Centrale 2015 - Méthodes d'Euler

Informatique pour tou(te)s - semaines 48 & 49

Guillaume CONNAN

Lycée Clemenceau - MP / MP*

novembre - décembre 2015

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

Soient y une fonction de classe C^2 sur $\mathbb{R}$ et $t_{\min}$ et $t_{\max}$ deux réels tels que $t_{\min} < t_{\max}$. On note I l'intervalle $[t_{\min}, t_{\max}]$. On s'intéresse à une équation différentielle du second ordre de la forme :

$$\forall t \in I \qquad y''(t) = f(y(t)) \tag{II.1}$$

où f est une fonction donnée, continue sur $\mathbb{R}$. De nombreux systèmes physiques peuvent être décrits par une équation de ce type.

On suppose connues les valeurs $y_0 = y(t_{\min})$ et $z_0 = y'(t_{\min})$. On suppose également que le système physique étudié est conservatif. Ce qui entraine l'existence d'une quantité indépendante du temps (énergie, quantité de mouvement, ...), notée E, qui vérifie l'équation (II.2) où $g' = -f$.

$$\forall t \in I \qquad \frac{1}{2}y'(t)^2 + g(y(t)) = E \tag{II.2}$$

Question II.A.1

- $y''(t) = f(y(t))$
- On introduit $z : I \to \mathbb{R},\ z : t \mapsto y'(t)$
- Montrer que l'équation peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$ noté (S).

Question II.A.1

- $y''(t) = f(y(t))$
- On introduit $z : I \to \mathbb{R},\ z : t \mapsto y'(t)$
- Montrer que l'équation peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$ noté (S).

Question II.A.1

- $y''(t) = f(y(t))$
- On introduit $z : I \to \mathbb{R},\ z : t \mapsto y'(t)$
- Montrer que l'équation peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$ noté (S).

Question II.A.1

- $y''(t) = f(y(t))$
- On introduit $z : I \to \mathbb{R},\ z : t \mapsto y'(t)$
- Montrer que l'équation peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$ noté (S).

$$\begin{cases} y'(t) = z(t) \\ z'(t) = f\big(y(t)\big) \end{cases}$$

Question II.A.2

$n > 1,\ J_n = [\![0, n-1]\!],\ t = \frac{t_{\max} - t_{\min}}{n-1},\ t_i = t_{\min} + ih$
Montrer que pour tout entier $i \in [\![0, n-2]\!]$

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} z(t)\,\mathrm{d}t \qquad z(t_{i+1}) = z(t_i) + \int_{t_i}^{t_{i+1}} f\big(y(t)\big)\,\mathrm{d}t$$

$$\int_{t_i}^{t_{i+1}} z(t)\,\mathrm{d}t = \int_{t_i}^{t_{i+1}} y'(t)\,\mathrm{d}t = y(t_{i+1}) - y(t_i)$$

$$\int_{t_i}^{t_{i+1}} f\big(y(t)\big)\,\mathrm{d}t = \int_{t_i}^{t_{i+1}} z'(t)\,\mathrm{d}t = z(t_{i+1}) - z(t_i)$$

$$\int_{t_i}^{t_{i+1}} z(t)\,\mathrm{d}t = \int_{t_i}^{t_{i+1}} y'(t)\,\mathrm{d}t = y(t_{i+1}) - y(t_i)$$

$$\int_{t_i}^{t_{i+1}} f\big(y(t)\big)\,\mathrm{d}t = \int_{t_i}^{t_{i+1}} z'(t)\,\mathrm{d}t = z(t_{i+1}) - z(t_i)$$

Sommaire

Étude de schémas numériques
Schéma d'Euler explicite
Schéma de Verlet
Approximation polynomiale
Inter(extra)polation
Lagrange, Waring, Gauß, Newton
Newton / Différences divisées
Cherchez l'erreur
Choix des points
Instabilité
Méthodes de Newton-Cotes
Newton-Cotes d'ordre 0
Newton-Cotes d'ordre 1
Newton-Cotes d'ordre 2
L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)
Algo ≠ Programme : approximation de π
Méthode des rectangles
Méthode des trapèzes
Méthode de Simpson
Bye bye π ?

Schéma d'Euler explicite

Dans le schéma d'Euler explicite, chaque terme sous le signe intégrale est remplacé par sa valeur prise en la borne inférieure.

Question II.B.1

Dans ce schéma, montrer que les équations précédentes permettent de définir deux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$ où y_i et z_i sont des valeurs approchées de $y(t_i)$ et $z(t_i)$.
Donner les relations de récurrence permettant de déterminer les valeurs de y_i et z_i connaissant y_0 et z_0.

Schéma d'Euler explicite

Dans le schéma d'Euler explicite, chaque terme sous le signe intégrale est remplacé par sa valeur prise en la borne inférieure.

Question II.B.1

Dans ce schéma, montrer que les équations précédentes permettent de définir deux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$ où y_i et z_i sont des valeurs approchées de $y(t_i)$ et $z(t_i)$.
Donner les relations de récurrence permettant de déterminer les valeurs de y_i et z_i connaissant y_0 et z_0.

$$\int_{t_i}^{t_{i+1}} z(t_i)\,\mathrm{d}t = z_i h \qquad \int_{t_i}^{t_{i+1}} f\big(y(t_i)\big)\,\mathrm{d}t = f(y_i)h$$

$$y_{i+1} = y_i + z_i h \qquad z_{i+1} = z_i + f(y_i)h$$

$$\int_{t_i}^{t_{i+1}} z(t_i)\,\mathrm{d}t = z_i h \qquad \int_{t_i}^{t_{i+1}} f\big(y(t_i)\big)\,\mathrm{d}t = f(y_i)h$$

$$y_{i+1} = y_i + z_i h \qquad z_{i+1} = z_i + f(y_i)h$$

EULER EXPLICITE = MÉTHODE DES RECTANGLES

Question II.B.2

Écrire une fonction `euler` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$.
Vous justifierez le choix des paramètres transmis à la fonction.

```
def euler(f, y0, z0, h, n) :
    y, z = y0, z0
    ys, zs = [y], [z]
    for i in range(n - 1) :
        fy = f(y)
        y, z  = y + z*h, z + fy*h
        ys.append(y)
        zs.append(z)
    return ys, zs
```

Pour illustrer cette méthode, on considère l'équation différentielle

$$y''(t) = -\omega^2 y(t)$$

dans laquelle ω est un nombre réel.

Question II.B.3.a

Montrer qu'on peut définir une quantité E, indépendante du temps, vérifiant une équation de la forme :

$$\frac{1}{2}y'(t)^2 + g(y(t)) = E$$

Pour illustrer cette méthode, on considère l'équation différentielle

$$y''(t) = -\omega^2 y(t)$$

dans laquelle ω est un nombre réel.

Question II.B.3.a

Montrer qu'on peut définir une quantité E, indépendante du temps, vérifiant une équation de la forme :

$$\frac{1}{2}y'(t)^2 + g\big(y(t)\big) = E$$

$$y''(t) = -\omega^2 y(t)$$

$$\frac{1}{2} 2y'(t)y''(t) = -\frac{1}{2}\omega^2 2y'(t)y(t)$$

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$E = cte \qquad g = x \mapsto \frac{\omega^2}{2}x^2$$

$$y''(t) = -\omega^2 y(t)$$

$$\frac{1}{2} 2y'(t)y''(t) = -\frac{1}{2}\omega^2 2y'(t)y(t)$$

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$E = cte \qquad g = x \mapsto \frac{\omega^2}{2}x^2$$

$$y''(t) = -\omega^2 y(t)$$

$$\frac{1}{2} 2y'(t)y''(t) = -\frac{1}{2}\omega^2 2y'(t)y(t)$$

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$E = cte \qquad g = x \mapsto \frac{\omega^2}{2}x^2$$

$$y''(t) = -\omega^2 y(t)$$

$$\frac{1}{2} 2y'(t)y''(t) = -\frac{1}{2}\omega^2 2y'(t)y(t)$$

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$E = cte \qquad g = x \mapsto \frac{\omega^2}{2}x^2$$

Question II.B.3.b

On note E_i la valeur approchée de E à l'instant t_i , $i \in J_n$, calculée en utilisant les valeurs approchées de $y(t_i)$ et $z(t_i)$ obtenues à la question II.B.1. Montrer que

$$E_{i+1} - E_i = h^2\omega^2 E_i$$

On fait les calculs et ça marche avec $z_{i+1} = z_i - \omega^2 y_i h$

Question II.B.3.b

On note E_i la valeur approchée de E à l'instant t_i , $i \in J_n$, calculée en utilisant les valeurs approchées de $y(t_i)$ et $z(t_i)$ obtenues à la question II.B.1. Montrer que

$$E_{i+1} - E_i = h^2\omega^2 E_i$$

On fait les calculs et ça marche avec $z_{i+1} = z_i - \omega^2 y_i h$

Question II.B.3.b

On note E_i la valeur approchée de E à l'instant t_i , $i \in J_n$, calculée en utilisant les valeurs approchées de $y(t_i)$ et $z(t_i)$ obtenues à la question II.B.1. Montrer que

$$E_{i+1} - E_i = h^2\omega^2 E_i$$

On fait les calculs et ça marche avec $z_{i+1} = z_i - \omega^2 y_i h$

Question II.B.3.c

Qu'aurait donné un schéma numérique qui satisfait à la conservation de E ?

Ben...l'énergie se conserve. On devrait avoir $\Delta E = 0$

Question II.B.3.c

Qu'aurait donné un schéma numérique qui satisfait à la conservation de E ?

Ben...l'énergie se conserve. On devrait avoir $\Delta E = 0$

Question II.B.3.c

Qu'aurait donné un schéma numérique qui satisfait à la conservation de E ?

Ben...l'énergie se conserve. On devrait avoir $\Delta E = 0$

Question II.B.3.d

En portant les valeurs de y_i et z_i sur l'axe des abscisses et l'axe des ordonnées respectivement, quelle serait l'allure du graphe qui respecte la conservation de E ?

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$\frac{1}{2}z^2 + \frac{\omega^2}{2}y^2 = E$$

Ellipse

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$\frac{1}{2}z^2 + \frac{\omega^2}{2}y^2 = E$$

Ellipse

$$\frac{1}{2}y'(t)^2 + \frac{\omega^2}{2}y(t)^2 = E$$

$$\frac{1}{2}z^2 + \frac{\omega^2}{2}y^2 = E$$

Ellipse

Question II.B.3.e

La mise en œuvre de la méthode d'Euler explicite génère le résultat graphique donné figure 1 à gauche. Dans un système d'unités adapté, les calculs ont été menés en prenant $y_0 = 3$, $z_0 = 0$, $t_{\text{min}} = 0$, $t_{\text{max}} = 3$, $\omega = 2\pi$ et $n = 100$

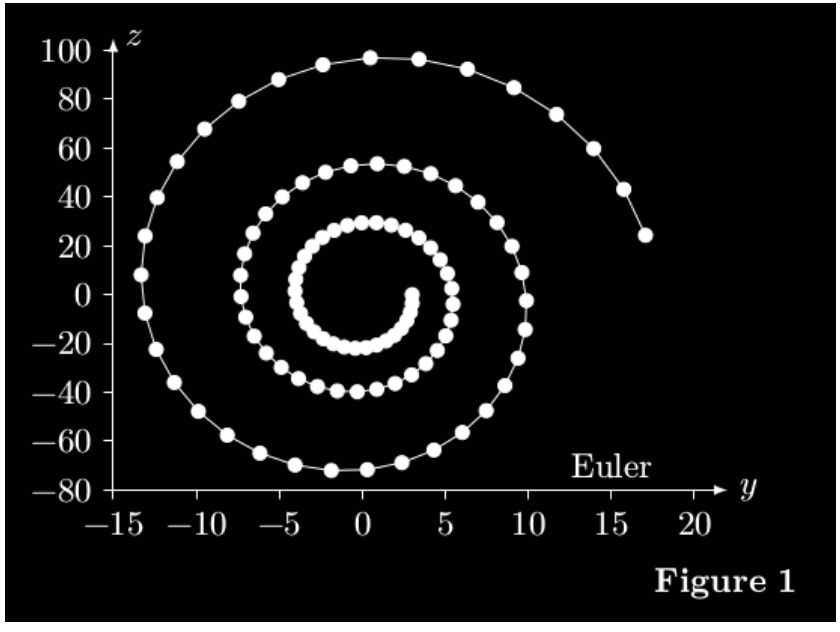


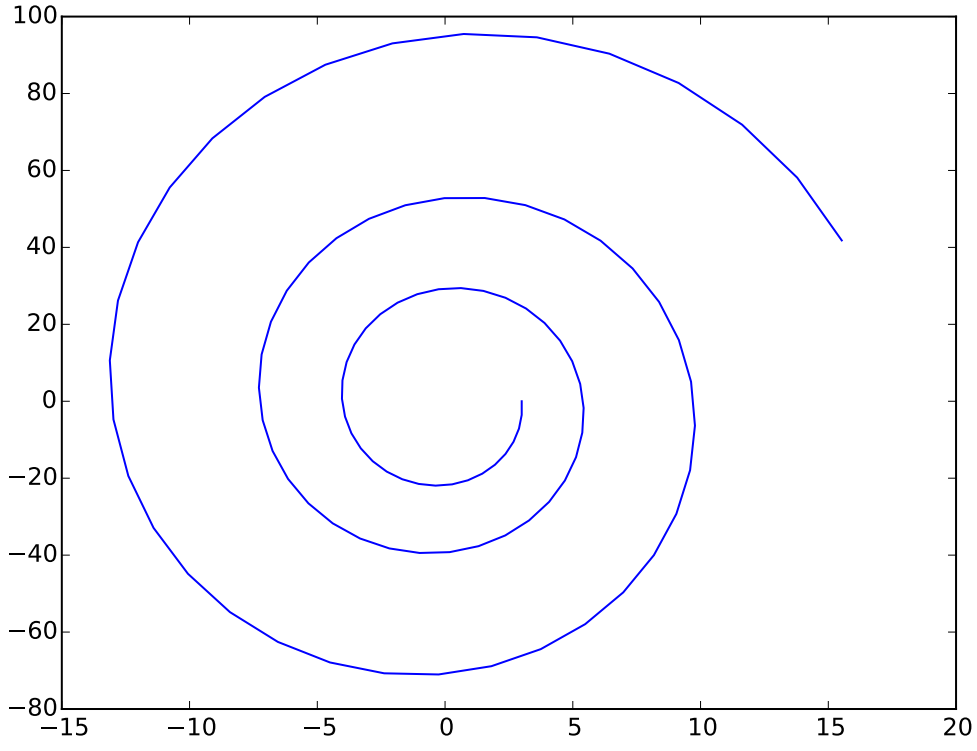
Figure 1

En quoi ce graphe confirme-t-il que le schéma numérique ne conserve pas E ?
Pouvez-vous justifier son allure ?

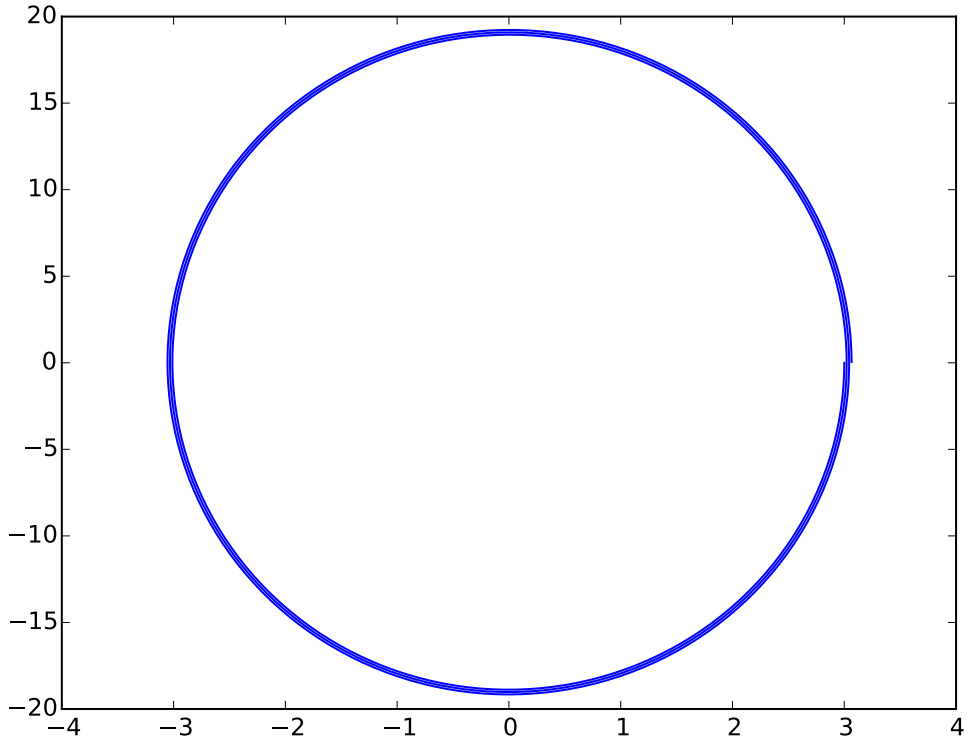
```python
import matplotlib.pyplot as plt

def phase_euler(f, y0, z0, h, n) :
    ys, zs = euler(f, y0, z0, h, n)
    plt.plot(ys, zs)
    plt.savefig("Graph.pdf")
```

```
In [46]: phase_euler(lambda x: -(2*pi)**2*x, 3, 0, 3/100,100)
```



```
In [46]: phase_euler(lambda x: -(2*pi)**2*x, 3, 0, 3/2**13,2**13)
```



Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

Le physicien français Loup Verlet a proposé en 1967 un schéma numérique d'intégration d'une équation de la forme (II.1) dans lequel, en notant $f_i = f(y_i)$ et $f_{i+1} = f(y_{i+1})$, les relations de récurrence s'écrivent

$$y_{i+1} = y_i + hz_i + \frac{h^2}{2} f_i \qquad z_{i+1} = z_i + \frac{h}{2}(f_i + f_{i+1})$$

Question II.C.1

Écrire une fonction `verlet` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i \in J_n}$ et $(z_i)_{i \in J_n}$.

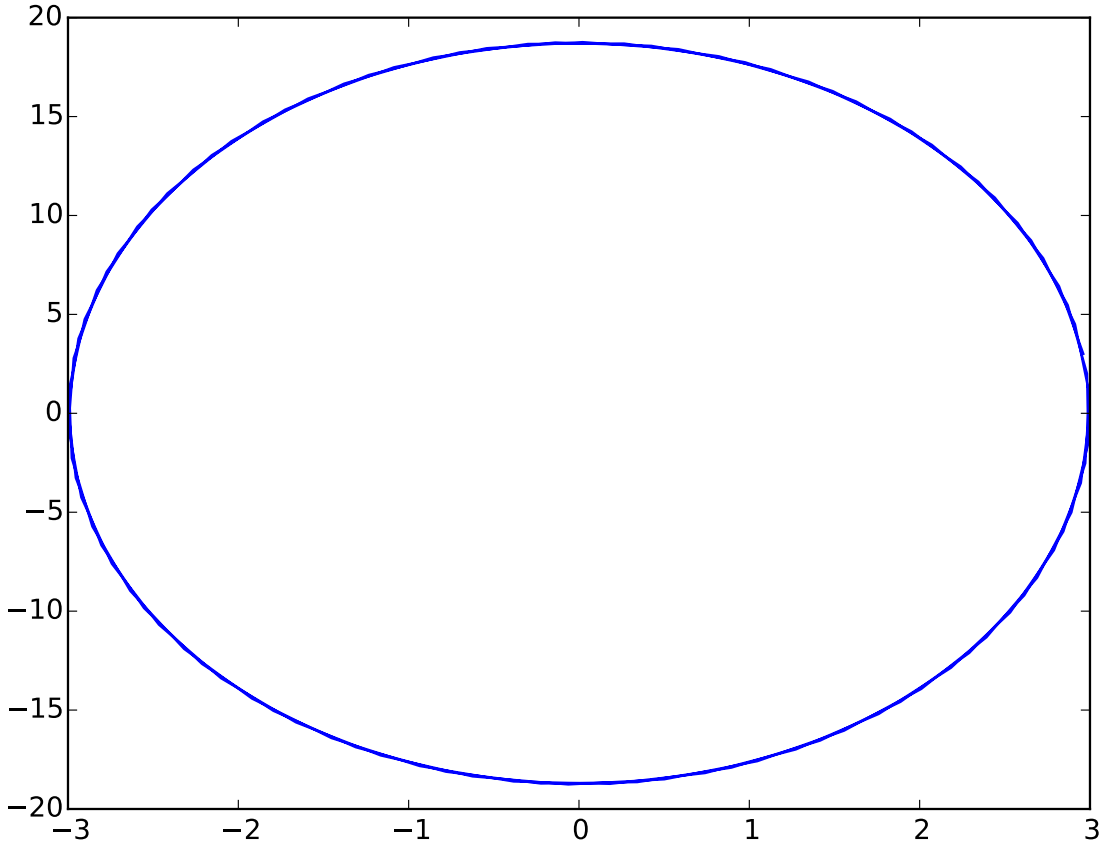
```python
def verlet(f, y0, z0, h, n) :
    y, z = y0, z0
    ys, zs = [y], [z]
    for i in range(n - 1) :
        fy     = f(y)
        y      = y + h*z + 0.5*h**2*fy
        fy_new = f(y)
        z      = z + 0.5*h*(fy + fy_new)
        ys.append(y)
        zs.append(z)
    return ys, zs
```

```
def phase(methode, f, y0, z0, h, n, no) :
    ys, zs = methode(f, y0, z0, h, n)
    plt.plot(ys, zs)
    plt.savefig('Graph' + str(no) + '.pdf')
```

```
In [18]: phase(verlet, lambda x: -(2*pi)**2*x, 3, 0, 3/100, 100, 3)
```

```
def phase(methode, f, y0, z0, h, n, no) :
    ys, zs = methode(f, y0, z0, h, n)
    plt.plot(ys, zs)
    plt.savefig('Graph' + str(no) + '.pdf')
```

```
In [18]: phase(verlet, lambda x: -(2*pi)**2*x, 3, 0, 3/100, 100, 3)
```


20
15
10
5
0
−5
−10
−15
−20
−3
−2
−1
0
1
2
3

D'où sortent ces formules ?

Verlet / Leapfrog / Point milieu

Point milieu

D'où sortent ces formules ?

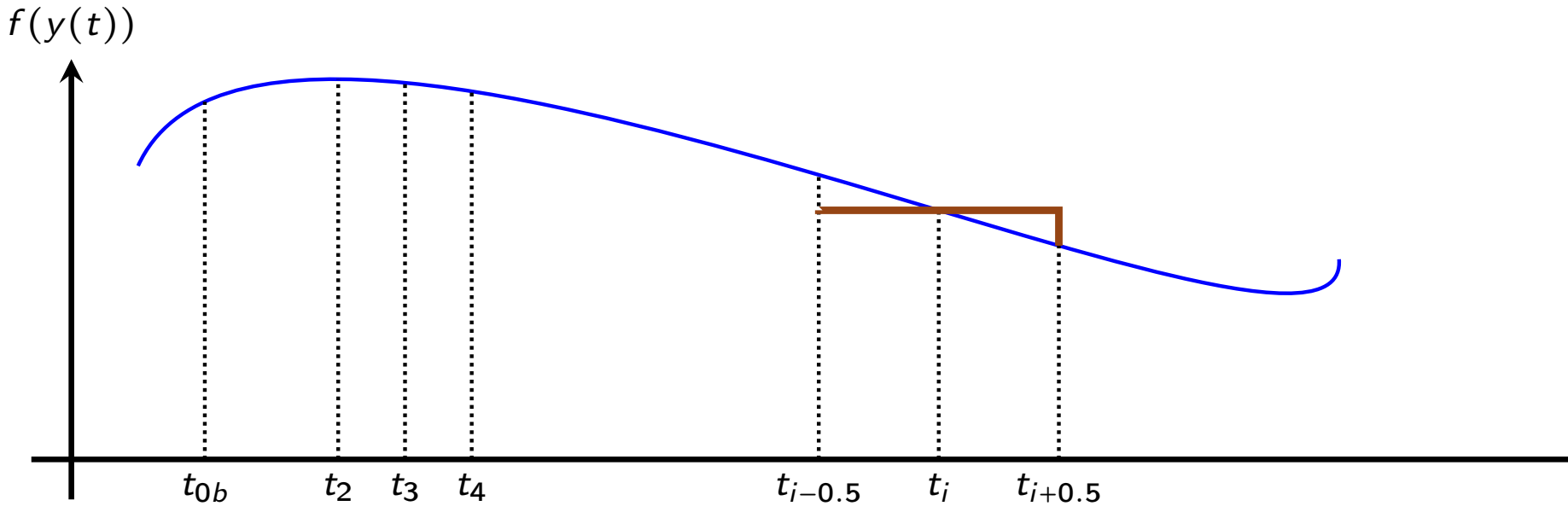
Verlet / Leapfrog / Point milieu

Point milieu

D'où sortent ces formules ?

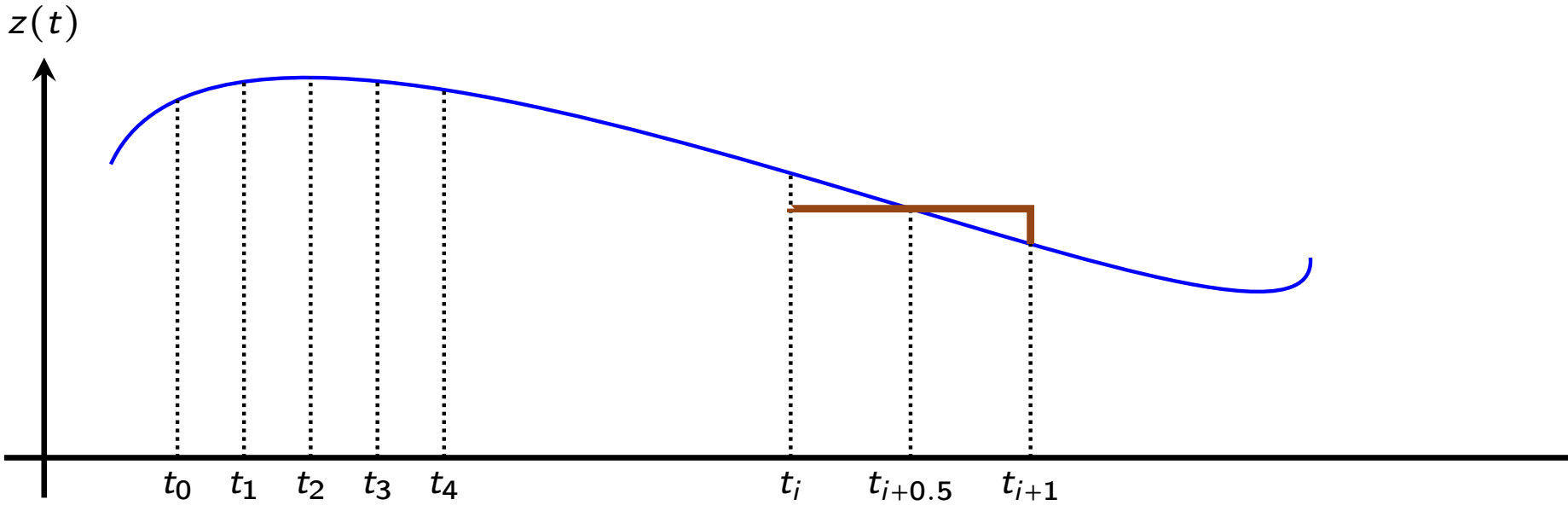
Verlet / Leapfrog / Point milieu

Point milieu



$$z_{i+1/2} - z_{i-1/2} = \int_{t_{i-\mathbf{1/2}}}^{t_{i+\mathbf{1/2}}} f(y(t))\,\mathrm{d}t = hf_i$$

Point milieu



$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} z(t)\,\mathrm{d}t = h z_{i+1/2}$$

$$\begin{aligned} z_{i+3/2} &= z_{i+1/2} + h f_{i+1} \\ &= z_{i-1/2} + h(f_i + f_{i+1}) \end{aligned}$$

$$\frac{z_{i+3/2} + z_{i+1/2}}{2} = \frac{z_{i+3/2} + z_{i+1/2}}{2} + \frac{h(f_i + f_{i+1})}{2}$$

$$z_{i+1} = z_i + h\frac{f_i + f_{i+1}}{2}$$

$$\begin{aligned} z_{i+3/2} &= z_{i+1/2} + h f_{i+1} \\ &= z_{i-1/2} + h(f_i + f_{i+1}) \end{aligned}$$

$$\frac{z_{i+3/2} + z_{i+1/2}}{2} = \frac{z_{i+3/2} + z_{i+1/2}}{2} + \frac{h(f_i + f_{i+1})}{2}$$

$$z_{i+1} = z_i + h\frac{f_i + f_{i+1}}{2}$$

$$\begin{aligned} z_{i+3/2} &= z_{i+1/2} + h f_{i+1} \\ &= z_{i-1/2} + h(f_i + f_{i+1}) \end{aligned}$$

$$\frac{z_{i+3/2} + z_{i+1/2}}{2} = \frac{z_{i+3/2} + z_{i+1/2}}{2} + \frac{h(f_i + f_{i+1})}{2}$$

$$z_{i+1} = z_i + h\frac{f_i + f_{i+1}}{2}$$

$$y_{i+1} = y_i + h z_{i+1/2}$$

$$z_i = \frac{z_{i-1/2} + z_{i+1/2}}{2} \qquad z_{i-1/2} = z_{i+1/2} - h f_i$$

$$z_{i+1/2} = z_i + h\frac{f_i}{2}$$

$$y_{i+1} = y_i + h\left(z_i + \frac{h f_i}{2}\right)$$

$$y_{i+1} = y_i + h z_{i+1/2}$$

$$z_i = \frac{z_{i-1/2} + z_{i+1/2}}{2} \qquad z_{i-1/2} = z_{i+1/2} - h f_i$$

$$z_{i+1/2} = z_i + h\frac{f_i}{2}$$

$$y_{i+1} = y_i + h\left(z_i + \frac{h f_i}{2}\right)$$

$$y_{i+1} = y_i + h z_{i+1/2}$$

$$z_i = \frac{z_{i-1/2} + z_{i+1/2}}{2} \qquad z_{i-1/2} = z_{i+1/2} - h f_i$$

$$z_{i+1/2} = z_i + h\frac{f_i}{2}$$

$$y_{i+1} = y_i + h\left(z_i + \frac{h f_i}{2}\right)$$

$$y_{i+1} = y_i + h z_{i+1/2}$$

$$z_i = \frac{z_{i-1/2} + z_{i+1/2}}{2} \qquad z_{i-1/2} = z_{i+1/2} - h f_i$$

$$z_{i+1/2} = z_i + h\frac{f_i}{2}$$

$$y_{i+1} = y_i + h\left(z_i + \frac{h f_i}{2}\right)$$

$$y_{i+1} = y_i + h z_{i+1/2}$$

$$z_i = \frac{z_{i-1/2} + z_{i+1/2}}{2} \qquad z_{i-1/2} = z_{i+1/2} - h f_i$$

$$z_{i+1/2} = z_i + h\frac{f_i}{2}$$

$$y_{i+1} = y_i + h\left(z_i + \frac{h f_i}{2}\right)$$

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

On connaît les valeurs d'une fonction f en des points x_i et on cherche à en déduire une approximation polynomiale P de f sur $[a, b]$.

Interpolation
Extrapolation
Intégration
Équations différentielles

On connaît les valeurs d'une fonction f en des points x_i et on cherche à en déduire une approximation polynomiale P de f sur $[a, b]$.

Interpolation

Extrapolation

Intégration

Équations différentielles

On connaît les valeurs d'une fonction f en des points x_i et on cherche à en déduire une approximation polynomiale P de f sur $[a, b]$.

Interpolation

Extrapolation

Intégration

Équations différentielles

On connaît les valeurs d'une fonction f en des points x_i et on cherche à en déduire une approximation polynomiale P de f sur $[a, b]$.

Interpolation

Extrapolation

Intégration

Équations différentielles

On connaît les valeurs d'une fonction f en des points x_i et on cherche à en déduire une approximation polynomiale P de f sur $[a, b]$.
Interpolation
Extrapolation
Intégration
Équations différentielles

Sommaire

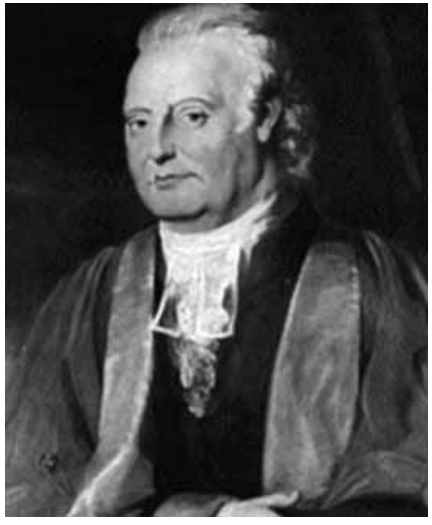
Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

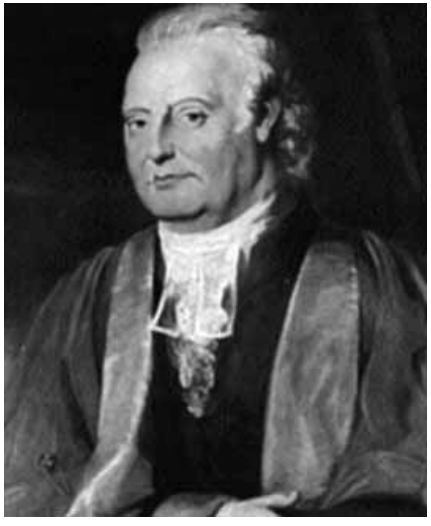


Edward Waring
(1736 - 1798)

1779 : *Problems Concerning Interpolations*

Giuseppe Lodovico de Lagrangia
(1736 - 1813)

1795 : *Leçons Élémentaires sur les Mathématiques Données a l'École Normale*

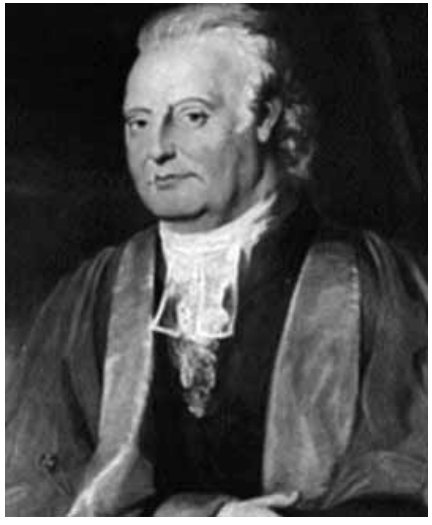


Edward Waring
(1736 - 1798)

Giuseppe Lodovico de Lagrangia
(1736 - 1813)

1779 : *Problems Concerning Interpolations*

1795 : *Leçons Élémentaires sur les Mathématiques Données a l'École Normale*

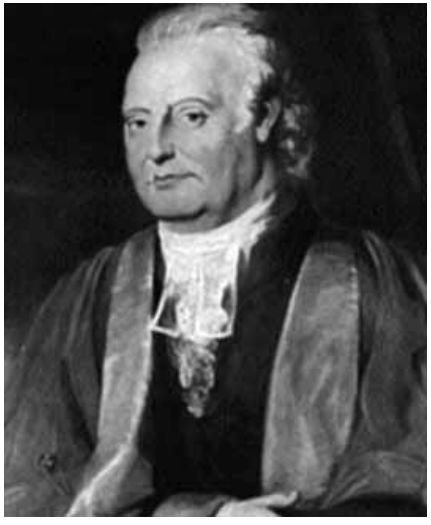


Edward Waring
(1736 - 1798)

Giuseppe Lodovico de Lagrangia
(1736 - 1813)

1779 : *Problems Concerning Interpolations*

1795 : *Leçons Élémentaires sur les Mathématiques Données a l'École Normale*



Edward Waring
(1736 - 1798)

1779 : *Problems Concerning Interpolations*

Giuseppe Lodovico de Lagrangia
(1736 - 1813)

1795 : *Leçons Élémentaires sur les Mathématiques Données a l'École Normale*

On travaillera dans cette section avec des polynômes de $\mathbb{R}_n[X]$, une fonction numérique réelle f définie sur un intervalle $[a, b]$ et une famille $\{x_0, x_1, ..., x_n\}$ de $n+1$ points distincts de $[a, b]$.

Théorème 1

Il existe un polynôme P_n de $\mathbb{R}_n[X]$ et un seul tel que :

$$\forall i \in \{0, 1, ..., n\} \qquad \widetilde{P_n}(x_i) = f(x_i)$$

Ce polynôme s'écrit :

$$P_n(X) = \sum_{i=0}^{n} f(x_i) L_i(X)$$

avec :

$$L_i(X) = \prod_{j=0, j\neq i}^{n} \frac{X - x_j}{x_i - x_j}$$

$$Q(X) = \prod_{j=0}^{n} (X - x_j)$$

$$Q'(X) = \sum_{i=0}^{n} \prod_{j=0, j\neq i}^{n} (X - x_j)$$

$$Q'(x_i) = \prod_{j=0, j\neq i}^{n} (x_i - x_j)$$

$$L_i(X) = \frac{Q(X)}{Q'(x_i)(X - x_i)}$$

$$Q(X) = \prod_{j=0}^{n} (X - x_j)$$

$$Q'(X) = \sum_{i=0}^{n} \prod_{j=0, j\neq i}^{n} (X - x_j)$$

$$Q'(x_i) = \prod_{j=0, j\neq i}^{n} (x_i - x_j)$$

$$L_i(X) = \frac{Q(X)}{Q'(x_i)(X - x_i)}$$

$$Q(X) = \prod_{j=0}^{n} (X - x_j)$$

$$Q'(X) = \sum_{i=0}^{n} \prod_{j=0, j\neq i}^{n} (X - x_j)$$

$$Q'(x_i) = \prod_{j=0, j\neq i}^{n} (x_i - x_j)$$

$$L_i(X) = \frac{Q(X)}{Q'(x_i)(X - x_i)}$$

$$Q(X) = \prod_{j=0}^{n} (X - x_j)$$

$$Q'(X) = \sum_{i=0}^{n} \prod_{j=0, j\neq i}^{n} (X - x_j)$$

$$Q'(x_i) = \prod_{j=0, j\neq i}^{n} (x_i - x_j)$$

$$L_i(X) = \frac{Q(X)}{Q'(x_i)(X - x_i)}$$

Choix de la base :

- base canonique : matrice de Vandermonde
- base des polynômes de Lagrange : matrice diagonale
- base des différences divisées : 1, $X - x_0$, $(X - x_0)(X - x_1)$,...,$(X - x_0)(X - x_1)\cdots(X - x_{n-1})$

Choix de la base :

- base canonique : matrice de Vandermonde
- base des polynômes de Lagrange : matrice diagonale
- base des différences divisées : 1, $X - x_0$, $(X - x_0)(X - x_1)$,...,$(X - x_0)(X - x_1)\cdots(X - x_{n-1})$

Choix de la base :

- base canonique : matrice de Vandermonde
- base des polynômes de Lagrange : matrice diagonale
- base des différences divisées : 1, $X - x_0$, $(X - x_0)(X - x_1)$,...,$(X - x_0)(X - x_1)\cdots(X - x_{n-1})$

D'un point de vue algorithmique, que se passe-t-il lorsqu'on rajoute un point pour améliorer l'interpolation ?

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$f[x_0, x_2, ..., x_k] = \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0}$$

$f[x_0]$ $f[x_0, x_1]$ $f[x_0, x_1, x_2]$ $f[x_0, x_1, x_2, x_3]$
$f[x_1]$ $f[x_1, x_2]$ $f[x_1, x_2, x_0]$
$f[x_2]$ $f[x_2, x_3]$
$f[x_3]$

Coefficients du polynôme interpolateur dans la base des différences divisées :

$$P_n(t) = f[x_0] + \sum_{k=1}^{n} f[x_0, ..., x_k](t - x_0)(t - x_1)\cdots(t - x_{k-1})$$

$$f[x_0, x_2, ..., x_k] = \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0}$$

$\boldsymbol{f[x_0]}$	$\boldsymbol{f[x_0, x_1]}$	$\boldsymbol{f[x_0, x_1, x_2]}$	$\boldsymbol{f[x_0, x_1, x_2, x_3]}$
$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_0]$	
$f[x_2]$	$f[x_2, x_3]$		
$f[x_3]$			

Coefficients du polynôme interpolateur dans la base des différences divisées :

$$P_n(t) = f[x_0] + \sum_{k=1}^{n} f[x_0, ..., x_k](t - x_0)(t - x_1)\cdots(t - x_{k-1})$$

$$f[x_0, x_2, ..., x_k] = \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0}$$

$\boldsymbol{f[x_0]}$	$\boldsymbol{f[x_0, x_1]}$	$\boldsymbol{f[x_0, x_1, x_2]}$	$\boldsymbol{f[x_0, x_1, x_2, x_3]}$
$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_0]$	
$f[x_2]$	$f[x_2, x_3]$		
$f[x_3]$			

Coefficients du polynôme interpolateur dans la base des différences divisées :

$$P_n(t) = f[x_0] + \sum_{k=1}^{n} f[x_0, ..., x_k](t - x_0)(t - x_1)\cdots(t - x_{k-1})$$

D'un point de vue algorithmique, que se passe-t-il lorsqu'on rajoute un point pour améliorer l'interpolation ?

$$
\begin{array}{ccccc}
f[x_0] & f[x_0,x_1] & f[x_0,x_1,x_2] & f[x_0,x_1,x_2,x_3] & f[x_0,x_1,x_2,x_3,x_4] \\
f[x_1] & f[x_1,x_2] & f[x_1,x_2,x_0] & f[x_1,x_2,x_3,x_4] & \\
f[x_2] & f[x_2,x_3] & f[x_2,x_3,x_4] & & \\
f[x_3] & f[x_3,x_4] & & & \\
f[x_4] & & & &
\end{array}
$$

D'un point de vue algorithmique, que se passe-t-il lorsqu'on rajoute un point pour améliorer l'interpolation ?

$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3, x_4]$
$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_0]$	$f[x_1, x_2, x_3, x_4]$	
$f[x_2]$	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$		
$f[x_3]$	$f[x_3, x_4]$			
$f[x_4]$				

D'un point de vue algorithmique, que se passe-t-il lorsqu'on rajoute un point pour améliorer l'interpolation ?

$f[x_0]$	$f[x_0,x_1]$	$f[x_0,x_1,x_2]$	$f[x_0,x_1,x_2,x_3]$	$f[x_0,x_1,x_2,x_3,x_4]$
$f[x_1]$	$f[x_1,x_2]$	$f[x_1,x_2,x_0]$	$f[x_1,x_2,x_3,x_4]$	
$f[x_2]$	$f[x_2,x_3]$	$f[x_2,x_3,x_4]$		
$f[x_3]$	$f[x_3,x_4]$			
$f[x_4]$				

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

Soit P_n un polynôme interpolateur en x_0, x_2,...,x_n points distincts de $[a, b]$ d'une fonction f définie sur $[a, b]$.

Soit $t \in [a, b]$ distinct des x_i (sinon...). Posons $e_n(t) = P_n(t) - f(t)$ et P_{n+1} le polynôme qui interpole f en x_0,...,x_n, t.

$$P_{n+1}(t) = f(t) = P_n(t) + f[x_0, x_2, ..., x_n, t] \prod_{i=0}^{n} (t - x_i)$$

$$e_n(t) = -f[x_0, x_2, ..., x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Une n^e formule de Cauchy montre que si f est de classe C^{n+1} sur $[a, b]$, il existe $\xi_t \in]a, b[$ tel que :

$$e_n(t) = -\frac{f^{(n+1)}(\xi_t)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \quad \text{soit} \quad |e_n(t)| \leqslant \max_{\xi \in]a,b[} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \right|$$

Soit P_n un polynôme interpolateur en x_0, x_2,...,x_n points distincts de $[a, b]$ d'une fonction f définie sur $[a, b]$.
Soit $t \in [a, b]$ distinct des x_i (sinon...). Posons $e_n(t) = P_n(t) - f(t)$ et P_{n+1} le polynôme qui interpole f en x_0,...,x_n, t.

$$P_{n+1}(t) = f(t) = P_n(t) + f[x_0, x_2, ..., x_n, t] \prod_{i=0}^{n} (t - x_i)$$

$$e_n(t) = -f[x_0, x_2, ..., x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Une n^e formule de CAUCHY montre que si f est de classe C^{n+1} sur $[a, b]$, il existe $\xi_t \in]a, b[$ tel que :

$$e_n(t) = -\frac{f^{(n+1)}(\xi_t)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \quad \text{soit} \quad |e_n(t)| \leqslant \max_{\xi \in]a,b[} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \right|$$

Soit P_n un polynôme interpolateur en x_0, x_2,...,x_n points distincts de $[a, b]$ d'une fonction f définie sur $[a, b]$.
Soit $t \in [a, b]$ distinct des x_i (sinon...). Posons $e_n(t) = P_n(t) - f(t)$ et P_{n+1} le polynôme qui interpole f en x_0,...,x_n, t.

$$P_{n+1}(t) = f(t) = P_n(t) + f[x_0, x_2, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

$$e_n(t) = -f[x_0, x_2, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Une n^e formule de CAUCHY montre que si f est de classe C^{n+1} sur $[a, b]$, il existe $\xi_t \in]a, b[$ tel que :

$$e_n(t) = -\frac{f^{(n+1)}(\xi_t)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \quad \text{soit} \quad |e_n(t)| \leqslant \max_{\xi \in]a,b[} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \right|$$

Soit P_n un polynôme interpolateur en x_0, x_2,...,x_n points distincts de $[a, b]$ d'une fonction f définie sur $[a, b]$.
Soit $t \in [a, b]$ distinct des x_i (sinon...). Posons $e_n(t) = P_n(t) - f(t)$ et P_{n+1} le polynôme qui interpole f en x_0,...,x_n, t.

$$P_{n+1}(t) = f(t) = P_n(t) + f[x_0, x_2, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

$$e_n(t) = -f[x_0, x_2, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Une n^e formule de Cauchy montre que si f est de classe C^{n+1} sur $[a, b]$, il existe $\xi_t \in]a, b[$ tel que :

$$e_n(t) = -\frac{f^{(n+1)}(\xi_t)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \quad \text{soit} \quad |e_n(t)| \leqslant \max_{\xi \in]a,b[} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \right|$$

Soit P_n un polynôme interpolateur en x_0, x_2,...,x_n points distincts de $[a,b]$ d'une fonction f définie sur $[a,b]$.
Soit $t \in [a,b]$ distinct des x_i (sinon...). Posons $e_n(t) = P_n(t) - f(t)$ et P_{n+1} le polynôme qui interpole f en x_0,...,x_n, t.

$$P_{n+1}(t) = f(t) = P_n(t) + f[x_0, x_2, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

$$e_n(t) = -f[x_0, x_2, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Une n^{e} formule de Cauchy montre que si f est de classe C^{n+1} sur $[a,b]$, il existe $\xi_t \in]a,b[$ tel que :

$$e_n(t) = -\frac{f^{(n+1)}(\xi_t)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \quad \text{soit} \quad |e_n(t)| \leqslant \max_{\xi \in]a,b[} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (t - x_i) \right|$$

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$|e_n(t)| \leqslant \frac{1}{(n+1)!} \left\| \prod_{i=0}^{n} (t - x_i) \right\|_\infty \left\| f^{(n+1)} \right\|_\infty$$

$$\left\| \prod_{i=0}^{n} (t - x_i) \right\|_\infty$$

Ce n'est qu'un début...

$$|e_n(t)| \leqslant \frac{1}{(n+1)!} \left\| \prod_{i=0}^{n} (t - x_i) \right\|_\infty \left\| f^{(n+1)} \right\|_\infty$$

$$\left\| \prod_{i=0}^{n} (t - x_i) \right\|_\infty$$

Ce n'est qu'un début...

$$|e_n(t)| \leqslant \frac{1}{(n+1)!} \left\| \prod_{i=0}^{n} (t - x_i) \right\|_\infty \left\| f^{(n+1)} \right\|_\infty$$

$$\left\| \prod_{i=0}^{n} (t - x_i) \right\|_\infty$$

Ce n'est qu'un début...

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

Juste pour information...On appele n^e constante de Lebesgue le réel

$$\Lambda_n = \max_{t\in[a,b]} \sum_{i=0}^{n} |L_i(t)|$$

Alors

Théorème 2

Soit f une fonction continue sur $[a, b]$.

$$\|e_n\|_\infty \leqslant (1 + \Lambda_n) \min_{Q\in\mathbb{R}[X]} \|f - \widetilde{Q}\|_\infty$$

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$\int_a^b f(t)\,\mathrm{d}t = \sum_{i=1}^{n} w_i f(x_i) + E_n(f)$$

les w_i sont les *poids*, les x_i sont les pivots, $E_n(f)$ l'erreur commise.

$$\int_a^b f(t)\,\mathrm{d}t = \sum_{i=1}^{n} w_i f(x_i) + E_n(f)$$

les w_i sont les *poids*, les x_i sont les pivots, $E_n(f)$ l'erreur commise.

$$\int_a^b f(t)\,\mathrm{d}t = \sum_{i=1}^{n} w_i f(x_i) + E_n(f)$$

les w_i sont les *poids*, les x_i sont les pivots, $E_n(f)$ l'erreur commise.

$$\int_a^b f(t)\,\mathrm{d}t = \sum_{i=1}^{n} w_i f(x_i) + E_n(f)$$

les w_i sont les *poids*, les x_i sont les pivots, $E_n(f)$ l'erreur commise.

$$\int_a^b f(t)\,\mathrm{d}t \approx \int_a^b P_n(t)\,\mathrm{d}t \quad = \quad \int_a^b \sum_{i=0}^{n} f(x_i) L_i(t)\,\mathrm{d}t$$

$$\int_a^b f(t)\,\mathrm{d}t \approx \int_a^b P_n(t)\,\mathrm{d}t \quad = \quad \int_a^b \sum_{i=0}^{n} f(x_i) L_i(t)\,\mathrm{d}t$$

$$\begin{aligned}\int_a^b f(t)\,\mathrm{d}t \approx \int_a^b P_n(t)\,\mathrm{d}t &= \int_a^b \sum_{i=0}^{n} f(x_i) L_i(t)\,\mathrm{d}t \\ &= \sum_{i=0}^{n} \left(\int_a^b L_i(t)\,\mathrm{d}t \right) f(x_i)\end{aligned}$$

$$\begin{aligned}\int_a^b f(t)\,\mathrm{d}t \approx \int_a^b P_n(t)\,\mathrm{d}t &= \int_a^b \sum_{i=0}^{n} f(x_i) L_i(t)\,\mathrm{d}t \\ &= \sum_{i=0}^{n} \underbrace{\left(\int_a^b L_i(t)\,\mathrm{d}t\right)}_{w_i} f(x_i)\end{aligned}$$

Dans la base des différences divisées :

$$\int_a^b f(t)\,\mathrm{d}t \approx \int_a^b f[x_0] + \sum_{k=1}^{n} \int_a^b f[x_0,\ldots,x_k](t-x_0)(t-x_1)\cdots(t-x_{k-1})\,\mathrm{d}t$$

On notera :

- $I(f) = \int_a^b f(t)\,\mathrm{d}t$
- $I_n(f) = \sum_{i=0}^{n} w_i f(x_i)$
- $E_n(f) = I(f) - I_n(f)$

On notera :

- $I(f) = \int_a^b f(t)\,\mathrm{d}t$
- $I_n(f) = \sum_{i=0}^{n} w_i f(x_i)$
- $E_n(f) = I(f) - I_n(f)$

On notera :

- $I(f) = \int_a^b f(t)\,\mathrm{d}t$
- $I_n(f) = \sum_{i=0}^{n} w_i f(x_i)$
- $E_n(f) = I(f) - I_n(f)$

Définition 3

Une méthode de quadrature est **exacte** sur un ensemble A si pour toute fonction f de A :

$$E_n(f) = 0$$

Définition 4

Une méthode de quadrature est d'**ordre** m si elle est exacte pour tout x^k avec $k \in \{0...m\}$ mais n'est pas exacte pour x^{m+1}.

Définition 3

Une méthode de quadrature est **exacte** sur un ensemble A si pour toute fonction f de A :

$$E_n(f) = 0$$

Définition 4

Une méthode de quadrature est d'**ordre** m si elle est exacte pour tout x^k avec $k \in \{0...m\}$ mais n'est pas exacte pour x^{m+1}.

Définition 3

Une méthode de quadrature est **exacte** sur un ensemble A si pour toute fonction f de A :

$$E_n(f) = 0$$

Définition 4

Une méthode de quadrature est d'**ordre** m si elle est exacte pour tout x^k avec $k \in \{0...m\}$ mais n'est pas exacte pour x^{m+1}.

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,\mathrm{d}t = (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a, b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,dt \approx (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a, b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,dt \approx (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a,b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,dt \approx (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a,b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,dt \approx (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a, b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,dt \approx (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a, b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

$$x_0 = a \qquad OU \qquad x_0 = b$$

$$I_0(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t \qquad OU \qquad f(b) \int_a^b L_0(t)\,\mathrm{d}t$$

$$L_0(t) = 1$$

Théorème 5 (Formule des rectangles)

$$\int_a^b f(t)\,dt \approx (b-a)f(a) \qquad OU \qquad (b-a)f(b)$$

Si f est C^1, il existe $\xi \in]a, b[$ tel que

$$R_0(f) = \pm \frac{(b-a)^2}{2} f'(\xi)$$

Démonstration ?

```python
def rect( f, a, b, n ):
    s = 0
    h = ( b - a ) / n
    for k in range( n ):
        s += h * f( a + k*h )
    return s
```

Cette méthode est-elle bien d'ordre 0 ?

$$\int_a^b 1 \, \mathrm{d}t = (b-a) = f(a)(b-a)$$

$$\int_a^b t \, \mathrm{d}t = \frac{b^2 - a^2}{2} \neq f(a)(b-a)$$

Cette méthode est-elle bien d'ordre 0 ?

$$\int_a^b 1 \, \mathrm{d}t = (b-a) = f(a)(b-a)$$

$$\int_a^b t \, \mathrm{d}t = \frac{b^2 - a^2}{2} \neq f(a)(b-a)$$

Cette méthode est-elle bien d'ordre 0 ?

$$\int_a^b 1 \, \mathrm{d}t = (b-a) = f(a)(b-a)$$

$$\int_a^b t \, \mathrm{d}t = \frac{b^2 - a^2}{2} \neq f(a)(b-a)$$

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$x_0 = a \qquad x_1 = b$$

$$I_1(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t + f(b) \int_a^b L_1(t)\,\mathrm{d}t$$

$$L_0(t) = \frac{t-b}{a-b} \qquad L_1(t) = \frac{t-a}{b-a}$$

Calculez....

$$x_0 = a \qquad x_1 = b$$

$$I_1(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t + f(b) \int_a^b L_1(t)\,\mathrm{d}t$$

$$L_0(t) = \frac{t-b}{a-b} \qquad L_1(t) = \frac{t-a}{b-a}$$

Calculez....

Théorème 6 (Formule des trapèzes)

$$\int_a^b f(t)\,\mathrm{d}t \approx \frac{b-a}{2}(f(a)+f(b))$$

$$x_0 = a \qquad x_1 = b$$

$$I_1(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t + f(b) \int_a^b L_1(t)\,\mathrm{d}t$$

$$L_0(t) = \frac{t-b}{a-b} \qquad L_1(t) = \frac{t-a}{b-a}$$

Calculez....

Théorème 6 (Formule des trapèzes)

$$\int_a^b f(t)\,dt \approx \frac{b-a}{2}(f(a)+f(b))$$

$$x_0 = a \qquad x_1 = b$$

$$I_1(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t + f(b) \int_a^b L_1(t)\,\mathrm{d}t$$

$$L_0(t) = \frac{t-b}{a-b} \qquad L_1(t) = \frac{t-a}{b-a}$$

Calculez....

Théorème 6 (Formule des trapèzes)

$$\int_a^b f(t)\,dt \approx \frac{b-a}{2}\left(f(a)+f(b)\right)$$

$$x_0 = a \qquad x_1 = b$$

$$I_1(f) = f(a) \int_a^b L_0(t)\,\mathrm{d}t + f(b) \int_a^b L_1(t)\,\mathrm{d}t$$

$$L_0(t) = \frac{t-b}{a-b} \qquad L_1(t) = \frac{t-a}{b-a}$$

Calculez....

Théorème 6 (Formule des trapèzes)

$$\int_a^b f(t)\,dt \approx \frac{b-a}{2}\big(f(a)+f(b)\big)$$

Si f est C^2, il existe $\xi \in]a, b[$ tel que :

$$|R_1(f)| = \frac{(b-a)^3}{12}|f''(\xi)|$$

Ipp de $\int_a^b \left(t - \frac{a+b}{2}\right) f'(t)\,\mathrm{d}t$ + théorème de la moyenne.
Ordre 1 ?

Si f est C^2, il existe $\xi \in]a, b[$ tel que :

$$|R_1(f)| = \frac{(b-a)^3}{12}|f''(\xi)|$$

Ipp de $\int_a^b \left(t - \frac{a+b}{2}\right) f'(t)\,\mathrm{d}t$ + théorème de la moyenne.

Ordre 1 ?

Si f est C^2, il existe $\xi \in]a, b[$ tel que :

$$|R_1(f)| = \frac{(b-a)^3}{12}|f''(\xi)|$$

Ipp de $\int_a^b \left(t - \frac{a+b}{2}\right) f'(t)\,\mathrm{d}t$ + théorème de la moyenne.
Ordre 1 ?

Exercice 1

Méthode du point milieu : quel est son ordre ?

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b$$

Pour changer, allons dans la base des différences divisées :

$$P_2(t) = f[x_0] + f[x_0, x_1](t - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Calculez...

Théorème 7 (Formule de Simpson)

$$\int_a^b f(t)\,\mathrm{d}t = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b$$

Pour changer, allons dans la base des différences divisées :

$$P_2(t) = f[x_0] + f[x_0, x_1](t - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Calculez...

Théorème 7 (Formule de Simpson)

$$\int_a^b f(t)\,dt \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+}{2}\right) + f(b)\right)$$

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b$$

Pour changer, allons dans la base des différences divisées :

$$P_2(t) = f[x_0] + f[x_0, x_1](t - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Calculez...

Théorème 7 (Formule de Simpson)

$$\int_a^b f(t)\,dt \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+}{2}\right) + f(b)\right)$$

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b$$

Pour changer, allons dans la base des différences divisées :

$$P_2(t) = f[x_0] + f[x_0, x_1](t - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Calculez...

Théorème 7 (Formule de Simpson)

$$\int_a^b f(t)\, dt \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+}{2}\right) + f(b)\right)$$

ordre ?

$$|R_2(f)| = \frac{(b-a)^5}{2880}|f^{(4)}(\xi)|$$

ordre ?

$$|R_2(f)| = \frac{(b-a)^5}{2880}|f^{(4)}(\xi)|$$

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$y'(x) = f(y(x), x)$$

$$\int_{x_n}^{x_{n+1}} y'(x)\,\mathrm{d}x \approx y'(x_n) \times (x_{n+1} - x_n) = f(y_n, x_n) \times (x_{n+1} - x_n) = h \times f(y_n, x_n)$$

$$y'(x) = f(y(x), x)$$

$$\int_{x_n}^{x_{n+1}} y'(x)\,\mathrm{d}x \approx y'(x_n) \times (x_{n+1} - x_n) = f(y_n, x_n) \times (x_{n+1} - x_n) = h \times f(y_n, x_n)$$

$$y'(x) = f(y(x), x)$$

$$\int_{x_n}^{x_{n+1}} y'(x)\,\mathrm{d}x \approx y'(x_n) \times (x_{n+1} - x_n) = f(y_n, x_n) \times (x_{n+1} - x_n) = h \times f(y_n, x_n)$$

$$y'(x) = f(y(x), x)$$

$$\int_{x_n}^{x_{n+1}} y'(x)\,\mathrm{d}x \approx y'(x_n) \times (x_{n+1} - x_n) = f(y_n, x_n) \times (x_{n+1} - x_n) = h \times f(y_n, x_n)$$

$$y'(x) = f(y(x), x)$$

$$\int_{x_n}^{x_{n+1}} y'(x)\,\mathrm{d}x \approx y'(x_n) \times (x_{n+1} - x_n) = f(y_n, x_n) \times (x_{n+1} - x_n) = h \times f(y_n, x_n)$$

```python
def rect(f, a, b, n):
    s = 0
    h = ( b - a ) / n
    for k in range( n ):
        s += h * f( a + k*h )
    return s
```

```python
def euler_exp(f, a, b, y0, n):
    y = y0
    h  = ( b - a ) / n
    ys  = [y]
    for k in range( n ):
        y += h * f( y,  a + k*h )
        ys.append(y)
    return ys
```

```python
def rect(f, a, b, n):
    s = 0
    h = ( b - a ) / n
    for k in range( n ):
        s += h * f( a + k*h )
    return s
```

```python
def euler_exp(f, a, b, y0, n):
    y = y0
    h  = ( b - a ) / n
    ys  = [y]
    for k in range( n ):
        y += h * f( y,  a + k*h )
        ys.append(y)
    return ys
```

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

$$\pi = 4 \int_0^1 \sqrt{1 - t^2}\, dt$$

```
from math import sqrt, pi

def f(t):
    return sqrt(1-t*t)
```

$$\pi = 4 \int_0^1 \sqrt{1 - t^2}\, dt$$

```
from math import sqrt, pi

def f(t):
    return sqrt(1-t*t)
```

Sommaire

Étude de schémas numériques
Schéma d'Euler explicite
Schéma de Verlet
Approximation polynomiale
Inter(extra)polation
Lagrange, Waring, Gauß, Newton
Newton / Différences divisées
Cherchez l'erreur
Choix des points
Instabilité
Méthodes de Newton-Cotes
Newton-Cotes d'ordre 0
Newton-Cotes d'ordre 1
Newton-Cotes d'ordre 2
L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)
Algo ≠ Programme : approximation de π
Méthode des rectangles
Méthode des trapèzes
Méthode de Simpson
Bye bye π ?

```python
def int_rec_droite(f,a,b,N):
    S = 0
    t = a
    dt = (b-a)/N
    while t < b :
        S += f(t)*dt
        t += dt
    return S
```

```python
def int_rec_gauche(f,a,b,N):
    S = 0
    t = a
    dt = (b-a)/N
    while t+dt < b :
        S += f(t+dt)*dt
        t += dt
    return S
```

```python
def int_rec_droite(f,a,b,N):
    S = 0
    t = a
    dt = (b-a)/N
    while t < b :
        S += f(t)*dt
        t += dt
    return S
```

```python
def int_rec_gauche(f,a,b,N):
    S = 0
    t = a
    dt = (b-a)/N
    while t+dt < b :
        S += f(t+dt)*dt
        t += dt
    return S
```

```
>>> 4*int_rec_gauche(f,0,1,100)
3.12041703177904 54
>>> 4*int_rec_gauche(f,0,1,10000)
3.1413914777848557
```

```
>>> pi - 4*int_rec_gauche(f,0,1,10000)
0.00020117580493739 24
```

```
>>> 4*int_rec_gauche(f,0,1,100)
3.12041703177904 54
>>> 4*int_rec_gauche(f,0,1,10000)
3.1413914777848557
```

```
>>> pi - 4*int_rec_gauche(f,0,1,10000)
0.00020117580493739 24
```

```
>>> for k in range(8): print(pi - 4*int_rec_gauche(f,0,1,10**k))
...
3.141592653589793
0.23707432138101003
0.021175621810747725
0.0020371866787676 22
0.00020117580493739 24
2.003710526476965e-05
2.001187481948108e-06
1.9943306694969465e-07
```

Sommaire

Étude de schémas numériques
Schéma d'Euler explicite
Schéma de Verlet
Approximation polynomiale
Inter(extra)polation
Lagrange, Waring, Gauß, Newton
Newton / Différences divisées
Cherchez l'erreur
Choix des points
Instabilité
Méthodes de Newton-Cotes
Newton-Cotes d'ordre 0
Newton-Cotes d'ordre 1
Newton-Cotes d'ordre 2
L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)
Algo ≠ Programme : approximation de π
Méthode des rectangles
Méthode des trapèzes
Méthode de Simpson
Bye bye π ?

```
def int_trap(f,a,b,N):
    S = 0
    t = a
    dt = (b-a)/N
    while t+dt <= b :
        S += (f(t)+f(t+dt))*dt/2
        t += dt
    return S
```

```
>>> for k in range(8): print(pi-4*int_trap(f,0,1,10**k))
...
1.1415926535897931
0.03707432436124236
0.0039969690066827 82
0.00012660703439637544
1.1758915650084134e-06
3.7144431530578004e-08
4.015919596866979e-09
-5.628750798791771e-10
```

Dernier résultat ??

```
>>> for k in range(8): print(pi-4*int_trap(f,0,1,10**k))
...
1.1415926535897931
0.03707432436124236
0.0039969690066827 82
0.00012660703439637544
1.1758915650084134e-06
3.7144431530578004e-08
4.015919596866979e-09
-5.628750798791771e-10
```

Dernier résultat ??

```
>>> def int_trap(f,a,b,N):
    S=0
    dt=(b-a)/N
    for k in range(N):
        S += (f(a + k*dt) + f(a + (k+1)*dt)) * dt/2
    return S
... ... ... ... ... ...
>>> for k in range(8): print(pi - 4*int_trap(f,0,1,10**k))
...
1.1415926535897931
0.0370743273414748 66
0.0011756218107490 39
3.7186678768730275e-05
1.1759784679377105e-06
3.7187829260432 18e-08
1.1759344609174605e-09
3.643840784661734e-11
```

$$\pi = 12\left(\int_0^{\frac{1}{2}} f(t)\,dt - \frac{\sqrt{3}}{8}\right)$$

```
>>> for k in range(7): print(pi-12*(int_trap(f,0,1/2,10**k)-sqrt(3)/8))
...
0.14159265358979312
0.0014430555975519788
0.05206198505358994
0.0051971622085238 42
1.4433143569192453e-09
1.3836487511298401e-11
5.196153916919854 5e-06
```

?????

$$\pi = 12\left(\int_0^{\frac{1}{2}} f(t)\,dt - \frac{\sqrt{3}}{8}\right)$$

```
>>> for k in range(7): print(pi-12*(int_trap(f,0,1/2,10**k)-sqrt(3)/8))
...
0.14159265358979312
0.0014430555975519788
0.05206198505358994
0.0051971622085238 42
1.4433143569192453e-09
1.3836487511298401e-11
5.1961539169198545e-06
```

?????

$$\pi = 12\left(\int_0^{\frac{1}{2}} f(t)\,dt - \frac{\sqrt{3}}{8}\right)$$

```
>>> for k in range(7): print(pi-12*(int_trap(f,0,1/2,10**k)-sqrt(3)/8))
...
0.14159265358979312
0.0014430555975519788
0.05206198505358994
0.0051971622085238 42
1.4433143569192453e-09
1.3836487511298401e-11
5.1961539169198545e-06
```

?????

```
>>> for k in range(7): print(pi-12*(int_trap(f,0,1/2,10**k)-sqrt(3)/8))
...
0.14159265358979223
0.0014430555975519788
1.4433724657703095e-05
1.4433755703402694e-07
1.4433609862862795e-09
1.4393375380450 38e-11
1.1457501614131615e-13
```

Sommaire

Étude de schémas numériques
Schéma d'Euler explicite
Schéma de Verlet
Approximation polynomiale
Inter(extra)polation
Lagrange, Waring, Gauß, Newton
Newton / Différences divisées
Cherchez l'erreur
Choix des points
Instabilité
Méthodes de Newton-Cotes
Newton-Cotes d'ordre 0
Newton-Cotes d'ordre 1
Newton-Cotes d'ordre 2
L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)
Algo ≠ Programme : approximation de π
Méthode des rectangles
Méthode des trapèzes
Méthode de Simpson
Bye bye π ?

```python
def int_simps(f,a,b,N):
    S = 0
    t = a
    dt = (b-a)/N
    while t+dt <= b:
        S += ( f(t) + 4*f(t + dt/2) + f(t + dt) ) * dt/6
        t += dt
    return S
```

```
>>> for k in range(8): print(pi-4*int_simps(f,0,1,10**k))
...
0.16552491016462367
0.00514558833917933
0.00376618470947676 1
0.00011925832541592 74
1.6234044686314064e-07
5.117280821309578e-09
3.783345636776403e-09
-5.92078830408127e-10
```

```
>>> for k in range(8): print(pi -
        ↪   12*(int_simps(f,0,1/2,10**k)-sqrt(3)/8))
...
0.0006601149512537319
7.997674877913141e-08
0.0520477428322228
0.005197018063311631
-2.042810365310288e-14
-5.88862292261183e-13
5.1961537730349505e-06
-8.213874025386758e-11
```

```
>>> for k in range(8): print(pi -
        ↪   12*(int_simps(f,0,1/2,10**k)-sqrt(3)/8))
...
0.0006601149512537319
7.997674877913141e-08
8.021139308311831e-12
5.329070518200751e-15
2.1316282072803006e-14
-3.907985046680551e-14
-3.375077994860476e-14
-5.284661597215745e-13
```

Ah oui, j'ai eu un cours sur les flottants

```
>>> for k in range(13):
...     print(pi-12*(int_simps(f,0,1/2,2**k)-sqrt(3)/8))
...
0.0006601149512537319
4.722142667734 63e-05
3.0824728671774437e-06
1.9496909775540416e-07
1.222303502856903e-08
7.645306610015723e-10
4.779110440722434e-11
2.9887203822909214e-12
1.8474111129762605e-13
7.993605777301127e-15
-4.440892098500626e-16
-1.3766765505351941e-14
-8.881784197001252e-15
```

```
>>> f(1e-8)
1.0
```

while et for

while t+dt<=b

```
>>> 1 + 0.1 + 0.1 + 0.1 - 0.3 <= 1
False
```

```
>>> f(1e-8)
1.0
```

while et for

while t+dt<=b

```
>>> 1 + 0.1 + 0.1 + 0.1 - 0.3 <= 1
False
```

```
>>> f(1e-8)
1.0
```

while et for

while t+dt<=b

```
>>> 1 + 0.1 + 0.1 + 0.1 - 0.3 <= 1
False
```

```
>>> f(1e-8)
1.0
```

while et for

while t+dt<=b

```
>>> 1 + 0.1 + 0.1 + 0.1 - 0.3 <= 1
False
```

```
print('-'*75)
print('{:22s} | {:22s} | {:22s}'.format('rectangle gauche', 'trapeze',
        ↪  'simpson'))
print('-'*75)

for k in range(10):
    print("{:1.16e} | {:1.16e} | {:1.16e}".format(\
        pi-12*(int_rec_gauche(f, 0, 1/2, 2**k)-sqrt(3)/8), \
        pi-12*(int_trap(f, 0, 1/2, 2**k)-sqrt(3)/8), \
        pi-12*(int_simps(f, 0, 1/2, 2**k)-sqrt(3)/8)))
```

rectangle gauche	trapeze	simpson
5.4351644223647e-01	1.4159265358979e-01	6.6011495125373e-04
2.3685514393423e-01	3.5893249610888e-02	4.7221426677346e-05
1.0948967563440e-01	9.0087284727298e-03	3.0824728671774e-06
5.2494967553668e-02	2.2544939728321e-03	1.9496909775540e-07
2.5684006510449e-02	5.6376972003269e-04	1.2223035028569e-08
1.2701069992491e-02	1.4095159728411e-04	7.6453066100157e-10
6.3152976703237e-03	3.5238472716692e-05	4.7791104407224e-11
3.1488392528258e-03	8.8096540218352e-06	2.9887203822909e-12
1.5722172151479e-03	2.2024157457778e-06	1.8474111129762e-13
7.8555800377522e-04	5.5060407389007e-07	7.9936057773011e-15

démonstration mathématique et domination de la machine valent mieux qu'observation et utilisation au petit bonheur

Sommaire

Étude de schémas numériques
- Schéma d'Euler explicite
- Schéma de Verlet

Approximation polynomiale
- Inter(extra)polation
- Lagrange, Waring, Gauß, Newton
- Newton / Différences divisées
- Cherchez l'erreur
- Choix des points
- Instabilité

Méthodes de Newton-Cotes
- Newton-Cotes d'ordre 0
- Newton-Cotes d'ordre 1
- Newton-Cotes d'ordre 2

L'A et l'Ω de l'ingénieur : la méthode d'Euler... :-)

Algo ≠ Programme : approximation de π
- Méthode des rectangles
- Méthode des trapèzes
- Méthode de Simpson
- Bye bye π ?

```
>>> from decimal import*
# on regle la precision a 30 chiffres
>>> getcontext().prec = 30
# on rentre les nombres entre apostrophes, comme des chaines
>>> deux=Decimal('2')
>>> deux.sqrt()
Decimal('1.41421356237309504880168872421')
>>> getcontext().prec = 50
>>> deux.sqrt()
Decimal('1.4142135623730950488016887242096980785696718753769')
```

James Gregory 1671 :

$$Arctan(x) = \sum_{k=0}^{+\infty} \frac{(-1)^k x^{2k+1}}{2k+1}$$

Wilhelm Leibniz 1682 : critère spécial des séries alternées :

$$\left| Arctan(x) - \sum_{k=0}^{n} \frac{(-1)^k x^{2k+1}}{2k+1} \right| < \frac{x^{2n+3}}{2n+3}$$

Valeur de n telle que $\frac{a^{2n+3}}{2n+3} \leqslant 10^{-d}$?

$$\left(2 + \frac{3}{n}\right) \log \frac{1}{a} \geqslant \frac{d}{n} - \frac{\log(2n+3)}{n}$$

James Gregory 1671 :

$$Arctan(x) = \sum_{k=0}^{+\infty} \frac{(-1)^k x^{2k+1}}{2k+1}$$

Wilhelm Leibniz 1682 : critère spécial des séries alternées :

$$\left| Arctan(x) - \sum_{k=0}^{n} \frac{(-1)^k x^{2k+1}}{2k+1} \right| < \frac{x^{2n+3}}{2n+3}$$

Valeur de n telle que $\frac{a^{2n+3}}{2n+3} \leqslant 10^{-d}$?

$$\left(2 + \frac{3}{n}\right) \log \frac{1}{a} \geqslant \frac{d}{n} - \frac{\log(2n+3)}{n}$$

James Gregory 1671 :

$$Arctan(x) = \sum_{k=0}^{+\infty} \frac{(-1)^k x^{2k+1}}{2k+1}$$

Wilhelm Leibniz 1682 : critère spécial des séries alternées :

$$\left| Arctan(x) - \sum_{k=0}^{n} \frac{(-1)^k x^{2k+1}}{2k+1} \right| < \frac{x^{2n+3}}{2n+3}$$

Valeur de n telle que $\frac{a^{2n+3}}{2n+3} \leqslant 10^{-d}$?

$$\left(2 + \frac{3}{n}\right) \log \frac{1}{a} \geqslant \frac{d}{n} - \frac{\log(2n+3)}{n}$$

James Gregory 1671 :

$$Arctan(x) = \sum_{k=0}^{+\infty} \frac{(-1)^k x^{2k+1}}{2k+1}$$

Wilhelm Leibniz 1682 : critère spécial des séries alternées :

$$\left| Arctan(x) - \sum_{k=0}^{n} \frac{(-1)^k x^{2k+1}}{2k+1} \right| < \frac{x^{2n+3}}{2n+3}$$

Valeur de n telle que $\frac{a^{2n+3}}{2n+3} \leqslant 10^{-d}$?

$$\left(2 + \frac{3}{n}\right) \log \frac{1}{a} \geqslant \frac{d}{n} - \frac{\log(2n+3)}{n}$$

John Machin 1706 :

$$4 \arctan \frac{1}{5} - \arctan \frac{1}{239} = \frac{\pi}{4}$$

Polynôme de Gregory

$$G_n(X) = \sum_{k=0}^{n} \frac{(-1)^k X^{2k+1}}{2k+1}$$

John Machin 1706 :

$$4 \arctan \frac{1}{5} - \arctan \frac{1}{239} = \frac{\pi}{4}$$

Polynôme de Gregory

$$G_n(X) = \sum_{k=0}^{n} \frac{(-1)^k X^{2k+1}}{2k+1}$$

```
def greg(a,N):
    ad = Decimal('1')/Decimal(str(a))
    nd = ad
    dd = Decimal('1')
    s  = nd/dd
    for k in range(1,N):
        nd *= Decimal('-1')*ad*ad
        dd += Decimal('2')
        s  += nd/dd
    return s
```

```
def pi_machin(d):
    getcontext().prec = d+2
    rap=2*log(5)/log(10)
    N=int(d/rap)+1
    return 4*(4*greg(5,N)-greg(239,N))
```

```
def greg(a,N):
    ad = Decimal('1')/Decimal(str(a))
    nd = ad
    dd = Decimal('1')
    s  = nd/dd
    for k in range(1,N):
        nd *= Decimal('-1')*ad*ad
        dd += Decimal('2')
        s  += nd/dd
    return s
```

```
def pi_machin(d):
    getcontext().prec = d+2
    rap=2*log(5)/log(10)
    N=int(d/rap)+1
    return 4*(4*greg(5,N)-greg(239,N))
```

```
>>> PI-pi_machin(1000)
Decimal('-1.5E-1000')
```

```
>>> pi_machin(1000)
Decimal('3.14159265358979323846264338327950288419716939937510582097494459230781
6208998628034825342117067982148086513282306647093844609550582231725359408128481
2841027019385211055596446229489549303819644288109756659334461284756482337867831
0190914564856692346034861045432664821339360726024914127372458700660631558817488
0962829254091715364367892590360011330530548820466521384146951941511609433057270
5919530921861173819326117931051185480744623799627495673518857527248912279381830
2983367336244065664308602139494639522473719070217986094370277053921717629317675
4818467669405132000568127145263560827785771342757789609173637178721468440901224
1465495853710507922796892589235420199561121290219608640344181598136297747713099
0721134999999837297804995105973173281609631859502445945534690830264252230825334
5261931188171010003137838752886587533208381420617177669147303598253490428755468
5628638823537875937519577818577805321712268066130019278766111959092164201991 5')
```

```
>>> PI-pi_machin(1000)
Decimal('-1.5E-1000')
```

```
>>> pi_machin(1000)
Decimal('3.14159265358979323846264338327950288419716939937510582097494459230781
62089986280348253421170679821480865132823066470938446095505822317253594081284811
28410270193852110555964462294895493038196442881097566593344612847564823378678310
01909145648566923460348610454326648213393607260249141273724587006606315588174881
09628292540917153643678925903600113305305488204665213841469519415116094330572703
59195309218611738193261179310511854807446237996274956735188575272489122793818301
29833673362440656643086021394946395224737190702179860943702770539217176293176752
48184676694051320005681271452635608277857713427577896091736371787214684409012249
14654958537105079227968925892354201995611212902196086403441815981362977477130990
07211349999998372978049951059731732816096318595024459455346908302642522308253344
52619311881710100031378387528865875332083814206171776691473035982534904287554681
56286388235378759375195778185778053217122680661300192787661119590921642019915')
```

Et sinon :