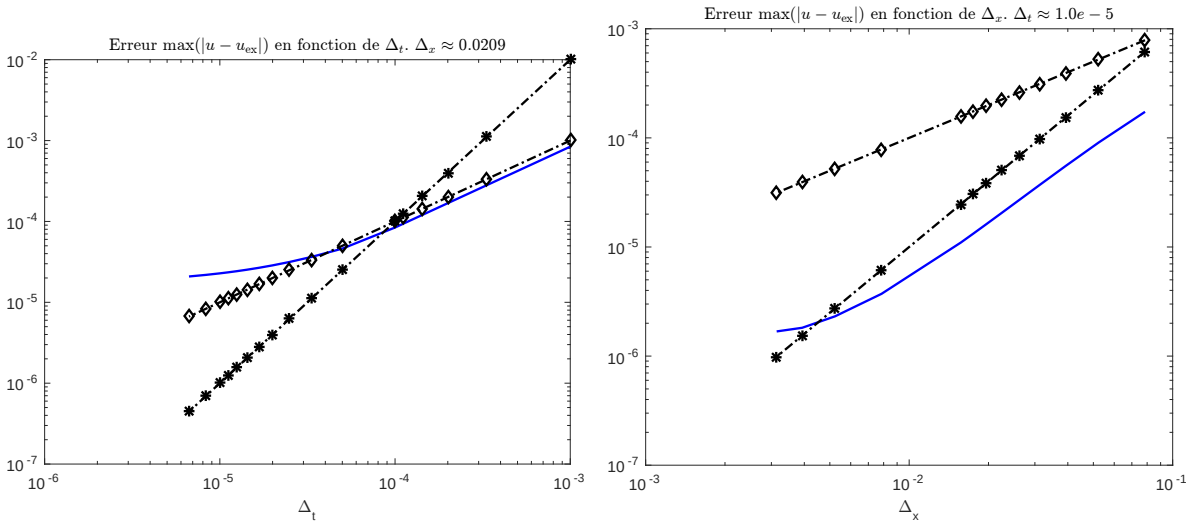


Figure 4.17: Equation de la chaleur 1D avec solution exacte. Ordres pour le schéma implicite.

On peut noter qu'aucun phénomène d'instabilité ne vient perturber les résultats. En effet, Une étude de stabilité au sens de Von Neumann du schéma implicite pour l'équation de la chaleur (4.56) permet de montrer le caractère **inconditionnellement stable** du schéma. Pour plus de détails se référer à [**?**] (section 2.2.3, page 37-42).

Liste des algorithmes