

Méthode d'Euler

Ivan Noyer

Lycée Thiers

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

Un théorème d'existence

Théorème (Théorème de Cauchy-Lipschitz)

Soit U un ouvert de $\mathbb{R}^2$ et $f : U \rightarrow \mathbb{R}$ une fonction de classe $\mathcal{C}^1$ sur U . Pour tout point (x_0, y_0) il existe une unique solution maximale (c.a.d. l'intervalle de définition est le plus grand possible) au problème de Cauchy :

$$\begin{cases} (E) : y' &= f(x, y) \\ y(x_0) &= y_0 \end{cases}$$

Observations

Le théorème de Cauchy-Lipschitz est un th. d'existence. Il ne donne aucune indication sur la méthode. La *méthode d'Euler* donne une approximation numérique de la solution.

Linéarité

f souvent linéaire selon y :

$$\begin{aligned} f(x, \lambda y + z) &= \\ f(x, y) + \lambda f(x, z). \end{aligned}$$

Euler

Figure – Portrait par
Johann Georg Brucker
(1756)



Sa vie

Leonhard Euler, né le 15 avril 1707 à Bâle (Suisse) et mort à 76 ans le 18 septembre 1783 à Saint-Pétersbourg (Empire russe), : mathématicien et physicien suisse membre de l'Académie royale des sciences de Prusse à Berlin.

Euler

Figure – Portrait par
Johann Georg Brucker
(1756)



Son œuvre

Principale découvertes en mathématiques : calcul infinitésimal, théorie des graphes. Introduit terminologie et notation modernes, en particulier pour l'analyse mathématique, comme la notion de fonction mathématique.

Travaux en mécanique, en dynamique des fluides, en optique et en astronomie...

Euler

Figure – Portrait par
Johann Georg Brucker
(1756)



Sa renommée

Un des plus grands et des plus prolifiques mathématicien de tous les temps. « Lisez Euler, lisez Euler, c'est notre maître à tous » (Pierre-Simon de Laplace).

Euler

Figure – Portrait par
Johann Georg Brucker
(1756)



Sa religion

Fervent chrétien, croyant en
l'inerrance biblique (i.e. *prendre la bible au premier degré*). S'opposa
aux athées éminents de son temps.

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

Principe de la méthode d'Euler explicite

Avec $y' = f(x, y)$ et $y(a) = y_0$.

- On approche la fonction solution sur $[a, b]$ par une fonction affine par morceaux.

Principe de la méthode d'Euler explicite

Avec $y' = f(x, y)$ et $y(a) = y_0$.

- On approche la fonction solution sur $[a, b]$ par une fonction affine par morceaux.
- On se fixe un *pas* $h = \frac{b - a}{n}$, $n \in \mathbb{N}^*$.

Principe de la méthode d'Euler explicite

Avec $y' = f(x, y)$ et $y(a) = y_0$.

- On approche la fonction solution sur $[a, b]$ par une fonction affine par morceaux.
 - On se fixe un *pas* $h = \frac{b-a}{n}$, $n \in \mathbb{N}^*$.
 - On pose $t_0 = a$, $t_1 = t_0 + h$, $\dots$ $t_k = t_{k-1} + h$ et $t_n = b$. (subdivision de $[a, b]$).
- Si y est la solution au pb de Cauchy, on pose $y_0 = y(t_0) = y(a)$.

Principe de la méthode d'Euler explicite

Avec $y' = f(x, y)$ et $y(a) = y_0$.

- On approche la fonction solution sur $[a, b]$ par une fonction affine par morceaux.
- On se fixe un *pas* $h = \frac{b-a}{n}$, $n \in \mathbb{N}^*$.
- On pose $t_0 = a$, $t_1 = t_0 + h$, $\dots$ $t_k = t_{k-1} + h$ et $t_n = b$. (subdivision de $[a, b]$).
Si y est la solution au pb de Cauchy, on pose $y_0 = y(t_0) = y(a)$.
- On fait un DL1 en t_k :

$$y(t_{k+1}) - y(t_k) \simeq y'(t_k) \overbrace{(t_{k+1} - t_k)}^h = hf(t_k, y(t_k)).$$

Donc $y(t_{k+1}) \simeq y(t_k) + hf(t_k, y(t_k))$.

On pose par analogie $y_{k+1} = y_k + hf(t_k, y_k)$.

Graphiques

Figure – Euler, premier itéré

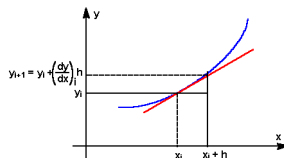
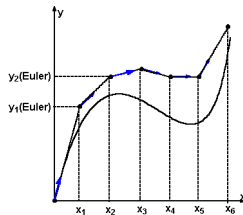


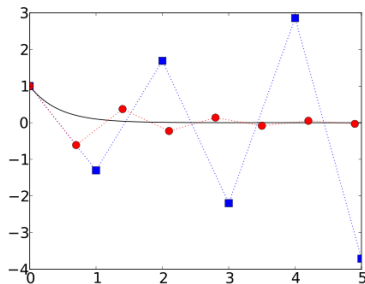
Figure – Approximation



Instabilité numérique

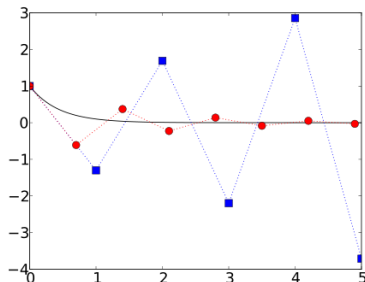
Figure – bleu $h = 1$, rouge $h = 0.7$

- Il faut choisir un pas assez petit.



Instabilité numérique

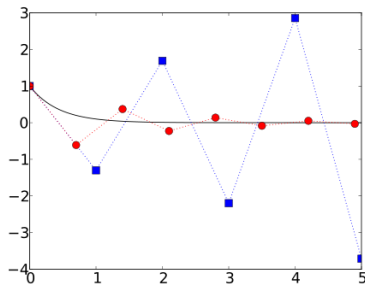
Figure – bleu $h = 1$, rouge $h = 0.7$



- Il faut choisir un pas assez petit.
- Avec $y' = -2.3y$ et $y(0) = 1$. Solution exacte est $y = e^{-2.3t}$. Tend vers 0 en $+\infty$.

Instabilité numérique

Figure – bleu $h = 1$, rouge $h = 0.7$



- Il faut choisir un pas assez petit.
- Avec $y' = -2.3y$ et $y(0) = 1$. Solution exacte est $y = e^{-2.3t}$. Tend vers 0 en $+\infty$.
- Avec un pas de 1 : La solution numérique oscille et l'écart croît : *instabilité*. Avec un pas de 0.7, l'approximation est correcte.

Instabilité numérique

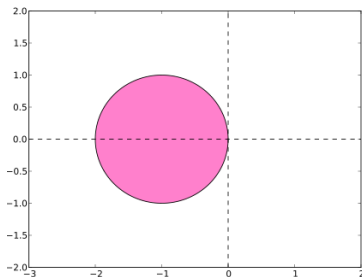
On peut montrer que pour une équation $y' = ky$,

$$\operatorname{Re}(k) < 0$$

et un pas de h , la solution numérique est instable si

$$hk \notin \{z \in \mathbb{C} \mid |z + 1| \leq 1\}.$$

Figure – Disque de stabilité



Un exemple

- PB de Cauchy $y' = y$, $y(0) = 1$. Solution exacte $t \mapsto e^t$.

Figure – Présentation sous forme de tableau

n	y_n	t_n	$f(t_n, y_n)$	h	Δy	y_{n+1}
0	1	0	1	1	1	2
1	2	1	2	1	2	4
2	4	2	4	1	4	8
3	8	3	8	1	8	16

Un exemple

- PB de Cauchy $y' = y$, $y(0) = 1$. Solution exacte $t \mapsto e^t$.
- On pose $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $(a, b) \mapsto b$. Donc $y' = f(t, y)$

Figure – Présentation sous forme de tableau

n	y_n	t_n	$f(t_n, y_n)$	h	Δy	y_{n+1}
0	1	0	1	1	1	2
1	2	1	2	1	2	4
2	4	2	4	1	4	8
3	8	3	8	1	8	16

Un exemple

- PB de Cauchy $y' = y$, $y(0) = 1$. Solution exacte $t \mapsto e^t$.
- On pose $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $(a, b) \mapsto b$. Donc $y' = f(t, y)$
- Avec un pas $h = 1$.

$$\begin{aligned}
 y_1 &= y_0 + hf(t_0, y_0) = 1 + 1 \cdot 1 = 2 \\
 y_2 &= y_1 + hf(t_1, y_1) = 2 + 1 \cdot 2 = 4, \\
 y_3 &= y_2 + hf(t_2, y_2) = 4 + 1 \cdot 4 = 8, \\
 y_4 &= y_3 + hf(t_3, y_3) = 8 + 1 \cdot 8 = 16.
 \end{aligned}$$

Figure – Présentation sous forme de tableau

n	y_n	t_n	$f(t_n, y_n)$	h	Δy	y_{n+1}
0	1	0	1	1	1	2
1	2	1	2	1	2	4
2	4	2	4	1	4	8
3	8	3	8	1	8	16

Un exemple

- PB de Cauchy $y' = y$, $y(0) = 1$. Solution exacte $t \mapsto e^t$.
- On pose $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $(a, b) \mapsto b$. Donc $y' = f(t, y)$
- Avec un pas $h = 1$.

$$\begin{aligned} y_1 &= y_0 + hf(t_0, y_0) = 1 + 1 \cdot 1 = 2 \\ y_2 &= y_1 + hf(t_1, y_1) = 2 + 1 \cdot 2 = 4, \\ y_3 &= y_2 + hf(t_2, y_2) = 4 + 1 \cdot 4 = 8, \\ y_4 &= y_3 + hf(t_3, y_3) = 8 + 1 \cdot 8 = 16. \end{aligned}$$

- Ici, $y_4 = 16$ et la solution véritable : $\exp(4) = e^4 \simeq 54.598$

Figure – Présentation sous forme de tableau

n	y_n	t_n	$f(t_n, y_n)$	h	Δy	y_{n+1}
0	1	0	1	1	1	2
1	2	1	2	1	2	4
2	4	2	4	1	4	8
3	8	3	8	1	8	16

Deux pas différents

Rouge : $t \mapsto \exp(t)$, bleu : Euler ; vert : midpoint.

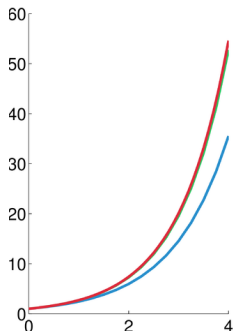
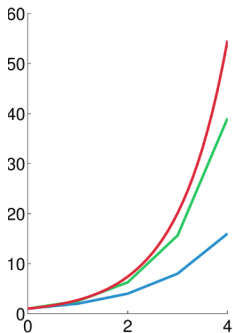


Figure – Gauche : pas de 1 ; Droite : pas de 0.25

Précision croissante

- Grossièrement, l'erreur est ici proportionnelle au pas. Ce qui est à peu près vrai aussi en général.

Précision croissante

- Grossièrement, l'erreur est ici proportionnelle au pas. Ce qui est à peu près vrai aussi en général.
- On compare la valeur véritable de $y(4) = e^4 \simeq 54.598$ avec l'approximation obtenue par Euler :

Figure – Évolution de l'erreur avec le pas

step size	result of Euler's method	error
1	16	38.598
0.25	35.53	19.07
0.1	45.26	9.34
0.05	49.56	5.04
0.025	51.98	2.62
0.0125	53.26	1.34

Dans le tableau, à part pour la première ligne, les quotients Erreur/Pas sont $\simeq 100$.

Erreur de troncature locale

- C'est l'erreur (différence avec la véritable solution) à une étape t_0 .

Erreur de troncature locale

- C'est l'erreur (différence avec la véritable solution) à une étape t_0 .
- $y_1 = y_0 + hf(t_0, y_0) = y_0 + hy'(t_0)$ par Euler.

Erreur de troncature locale

- C'est l'erreur (différence avec la véritable solution) à une étape t_0 .
- $y_1 = y_0 + hf(t_0, y_0) = y_0 + hy'(t_0)$ par Euler.
- Si y est de classe $\mathcal{C}^2$, par la formule de Taylor-Young :

$$y(t_0 + h) = y(t_0) + hy'(t_0) + O(h^2)$$

Erreur de troncature locale

- C'est l'erreur (différence avec la véritable solution) à une étape t_0 .
- $y_1 = y_0 + hf(t_0, y_0) = y_0 + hy'(t_0)$ par Euler.
- Si y est de classe $\mathcal{C}^2$, par la formule de Taylor-Young :

$$y(t_0 + h) = y(t_0) + hy'(t_0) + O(h^2)$$

- Par soustraction :

$$y(t_0 + h) - y_1 = O(h^2)$$

Erreur de troncature locale

- C'est l'erreur (différence avec la véritable solution) à une étape t_0 .
- $y_1 = y_0 + hf(t_0, y_0) = y_0 + hy'(t_0)$ par Euler.
- Si y est de classe $\mathcal{C}^2$, par la formule de Taylor-Young :

$$y(t_0 + h) = y(t_0) + hy'(t_0) + O(h^2)$$

- Par soustraction :

$$y(t_0 + h) - y_1 = O(h^2)$$

- L'erreur de troncature locale est en $O(h^2)$, c'est à dire majorée par une constante fois h^2 .

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.
- L'erreur locale est en gros proportionnelle à h^2 .

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.
- L'erreur locale est en gros proportionnelle à h^2 .
- Donc l'erreur de troncature globale est en gros proportionnelle à $h^2 \frac{1}{h} = h$.

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.
- L'erreur locale est en gros proportionnelle à h^2 .
- Donc l'erreur de troncature globale est en gros proportionnelle à $h^2 \frac{1}{h} = h$.
- C'est la raison pour laquelle on dit que la méthode d'Euler est du premier ordre.

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.
- L'erreur locale est en gros proportionnelle à h^2 .
- Donc l'erreur de troncature globale est en gros proportionnelle à $h^2 \frac{1}{h} = h$.
- C'est la raison pour laquelle on dit que la méthode d'Euler est du premier ordre.
- Plus la méthode est performante, plus l'ordre est élevé.

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.
- L'erreur locale est en gros proportionnelle à h^2 .
- Donc l'erreur de troncature globale est en gros proportionnelle à $h^2 \frac{1}{h} = h$.
- C'est la raison pour laquelle on dit que la méthode d'Euler est du premier ordre.
- Plus la méthode est performante, plus l'ordre est élevé.
 - ❶ Méthode de Heun : ordre 2

Erreur de troncature globale

- C'est le cumul des erreurs de troncature locale, c.a.d à chaque étape. (appelée aussi erreur de consistance). La somme des valeurs absolues des erreurs locales.
- Nombre de pas effectués au moment t : $\frac{t - t_0}{h}$, proportionnel à $\frac{1}{h}$.
- L'erreur locale est en gros proportionnelle à h^2 .
- Donc l'erreur de troncature globale est en gros proportionnelle à $h^2 \frac{1}{h} = h$.
- C'est la raison pour laquelle on dit que la méthode d'Euler est du premier ordre.
- Plus la méthode est performante, plus l'ordre est élevé.
 - ➊ Méthode de Heun : ordre 2
 - ➋ Méthode de Runge-Kutta classique : ordre 4.

Erreurs d'arrondis VS erreur de troncature

Calcul informels :

- Notre calcul de l'erreur de troncature, ne tient pas compte de l'erreur d'arrondi.

Erreurs d'arrondis VS erreur de troncature

Calcul informels :

- Notre calcul de l'erreur de troncature, ne tient pas compte de l'erreur d'arrondi.
- À l'étape n , on peut estimer que l'erreur d'arrondi est à peu près de εy_n , où ε est l'epsilon machine (Wikipedia).

Erreurs d'arrondis VS erreur de troncature

Calcul informels :

- Notre calcul de l'erreur de troncature, ne tient pas compte de l'erreur d'arrondi.
- À l'étape n , on peut estimer que l'erreur d'arrondi est à peu près de εy_n , où ε est l'epsilon machine (Wikipedia).
- Si les erreurs d'arrondis aux différentes étapes sont toutes approximativement égales, l'erreur d'arrondis combinés au bout de N étapes est de l'ordre de $N\varepsilon y_0$.

Puisque le nombre d'étapes est proportionnel à $\frac{1}{h}$, l'arrondi combiné est de l'ordre de $\frac{\varepsilon}{h}$. Or cette quantité tend vers $+\infty$ quand h tend vers 0.

Erreurs d'arrondis VS erreur de troncature

Calcul informels :

- Notre calcul de l'erreur de troncature, ne tient pas compte de l'erreur d'arrondi.
- À l'étape n , on peut estimer que l'erreur d'arrondi est à peu près de εy_n , où ε est l'epsilon machine (Wikipedia).
- Si les erreurs d'arrondis aux différentes étapes sont toutes approximativement égales, l'erreur d'arrondi combinée est en $O(\frac{\varepsilon}{h})$.
- En réalité, les erreurs ne pointent pas toutes dans la même direction, et on peut considérer les erreurs d'arrondis aux différentes étapes comme des variables aléatoires indépendantes.

On peut alors montrer que l'erreur d'arrondis combinés est plutôt en $\frac{\varepsilon}{\sqrt{h}}$, ce qui tend toujours vers $+\infty$ mais moins vite que précédemment.

Erreurs d'arrondis VS erreur de troncature

Calcul informels :

- Notre calcul de l'erreur de troncature, ne tient pas compte de l'erreur d'arrondi.
- À l'étape n , on peut estimer que l'erreur d'arrondi est à peu près de εy_n , où ε est l'epsilon machine (Wikipedia).
- Si les erreurs d'arrondis aux différentes étapes sont toutes approximativement égales, l'erreur d'arrondi combinée est en $O(\frac{\varepsilon}{h})$.
- En réalité, les erreurs ne pointent pas toutes dans la même direction, l'erreur d'arrondis combinés est plutôt en $\frac{\varepsilon}{\sqrt{h}}$, ce qui tend toujours vers $+\infty$ mais moins vite que précédemment.
- Pour de toutes petites valeur du pas h , l'erreur de troncature globale (la seule erreur prise en compte par le prof de maths) peut-être petite mais les effets de l'arrondi peuvent être importants : il faut alors utiliser des méthodes comme la *sommation compensée*.

Algorithme

```
1 def euler(f, a, b, y0, h):
2     """ solutions de  $y'=f(t,y)$  sur  $[a,b]$ 
3     avec  $y(a)=y0$  et pas de  $h$  """
4     #ordonnée et abscisse initiales :
5     y,t = y0, a
6     #liste des valeurs  $y_i$ , des temps  $t_i$  :
7     les_y, les_t = [y0],[a]
8     while t+h <= b:
9         y += h * f(t, y)
10        les_y.append(y)
11        t += h
12        les_t.append(t)
13    return les_t, les_y
```

Algorithme

Retour obtenu

```
>>> euler(lambda t,y :y,0,1,1,1./3)[1]  
[1, 1.3333333333333333,  
1.7777777777777777,  
2.3703703703703702]
```

Autres retours

Valeur approchée en 1 (approximation de e) pour des pas en 10^{-k}

```
>>> [euler(lambda t,y :y,0,1,1,1./10**k)[1][-1]\  
      for k in range(1,5)]  
[2.5937424601, 2.6780334944767583,  
2.7142097225133828, 2.7181459268252266]
```

- 1 Introduction
- 2 Principe
- 3 Intégration**
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

Principe

- Rappel : la méthode d'Euler résoud numériquement des équations de la forme $y' = f(x, y)$ où $f : U \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ est continue.

Principe

- Rappel : la méthode d'Euler résoud numériquement des équations de la forme $y' = f(x, y)$ où $f : U \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ est continue.
- Si $f(x, y)$ ne dépend pas de y , on peut écrire $g(x) = f(x, y)$ et on a donc $y' = g(x)$.

Principe

- Rappel : la méthode d'Euler résoud numériquement des équations de la forme $y' = f(x, y)$ où $f : U \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ est continue.
- Si $f(x, y)$ ne dépend pas de y , on peut écrire $g(x) = f(x, y)$ et on a donc $y' = g(x)$.
- Résoudre cette équation différentielle avec la condition $y(a) = y_0$ revient alors à donner la primitive de g qui vaut y_0 en a .

Méthode des rectangles

Equation $y' = g(x)$ avec $g : [a, b] \rightarrow \mathbb{R}$ continue.



$$\begin{aligned}
 y_3 &= y_2 + hg(t_2) \\
 &= y_2 + hg(a + 2h) \\
 &= hg(a + 2h) + y_1 + hg(a + h) \\
 &= hg(a + 2h) + hg(a + h) + hg(a) + y_0
 \end{aligned}$$

On en déduit $y_n - y_0 = h \sum_{k=0}^{n-1} g(a + kh)$

Méthode des rectangles

Equation $y' = g(x)$ avec $g : [a, b] \rightarrow \mathbb{R}$ continue.



$$\begin{aligned}
 y_3 &= y_2 + hg(t_2) \\
 &= y_2 + hg(a + 2h) \\
 &= hg(a + 2h) + y_1 + hg(a + h) \\
 &= hg(a + 2h) + hg(a + h) + hg(a) + y_0
 \end{aligned}$$

On en déduit $y_n - y_0 = h \sum_{k=0}^{n-1} g(a + kh)$

- On reconnaît la méthode des rectangles bords gauche pour le calcul de :

$$\int_a^b g(x) dx.$$

Méthode des rectangles

Equation $y' = g(x)$ avec $g : [a, b] \rightarrow \mathbb{R}$ continue.



$$\begin{aligned}
 y_3 &= y_2 + hg(t_2) \\
 &= y_2 + hg(a + 2h) \\
 &= hg(a + 2h) + y_1 + hg(a + h) \\
 &= hg(a + 2h) + hg(a + h) + hg(a) + y_0
 \end{aligned}$$

On en déduit $y_n - y_0 = h \sum_{k=0}^{n-1} g(a + kh)$

- On reconnaît la méthode des rectangles bords gauche pour le calcul de :

$$\int_a^b g(x) dx.$$

- Ainsi, la méthode d'Euler est un prolongement de celle des rectangles aux équations différentielles.

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n**
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

Objectif

Adapter le schéma d'Euler explicite pour résoudre des équations d'ordre supérieur ou égal à 2, c'est à dire de la forme :

$$y^{(n)} = f(t, y, y', \dots, y^{(n-1)})$$

Exemple

- Soit l'équation $y'' = -k \sin(t^2 y)$ d'ordre 2 avec CC $y(0) = 1; y'(0) = 0.5$.

Exemple

- Soit l'équation $y'' = -k \sin(t^2 y)$ d'ordre 2 avec CC $y(0) = 1; y'(0) = 0.5$.
- Avec $Y = (y, y')$, on a $Y' = (y', y'') = (y', -k \sin(t^2 y)) = F(t, Y)$ où $F(t, (a, b)) = (b, -k \sin(t^2 a))$.

Exemple

- Soit l'équation $y'' = -k \sin(t^2 y)$ d'ordre 2 avec CC $y(0) = 1; y'(0) = 0.5$.
- Avec $Y = (y, y')$, on a $Y' = (y', y'') = (y', -k \sin(t^2 y)) = F(t, Y)$ où $F(t, (a, b)) = (b, -k \sin(t^2 a))$.
- L'ordre de l'équation a baissé d'un cran, mais F est à valeurs *vectorielles* dans $\mathbb{R}^2$ et plus dans $\mathbb{R}$.
On obtient une équation vectorielle d'ordre 1 : $Y' = F(t, Y)$ qu'on sait résoudre par Euler. La CC est $Y_0 = (1, 0.5)$

Principe

On ramène une équation d'ordre n : $y^{(n)} = f(t, y', \dots, y^{(n-1)})$ à une équation d'ordre 1 mais à valeurs dans $\mathbb{R}^n$.

On part de :

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n-1)} \\ y^{(n)} \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n-1)} \\ f(t, y', \dots, y^{(n-1)}) \end{pmatrix}$$

Et on résoud $Y' = F(t, Y)$ où

$$F : \begin{pmatrix} t \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} x_2 \\ \vdots \\ x_n \\ f(t, x_1, \dots, x_{n-1}) \end{pmatrix}$$

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n**
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite

Questions

- On veut adapter le code pour résoudre des équations vectorielles.

Questions

- On veut adapter le code pour résoudre des équations vectorielles.
- En quoi consistent exactement les changements ?

Questions

- On veut adapter le code pour résoudre des équations vectorielles.
- En quoi consistent exactement les changements ?
- Rien ne change, on peut garder quasiment la même fonction.

Questions

- On veut adapter le code pour résoudre des équations vectorielles.
- En quoi consistent exactement les changements ?
- Rien ne change, on peut garder quasiment la même fonction.
- MAIS le type de données doit être adapté pour implémenter l'addition des matrices (donc coordonnée par coordonnée).

Questions

- On veut adapter le code pour résoudre des équations vectorielles.
- En quoi consistent exactement les changements ?
- Rien ne change, on peut garder quasiment la même fonction.
- MAIS le type de données doit être adapté pour implémenter l'addition des matrices (donc coordonnée par coordonnée).
- DE PLUS, la ligne 9 du code **$\mathbf{y} += \mathbf{h} * \mathbf{f}(\mathbf{t}, \mathbf{y})$** doit être modifiée car elle est bien adaptée au cas où **$\mathbf{y}$** désigne un nombre mais pas au cas où **$\mathbf{y}$** désigne un tableau.

Explication des modifications

Dans la ligne 9

y est un nombre donc immutable. Alors $y+=h*F(t,y)$ produit une nouvelle variable y (les **y** à droite et à gauche de l'opérateur d'affectation sont différents).

Quand on l'ajoute à `les_y`, on obtient un tableau dont chaque case a une valeur différente des autres (les valeurs des **y** successifs).

Explication des modifications

Si, dans la méthode vectorielle, on garde $y += h * F(t, y)$

comme y est de type mutable `np.array`, on n'a pas changé l'adresse de la variable, juste son contenu.

Quand on ajoute **y** à `les_y`, on obtient le tableau **`les_y`** qui a grossi mais dont toutes les cases sont occupées par le même y , donc ont la même valeur.

Explication des modifications

C'est pourquoi, il faut écrire $y = y + h * F(t, y)$ ligne 9

(le y de gauche et celui de droite désignent deux objets distincts). Quand on ajoute ce nouvel y au tableau, les cases du tableau contiennent des objets distincts : toutes les valeurs prises successivement par y .

C'est ce que nous voulons et c'est la seule modification à faire dans le corps de la fonction **euler** !

En revanche, le paramètre **y0** sera, lors de l'appel de la fonction, un tableau numpy et non plus un nombre.

Euler avec vectorialisation

```
1 def euler(f, a, b, y0, h):
2     """ solutions de  $y'=f(t,y)$  sur  $[a,b]$ 
3     avec  $y(a)=y0$  : un np.array et un pas de  $h$  """
4     #ordonnée et abscisse initiales :
5     y,t = y0, a
6     #liste des valeurs  $y_i$ , des temps  $t_i$  :
7     les_y, les_t = [y0],[a]
8     while t+h <= b:
9         y = y+ h * f(t, y) #seule modification
10        les_y.append(y)
11        t += h
12        les_t.append(t)
13    return les_t, les_y
```

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique**
- 6 Le schéma d'Euler implicite

Retour de la fonction euler

On teste le programme sur l'équation $y'' + y = 0$, avec les conditions initiales $(y(0), y'(0)) = (0, 1)$. Mathématiquement, la solution est connue : c'est sinus.

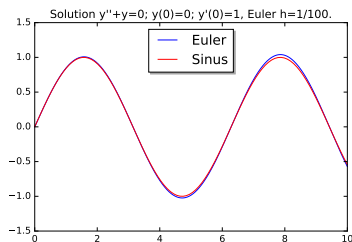
```

1 #resoudre y''+y=0 avec (y(0)=0 et y'(0)=1)
2 def F(t,X):
3     return np.array([X[1], -X[0]])
4
5 t, theta = euler(F, 0, 10, np.array([0,1]),\
6                 10**(-2))

```


Retour de la fonction euler

Figure – Comparaison des graphes du sinus et de la fonction retournée par Euler



On voit que pour un pas $h = \frac{1}{100}$, les approximations conduisent à s'écarter de la véritable solution, puisqu'on quitte l'intervalle $[-1, 1]$.

Fonctions de bibliothèques

odeint (scipy) et **dsolve** (sympy)

```
1 from scipy.integrate import odeint#calc. num.  
2 t = np.linspace(0,1,5)# 5 valeurs entre 0 et 1  
3 odeint(lambda y,t :y,1,t)# résoudre  $y'=y$  ;  $y_0=1$ 
```

```
array([[ 1.          ],  
       [ 1.28402541 ],  
       [ 1.64872127 ],  
       [ 2.11700009 ],  
       [ 2.7182819  ]])
```

Fonctions de bibliothèques

odeint (scipy) et **dsolve** (sympy)

```

1 from sympy import dsolve # solveur d'EDL
2 from sympy import symbols # var. symboliques
3 from sympy import Function # fonctions symb.
4 from sympy import Derivative# dérivées
5 x = symbols('x')#variable symbolique x
6 y = Function('y')(x)# expression symbolique y(x)
7 y__ = Derivative(y,x,x)## expression y''(x)
8 dsolve(y__+y)#résolution symbolique de y''+y=0

```

$$\text{Eq}(y(x), C1*\sin(x) + C2*\cos(x))$$

ordre supérieur

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.integrate import odeint
4
5 # Définir le système
6 def system(Y, t):
7     u, v = Y #  $Y[0]=u=y$ ,  $Y[1]=v=y'$ 
8     du_dt = v
9     dv_dt = -u
10    return [du_dt, dv_dt]
11
12 # Conditions initiales
13 Y0 = [0, 1] #  $y(0)=0$ ,  $y'(0)=1$ 
14
15 # Discrétisation du temps
16 t = np.linspace(0, 10, 1000)
```

ordre supérieur

```
1 # Résolution avec odeint
2 sol = odeint(system, Y0, t) # tableau numpy à 2 lignes
3
4 # Extraire les solutions
5 y = sol[:, 0] # y(t)
6 y_prime = sol[:, 1] # y'(t)
7
8 # Affichage
9 plt.figure(figsize=(8,4))
10 plt.plot(t, y, label="y(t)")
11 plt.plot(t, y_prime, label="y'(t)")
12 plt.xlabel("t")
13 plt.ylabel("Valeur")
14 plt.title("Solution de y'' + y = 0")
15 plt.legend()
16 plt.grid(True)
17 plt.show()
```

- 1 Introduction
- 2 Principe
- 3 Intégration
- 4 Équations différentielles d'ordre n
 - Présentation
 - Adaptation du code
- 5 La pratique
- 6 Le schéma d'Euler implicite**

Motivation

Instabilité du schéma explicite

Des problèmes de stabilité numérique peuvent se poser avec le schéma d'Euler explicite : la solution trouvée oscille autour de la véritable solution avec un écart non majoré. Il faut donc prendre un pas très petit. Mais le nombre d'étape de calcul est alors grand et les erreurs d'arrondis aussi.

Motivation

Plus grande stabilité du schéma implicite

On préfère alors utiliser la *méthode implicite*. Plus difficile à mettre en œuvre, elle nécessite une résolution d'équation à chaque étape.

Mais le risque d'instabilité est plus faible. Et donc le pas n'a pas besoin d'être aussi petit que dans la méthode explicite.

Motivation

Quelle méthode choisir ?

Le choix de la méthode explicite/implicite dépend du type de problème étudié, suivant qu'il conduit facilement ou non à des problèmes d'instabilité. Une étude préalable du problème est donc à réaliser.

Schéma d'Euler implicite

$$y' = f(t, y(t))$$

DL1

Pour une fonction y de classe $\mathcal{C}^1$, on a par DL1

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i) \underbrace{y'(t_i)}_{=f(t_i, y(t_i))} + \underbrace{o(h)}_{=o(t_{i+1}-t_i)} .$$

Schéma d'Euler implicite

$$y' = f(t, y(t))$$

Explicite

La méthode d'Euler présentée jusqu'ici est dite *explicite* :

$$y_{i+1} = y_i + hf(t_i, y_i)$$

Schéma d'Euler implicite

$$y' = f(t, y(t))$$

Implicite

Mais on a aussi, toujours par DL1 :

$$y(t_i) = y(t_{i+1}) + \underbrace{(t_i - t_{i+1})}_{=-h} \times \underbrace{y'(t_{i+1})}_{=f(t_{i+1}, y(t_{i+1}))} + o(h)$$

Par analogie, on en tire la relation de récurrence :

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) \text{ avec } y_{i+1} \text{ dans les deux membres.}$$

On est alors amené à utiliser des méthodes de résolution numérique d'équations comme la méthode de Newton.