

Méthodes numériques II (cours du 5 janv. 2016)

François Cuvelier

Laboratoire d'Analyse Géométrie et Applications
Institut Galilée
Université Paris XIII.

2018/01/08

- Volume horaire : $8 \times 3h$ cours/TD, $2 \times 8h$ TP.
- Prérequis des cours :
 - ▶ *Cours commun Mathématiques pour l'ingénieur* (L. El Alaoui), ...
 - ▶ *Algorithmique*
 - ▶ Langage de programmation : *Matlab/Octave*
- Note finale : $(P_1 + P_2 + TP)/3$ si possible.
 - ▶ P_1 : partiel 1 du 13 février 2018 (1h30/2h00),
 - ▶ P_2 : partiel 2 du 27 mars 2018 (1h30/2h00),
 - ▶ TP : travaux pratiques en mai et juin.

- Algorithmique numérique.
- Poursuite de l'apprentissage de Matlab/Octave.
- Résolution numériques d'équations différentielles ordinaires (E.D.O.[fr] ou O.D.E.[en]).
- Résolution numériques d'équations aux dérivées partielles (E.D.P.[fr] ou P.D.E.[en]) par des méthodes de différences finies.

- Chapitre I : Algorithmique numérique
- Chapitre II : Dérivation numérique
- Chapitre III : Résolution numérique des E.D.O.
- Chapitre IV : Résolution numérique des E.D.P.

Part I

Algorithmique numérique

Plan

- 1 Introduction
- 2 Pseudo-langage algorithmique
- 3 Méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)
- 5 Histoire de ponts

Definition 1.1 (Petit Robert 97)

Algorithmique : Enchaînement d'actions nécessaires à l'accomplissement d'une tâche.

Exemple 1 : permutation

Nous voulons permuter deux voitures sur un parking de trois places numérotées de 1 à 3 et ceci sans gêner la circulation.

La première voiture, une Saxo, est sur l'emplacement 2, la seconde, une Clio, est sur l'emplacement 3.

Donner un algorithme permettant de résoudre cette tâche.

Exemple 2 : équation du premier degré

Donner un algorithme permettant de résoudre

$$ax = b$$

Caractéristiques d'un *bon* algorithme

- Il ne souffre d'aucune ambiguïté $\Rightarrow$ très clair.
- Combinaison d'opérations (actions) élémentaires.
- Pour toutes les données d'entrée, l'algorithme doit fournir un résultat en un nombre fini d'opérations.

Etape 1 : Définir clairement le problème.

Première approche méthodologique

Etape 1 : Définir clairement le problème.

Etape 2 : Rechercher une méthode de résolution (formules, ...)

Etape 1 : Définir clairement le problème.

Etape 2 : Rechercher une méthode de résolution (formules, ...)

Etape 3 : Ecrire l'algorithme (par raffinement successif pour des algorithmes *compliqués*).

Plan

- 1 Introduction
- 2 Pseudo-langage algorithmique**
 - Les bases
 - Les instructions structurées
- 3 Méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)
- 5 Histoire de ponts

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

- Donnée $\Rightarrow$ introduite par l'utilisateur
- Constante $\Rightarrow$ symbole, identificateur non modifiable

Definition 2.1

Une variable est un objet dont la valeur est modifiable, qui possède un nom et un type (entier, caractère, réel, complexe, tableau, matrice, vecteur...).

Opérateurs arithmétiques

Nom	Symbole	Exemple
addition	+	$a + b$
soustraction	—	$a - b$
opposé	—	$-a$
produit	*	$a * b$
division	/	a/b

Opérateurs relationnels

Nom	Symbole	Exemple
identique	$==$	$a == b$
différent	$\sim =$	$a \sim = b$
inférieur	$<$	$a < b$
supérieur	$>$	$a > b$
inférieur ou égal	$<=$	$a <= b$
supérieur ou égal	$>=$	$a >= b$

Opérateurs logiques

Nom	Symbole	Exemple
négation	$\sim$	$\sim a$
ou	$ $	$a b$
et	$\&$	$a\&b$

Opérateur d'affectation

Nom	Symbole	Exemple
affectation	$\leftarrow$	$a \leftarrow b$

Definition 2.2

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)

Definition 2.2

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 d'expression numérique

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

Definition 2.2

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 d'expression numérique

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

Opérandes $\Rightarrow$ identifiants a, b, c ,
constantes 4 et 2.

Definition 2.2

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 d'expression numérique

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

Opérandes $\Rightarrow$ identifiants a, b, c ,
constantes 4 et 2.

Opérateurs $\Rightarrow$ symboles $*, -, \text{ et } /$

Definition 2.3

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 d'expression booléenne

$$(x < 3.14)$$

Definition 2.3

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 d'expression booléenne

$$(x < 3.14)$$

Opérandes $\Rightarrow$ identifiants x et constantes 3.14

Definition 2.3

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 d'expression booléenne

$$(x < 3.14)$$

Opérandes $\Rightarrow$ identifiants x et contantes 3.14

Opérateurs $\Rightarrow$ symboles $<$

Definition 2.4

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.

Instructions simples

- affectation d'une valeur a une variable.
- appel d'une fonction (procedure, subroutine, ... suivant les langages).

- ① les instructions composées, groupe de plusieurs instructions simples,
- ② les instructions répétitives, permettant l'exécution répétée d'instructions simples, (i.e. boucles «pour», «tant que»)
- ③ les instructions conditionnelles, lesquels ne sont exécutées que si une certaine condition est respectée (i.e. «si»)

Exemple : boucle «pour»

Algorithme 1 Exemple boucle «pour»

Données : n : un entier positif

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

Mais que fait-il?

Exemple : boucle «pour»

Algorithme 2 Exemple boucle «pour»

Données : n : un entier positif

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

Mais que fait-il?

Calcul de $S = \sum_{i=1}^n \cos(i^2)$

Exemple : boucle «tant que»

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

Mais que fait-il?

Exemple : boucle «tant que»

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

Mais que fait-il?

Calcul de $x = 1 + \sum_{i=0}^{999} i^2$

Exemple : instructions conditionnelles «si»

Données :

age : un réel.

- 1: **Si** *age* $\geq$ 18 **alors**
- 2: affiche('majeur')
- 3: **Sinon Si** *age* $\geq$ 0 **alors**
- 4: affiche('mineur')
- 5: **Sinon**
- 6: affiche('en devenir')
- 7: **Fin Si**

Plan

- 1 Introduction
- 2 Pseudo-langage algorithmique
- 3 Méthodologie de construction**
 - Principe
 - Exercices
- 4 Pseudo-langage algorithmique (suite)
- 5 Histoire de ponts

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

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.

- Le type des données et des résultats doivent être précisés.
- L'algorithme doit fournir au moins un résultat.
- L'algorithme doit être exécuté en un nombre fini d'opérations.
- L'algorithme doit être spécifié clairement, sans la moindre ambiguïté.



Exercise 3.1

On cherche les solutions réelles de l'équation

$$ax^2 + bx + c = 0, \quad (1)$$

en supposant que $a \in \mathbb{R}^*$, $b \in \mathbb{R}$ et $c \in \mathbb{R}$ sont donnés.

Ecrire un algorithme permettant de résoudre cette équation.



Exercise 3.2

Ecrire un algorithme permettant de calculer

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





Exercise 3.3

Ecrire un algorithme permettant de calculer

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





Exercise 3.4

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

Plan

- 1 Introduction
- 2 Pseudo-langage algorithmique
- 3 Méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)**
 - Les fonctions
 - Exemple : résolution d'une équation
 - Exemple : résolution d'une équation du second degré
- 5 Histoire de ponts

Les fonctions permettent

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

- les fonctions d'affichage et de lecture : **Affiche**, **Lit**

Les fonctions prédéfinies

- les fonctions d'affichage et de lecture : **Affiche**, **Lit**
- les fonctions mathématiques :

$\sin$, $\cos$, $\exp$, ...

Les fonctions prédéfinies

- les fonctions d'affichage et de lecture : **Affiche**, **Lit**
- les fonctions mathématiques :

$\sin$, $\cos$, $\exp$, ...

- les fonctions de gestion de fichiers
- ...

Ecrire ses propres fonctions

- 1 Que doit-on calculer/réaliser précisément (but)?
- 2 Quelles sont les données (avec leurs limitations)?

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

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

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
- Données :
- Résultats :

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
trouver $x \in \mathbb{R}$ solution de $ax + b = 0$.
- Données :
- Résultats :

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
trouver $x \in \mathbb{R}$ solution de $ax + b = 0$.
- Données :
 $a \in \mathbb{R}^*$ et $b \in \mathbb{R}$.
- Résultats :

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
trouver $x \in \mathbb{R}$ solution de $ax + b = 0$.
- Données :
 $a \in \mathbb{R}^*$ et $b \in \mathbb{R}$.
- Résultats :
 $x \in \mathbb{R}$.

Algorithme 3 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** $x \leftarrow \text{REPD}(a, b)$
 - 2: $x \leftarrow -b/a$
 - 3: **Fin Fonction**
-

équation du second degré

On cherche les solutions réelles de l'équation

$$ax^2 + bx + c = 0, \quad (2)$$

Pour celà, on pose $\Delta = b^2 - 4ac$

- 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 $x_1 = \frac{-b - \sqrt{\Delta}}{2a}$ et $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$.

Exercice

- 1 Ecrire la fonction discriminant permettant de calculer le discriminant de l'équation (2).
- 2 Ecrire la fonction RESD permettant de résoudre l'équation (2) en utilisant la fonction discriminant.
- 3 Ecrire un programme permettant de valider ces deux fonctions.

Plan

- 1 Introduction
- 2 Pseudo-langage algorithmique
- 3 Méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)
- 5 Histoire de ponts**

Mais avant de poursuivre ...



(a) Pont de la Basse-Chaine, Angers
(1850)



(b) Takoma Narrows Bridge,
Washington (1940)



(c) Millenium Bridge, *London* (2000)

Figure : Une histoire de ponts

Part II

Dérivation numérique

♥ Definition 5.1

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

On note $t^n = a + nh$, $n \in \llbracket 0, N \rrbracket$, une **discrétisation régulière de $[a, b]$** et $(Dy)_n$ une approximation de $y'(t^n)$. On appelle

- **différence finie progressive** l'approximation

$$(Dy)_n^P = \frac{y(t^{n+1}) - y(t^n)}{h}, \quad \forall n \in \llbracket 0, N - 1 \rrbracket \quad (3)$$

- **différence finie rétrograde** l'approximation

$$(Dy)_n^R = \frac{y(t^n) - y(t^{n-1})}{h}, \quad \forall n \in \llbracket 1, N \rrbracket \quad (4)$$

- **différence finie centrée** l'approximation

$$(Dy)_n^C = \frac{y(t^{n+1}) - y(t^{n-1}))}{2h}, \quad \forall n \in \llbracket 1, N - 1 \rrbracket \quad (5)$$



Proposition 5.2: Développement de Taylor

Soit f une application $f : [a, b] \longrightarrow \mathbb{R}$. Si $f \in \mathcal{C}^{r+1}([a, b])$ alors

- $\forall (x, y) \in [a, b]^2$ il existe un $\xi \in]x, y[$ tel que

$$f(x) = f(y) + \sum_{k=1}^r \frac{f^{(k)}(y)}{k!} (x - y)^k + \frac{f^{(r+1)}(\xi)}{(r+1)!} (x - y)^{r+1} \quad (6)$$

- $\forall x \in [a, b]$, $\forall h \in \mathbb{R}^*$ vérifiant $x + h \in [a, b]$, il existe $\xi \in]\min(x, x + h), \max(x, x + h)[$ tel quel

$$f(x + h) = f(x) + \sum_{k=1}^r \frac{f^{(k)}(x)}{k!} h^k + \frac{f^{(r+1)}(\xi)}{(r+1)!} h^{r+1} \quad (7)$$



Exercise 5.1

Q.1 Soit $y \in \mathcal{C}^2([a, b])$.

- ➊ Montrer qu'il existe $\eta_P^n \in]t^n, t^{n+1}[$ et $\eta_R^n \in]t^{n-1}, t^n[$ tels que

$$(Dy)_n^P = y^{(1)}(t^n) + \frac{h}{2} y^{(2)}(\eta_P^n)$$

et

$$(Dy)_n^R = y^{(1)}(t^n) - \frac{h}{2} y^{(2)}(\eta_R^n)$$

- ➋ En déduire que

$$|y^{(1)}(t^n) - (Dy)_n^P| \leq C_1 h, \text{ avec } C_1 = \frac{1}{2} \max_{t \in [t^n, t^{n+1}]} |y^{(2)}(t)|$$

et

$$|y'(t^n) - (Dy)_n^R| \leq C_2 h, \text{ avec } C_2 = \frac{1}{2} \max_{t \in [t^{n-1}, t^n]} |y^{(2)}(t)|$$

Q.2 Soit $y \in \mathcal{C}^3([a, b])$.

- ➊ Montrer qu'il existe $\eta_1^n \in]t^n, t^{n+1}[$ et $\eta_2^n \in]t^{n-1}, t^n[$ tels que

$$(Dy)_n^C = y^{(1)}(t^n) - \frac{h^2}{12} (y^{(3)}(\eta_1^n) + y^{(3)}(\eta_2^n))$$

- ➋ En déduire que

$$|y^{(1)}(t^n) - (Dy)_n^C| \leq E h^2, \text{ avec } E = \frac{1}{6} \max_{t \in [t^{n-1}, t^{n+1}]} |y^{(3)}(t)|$$

♥ Definition 5.3

Soit g une fonction. 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

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

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

♥ Definition 5.4

La différence $|y'(t^n) - (Dy)_n|$ est appelée **erreur de troncature au point t^n** . On dira que $|y'(t^n) - (Dy)_n|$ est d'ordre $p > 0$ si il existe une constante $C > 0$ telle que

$$|y'(t^n) - (Dy)_n| \leq Ch^p \iff |y'(t^n) - (Dy)_n| = \mathcal{O}(h^p).$$

On a démontré le lemme suivant

Lemma 5.5

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

$$\left| y'(t^n) - \frac{y(t^{n+1}) - y(t^n)}{h} \right| = \mathcal{O}(h), \quad \forall n \in \llbracket 0, N-1 \rrbracket, \quad (8)$$

$$\left| y'(t^n) - \frac{y(t^n) - y(t^{n-1})}{h} \right| = \mathcal{O}(h), \quad \forall n \in \llbracket 1, N \rrbracket. \quad (9)$$

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

$$\left| y'(t^n) - \frac{y(t^{n+1}) - y(t^{n-1}))}{2h} \right| = \mathcal{O}(h^2), \quad \forall n \in \llbracket 1, N-1 \rrbracket. \quad (10)$$



Exercise 5.2

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

Q.1

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

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

- 2 Ecrire une fonction algorithmique permettant, à partir du vecteur $\mathbf{F}$ et de la discrétisation régulière, de calculer le vecteur $\mathbf{V}$ précédent.

Q.2

- 1 Connaissant uniquement le vecteur $\mathbf{F}$, déterminer un vecteur $\mathbf{W} \in \mathbb{R}^{N+1}$ vérifiant

$$\mathbf{W}_n = f'(t^n) + \mathcal{O}(h^2), \quad \forall n \in \llbracket 0, N \rrbracket$$

- 2 Ecrire une fonction algorithmique permettant, à partir du vecteur $\mathbf{F}$ et de la discrétisation régulière, de calculer le vecteur $\mathbf{W}$ précédent.

Application

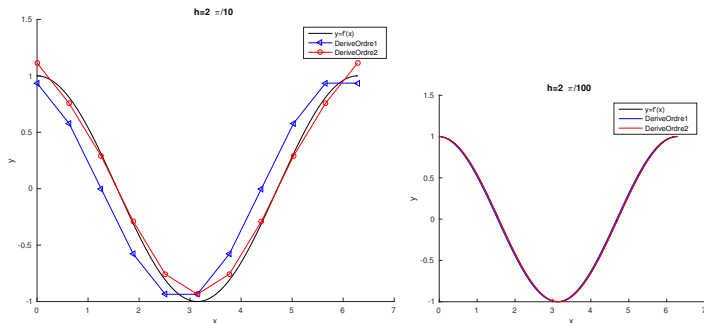


Figure : Représentation des dérivées numériques avec $f(x) := \sin(x)$, $a = 0$, $b = 2\pi$. À gauche $h = \frac{2\pi}{10}$, à droite $h = \frac{2\pi}{100}$.

Application

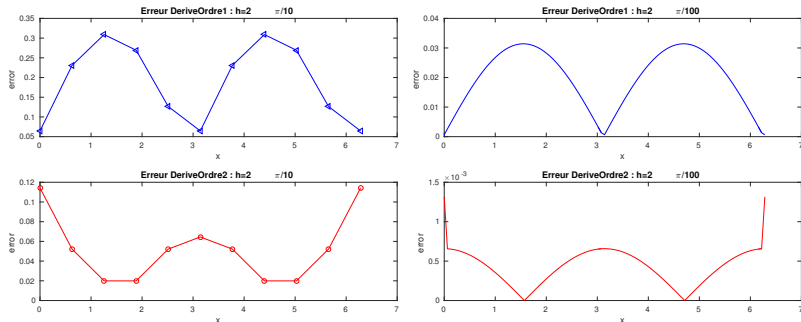


Figure : Erreur des dérivées numériques numériques avec $f(x) := \sin(x)$, $a = 0$, $b = 2\pi$. A gauche $h = \frac{2\pi}{10}$, à droite $h = \frac{2\pi}{100}$.

- Dérivée ordre 1 :

h a été divisé par 10 $\implies$ l'erreur, $\mathcal{O}(h)$, divisée par 10.

- Dérivée ordre 2 :

h a été divisé par 10 $\implies$ l'erreur, $\mathcal{O}(h^2)$, divisée par 100.

Application

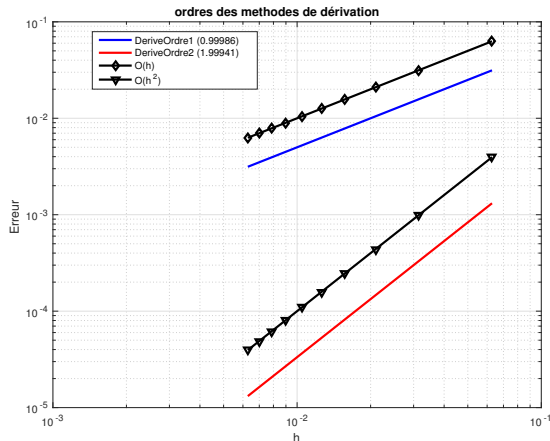


Figure : Dérivation numérique : mise en évidence de l'ordre des méthodes

Figure en échelle logarithmique : intérêt de *monter* en ordre.