Sorbonne Université **Master 1 SPE M3E**

Méthodes mathématiques
et modélisation de l'environnement (M3E)

Résolution numérique des équations différentielles ordinaires (EDO)

2020–2021

`Jacques.Lefrere@upmc.fr`

Table des matières

1 Dérivation

1.1 Rappels : définition et propriétés

Définition

$$f'(t) = \frac{\mathrm{d}f}{\mathrm{d}t}(t) = \lim_{h\to 0} \frac{f(t+h)-f(t)}{h} \tag{1}$$

Quelques dérivées usuelles (a = constante)

$f(t)$	t^a	$\ln\lvert t+a\rvert$	$\exp(at)$	$\sin(at)$	$\cos(at)$
$f'(t)$	at^{a-1}	$1/(t+a)$	$a\exp(at)$	$a\cos(at)$	$-a\sin(at)$

Linéarité : $(f+g)' = f' + g' \qquad (af(t))' = af'(t)$

Dérivation et composition de fonctions

$$h(t) = f[g(t)] \implies h'(t) = g'(t) \times f'[g(t)]$$

1.2 Dérivation numérique : erreurs de troncature et d'arrondi

Objectif : estimer numériquement **la dérivée première** $f'(t)$ d'une fonction f en t à partir des échantillons de la fonction f aux instants $t+ih$, où h est le pas d'échantillonnage de f.

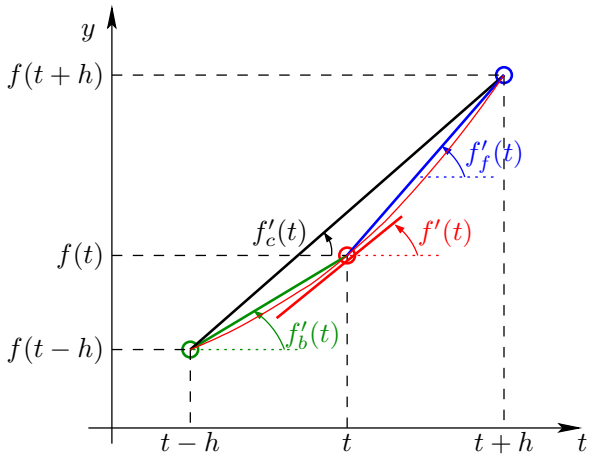
Plusieurs approximations de $f'(t)$ ou schémas aux différences finies envisageables. Les plus simples sont les schémas à deux termes :

$$f'_b(t) = \frac{f(t)-f(t-h)}{h} \quad \text{schéma arrière} \quad \textit{backward}$$

$$f'_f(t) = \frac{f(t+h)-f(t)}{h} \quad \text{schéma avant} \quad \textit{forward}$$

$$f'_c(t) = \frac{f(t+h)-f(t-h)}{2h} \quad \text{schéma centré} \quad \textit{centered}$$

Préférer le schéma aux différences finies **centré**



Schémas de dérivation aux différences finies à 2 termes

f'_f avant

f'_b arrière

f'_c centré

1.2.1 Erreur d'estimation associée

L'erreur d'estimation, $f'_c(t) - f'(t)$ pour le schéma centré, comporte deux contributions qui (au pire) s'ajoutent en valeur absolue :

— l'**erreur systématique de troncature** déterministe de valeur absolue e_t liée au **nombre fini de termes** dans l'estimateur (2 termes dans ce schéma).
Dérivation théorique $\Longleftrightarrow$ multiplication par $\boldsymbol{i\omega}$ dans l'espace de Fourier
Dérivation numérique centrée à 2 termes $\Longrightarrow \times \boldsymbol{i \sin(\omega h)/h}$
qui n'est proche de $\boldsymbol{i\omega}$ que pour $h \to 0$.
e_t croît quand h croît

— l'**erreur aléatoire d'arrondi** de valeur absolue e_a liée à la précision de la **représentation approximative des flottants** en machine et essentiellement due au calcul de la différence faible de deux termes proches.
e_a croît quand h décroît

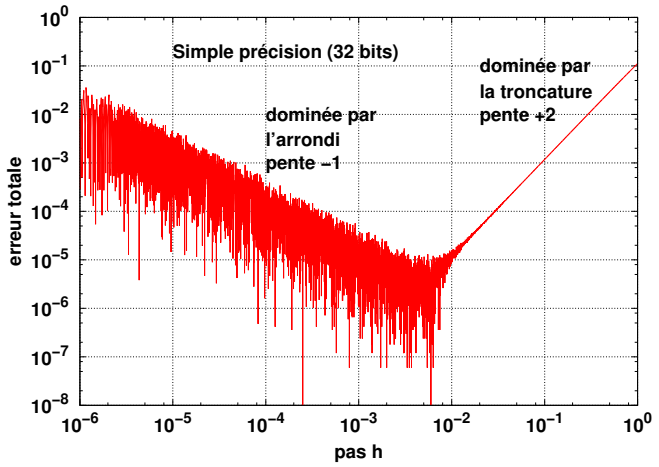


FIGURE 1: **Erreur totale** e de l'estimateur centré à 2 termes de la dérivée première de la fonction sinus en $t = \pi/4$ en fonction du pas h **en échelle log-log** .

1.2.2 Estimation de l'erreur de troncature

Développement en série de Taylor avec reste de $f(t \pm h)$ au deuxième ordre autour de t :

$$f(t+h) = f(t) + h\frac{\mathrm{d}f}{\mathrm{d}t}(t) + \frac{h^2}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}t^2}(t) + \frac{h^3}{6}\frac{\mathrm{d}^3 f}{\mathrm{d}t^3}(t+\theta h)$$

$$f(t-h) = f(t) - h\frac{\mathrm{d}f}{\mathrm{d}t}(t) + \frac{h^2}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}t^2}(t) - \frac{h^3}{6}\frac{\mathrm{d}^3 f}{\mathrm{d}t^3}(t-\theta' h)$$

où θ et θ' sont dans l'intervalle $[0,\ 1]$.

Estimateur centré à 2 termes de la dérivée :

$$\frac{f(t+h) - f(t-h)}{2h} = f'(t) + \frac{h^2}{12}\left[f'''(t+\theta h) + f'''(t-\theta' h)\right]$$

Erreur absolue de troncature liée à la dérivée troisième :

$$e_t \approx \frac{\boldsymbol{h^2}}{6}|f'''(t)|$$

1.2.3 Rappel : domaine et précision des réels en virgule flottante

Représentation approchée des réels en **virgule flottante** pour concilier grande dynamique et précision relative presque constante.
Par exemple en base 10, avec 4 chiffres après la virgule, comparer les représentations approchées (par troncature) en virgule fixe et flottante :

nombre exact	virgule fixe	virgule flottante
	par troncature	
0.0000123456789	⚠ .0000	0.1234×10^{-4}
0.000123456789	.0001	0.1234×10^{-3}
0.00123456789	.0012	0.1234×10^{-2}
0.0123456789	.0123	0.1234×10^{-1}
0.123456789	.1234	0.1234×10^{0}
1.23456789	1.2345	0.1234×10^{1}
12.3456789	12.3456	0.1234×10^{2}

Virgule flottante
en base 10

$$0.\underbrace{1234}_{\text{mantisse}} \times 10^{\overbrace{-1}^{\text{exposant}}}$$

En binaire, nombre de bits réparti entre **mantisse** (partie fractionnaire) et **exposant**

— m bits de mantisse ⇒ **précision** limitée

— q bits de l'exposant ⇒ **domaine** fini

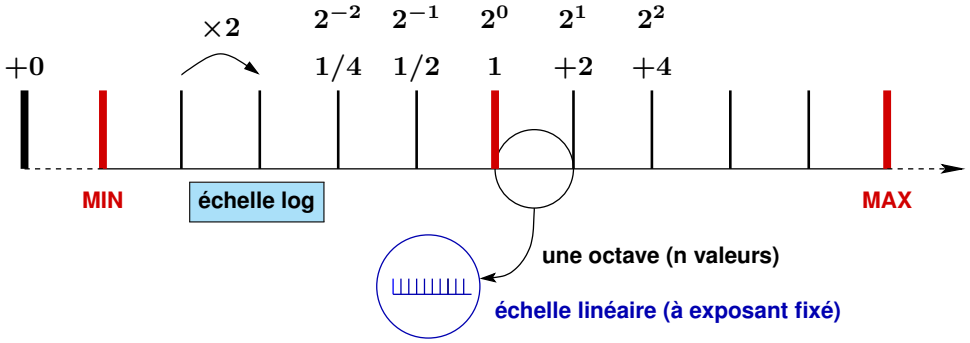
Ajouter 1 bit de signe ⇒ nombre de bits $= m + q + 1$

— À exposant (puissance de 2) fixé : progression arithmétique dans chaque octave
⇒ 2^m valeurs par octave
ε = la plus petite valeur telle que $1 + \varepsilon > 1$ donc $1 + \varepsilon =$ successeur de 1
ε est le pas des flottants dans l'octave $[1, 2[\Rightarrow \varepsilon = 1/2^m$
Précision relative $\leqslant \varepsilon = 1/2^m$
Flottants sur 64 bits : $m =$ **52 bits de mantisse** $\Rightarrow \varepsilon =$ $2^{-52} \approx 10^{-15}/4$

— Octaves en progression géométrique de raison 2
q bits d'exposant ⇒ $2^q - 2$ octaves (+ codes non numériques)
Flottants sur 64 bits : $q =$ **11 bits d'exposant** donc 2046 octaves
Domaine : MAX $\approx 2^{1024} \approx 1{,}8 \times 10^{308}$ et MIN $\approx 2^{-1022} \approx 2{,}2 \times 10^{-308}$

Domaine des flottants fixé par le nombre de bits q de l'exposant

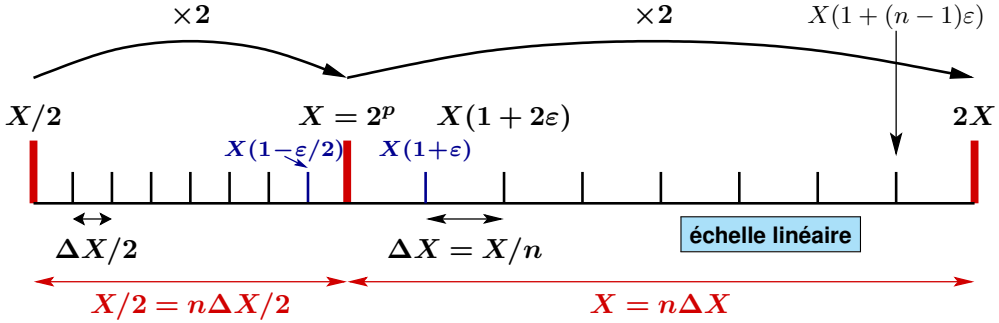
En virgule flottante, les octaves sont en **progression géométrique de raison 2** :
Nombre d'octaves $\approx 2^q$, réparties presque symétriquement autour de 1.



Domaine des flottants positifs normalisés (**échelle log**)

Précision des flottants fixée par le nombre de bits m de la mantisse

Dans chaque octave $[2^p, 2^{p+1}[= [X, 2X[$, l'exposant est constant
$\Rightarrow n = 2^m$ flottants en **progression arithmétique** de pas $\Delta X = X/n = \varepsilon X$



Exemple représenté ici : deux octaves de flottants positifs avec mantisse sur m=3 bits
$n = 2^m = 2^3 = 8$ intervalles et aussi 8 valeurs par octave

1.2.4 Caractéristiques numériques des flottants sur 32 et 64 bits

Interrogation via **`numpy.finfo(type).attribut`**

Le type par défaut des réels est **`float`**, ou encore **`numpy.float64`**.

voir norme IEEE 754-2008		`type`	
attribut	signification	**`numpy.float32`**	**`numpy.float64`**
`bits`	nombre de bits	32 bits	64 bits
`nmant`	nb de bits de la mantisse	23 bits	52 bits
`nexp`	nb de bits de l'exposant	8 bits	11 bits
`max`	valeur maximale	$3{,}4 \times 10^{38}$	$1{,}8 \times 10^{308}$
`tiny`	minimum positif	$1{,}18 \times 10^{-38}$	$2{,}2 \times 10^{-308}$
`eps`	précision relative	$2^{-23} \approx 1{,}2 \times 10^{-7}$	$2^{-52} \approx 2{,}2 \times 10^{-16}$

1.2.5 Estimation de l'erreur d'arrondi

Chacun des termes de la différence est représenté avec une précision relative ε imposée par le nombre de bits de la mantisse du type de flottant.

- Simple précision : flottants sur 32 bits dont 23 de mantisse
  $\varepsilon = 2^{-23} \approx 1,2\,10^{-7}$ donné par **`numpy.finfo(numpy.float32).eps`**
- Double précision : flottants sur 64 bits dont 52 de mantisse
  $\varepsilon = 2^{-52} \approx 2,2\,10^{-16}$ cas des flottants par défaut sous python
  ε donné par **`numpy.finfo(float).eps`** c'est-à-dire
  `numpy.finfo(numpy.float64).eps`

Erreur absolue d'arrondi sur $f(t \pm h)$:

$$|\delta_a f(t+h)| \approx |\delta_a f(t-h)| \leqslant \varepsilon |f(t)|$$

Erreur absolue d'arrondi sur $f'_c(t)$ majorée par e_a :

$$|\delta_a f'_c(t)| = \frac{|\delta_a \left[f(t+h) - f(t-h)\right]|}{2h} \leqslant e_a = \frac{2\varepsilon|f(t)|}{2\boldsymbol{h}}$$

1.2.6 Comparaison des erreurs pour schéma centré à 2 termes

Erreur d'arrondi

$$e_a \propto h^{-1}$$

Pente en log-log -1

Dominante pour h faible

Erreur de troncature

$$e_t \propto h^2$$

Pente en log-log $+2$

Dominante pour h grand

Majorant de l'erreur absolue totale e

$$e \leqslant \frac{\varepsilon|f(t)|}{h} + \frac{h^2}{6}|f'''(t)| = e_m$$

Les deux erreurs varient en sens inverse selon le pas h

$\Rightarrow$ **compromis** nécessaire pour minimiser la somme des erreurs

Pas optimal $\tilde{h} = \sqrt[3]{3\varepsilon \left|\frac{f(t)}{f'''(t)}\right|} \quad \Rightarrow \quad e(\tilde{h}) = |f(t)| \sqrt[3]{\frac{9\varepsilon^2}{8} \left|\frac{f'''(t)}{f(t)}\right|}$

1.2.7 Influence de la précision : réduction de l'erreur d'arrondi

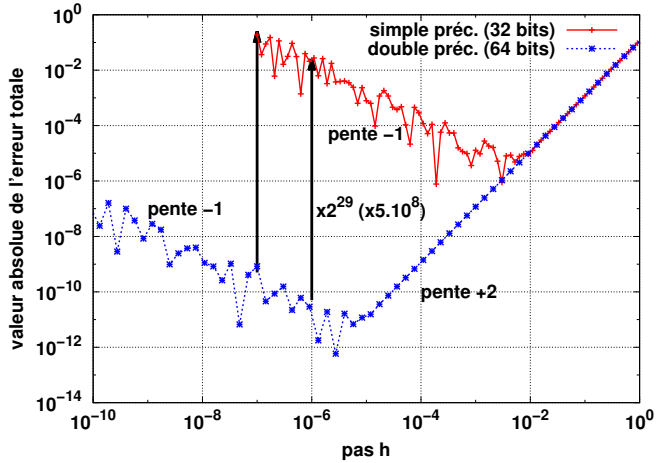


FIGURE 2: Erreur totale (valeur absolue) $|e|$ en simple et double précision de l'estimateur à 2 termes de la dérivée première en fonction du pas h **en échelle log-log**.

1.2.8 Influence du nombre de termes sur l'erreur de troncature

Objectif : éliminer les termes en h^3 dans le développement de Taylor de f
en compensant $h^3 f^{(3)}(t)$ issu des points à $\pm h$
par $(2h)^3 f^{(3)}(t) = 8h^3 f^{(3)}(t)$ issu des points à $\pm 2h$
avec une pondération relative de $-1/8$.

Schéma aux différences finies centré à **quatre termes**
pour estimer la dérivée première d'une fonction f :

$$f'(t) \approx f_{c_4}(t) = \frac{-f(t+2h) + 8f(t+h) - 8f(t-h) + f(t-2h)}{12h}$$

Les termes en puissances paires de h se compensent par symétrie
Erreur de troncature issue du terme en $h^5 f^{(5)}$ dans le développement de f donc :

$$e_t \propto h^4$$

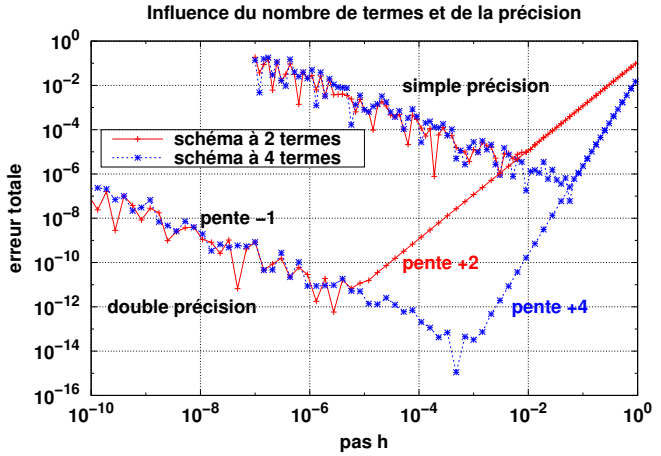


FIGURE 3: Erreur totale e des estimateurs à deux termes (rouge) et à quatre termes (bleu) de la dérivée première pour les précisions 32 et 64 bits en fonction du pas h.

1.3 Plus généralement...

Schémas d'ordre supérieur Passer d'un schéma à 2 termes à un schéma à 4 termes **améliore nettement l'erreur de troncature**, qui varie en $\boldsymbol{h^4}$ au lieu de $\boldsymbol{h^2}$.

L'erreur d'arrondi augmente mais très peu.

À précision des réels donnée, l'optimum est obtenu pour un pas plus grand et l'erreur totale est plus faible. Le schéma à 4 termes est donc préférable.

Dérivées d'ordre $\boldsymbol{n}$

$$y^{(n)} = \frac{\mathrm{d}^n y}{\mathrm{d}t^n} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}}\right) = \frac{\mathrm{d}y^{(n-1)}}{\mathrm{d}t}$$

Les schémas aux différences finies pour la dérivée d'ordre n

- présentent une **erreur d'arrondi** en $\boldsymbol{h^{-n}}$
- mais leur **erreur de troncature** dépend du nombre de termes utilisés, par ex. en h^2 pour une dérivée seconde avec un schéma centré à 3 termes.

2 Rappels sur les EDO

2.1 Définition, terminologie

EDO = **Équations Différentielles Ordinaires :** équations faisant intervenir des dérivées successives de la fonction recherchée et des fonctions de la variable indépendante (ici t).

NB. : EDP = Équations aux Dérivées Partielles avec plusieurs variables indépendantes (ex : x, y pour les coordonnées dans un plan et t pour le temps).

Dans le cas explicite, une EDO d'ordre n se met sous la forme :

$$\frac{\mathrm{d}^n y}{\mathrm{d}t^n} = f\left(t,\ y,\ \frac{\mathrm{d}y}{\mathrm{d}t},\ \ldots,\ \frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}}\right)$$

où f, connue, est appelée **le second membre**.

Une EDO d'ordre n possède une famille de solutions $y(t)$ à n paramètres.

EDO d'ordre 1 :

$$\boxed{\frac{\mathrm{d}y}{\mathrm{d}t} = f\,(t,\ y)}$$

Les solutions ne dépendent que d'un paramètre, souvent la condition initiale $y(t_0)$.
Résoudre une EDO = trouver les fonctions y de la variable indépendante t qui vérifient l'EDO.

2.2 Classification des EDO et résolution analytique

2.2.1 EDO à variables séparables du 1er ordre

Les EDO du 1er ordre **séparables** peuvent se mettre sous la forme

$$a(y)y' = b(t) \quad \text{soit} \quad \boxed{a(y)\,\mathrm{d}y = b(t)\,\mathrm{d}t} \qquad (2)$$

Si on connait des primitives $A(y)$ et $B(t)$ de $a(y)$ et $b(t)$, les solutions vérifient

$$A(y) = B(t) + k \quad \text{où } k \text{ est une constante}$$

Exemple Croissance ou extinction de population, désintégration radioactive...

$$\boxed{\frac{\mathrm{d}y}{y} = \pm\frac{\mathrm{d}t}{\tau}}$$

qui s'intègre en $\ln|y| = \pm t/\tau + k$ soit $|y| = k'\exp(\pm t/\tau)$
La constante k' est fixée par la condition initiale $y(t_0)$, donc

$$y(t) = y(t_0)\exp(\pm(t-t_0)/\tau)$$

2.2.2 EDO linéaires du 1er ordre

Une **EDO linéaire du 1er ordre** peut se mettre sous la forme

$$\boxed{y'(t) - a(t)y(t) = b(t)}$$

L'EDO **homogène** associée, dite «sans second membre» b,

$$y'(t) = a(t)y(t)$$

est séparable, donc facilement soluble si on connait une primitive de a.

De la linéarité, on déduit une propriété essentielle pour la résolution :
La solution générale de l'EDO «avec second membre» est la somme de la solution générale de l'EDO sans second membre et d'une solution particulière de l'EDO avec second membre. En pratique, la difficulté sera souvent de trouver **une** solution particulière de l'EDO avec second membre.

Recherche par «variation de la constante» :

L'EDO homogène s'écrit $y'/y = a(t)$. Elle s'intègre en $y = \mathrm{C^{ste}}e^{A(t)}$ où A est une primitive de a donc $A'(t) = a(t)$.

Rechercher une solution de l'EDO avec second membre sous la forme

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

$$y'(t) = \lambda'(t)e^{A(t)} + \lambda(t)a(t)e^{A(t)}$$

L'EDO prend la forme

$$y' - ay = b(t) = \lambda'(t)e^{A(t)} \quad \text{soit} \quad \lambda'(t) = b(t)e^{-A(t)}$$

$$\text{D'où} \quad \lambda(t) = \int_{t_0}^{t} b(u)e^{-A(u)}\ \mathrm{d}u$$

La solution particulière s'écrit donc

$$y(t) = e^{A(t)}\int_{t_0}^{t} b(u)e^{-A(u)}\ \mathrm{d}u$$

Et la solution générale de l'EDO

$$y(t) = e^{A(t)}\left[\mathrm{C^{ste}} + \int_{t_0}^{t} b(u)e^{-A(u)}\ \mathrm{d}u\right]$$

Reste à trouver une primitive de $b(u)e^{-A(u)}$...

Autre méthode sur un exemple

Décroissance d'une population avec apports externes périodiques

$$\boxed{\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{y}{\tau} + b\sin\omega t}$$

La solution générale peut ici se décomposer en une composante transitoire, en $\exp(-t/\tau)$, qui disparait au bout de quelques constantes de temps, plus une solution sinusoïdale permanente de réponse au forçage. On peut choisir cette dernière comme solution particulière.

2.2.3 EDO linéaires du 2e ordre à coefficients constants

EDO linéaire 2e ordre à coefficients a, b, et c constants :

$$\boxed{ay'' + by' + cy = d(t)}$$

où $d(t)$ est parfois appelé l'excitation ou forçage.

Équation homogène $d = 0$: En recherchant des solutions (dites **libres**) sous la forme de combinaisons linéaires d'exponentielles e^{rt}, où r est une constante éventuellement complexe, on obtient l' **équation caractéristique**

$$ar^2 + br + c = 0$$

Elle permet de discuter la nature des solutions selon le signe de $\Delta = b^2 - 4ac$.

— $\Delta > 0$ donc deux solutions réelles r_1 et r_2

$$y(t) = \lambda_1 e^{r_1 t} + \lambda_2 e^{r_2 t}$$

— $\Delta = 0$ donc une solution double r_0

$$y(t) = e^{r_0 t}\left[\lambda_1 + \lambda_2 t\right]$$

— $\Delta < 0$ donc deux solutions complexes conjuguées $r = \alpha \pm i\beta$

$$y(t) = e^{\alpha t}\left[\lambda_1\cos(\beta t) + \lambda_2\sin(\beta t)\right]$$

3 Introduction à la résolution numérique des EDO

3.1 Problème différentiel

— équation différentielle scalaire d'ordre $\boldsymbol{n}$

$$\boxed{\frac{\mathrm{d}^n y}{\mathrm{d}t^n} = f\left(t,\ y,\ \frac{\mathrm{d}y}{\mathrm{d}t},\ \ldots,\ \frac{\mathrm{d}^{n-1}y}{\mathrm{d}t^{n-1}}\right)}$$

où f est la fonction second membre donnée

$\Rightarrow$ **famille** de solutions $y(t)$ à $\boldsymbol{n}$ paramètres

— ensemble de $\boldsymbol{n}$ conditions imposées

$\Rightarrow$ choix d' **une** solution dans la famille

3.2 Deux types de problèmes différentiels à résoudre

— Conditions initiales données pour une seule valeur t_0 de t, par exemple

$$y(t_0) = y_0,\quad y'(t_0) = y'_0,\ \ldots\quad ,\quad y^{(n-1)}(t_0) = y_0^{(n-1)}$$

Problème de **conditions initiales** ou de **Cauchy**

— Conditions données pour des valeurs distinctes de la variable indépendante t, par exemple :

$$y(t_0) = y_0,\quad y(t_1) = y_1,\ \ldots\quad ,\quad y(t_{n-1}) = y_{n-1}$$

Problème de **conditions aux limites** (non traité, sauf problème de tir).

3.3 Équations différentielles scalaires du 1er ordre

Étudier d'abord les équations différentielles scalaires **du premier ordre**.
⇒ **famille de solutions** $y(t)$ à **un** paramètre (y_0)

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t,\ y(t)) \quad \text{avec} \quad y(t_0) = y_0$$ condition initiale

Résolution numérique approchée sur l'intervalle $[t_0, t_0 + L]$ de longueur L
⇒ approximation $u_i = u(t_i) \approx y(t_i)$ en n instants $t_0 < t_i \leqslant t_n = t_0 + L$.

Les EDO d'ordre supérieur se ramènent à des systèmes différentiels couplés du premier ordre (EDO vectorielles du premier ordre).

3.4 Unicité et problème bien posé : conditions suffisantes

La **condition de Lipschitz**

$$|f(t, y_2) - f(t, y_1)| \leqslant K\ |y_2 - y_1| \tag{3}$$

assure l'**unicité** de la solution.

$$\left|\frac{\partial f}{\partial y}(t, y)\right| \leqslant K \quad \text{dans un domaine convexe} \tag{4}$$

⇒ condition de Lipschitz vérifiée.

⚠ Les erreurs d'arrondi amènent à **toujours résoudre un problème perturbé**.

Problème bien posé si : le problème faiblement perturbé (second membre ou condition initiale) possède une solution proche de celle du problème original.
La condition de Lipschitz assure que le problème est bien posé.

3.5 Méthodes de résolution numérique et notations

Discrétisation par découpage de l'intervalle de longueur L selon un pas constant h

Échantillonnage de la solution aux instants $t_i = t_0 + ih$ pour $1 \leqslant i \leqslant n$.
Solution numérique : $u_i =$ approximation de $y(t_i)$

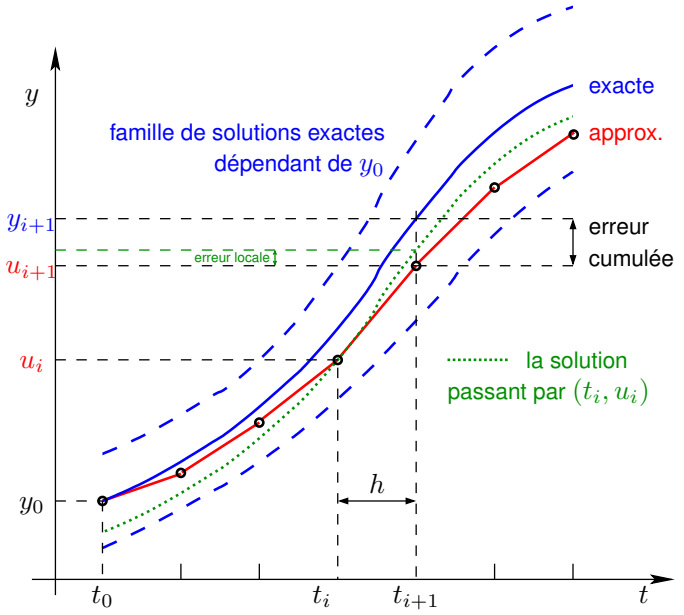
À partir de la condition initiale $u_0 = y(t_0)$ imposée,
faire une **boucle** sur les abscisses t_i pour calculer l'approximation u_{i+1} à t_{i+1}
→ approximer ainsi de proche en proche la solution sur l'intervalle L.

⇒ accumulation des erreurs dans la boucle

À chaque pas de la boucle, pour calculer u_{i+1}, on peut s'appuyer :

— sur **la dernière valeur** calculée u_i : **méthodes à un pas**

— sur **plusieurs valeurs** $u_{i-k} (k \geqslant 0)$ antérieurement calculées :
méthodes à plusieurs pas (initialisation nécessaire par méthode à un pas)



Méthode à pas constant

Découpage de l'intervalle de longueur L selon un pas fixe $h = L/n$.

$u_i =$ approximat. de $y(t_i)$

Un pas :
$t_i \to t_{i+1}$
$u_i \to u_{i+1}$

4 Méthodes à un pas

Constituent l'algorithme de base qui permet d'estimer la valeur de la solution à l'instant $t_{i+1} = t_i + h$, connaissant seulement u_i, celle à t_i.

La valeur à estimer peut être approchée par un développement limité de Taylor :

$$y(t_i + h) = y(t_i) + h\frac{\mathrm{d}y}{\mathrm{d}t}(t_i) + \frac{h^2}{2}\frac{\mathrm{d}^2y}{\mathrm{d}t^2}(t_i) + \cdots \tag{5}$$

Ordre n de la méthode = plus grande puissance de h prise en compte dans l'approximation.

— Somme des termes négligés = **erreur de troncature locale** $\propto h^{n+1}$ déterministe, augmente si le pas h augmente et si l'ordre de la méthode diminue

— Précision finie des opérations sur les réels ⇒ **erreur d'arrondi** aléatoire augmente lorsque les calculs se compliquent, en particulier si le pas h diminue.

Indépendamment du coût (en temps de calcul) des opérations, et des cas où la fonction est tabulée, **ne pas croire que diminuer le pas améliore toujours** la qualité du résultat : un **compromis** doit être trouvé entre ces deux types d'erreurs.

4.1 Méthodes du premier ordre

4.1.1 Méthode d'Euler progressive (explicite)

Méthode du premier ordre d'intérêt pédagogique, à éviter en pratique

$$u_{i+1} = u_i + hf(t_i, u_i) \tag{6}$$

Exemple : stabilité

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{y}{\tau} \quad \Rightarrow \quad \text{solution analytique} \quad y = y_0 e^{-t/\tau} \Rightarrow y_n = y_0(e^{-h/\tau})^n$$

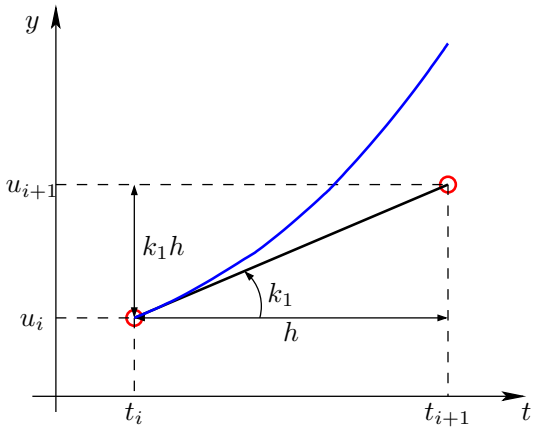
$$u_{i+1} = u_i - \frac{h}{\tau}u_i \quad \Rightarrow \quad \text{solution numérique} \quad u_n = y_0\,(1 - h/\tau)^n$$

Si $\tau > 0$, la solution exacte vérifie $y(\infty) = 0$,

Mais pour l'approximation, $u_n \to 0 \iff |1 - h/\tau| < 1 \iff 0 < h < 2\tau$.

Condition de **stabilité** : $h < 2\tau$ (pas h petit)

Mais, si $h > \tau$, alors $(1 - h/\tau) < 0$: alternance de signe de la solution u_n.



Méthode d'Euler

Méthode **explicite** qui ne nécessite qu'une seule évaluation de la fonction second membre f par pas :

$k_1 = f(t_i, u_i)$

facilement **instable**

$$\frac{u_{i+1} - u_i}{h} = f(t_i, u_i)$$

voir dérivée avant

4.1.2 Méthode d'Euler rétrograde (implicite)

$$u_{i+1} = u_i + hf(t_{i+1}, \boldsymbol{u_{i+1}}) \tag{7}$$

Méthode **implicite** : résolution itérative, plus difficile à mettre en œuvre, sauf si la forme de $f(t, u)$ permet le calcul analytique de u_{i+1} à partir de l'équation (7).

Avantage : meilleure **stabilité** que la méthode progressive explicite.

Exemple : stabilité

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{y}{\tau} \quad \Rightarrow \quad \text{solution analytique} \quad y = y_0 e^{-t/\tau} \Rightarrow y_n = y_0(e^{-h/\tau})^n$$

$$u_{i+1} = u_i - \frac{h}{\tau}u_{i+1} \quad \Rightarrow \quad \text{solution numérique} \quad u_{i+1} = \frac{u_i}{1 + h/\tau}$$

$$u_n = \frac{y_0}{(1 + h/\tau)^n}$$

Si $\tau > 0,\ y(\infty) = 0$, et aussi $u_n \to 0 \quad \forall\tau > 0,\ \forall h > 0$ solution **stable**

Mise en œuvre de la méthode d'Euler rétrograde : résolution par itération de l'équation implicite $u_{i+1} = u_i + hf(t_{i+1}, \boldsymbol{u_{i+1}})$

Itérer l'application g pour rechercher son **point fixe** où $v = g(v)$

$$\boxed{v_2' = g(v_2) = u_i + hf(t_2, v_2)}$$

Ce point fixe est la solution de l'équation implicite.

— Utilise plusieurs évaluations du second membre, sans calcul de ses dérivées.

— Très peu d'itérations nécessaires

Initialisation par le prédicteur avec Euler progressif

$$\begin{aligned} t_2 &= t_i + h \\ k_1 &= f(t_i, u_i) \\ v_2 &= u_i + hk_1 \end{aligned}$$

Boucle pour recherche du point fixe de $g(v_2) = v_2' = u_i + hf(t_2, v_2)$

$$\begin{aligned} k_2 &= f(t_2, v_2) \\ v_2' &= u_i + hk_2 \\ \delta v_2 &= v_2' - v_2 \\ &\text{arrêt si } |\delta v_2|^2 \leqslant \alpha^2 |v_2|^2 \quad (\alpha \text{ petit}) \\ v_2 &= v_2' \end{aligned}$$

La fonction g est contractante si $|g'(v_2)| = h\left|\frac{\partial f}{\partial v_2}\right| \leqslant 1,$

Si la majoration $\left|\frac{\partial f}{\partial v_2}\right| \leqslant K$ (souvent invoquée pour assurer la condition de Lipschitz) est vérifiée, g est contractante si le pas h est assez faible.

Critère d'arrêt : choisir α faible, mais $\alpha > \varepsilon$.

4.2 Méthodes du deuxième ordre

Première idée : augmenter le nombre de termes du développement de Taylor : rarement utilisé, car nécessite l'évaluation des dérivées partielles de f.

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t,\ y(t)) \qquad \Rightarrow \qquad \frac{\mathrm{d}^2y}{\mathrm{d}t^2} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y} \qquad (10)$$

Préférer utiliser **plusieurs évaluations du second membre** f en des points adaptés.

Centrer l'évaluation de la dérivée au point milieu $t_m = (t_i + t_{i+1})/2$.

$$y(t_i + h) = y(t_m) + \frac{h}{2}\frac{\mathrm{d}y}{\mathrm{d}t}(t_m) + \frac{1}{2}\frac{h^2}{4}\frac{\mathrm{d}^2y}{\mathrm{d}t^2}(t_m) + O(h^3) \qquad (11a)$$

$$y(t_i) = y(t_m) - \frac{h}{2}\frac{\mathrm{d}y}{\mathrm{d}t}(t_m) + \frac{1}{2}\frac{h^2}{4}\frac{\mathrm{d}^2y}{\mathrm{d}t^2}(t_m) + O(h^3) \qquad (11b)$$

Par différence, (approximation locale parabolique, voir aussi dérivée centrée à 2 termes)

$$y(t_i + h) - y(t_i) = h\frac{\mathrm{d}y}{\mathrm{d}t}(t_m) + O(h^3)$$

4.2.1 Méthode du point milieu

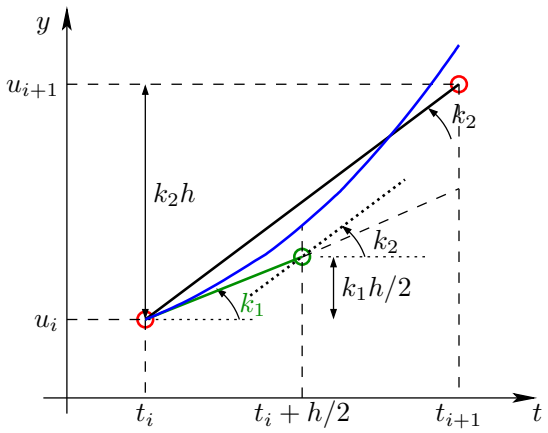
Nécessite l'évaluation du second membre f en 2 points :
en (t_i, u_i) et **au milieu** $(t_{i+1/2} = t_i + h/2,\ u_{i+1/2})$ d'un pas (hors grille).

$$\boxed{u_{i+1} = u_i + hf\left(t_i + \frac{h}{2},\ u_i + \frac{h}{2}f(t_i, u_i)\right)}$$

$$k_1 = f(t_i,\ u_i) \qquad (12a)$$

($u_{i+1/2}$ calculé via Euler) $\qquad k_2 = f(t_i + \frac{h}{2},\ u_i + k_1\frac{h}{2}) \qquad (12b)$

$$u_{i+1} = u_i + hk_2 \qquad (12c)$$



Méthode du point milieu

Méthode **explicite** qui nécessite deux évaluations du second membre par pas dont une hors grille.

4.2.2 Méthode d'Euler modifiée

En appliquant 11a et 11b à la dérivée et en faisant la somme, on peut remplacer la dérivée au milieu par la moyenne des dérivées aux extrémités de l'intervalle (voir méthode de quadrature dite des trapèzes) :

$$\frac{\mathrm{d}y}{\mathrm{d}t}(t_i) + \frac{\mathrm{d}y}{\mathrm{d}t}(t_{i+1}) = 2\frac{\mathrm{d}y}{\mathrm{d}t}(t_m) + O(h^2)$$

D'où une approximation n'utilisant pas la valeur de f au point milieu t_m :

$$u_{i+1} = u_i + \frac{h}{2}\left[f(t_i, u_i) + f(t_{i+1}, \boldsymbol{u_{i+1}})\right]$$

De nouveau, méthode a priori **implicite**, plus stable, mais plus lourde.

$\Rightarrow$ Contournement du problème en utilisant l'approximation d'Euler explicite (voir 6) pour évaluer u_{i+1} intervenant dans f.

$$u_{i+1} = u_i + \frac{h}{2}\left[f(t_i, u_i) + f(t_{i+1}, u_i + hf(t_i, u_i))\right]$$

Bilan : méthode de type **prédicteur-correcteur** équivalent à

— un demi-pas avec la pente initiale k_1

— et un demi-pas avec la pente k_2 du point prédit par Euler progressif.

$$k_1 = f(t_i,\ u_i) \tag{13a}$$

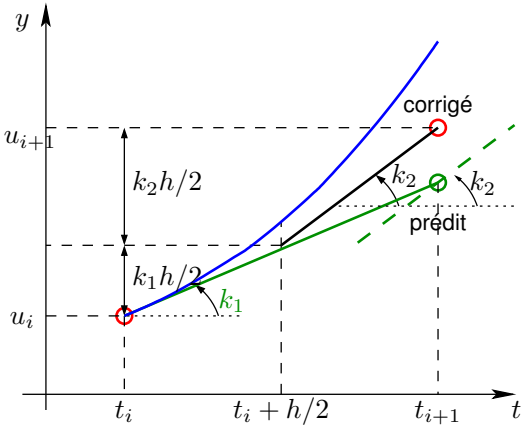
$$k_2 = f(t_{i+1},\ u_i + k_1 h) \tag{13b}$$

$$u_{i+1} = u_i + \frac{h}{2}\left[k_1 + k_2\right] \tag{13c}$$

Remarques

— deuxième ordre comme point milieu mais sans évaluation hors grille

— la résolution de l'équation implicite peut se faire en itérant la correction jusqu'à ce qu'elle devienne négligeable (voir Euler rétrograde avec $g' = \frac{h}{2}\frac{\partial f}{\partial y}$).



Méthode d'Euler modifiée

Méthode **explicite** qui nécessite deux évaluations de la fonction par pas en des points de la grille.

4.3 Méthodes de Runge Kutta

Plus généralement, avec r évaluations de f, on peut atteindre une méthode d'ordre r si $r \leqslant 4$. Pour atteindre l'ordre 5, six évaluations sont nécessaires.

$\Longrightarrow$ la méthode de Runge Kutta d'ordre 4 est très utilisée.

4.3.1 Méthode de Runge Kutta d'ordre 3

$$k_1 = f(t_i, u_i) \tag{14a}$$

$$k_2 = f(t_i + \frac{h}{2},\ u_i + k_1\frac{h}{2}) \tag{14b}$$

$$k_3 = f(t_i + h,\ u_i + (2k_2 - k_1)h) \tag{14c}$$

$$u_{i+1} = u_i + (k_1 + 4k_2 + k_3)\frac{h}{6} \tag{14d}$$

4.3.2 Méthode de Runge Kutta d'ordre 4

$$k_1 = f(t_i,\ u_i) \tag{15a}$$

$$k_2 = f(t_i + \frac{h}{2},\ u_i + k_1\frac{h}{2}) \tag{15b}$$

$$k_3 = f(t_i + \frac{h}{2},\ u_i + k_2\frac{h}{2}) \tag{15c}$$

$$k_4 = f(t_i + h,\ u_i + k_3 h) \tag{15d}$$

$$u_{i+1} = u_i + (k_1 + 2k_2 + 2k_3 + k_4)\frac{h}{6} \tag{15e}$$

4.4 Erreur absolue en fonction du pas et de l'ordre

nombre de pas $= L/h \quad \Longrightarrow \quad$ erreur globale $\sim$ erreur locale $\times\ L/h$

TABLE 1: Erreur de **troncature seule**

Méthode	ordre	erreur locale	erreur globale
Euler explicite	1	$\propto h^2$	$\propto h$
Point milieu – Euler modifiée	2	$\propto h^3$	$\propto h^2$
Runge-Kutta 3	3	$\propto h^4$	$\propto h^3$
Runge-Kutta 4	4	$\propto h^5$	$\propto h^4$

Erreur d'**arrondi** locale indépendante de $h \Rightarrow$ erreur d'arrondi globale $\propto 1/h$

4.5 Exemple de l'équation logistique

$$\boxed{\frac{\mathrm{d}y}{\mathrm{d}t} = ay\left(1 - \frac{y}{k}\right)} \tag{16}$$

Par séparation des variables, puis décomposition en éléments simples, elle s'écrit

$$a\,\mathrm{d}t = \frac{\mathrm{d}y}{y\,(1 - y/k)} = \frac{\mathrm{d}y}{y} + \frac{\mathrm{d}y}{k - y}$$

Par intégration des trois termes :

$$at + \text{constante} = \ln|y| - \ln|k - y|$$

On identifie la constante en supposant qu'à l'instant initial t_0, $0 < y_0 = y(t_0) < k$.

La solution analytique se met sous la forme :

$$y(t) = \frac{k}{1 + \dfrac{k - y_0}{y_0}\exp\left(-a(t - t_0)\right)} \tag{17}$$

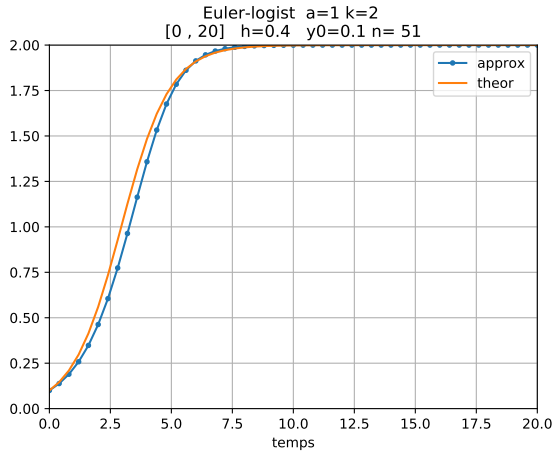


FIGURE 4: Solution analytique et approximation par la méthode d'Euler avec un pas $h = 0.4$ de l'équation logistique pour $t_0 = 0, y(t_0) = 0.1, a = 1$ et $k = 2$.

4.5.1 Erreurs en fonction du temps

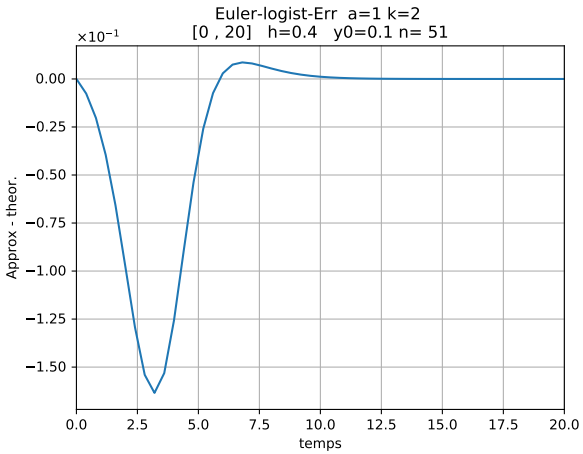


FIGURE 5: Erreur dans l'intégration de l'équation logistique avec la méthode d'Euler progressive avec $h = 0,4$. L'allure régulière montre que l'**erreur de troncature** domine. L'erreur de troncature locale ($0{,}163$ max) est liée à la courbure de la solution.

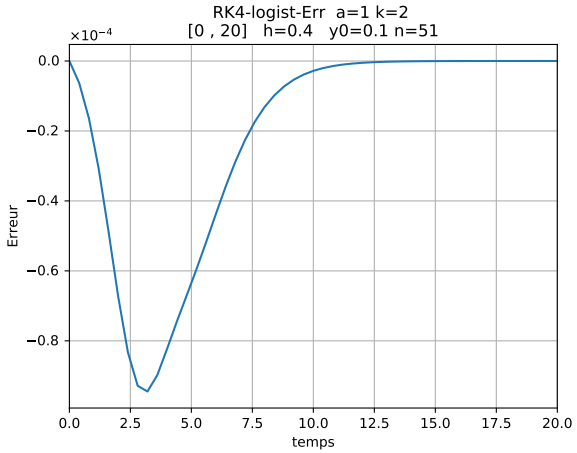


FIGURE 6: Erreur dans l'intégration de l'équation logistique avec la méthode de Runge Kutta d'ordre 4 et $h = 0,4$. Noter la diminution de l'erreur maximale d'un facteur 2000 environ par rapport à la méthode d'Euler pour un même pas.

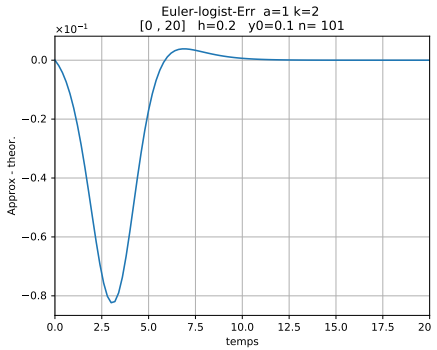


FIGURE 7: Erreur avec la méthode d'Euler et $h = 0,2$. Diviser le pas par 2 divise l'erreur maximale par 2.

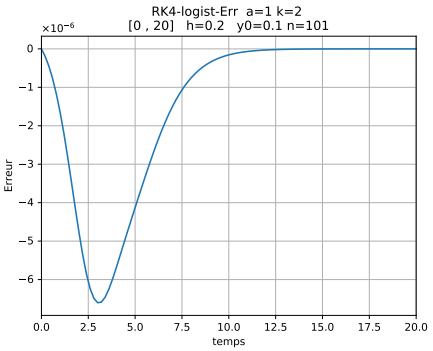


FIGURE 8: Erreur avec la méthode RK4 et $h = 0,2$. Diviser le pas par 2 divise l'erreur maximale par $2^4 = 16$.

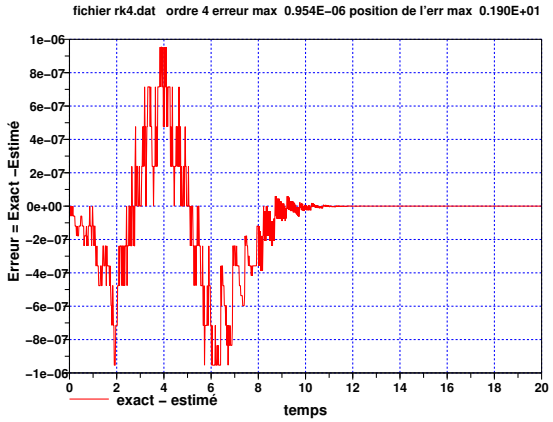


FIGURE 9: Erreur avec des **flottants sur 32 bits** avec la méthode de Runge Kutta d'ordre 4 pour $h = 0,02$. L'allure **bruitée** est caractéristique de l'**erreur d'arrondi** et on retrouve les niveaux de quantification des réels sur 32 bits ($\varepsilon_{32} \approx 1{,}2 \times 10^{-7}$). En python, par défaut les float sont sur 64 bits ($\varepsilon_{64} \approx 2{,}2 \times 10^{-16}$), donc l'erreur d'arrondi est négligeable avec ce pas.

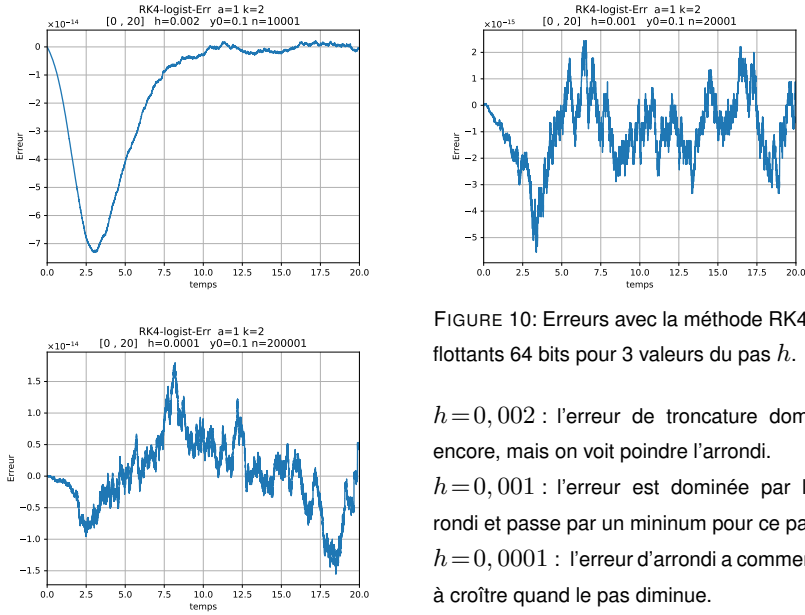
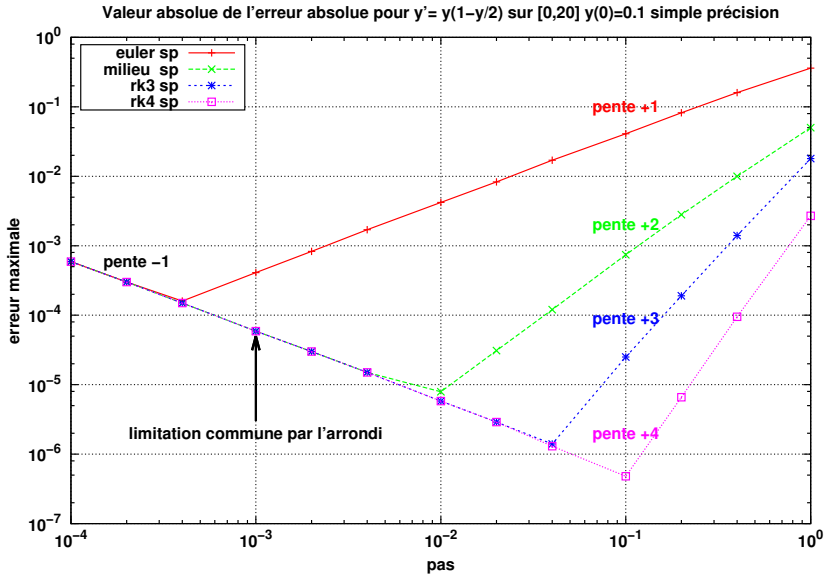


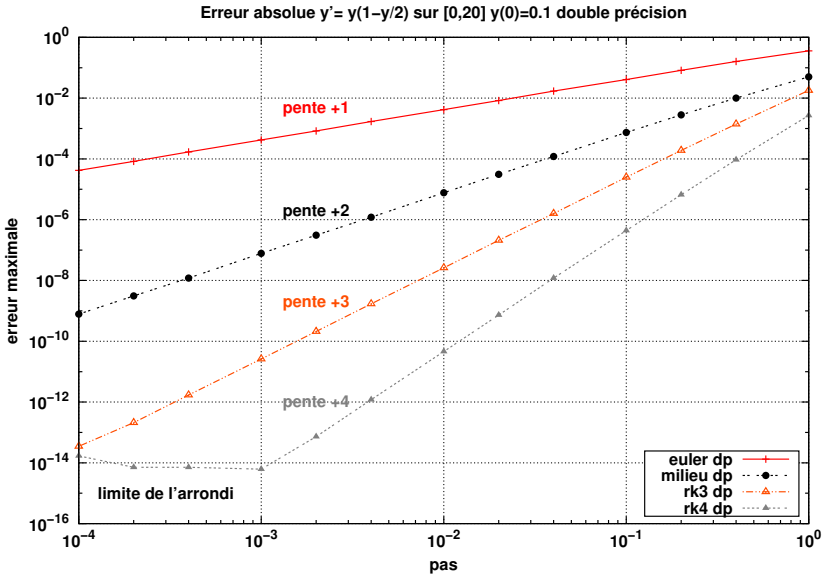
FIGURE 10: Erreurs avec la méthode RK4 en flottants 64 bits pour 3 valeurs du pas h.

$h = 0,002$: l'erreur de troncature domine encore, mais on voit poindre l'arrondi.
$h = 0,001$: l'erreur est dominée par l'arrondi et passe par un mininum pour ce pas.
$h = 0,0001$: l'erreur d'arrondi a commencé à croître quand le pas diminue.

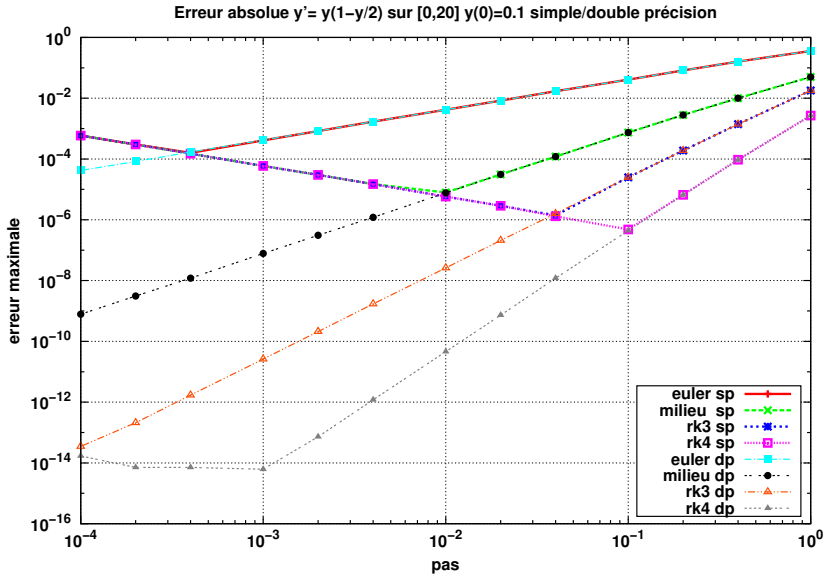
4.5.2 Erreur totale maximale en simple précision (32 bits) en fonction du pas



4.5.3 Erreur totale maximale en double précision (64 bits) en fonction du pas



4.5.4 Comparaison des erreurs maximales simple/double précision



4.5.5 Méthodes de prédicteur correcteur

Principe : bénéficier des qualités d'une méthode implicite mais l'appliquer à une estimation obtenue par une méthode explicite du même ordre (voir Euler modifiée).

— **prédiction** de u_{i+1} par une méthode **explicite**

— **correction** de u_{i+1} par une formule **implicite** où $f(t_{i+1}, y(t_{i+1}))$ a été approximé par la prédiction $f(t_{i+1}, u_{i+1})$.

Exemple : méthode d'Euler modifiée
Une itération de la partie correction est possible.
L'ordre est celui du correcteur, mais la stabilité dépend plus du prédicteur.
Ces méthodes permettent d'**estimer l'erreur de troncature** à partir de la différence entre prédicteur et correcteur $\Longrightarrow$ adaptation du pas

Plus généralement, **les méthodes adaptatives** sont celles où on ajuste **localement** le pas aux accidents de la solution pour obtenir une précision imposée.

5 Les EDO du premier ordre en pratique

5.1 Structure des programmes de résolution d'EDO

1. un module comportant les différentes **méthodes** (Euler, Point Milieu et RK4) : algorithmes de base s'appliquant à une fonction second membre f passée en argument permettant d'**avancer d'un pas** dans l'intégration de l'EDO
2. un module comportant **les fonctions seconds membres** de l'équation différentielle et les éventuelles solutions analytiques exactes ou approchées
3. une fonction d'**intégration** qui choisit la méthode, le second membre, les paramètres (début, fin, pas et conditions initiales par ex.). Elle déclenche et arrête la boucle d'intégration et stocke les résultats dans des tableaux.
4. un module d'**utilitaires** notamment pour écrire les résultats dans un fichier et tracer les solutions et les écarts avec l'analytique s'il existe.

5.2 Échelles de temps et problèmes raides

Ne pas oublier que chaque problème différentiel possède une ou plusieurs **échelles de temps propres** (périodes ou pseudo-périodes, constantes de temps).
La solution ne peut être représentée correctement qu'avec un pas assez inférieur au plus petit de ces temps propres.
Cette analyse impose donc une valeur maximale pour le pas.

Certains problèmes différentiels qualifiés de **raides** comportent des échelles de temps très différentes : leur intégration numérique s'avère délicate et coûteuse (pas faibles pour respecter le temps court, mais nombreux pour accéder au temps long).
Il existe des méthodes spécifiques des EDO raides qui ne sont pas présentées ici.

5.3 Validation des résultats

Validation via une solution analytique d'un problème simplifié

Lorsqu'une solution analytique est disponible (par exemple pour certaines valeurs de paramètres qui permettent de simplifier l'EDO), sa comparaison avec la solution numérique permet de tester la méthode. Le calcul de l'erreur dans le domaine où la troncature domine permet d'extrapoler l'effet d'un changement de pas connaissant l'ordre de la méthode.

Validation sans solution analytique

Dans le cas où aucune solution analytique de référence n'est disponible, la validation s'appuie sur les mêmes outils que les méthodes adaptatives :

— diminution du pas (division par 2)
— augmentation de l'ordre de la méthode
— calcul d'invariants (énergie par exemple)

5.4 Cas d'un second membre défini par intervalles

Dans le cas où le second membre est défini de façon différente suivant les intervalles de temps, il peut présenter des «accidents» aux bords de ces intervalles (non-dérivabilité, voire discontinuité). Un exemple classique est celui d'un second membre comportant un forçage constant par intervalle, tel un signal carré.

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f_0(t,y)+h(t) \qquad \text{où} \quad h(t) = \begin{cases} +c & \text{si} \quad t \in [nT_0, nT_0 + T_0/2[\\ -c & \text{si} \quad t \in [nT_0 + T_0/2, nT_0 + T_0[\end{cases}$$

La solution $y(t)$ peut être non-dérivable (ou discontinue) en ces instants, tout en restant dérivable (ou continue) à gauche et à droite (voir charge/décharge de condensateur).

Aussi bien analytiquement que numériquement, il faut donc **intégrer l'EDO indépendamment dans chaque intervalle**. La condition finale de l'intervalle de gauche devient la condition initiale de celui de droite.

⚠ Éviter absolument un pas d'intégration à cheval sur cette discontinuité.

6 Systèmes d'équations différentielles du 1er ordre

6.1 Extension des méthodes scalaires explicites aux vecteurs

Système de n équations différentielles couplées du premier ordre associées à n conditions initiales

considérer les **vecteurs** $\vec{\mathbf{y}}$ et $\vec{\mathbf{f}}$.

$$\begin{aligned} \frac{\mathrm{d}y_1}{\mathrm{d}t} &= f_1(t,\ y_1, y_2, \ldots, y_n) \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= f_2(t,\ y_1, y_2, \ldots, y_n) \\ \ldots &= \ldots \\ \frac{\mathrm{d}y_n}{\mathrm{d}t} &= f_n(t,\ y_1, y_2, \ldots, y_n) \end{aligned}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} f_1 \\ f_2 \\ \ldots \\ f_n \end{pmatrix}$$

Les **méthodes explicites** de résolution des équations différentielles scalaires du premier ordre **s'appliquent aux systèmes**.

$$\frac{\mathrm{d}\vec{\mathbf{y}}}{\mathrm{d}t} = \vec{\mathbf{f}}(t,\ \vec{\mathbf{y}})$$

À chaque étape, effectuer les calculs **sur chaque composante** avant de passer à l'étape suivante : **exemple avec la méthode du point milieu**

Étape 1 : vecteur des pentes au bord gauche t_i de l'intervalle $[t_i, t_i + h]$

$$\vec{\mathbf{k_1}} = \vec{\mathbf{f}}(t_i,\ \vec{\mathbf{u}_i})$$

$$\begin{aligned} k_{1,1} &= f_1(t_i,\ u_{i,1},\ u_{i,2},\ \ldots,\ u_{i,n}) \\ k_{1,2} &= f_2(t_i,\ u_{i,1},\ u_{i,2},\ \ldots,\ u_{i,n}) \\ \ldots &= \ldots \\ k_{1,n} &= f_n(t_i,\ u_{i,1},\ u_{i,2},\ \ldots,\ u_{i,n}) \end{aligned}$$

avant de calculer...

Étape 2 : vecteur des pentes au point milieu prédit en $t_i + h/2$

$$\vec{\mathbf{k}_2} = \vec{\mathbf{f}}\,(t_i + h/2,\ \vec{\mathbf{u}_i} + \vec{\mathbf{k}_1} h/2)$$

$$\begin{aligned}
k_{2,1} &= f_1(t_i + h/2,\ u_{i,1} + k_{1,1}h/2,\ u_{i,2} + k_{1,2}h/2,\ \ldots,\ u_{i,n} + k_{1,n}h/2)\\
k_{2,2} &= f_2(t_i + h/2,\ u_{i,1} + k_{1,1}h/2,\ u_{i,2} + k_{1,2}h/2,\ \ldots,\ u_{i,n} + k_{1,n}h/2)\\
\ldots &= \ldots\\
k_{2,n} &= f_n(t_1 + h/2,\ u_{i,1} + k_{1,1}h/2,\ u_{i,2} + k_{1,2}h/2,\ \ldots,\ u_{i,n} + k_{1,n}h/2)
\end{aligned}$$

Étape 3 : vecteur résultat au bord droit $t_i + h$ de l'intervalle

$$\vec{\mathbf{u}}_{i+1} = \vec{\mathbf{u}_i} + h\,\vec{\mathbf{k}_2}$$

$$u_{i+1,j} = u_{i,j} + hk_{2,j} \qquad 1 \leqslant j \leqslant n$$

6.2 Application des méthodes implicites aux vecteurs

⚠ Les **méthodes implicites** nécessitent de résoudre à chaque pas un système d'équations a priori non linéaires.

Cette résolution numérique se fait de façon itérative, souvent par linéarisation locale via la **matrice jacobienne** $\left(\left(\frac{\partial f_i}{\partial y_j}\right)\right)_{1\leqslant i,j\leqslant n}$.

6.3 Outils disponibles sous python

Les outils d'intégration sous python sont regroupés sous **`scipy.integrate`** et concernent les systèmes d'EDO avec la notation vectorielle.

La fonction **`scipy.integrate.odeint`** est une interface générale d'intégration qui choisit automatiquement la méthode suivant qu'il s'agit d'**EDO raides ou non**. On lui fournit les arguments obligatoires suivants :

- la fonction second membre (**`func`**) ;
- le vecteur des conditions initiales (**`y0`**) ;
- le vecteur (**`t`**) des instants où la solution doit être évaluée, mais c'est l'**intégrateur adaptatif qui choisit le pas** de façon optimale.

Le passage des **paramètres** des fonctions second membre se fait avec l'argument optionnel de mot clef **`args`** sous forme d'un t-uple comme pratiqué en TE.

`scipy.integrate` comporte aussi une interface plus orientée objet, la classe **`ode`** qui permet de choisir les méthodes d'intégration, parmi lesquelles :

- des méthodes explicites : par exemple **`dopri5`** qui utilise des schémas de Runge Kutta d'ordres 5 et 4 ;
- des méthodes implicites : par exemple **`vode`**. Dans ce cas, il vaut mieux fournir la **matrice jacobienne** analytique (argument optionnel **`jac`**), au lieu de laisser l'intégrateur l'évaluer approximativement par différences finies.

N.B : Les versions les plus récentes de `scipy` privilégient la nouvelle fonction **`solve_ivp`** (*solve initial value problem*).

6.4 Mise en œuvre vectorielle des méthodes à un pas

- Les méthodes d'intégration doivent fonctionner **quelle que soit la taille p des vecteurs** qui représentent la solution $\vec{\mathbf{y}}$ et le second membre $\vec{\mathbf{f}}$ de l'EDO.
- C'est **le programme principal qui fixera cette taille**.
  Il devra donc choisir un second membre de la même dimension.
- Il fixe aussi la condition initiale $\vec{\mathbf{y}(\mathbf{t}_0)}$ qui est un vecteur à p composantes .
- La solution approchée est représentée par un tableau 2D :
  - première dimension n : le nombre d'instants
  - la deuxième dimension p : le nombre de composantes de $\vec{\mathbf{y}}$.
- Les tailles des tableaux des seconds membres effectifs seront héritées du programme principal et **non déclarées explicitement**.
  Mais seules les p composantes effectives de $\vec{\mathbf{f}}$ (2 pour Lotka Volterra) seront calculées à partir des p composantes de $\vec{\mathbf{y}}$.

6.5 Exemple de système non-linéaire couplé du premier ordre : équations de Lotka-Volterra

Deux populations en conflit : modèle proies (y_1) – prédateurs (y_2)

$a_1 = 1/\tau_1$ = taux de croissance de y_1 (proies) en l'absence de y_2 (prédateurs)

$a_2 = 1/\tau_2$ = taux de décroissance de y_2 (prédateurs) en l'absence de y_1 (proies)

Termes de couplage non-linéaires en $y_1 y_2$ (rencontre des 2 espèces)

$\frac{a_1}{k_2} y_2$ = taux de destruction des proies par les prédateurs

$\frac{a_2}{k_1} y_1$ = taux de croissance des prédateurs au détriment des proies

$$\frac{\mathrm{d}y_1}{\mathrm{d}t} = +a_1 y_1 \left(1 - \frac{y_2}{k_2}\right) \qquad (18a)$$

$$\frac{\mathrm{d}y_2}{\mathrm{d}t} = -a_2 y_2 \left(1 - \frac{y_1}{k_1}\right) \qquad (18b)$$

Solutions **périodiques**

Lotka-Volterra : cycle dans le plan de phase

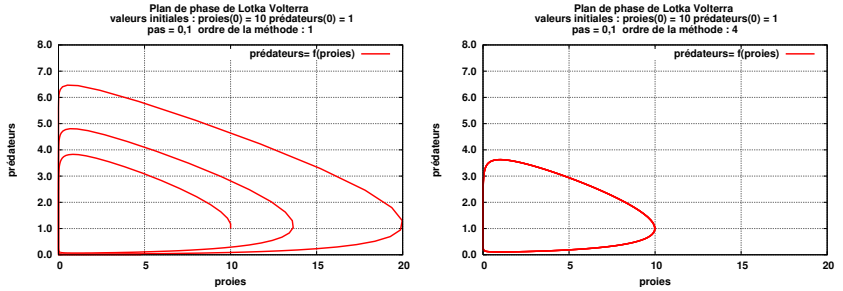
En éliminant le temps, on obtient un invariant, donc des solutions périodiques :

$$\frac{\mathrm{d}y_2}{\mathrm{d}y_1} = -\frac{a_2 y_2}{a_1 y_1}\,\frac{1 - y_1/k_1}{1 - y_2/k_2} \quad \Rightarrow \quad \boxed{y_1^{a_2} y_2^{a_1} \mathrm{e}^{-a_1 y_2/k_2 - a_2 y_1/k_1} = \mathrm{C^{te}}}$$

Tangentes horizontales pour $y_1 = k_1$ (ou $y_2 = 0$) : équilibre des prédateurs

Tangentes verticales pour $y_2 = k_2$ (ou $y_1 = 0$) : équilibre des proies

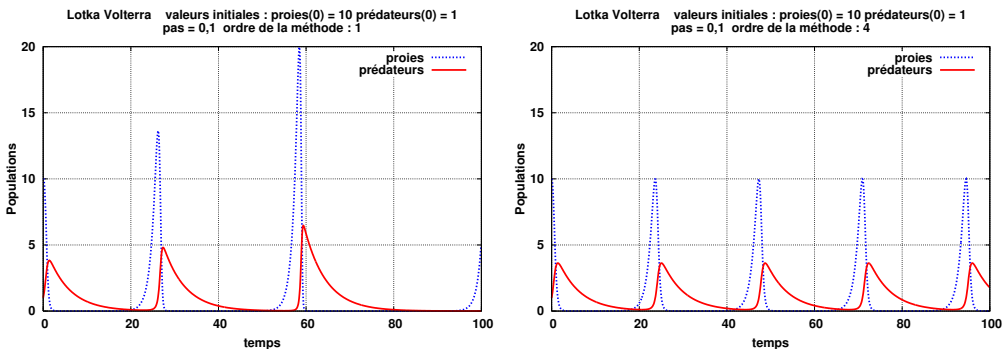
$$k_1 = k_2 = 1, a_1 = 1, a_2 = 0,2$$



Méthode d'Euler : non périodique — Runge Kutta d'ordre 4 : cycle correct

Résolution numérique de Lotka-Volterra : $k_1 = k_2 = 1, a_1 = 1, a_2 = 0,2, h = 0,1$

Échelles **linéaires**

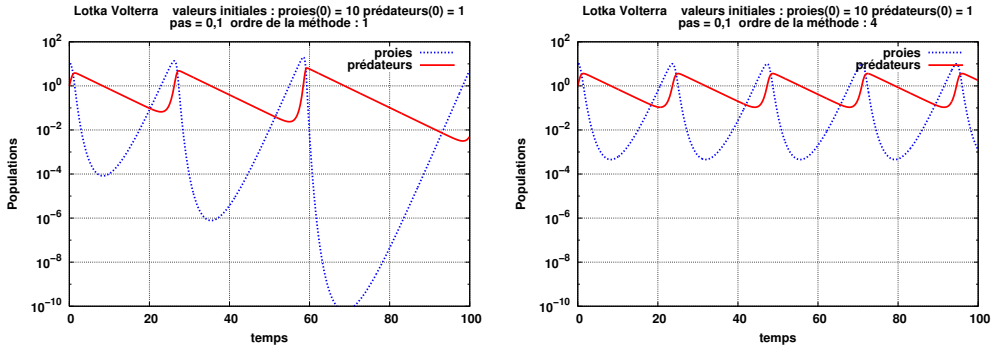


Méthode d'Euler progressive : **Les solutions divergent**

Méthode de Runge Kutta d'ordre 4 : **Cycle stable**

Résolution numérique de Lotka-Volterra : $k_1 = k_2 = 1, a_1 = 1$ **et** $a_2 = 0,2$.

Échelle **log** en ordonnée



Méthode d'Euler progressive : **divergente** — Méthode de Runge Kutta d'ordre 4 : **stable**

Pente avec peu de proies : $\frac{\mathrm{d}\ln y_2}{\mathrm{d}t} \approx -1/\tau_2$ d'où facteur $e^{-4} \approx 1/54$ sur une durée de $20 = 4\tau_2$.

Pente avec peu de prédateurs : $\frac{\mathrm{d}\ln y_1}{\mathrm{d}t} \approx 1/\tau_1$ d'où facteur 100 sur durée de $4,6 = 4,6\tau_1$

7 Équations différentielles d'ordre supérieur

$$\frac{\mathrm{d}^n y}{\mathrm{d}t^n} = f\left(t,\ y,\ \frac{\mathrm{d}y}{\mathrm{d}t},\ \ldots,\ \frac{\mathrm{d}^{n-1}y}{\mathrm{d}t^{n-1}}\right)$$

Une EDO scalaire d'ordre n se ramène à un système de n équations différentielles du premier ordre couplées en posant :

$$\begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{pmatrix} = \begin{pmatrix} y \\ y' \\ \ldots \\ y^{(n-1)} \end{pmatrix} \Longrightarrow \begin{pmatrix} y_1' \\ y_2' \\ \ldots \\ y_n' \end{pmatrix} = \begin{pmatrix} y_2 \\ y_3 \\ \ldots \\ f\left(t, y_1, y_2, \ldots, y_n\right) \end{pmatrix}$$

7.1 Exemple d'EDO linéaire d'ordre 2 avec forçage

Système linéaire du second ordre avec excitation $h(t)$

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} = a\frac{\mathrm{d}y}{\mathrm{d}t} + by + h(t) \qquad (19)$$

Poser

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix} \Rightarrow \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} y_2 \\ ay_2 + by_1 + h(t) \end{pmatrix}$$

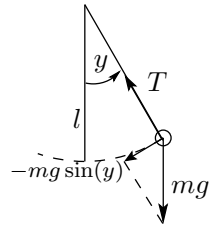
Condition initiale vectorielle : position $y(t_0)$ et vitesse $y'(t_0)$

Remarque Système différentiel d'ordre p de dimension n
$\Rightarrow$ système différentiel couplé du premier ordre à np dimensions.

7.2 Exemple d'EDO d'ordre 2 : le pendule

Pendule avec tige rigide de longueur l et masse ponctuelle m.
Pendule non linéaire (y = position angulaire)

$$\boxed{\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} = -k^2 \sin(y)} \quad \text{où} \quad k^2 = g/l \qquad (20)$$



Pendule linéarisé (cas des petites amplitudes) : $\sin(y) \approx y$

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} = -k^2 y \qquad (21)$$

l'équation linéarisée admet une solution analytique en $A\cos(kt) + B\sin(kt)$.

Pour les amplitudes assez faibles, une approximation s'appuyant sur un développement au troisième ordre du sinus donne la période T en fonction de celle $T_0 = 2\pi/k$ du cas linéaire :

$$T = T_0\left(1 + y_{\text{max}}^2/16\right) \quad \text{(formule de Borda)} \qquad (22)$$

Exprimer cette EDO du second ordre sous la forme d'un système différentiel couplé de dimension 2 mais du premier ordre.

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix} \Longrightarrow \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} y_2 \\ -k^2 \sin(y_1) \end{pmatrix}$$

On peut alors résoudre numériquement le système non-linéaire, avec pour **vecteur** des conditions initiales :

$$\begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} y(0) \\ \dfrac{\mathrm{d}y}{\mathrm{d}t}(0) = a \end{pmatrix} = \begin{pmatrix} \text{position angulaire} \\ \text{vitesse angulaire} \end{pmatrix}$$

Énergie mécanique conservée , soit, après division par ml^2 :

$$\frac{1}{2}\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2 + k^2(1-\cos y) = \text{constante}$$

Invariant qui permet de **diagnostiquer la qualité** de l'intégration numérique.

Cas où $y(0) = 0$ (départ en position d'équilibre stable)

$$\frac{1}{2}\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2 + k^2(1-\cos y) = \frac{1}{2}\left(\frac{\mathrm{d}y}{\mathrm{d}t}(0)\right)^2$$

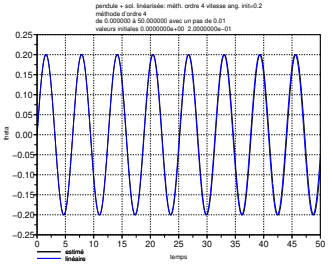
Vitesse angulaire minimale pour $y = \pi$ (position d'équilibre instable si atteinte).

Si $a = \dfrac{\mathrm{d}y}{\mathrm{d}t}(0) > 2k$ (seuil) $\Rightarrow$ la vitesse angulaire ne s'annule plus : on passe en régime apériodique, où le pendule tourne toujours dans le même sens.

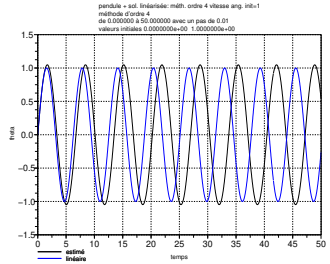
Étude numérique de la transition périodique–apériodique attendue pour $a = 2k$ avec les méthodes de Runge Kutta d'ordre 4 et d'Euler

À $k = 1$ fixé, étude pour les valeurs de a : $0{,}2$; 1 ; $1{,}98$ et $2{,}02$.

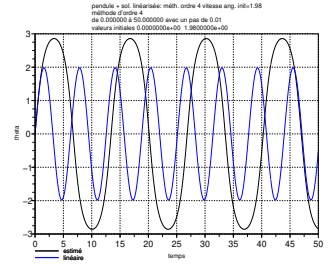
Comparaisons non-linéaire (Runge-Kutta 4)–analytique linéarisé : $y(t)$



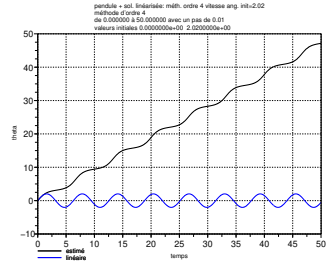
$a = 0.2 \ll 1$ linéarisable



$a = 1$ périodique non sinusoïdal

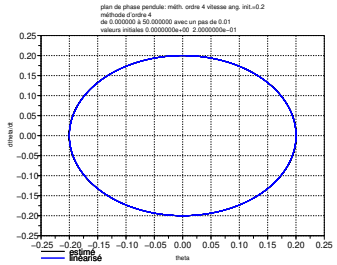


$a = 1.98$ périodique non sinusoïdal

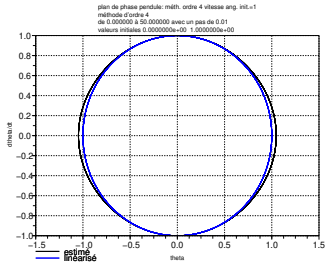


$a = 2.02$ apériodique

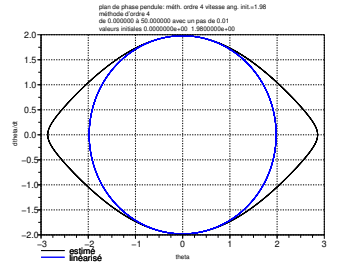
Comparaisons non-linéaire (RK 4)–analytique linéarisé : plan de phase $y'(y)$



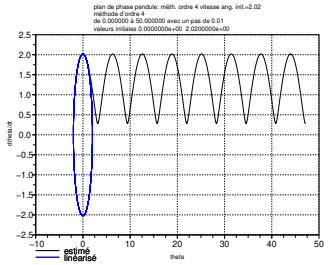
$a = 0.2 \ll 1$ linéarisable



$a = 1$ périodique non sinusoïdal

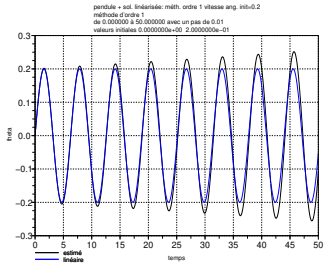


$a = 1.98$ périodique non sinusoïdal

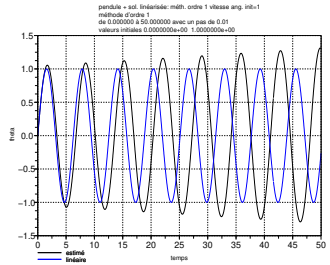


$a = 2.02$ apériodique

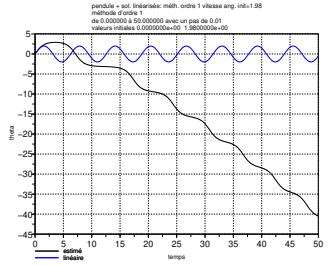
Comparaisons non-linéaire (Euler)–analytique linéarisé $y(t)$



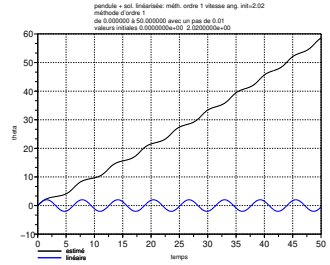
$a = 0.2$



$a = 1$

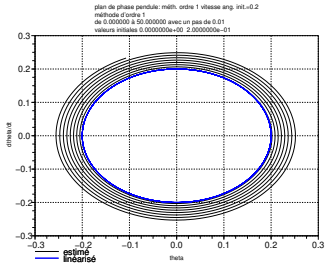


$a = 1.98$ apériodique selon Euler !

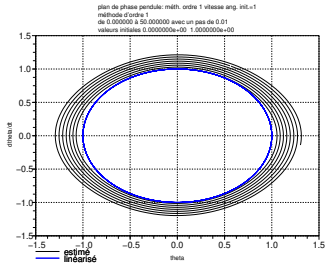


$a = 2.02$

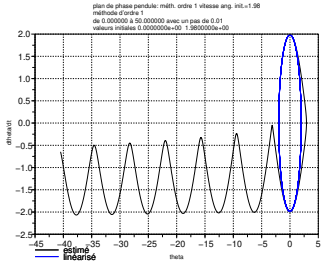
Comparaisons non-linéaire (Euler)–analytique linéarisé : plan de phase $y'(y)$



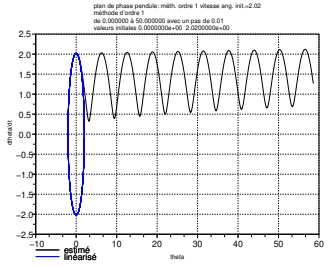
$a = 0.2$ divergence



$a = 1$ divergence



$a = 1.98$ apériodique selon Euler !



$a = 2.02$

7.3 Stabilité à long terme avec Euler progressive et rétrograde

Pendule linéarisé sans frottement représenté dans l'espace des phases : comportement à long terme d'un système non dissipatif

Même méthode sur les 2 composantes (position et vitesse) $h = 0.025$

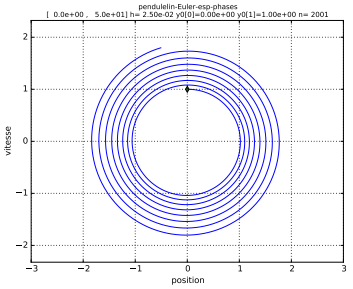


FIGURE 11: Euler progressive : instable, **amplification**

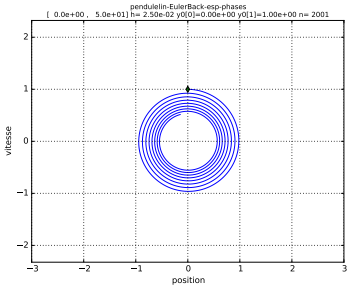


FIGURE 12: Euler rétrograde : stable, **contraction**

Alternance des méthodes progressive et rétrograde entre position et vitesse

⇒ **méthode symplectique** conservant mieux l'énergie

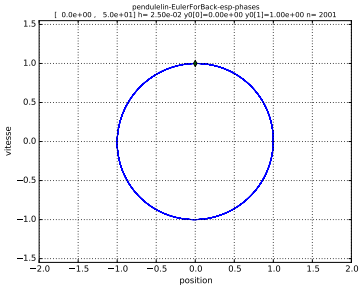


FIGURE 13: Euler mixte
— progressive sur position,
— rétrograde sur vitesse.

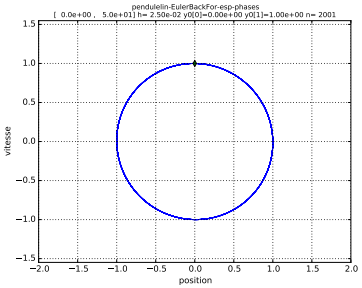


FIGURE 14: Euler mixte
— rétrograde sur position,
— progressive sur vitesse.

Références

AKAI, TERRENCE J., *Applied Numerical Methods for Engineers*, 410 pages (Wiley, 1994), ISBN 0-471-57523-2.

BURDEN, RICHARD L. et J. DOUGLAS FAIRES, *Numerical Analysis*, 875 pages (Brooks/Cole, 2011), neuvième édition, ISBN 0-538-73564-3.

DEMAILLY, J.-P., *Analyse numérique et équations différentielles*, 350 pages (EDP Sciences, 2006), troisième édition, ISBN 978-2-86883-891-9.

GUILPIN, CH., *Manuel de calcul numérique appliqué*, 577 pages (EDP Sciences, 1999), ISBN 2-86883-406-X.

RAPPAZ, JACQUES et MARCO PICASSO, *Introduction à l'analyse numérique*, 268 pages (Presses polytechniques et universitaires romandes, 2010), ISBN 978-2-88074-851-7.