Chapitre 1 : Introduction à L'Analyse Numérique

...

1. ## Définition

L'analyse numérique est la conception et l'étude d'algorithmes pour obtenir des solutions à des ensembles d'équations issus de modèles issus de la physique, de la biologie, de la finance ...

2. ## Motivations :

- Recherche et développement : études expérimentales coûteuses
- Les modèles considérés sont composés d'ensemble d'équations dont on ne sait pas déterminer de solutions explicites
- Proposer une solution approchée, calculée à l'aide de l'ordinateur.

3. ## Développer des algorithmes efficaces

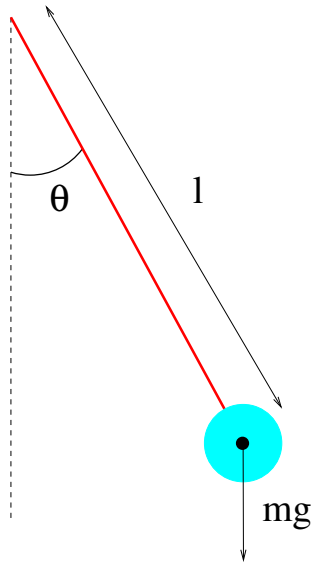
- Convergence et stabilité de la méthode numérique
- Coût algorithmique

Plan du cours

1. Introduction à l'analyse numérique
2. Interpolation et approximation
3. Intégration numérique
4. Résolution de systèmes linéaires
5. Equations non linéaires
6. Equations différentielles ordinaires
7. Equations aux dérivées partielles

Quelques exemples

Mouvement du pendule



- Equation

$$\begin{cases} \theta''(t) + \frac{g}{l}\sin(\theta(t)) = 0, \\ \theta(0) = \theta_0, \theta'(0) = \theta_1 \end{cases}$$

- Solution exacte ?

- Solution approchée pour $\theta << 1$

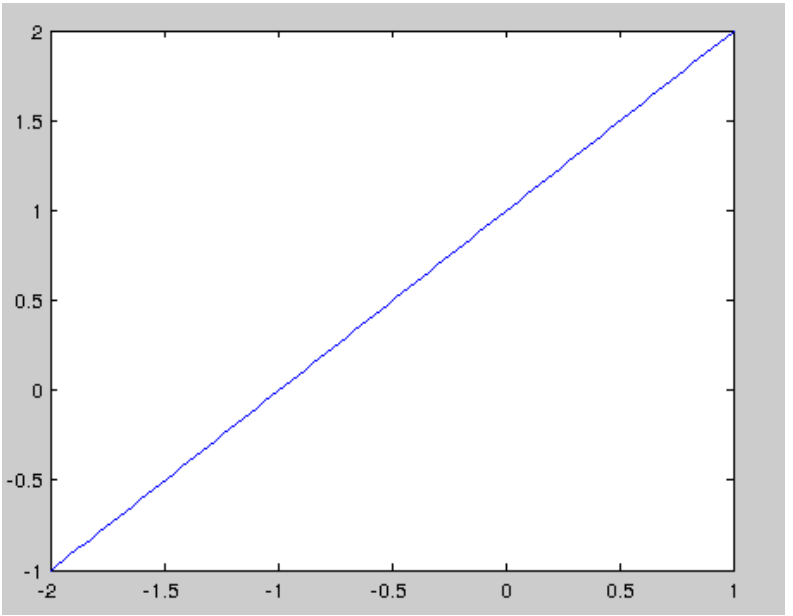
$$\begin{cases} \theta''(t) + \frac{g}{l}\theta(t) = 0, \\ \theta(t) = \theta_0 \cos\left(\sqrt{\frac{g}{l}}t\right) + \sqrt{\frac{l}{g}}\theta_1 \sin\left(\sqrt{\frac{g}{l}}t\right) \end{cases}$$

- Utiliser une approximation numérique de la solution !

Quelques exemples

Calculer les racines du polynôme $p(x) = ax^2 + bx + c$

```
>> a = 10^(-20);
>> b = 1;
>> c=1;
>> x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a)

x1 =

     0

>> x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a)

x2 =

  -1.0000e+20
```



Quelques exemples

Temps de calcul pour l'inversion d'une matrice

```
>>
>> N = 2^10

N =

        1024

>> N = 2^10;
>> A = rand(N,N);
>> tic;
A^(-1);
toc
Elapsed time is 1.033932 seconds.
>> B = rand(2*N,2*N);
>> tic;
B^(-1);
toc
Elapsed time is 9.313878 seconds.
>>
```

1 Représentation des réels sur l'ordinateur

2 Conditionnement, stabilité et complexité

Représentation des réels en base b

- Tous nombre réel x peut être représenté sous la forme

$$x = \pm m\, b^{\pm e}, \quad \text{avec} \quad b \geq 2.$$

  - avec la mantisse m :

$$m = m_1 b^{-1} + m_2 b^{-2} + ..., \quad \text{et} \quad m_i \in \{0, 1, ..., b-1\}$$

  - et l'exposant e :

$$e = e_0 b^0 + e_1 b^1 + ... e_{s-1} b^{s-1}, \quad \textit{avec} \quad s \in \mathbb{N}$$

- Représentation unique ?
- Ordinateur : mémoire finie !
  - La mantisse est tronquée au bout de r chiffres,
  - Valeur maximale pour s

Le standard IEE (754-1985)

- Float double précision : utilisation de 8 octets (64 bits), $b = 2$

- Pour la mantisse : utilisation de 52 bits

$$m = 2^{-1} + m_2 2^{-2} + m_3 2^{-3} + ... + m_{53} 2^{-53},$$

- Pour l'exposant : 11 bits : $e \in [-1022, 1025]$ avec

$$e = c_0 2^0 + c_1 2^1 + ... + c_{10} 2^{10} - 1022, \quad \text{avec} \quad c_i \in \{0, 1\}$$

- Les valeurs $e = -1022$ et $e = 1025$ sont réservées à la représentation de 0 et de Inf
- On appelle $\mathbb{F}$ l'ensemble fini de ces nombres

Exercices

Exercice

Calculer x_{max} le nombre le plus grand de $\mathbb{F}$ et x_{posmin}, le nombre positif le plus petit de $\mathbb{F}$

Exercice

Proposer deux algorithmes pour déterminer respectivement une approximation numérique de x_{max} et x_{posmin}

Erreur d'arrondi

- Seuls les éléments de $\mathbb{F}$ sont autorisés
- Un nombre réel $x \pm m\, 2^e$, $m = 0.1 m_2 m_3 ...$ pour $x_{posmin} < |x| < x_{max}$ est arrondi de la manière suivante

$$rd(x) = sign(x) \begin{cases} 0.1 m_2 ... m_{53} 2^e & \text{si} \quad m_{54} = 0, \\ (0.1 m_2 ... m_{53} + 2^{-53}) 2^e & \text{si} \quad m_{54} = 1, \end{cases}$$

Exercice

Calculer le nombre le plus petit tel que $rd(x+1) > 1$ et proposer un algorithme pour retrouver numériquement ce nombre

Erreur arrondi

- Si $x_{posmin} < |x| < x_{maxx}$, alors
  - L'erreur d'arrondi absolue est

$$|x - rd(x)| \leq \frac{1}{2} 2^{-53} 2^e.$$

  - L'erreur d'arrondi relative est

$$\left| \frac{x - rd(x)}{x} \right| \leq \frac{1}{2} \frac{2^{-53} 2^e}{|m| 2^e} \leq 2^{-53} \simeq 10^{-16}$$

1. Représentation des réels sur l'ordinateur

2. Conditionnement, stabilité et complexité

Conditionnement d'un problème numérique

Definition

Le conditionnement représente la sensibilité du résultat par rapport à de petites variations des données ...
On dit qu'un problème est
- *bien conditionné* si une petite variation des données entraîne une petite variation du résultat
- *mal conditionné* si une petite variation des données entraîne une grande variation du résultat

Exemple : soit $f : \mathbb{R} \to \mathbb{R}$

- Développement de Taylor

$$f(x + \delta_x) = f(x) + f'(x) * \delta_x + o(\delta_x)$$

$$\delta f := f(x + \delta_x) - f(x) \simeq f'(x)\delta_x$$

- Conditionnement

$$k = \frac{\delta f}{f}\frac{x}{\delta_x} \simeq \frac{xf'(x)}{f(x)}$$

Conditionnement des opérations arithmétiques élémentaires

- Soit $f \in C^2(\mathbb{R}^n, \mathbb{R})$. Quel est le conditionnement de f ?
- D'après le développement de Taylor de f,

$$\delta f = f(x + dx) - f(x) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \delta_{x_i} + 0(|\delta x|^2).$$

- Le conditionnement est donc déterminé par les nombres

$$k_i = \frac{\partial f}{\partial x_i} \frac{x_i}{f(x)}.$$

Exercice

Calculer le conditionnement de l'addition et de la multiplication

Stabilité d'un algorithme

Definition

La stabilité d'un algorithme se réfère à la propagation des erreurs au cours des étapes du calcul, à la capacité de l'algorithme de ne pas trop amplifier d'éventuels écarts, à la précision des résultats obtenus.

Exemple : l'algorithme qui calcule les racines du polynôme $p(x) = ax^2 + bx + c$ basé sur les formules

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

s'avère instable en pratique : exemple avec $a = 10^{-20}, b = 1, c = 1$

Exercice

Proposer un autre algorithme plus stable

Complexité algorithmique

Definition (Coût algorithmique)

Nombre d'opérations élémentaires effectuées par l'algorithme (+, *, sqrt, puissance ...)

Exemple : Evalutation du polynôme $p(x) = \sum_{i=0}^{n} a_i x^i$:

- Nombre de multiplications : $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$
- Nombre d'additions : n
- Coût algorithmique $\rightarrow O(n^2)$

Definition

$$\begin{cases} O(n^k) \rightarrow \text{coût polynomial} \\ O(a^n) \rightarrow \text{coût exponentiel} \\ O(n!) \rightarrow \text{coût factoriel} \end{cases}$$

Complexité algorithmique

Exemple : Evalutation du polynôme $p(x) = \sum_{i=0}^{n} a_i x^i$:

- Schéma de Horner

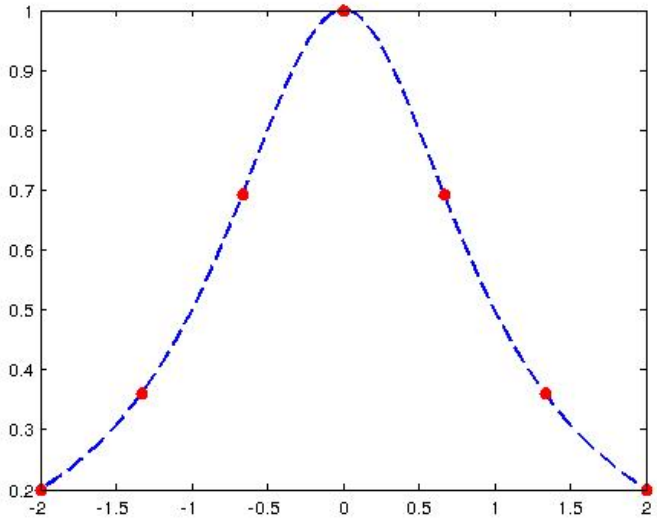
$$p(x) = a_0 + x\,(a_1 + x\,(a_2 + x\,(...a_n)))$$

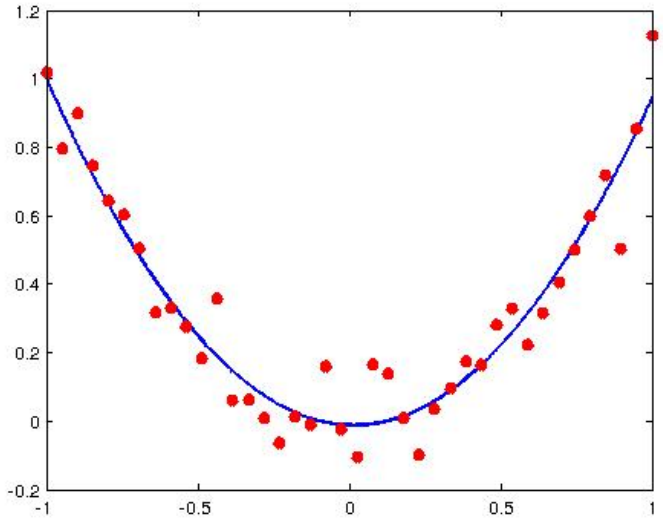
- Coût algorithmique ?

Chapitre 2 : Interpolation et approximation

...

A partir d'un ensemble de points $\{(x_i, y_i)\}_{i=0:n}$, on recherche dans un espace de fonctions,





Interpolation : une fonction s qui interpole les noeuds (x_i, y_i),

$$s(x_i) = yi, \quad \forall i = 0 : n$$

Approximation au sens des moindres carrés : une fonction s qui minimise l'énergie,

$$\sum_{i=0}^{n} |y_i - s(x_i)|^2$$

Exemples d'interpolation :

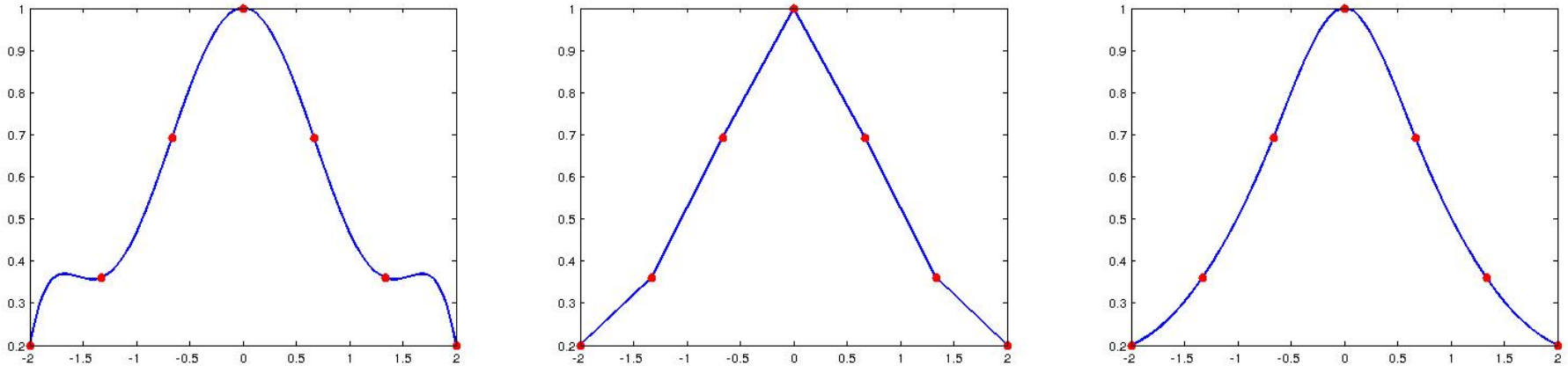


fig: Interpolation polynomiale, linéaire par morceaux et spline

Exemples d'approximation au sens des moindres carrés :

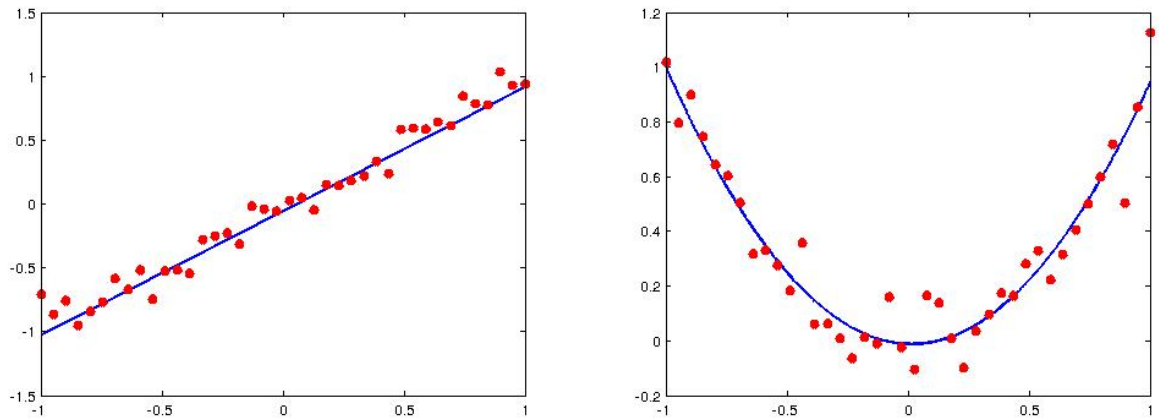


fig: Régression linéaire, quadratique

1. Interpolation polynomiale
2. Interpolation polynomiale par morceaux
3. Approximation au sens des moindres carrés

Interpolation polynomiale :

Définition

On note P_n l'ensemble des polynômes réels de degré n

$$P_n = \left\{ p(x) \; ; \; p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \right\}$$

avec $a_i \in \mathbb{R}$

On recherche un polynôme $p_n \in P_n$ tel que pour tout $i = 0 : n$,

$$p_n(x_i) = y_i.$$

Théorème

Il existe un unique polynôme $p_n \in P_n$ qui interpole les noeuds $\{(x_i, y_i)\}_{i=0:n}$.

Polynôme d'interpolation de Vandermonde :

On souhaite trouver un polynôme de degré 2 qui interpole les noeuds $\{(-1,2),(0,3),(1,6)\}$.

On cherche le polynôme p_2 sous la forme $p_2(x) = a_0 + a_1 x + a_2 x^2$ tel que $p_2(-1) = 2$, $p_2(0) = 3$ et $p_2(1) = 6$. Alors

$$\begin{cases} p_2(-1) = 2 \\ p_2(0) = 3 \\ p_2(1) = 6 \end{cases} \Longrightarrow \begin{cases} a_0 - a_1 + a_2 = 2 \\ a_0 = 3 \\ a_0 + a_1 + a_2 = 6 \end{cases} \Longrightarrow \begin{cases} a_0 = 3 \\ a_1 = 2 \\ a_2 = 1 \end{cases}$$

La solution du problème est donc $p_2(x) = 3 + 2x + x^2$

Polynôme d'interpolation de Vandermonde :

Plus généralement, P_n est un espace vectoriel dont la base canonique s'écrit $\left\{1, X, X^2, ..., X^n\right\}$ et

$$p_n(x) = \sum_{k=0}^{n} a_k x^k.$$

Alors

$$\begin{cases} p_n(x_0) & = y_0 \\ p_n(x_1) & = y_1 \\ \vdots & \\ p_n(x_n) & = y_n \end{cases} \Longrightarrow \begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + ... + a_n x_0^n & = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + ... + a_n x_1^n & = y_1 \\ \vdots & \\ a_0 + a_1 x_n + a_2 x_n^2 + ... + a_n x_n^n & = y_n \end{cases}$$

Polynôme d'interpolation de Vandermonde :

$$\begin{cases} p(x_0) & = y_0 \\ p(x_1) & = y_1 \\ \vdots & \\ p(x_n) & = y_n \end{cases} \Longrightarrow \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Les coefficients a_i du polynôme d'interpolation s'obtiennent donc en résolvant le système linéaire suivant

$$Ba = y,$$

où $(B)_{i,j} = x_j^{i-1}$, $(a)_i = a_i$ et $(y)_i = y_i$.

Propriété

Le déterminant de la matrice B vérifie

$$det(B) = \prod_{0 \leq i < j \leq n} (x_j - x_i).$$

Polynôme d'interpolation de Vandermonde :

Le polynôme d'interpolation p_n s'obtient donc par la résolution du système linéaire $Ba = y$! Mais en pratique

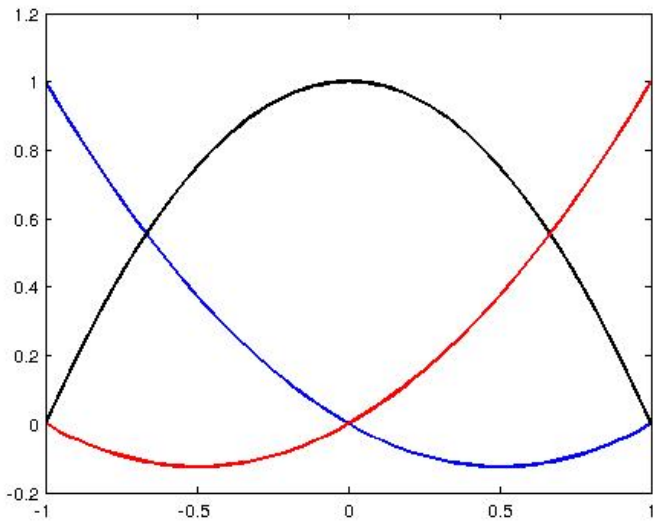
- B est une matrice mal conditionnée !
- Coût de la résolution du système linéaire en $O(n^3)$!

Idée : Exprimer le polynôme p_n dans une autre base de P_n et pour laquelle, la décomposition de p_n est explicite !

Polynôme d'interpolation de Lagrange

On souhaite trouver le polynôme d'interpolation associé aux noeuds $\{(-1,2),(0,3),(1,6)\}$.
On introduit alors les trois polynômes $\mathcal{L}_0$, $\mathcal{L}_1$ et $\mathcal{L}_2$ définis par

$$\mathcal{L}_0(x) = \frac{1}{2}x(x-1), \quad \mathcal{L}_1(x) = 1 - x^2, \quad \text{et} \quad \mathcal{L}_2(x) = \frac{1}{2}x(1+x),$$



qui vérifient

$$\begin{cases} \mathcal{L}_0(-1) = 1, \mathcal{L}_0(0) = 0, \mathcal{L}_0(1) = 0, \\ \mathcal{L}_1(-1) = 0, \mathcal{L}_1(0) = 1, \mathcal{L}_1(1) = 0, \\ \mathcal{L}_2(-1) = 0, \mathcal{L}_2(0) = 0, \mathcal{L}_2(1) = 1, \end{cases}$$

Exercice

Montrer que la famille $\{\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2\}$ est libre et expliciter la décomposition de p_2 dans cette base

Polynôme d'interpolation de Lagrange

Plus généralement, on introduit une base de P_n notée $\left\{\mathcal{L}_k^n\right\}_{k=0:n}$, où les polynômes $\mathcal{L}_k^n$ sont définis par la propriété suivante :

$$\mathcal{L}_k^n(x_i) = \begin{cases} 1 & \text{si } k = i \\ 0 & \text{sinon} \end{cases}$$

Exercice

Montrer que le polynôme $\mathcal{L}_k^n$ s'explicite sous la forme

$$\mathcal{L}_k^n(x) = \frac{\prod_{i=0, i\neq k}^n (x - x_i)}{\prod_{i=0, i\neq k}^n (x_k - x_i)}.$$

Exercice

Montrer que le polynôme d'interpolation p_n au noeuds $\{(x_i, y_i)\}_{i=0:n}$ vérifie

$$p_n = \sum_{i=0}^n y_i \mathcal{L}_i^n$$

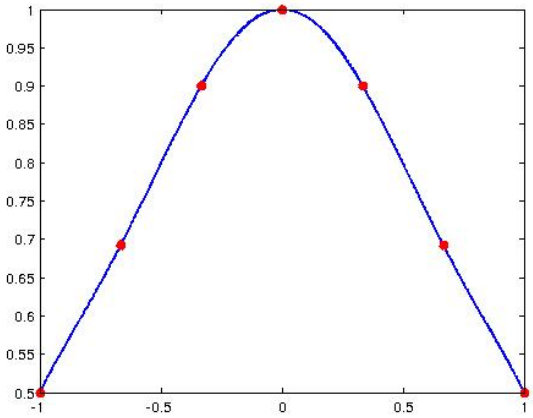
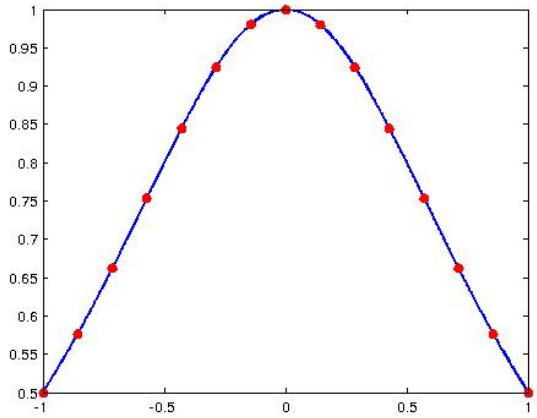
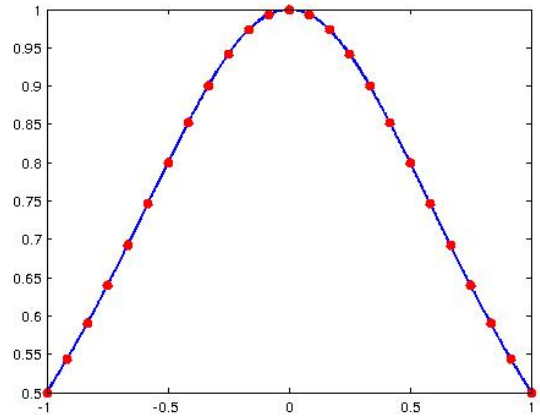




fig: Interpolation polynomiale de la fonction $f(x) = \frac{1}{1+x^2}$ sur $[-1, 1]$ avec respectivement $n = 6$, $n = 14$ et $n = 24$

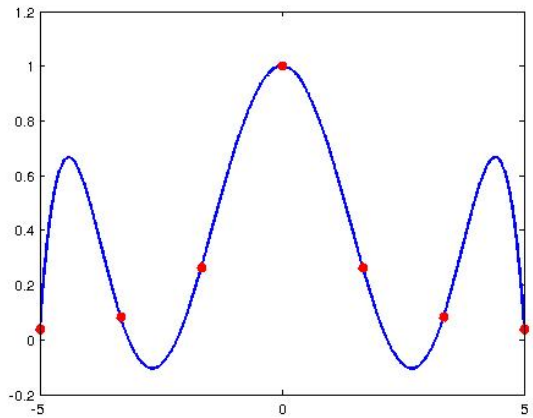
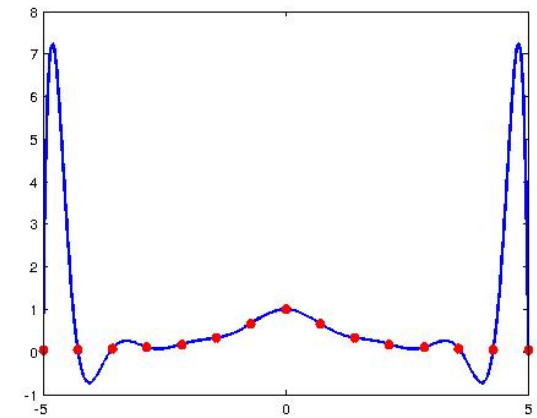
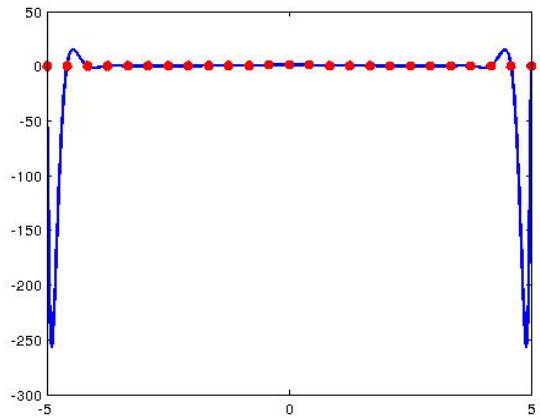




fig: Interpolation polynomiale de la fonction $f(x) = \frac{1}{1+x^2}$ sur $[-5, 5]$ avec respectivement $n = 6$, $n = 14$ et $n = 24$

Erreur d'interpolation

On suppose que $f : [a, b] \to \mathbb{R}$ est une fonction suffisamment régulière.

On vient d'expliquer comment obtenir le polynôme d'interpolation p_n au noeuds $\{(x_i, f(x_i))\}_{i=0:n}$.

On souhaite maintenant donner une estimation de l'erreur $E(x)$ commise entre p_n et f :

$$E(x) = f(x) - p_n(x).$$

Erreur d'interpolation

Théorème

Soit $f \in C^{n+1}[a,b]$ *avec* $a = x_0 < x_1 < x_2 ... < x_n = b$. *Alors* $\forall x \in [a,b]$, *il existe* $\xi_x \in [a,b]$ *tel que*

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

Démonstration : Appliquer le théorème de Rolle à la fonction ϕ définie par

$$\phi(t) = f(t) - P(t) - \frac{(f(x) - p_n(x)) \prod_{i=0}^{n} (t - x_i)}{\prod_{i=0}^{n} (x - x_i)}$$

Erreur d'interpolation

Si les noeuds sont équirépartis sur $[a, b]$ alors

$$\left| \prod_{i=0}^{n} (x - x_i) \right| \leq \frac{n!}{4} h^{n+1},$$

où $h = (b - a)/n$, et

$$|E| \leq \sup_{x \in [a,b]} \left\{ \left| f^{(n+1)}(x) \right| \right\} \left(\frac{b-a}{n} \right)^{n+1} \frac{1}{4(n+1)}.$$

A noter, rien n'assure que $E \to 0$ lorsque $n \to \infty$!

Exercice

Montrer le résultat précédent

Noeuds d'interpolation de tchebychev

Idée : Optimiser la position des noeuds $x_i \in [a, b]$ afin de minimiser

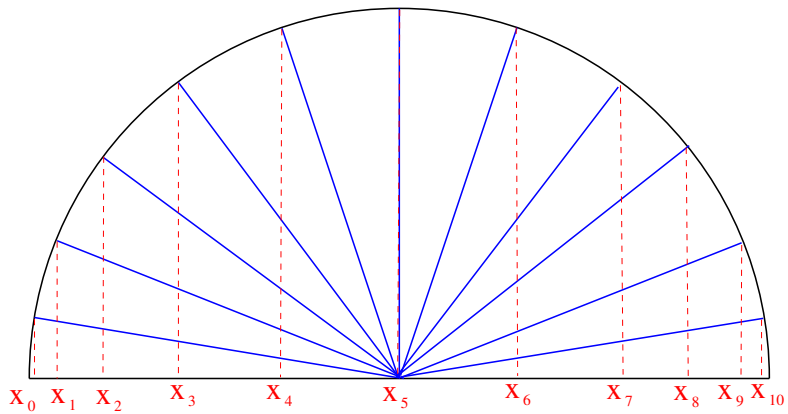
$$\max_{x\in[a,b]} \left| \prod_{i=0}^{n} (x - x_i) \right|$$

Solution :

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}_i,$$

où

$$\hat{x}_i = \cos\left(\frac{\pi}{2}\frac{2i+1}{n+1}\right)$$



On obtient alors l'estimation

$$\max |E(x)| \leq \frac{1}{(n+1)!} \frac{(b-a)^{n+1}}{2^{2n+1}} \sup_{x\in[a,b]} \left| f^{(n+1)}(x) \right|,$$

et $\lim_{n\to\infty} E = 0$.

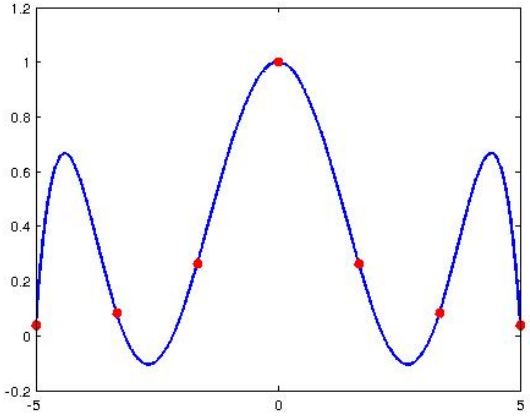
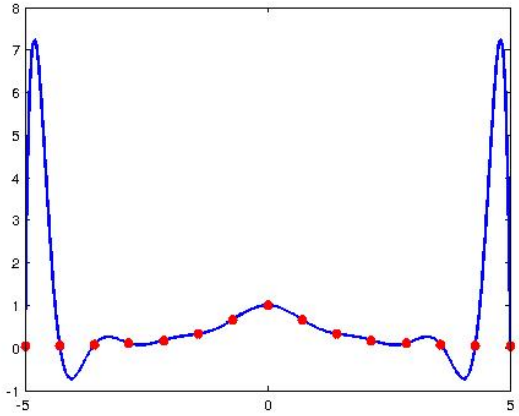
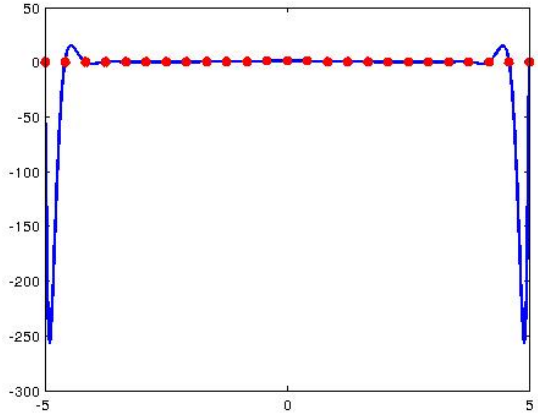




fig: Interpolation polynomiale de la fonction $f(x) = \frac{1}{1+x^2}$ sur $[-5, 5]$ avec respectivement $n = 6$, $n = 14$ et $n = 24$ et noeuds equirépartis

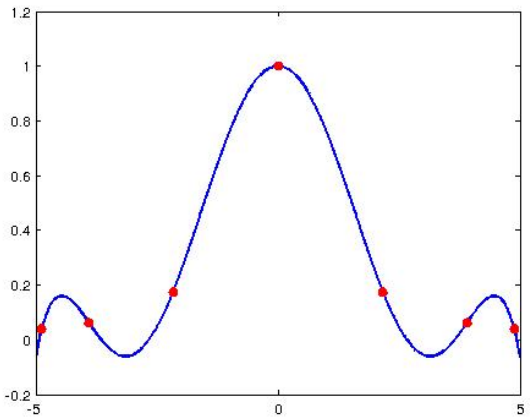
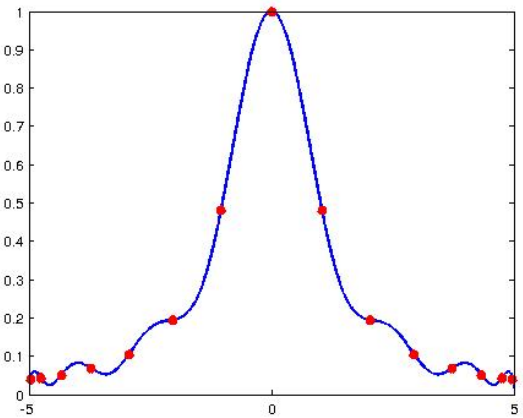
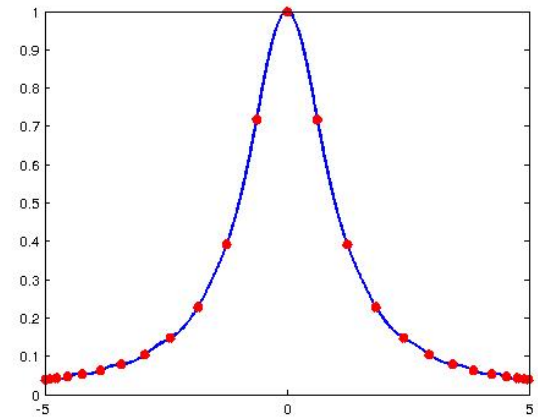




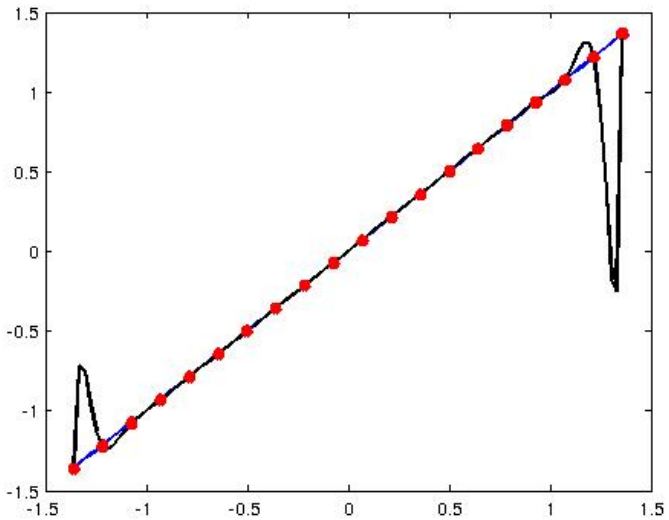
fig: Interpolation polynomiale de la fonction $f(x) = \frac{1}{1+x^2}$ sur $[-5, 5]$ avec respectivement $n = 6$, $n = 14$ et $n = 24$ et noeuds d'interpolation de tchebychev

1. Interpolation polynomiale
2. Interpolation polynomiale par morceaux
3. Approximation au sens des moindres carrés

Interpolation polynomiale par morceaux

Motivation :

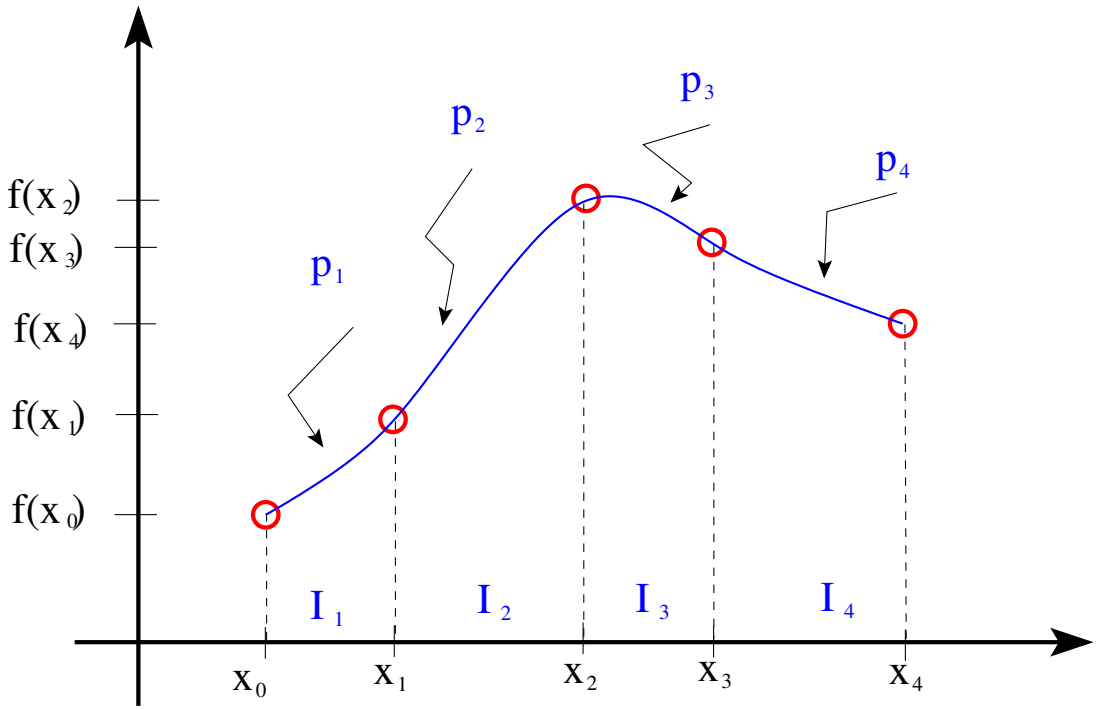
- L'interpolation polynomiale a tendance à produire des oscillations lorsque le degré des polynômes est trop élevé : c'est le phénomène de Runge ...
- -> Utiliser des polynômes de degré plus petit sur des sous-intervalles de $[a, b]$



Interpolation polynomiale par morceaux

- Noeuds d'interpolation $a = x_0 < x_1 < x_2 < ... < x_n = b$
- On note $I_i = [x_{i-1}, x_i]$ et $h_i = |I_i| = |x_i - x_{i-1}|$ pour tout $i = 1 : n$
- On recherche une interpolation de f dans les espaces $\mathcal{S}_h^{k,r}$ définis par

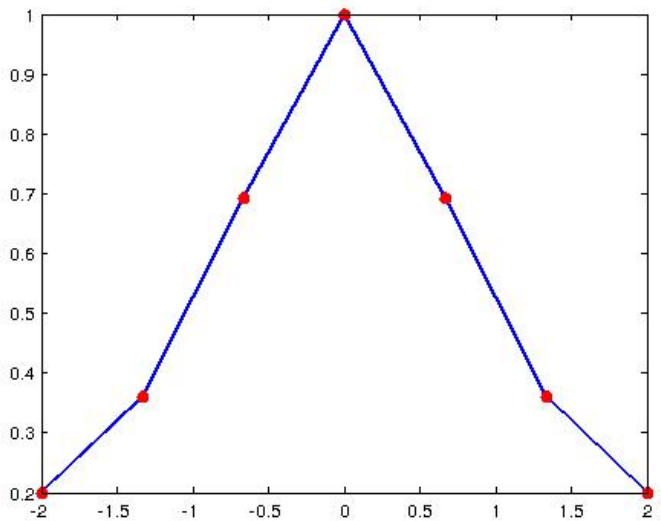
$$\mathcal{S}_h^{k,r}([a,b]) = \{s \in C^r([a,b]) \; ; \; s_{|I_i} \in P_k(I_i), \; \forall i = 1 : n\}$$



Interpolation linéaire par morceaux

- Espace $\mathcal{S}_h^{k,r}([a,b])$ avec $k = 1$ et $r = 0$
- Espace vectoriel de dimension $n + 1$
  - $2n$ coefficients associés aux n polynômes (sur chaque intervalle I_i)
  - $n - 1$ contraintes pour la continuité de s : $p_i(x_i) = p_{i+1}(x_i)$.
- Solution explicite

$$s_{|I_i}(x) = y_{i-1}\left(\frac{x - x_i}{x_{i-1} - x_i}\right) + y_i\left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)$$



Exercice

Avec ϕ_i définie par

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & si \quad x \in [x_{i-1}, x_i] \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & si \quad x \in [x_i, x_{i+1}] \\ 0 & sinon \end{cases}$$

Montrer que $\{\phi_i\}_{i=0:n}$ forme une base de $\mathcal{S}_h^{1,0}([a,b])$ et que

$$s = \sum_{i=0}^{n} y_i \phi_i.$$

Exercice

Soit $f \in C^2([a,b])$, montrer que

$$\max_{x \in [a,b]} \left| s(x) - f(x) \right| \leq \frac{1}{8} \max_{i=1:n} |h_i|^2 \max_{x \in [a,b]} \{|f''(x)|\}$$

Interpolation Spline

Motivation : On souhaite plus de régularité...
En particulier, on recherche parmi toutes les fonctions $C^2([a,b])$ qui interpolent f aux noeuds $\{x_i\}_{i=0:n}$, celle qui minimise l'énergie élastique

$$J(s) = \int_a^b \left|s''(x)\right|^2 dx.$$

Definition

La fonction s définie sur $[a,b]$ est une spline cubique (par rapport à la subdivision $a = x_0 < x_1 < ... < x_n$) si $s \in \mathcal{S}^{3,2}([a,b])$. Si de plus $s''(a) = s''(b) = 0$, alors s est une spline cubique naturelle.

Interpolation Spline

- L'espace vectoriel $\mathcal{S}^{3,2}([a,b])$ est de dimension $n+3$
  - n polynômes de degré 3 $\rightarrow 4n$ coefficients
  - contrainte continuité $\rightarrow n-1$ conditions
  - contrainte dérivée continue $\rightarrow n-1$ conditions
  - contrainte dérivée seconde continue $\rightarrow n-1$ conditions
- Il faut donc imposer 2 conditions supplémentaires pour espérer avoir l'unicité de l'interpolation spline

Théorème

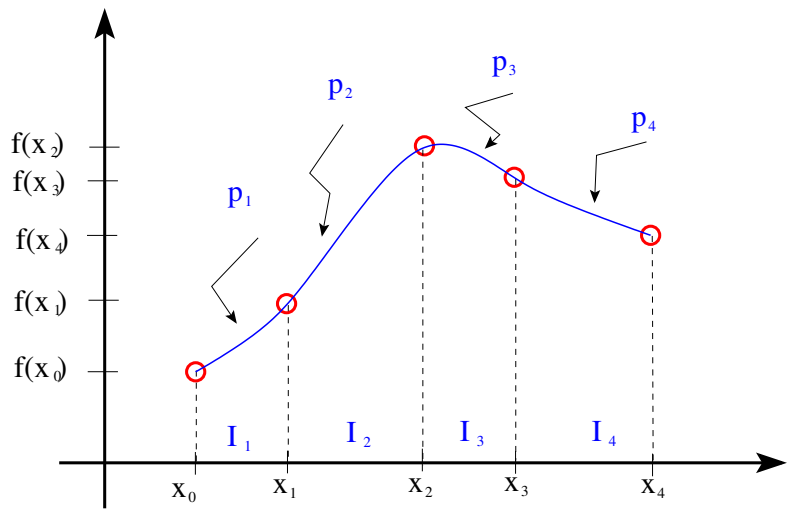
Il existe une spline cubique qui interpole les noeuds $\{(x_i, y_i)\}_{i=0:n}$. Si de plus les coefficients $s''(a)$ et $s''(b)$ sont fixés, alors cette spline est unique.

Une expression explicite de la spline interpolante s'obtient en résolvant un système linéaire de taille $n+1$. Pour simplifier les calculs, nous supposons que la répartition des noeuds est uniforme, c'est à dire

$$h_i = |x_i - x_{i-1}| = h = (b-a)/n$$

La restriction de s à I_i est un polynôme de degré 3 noté p_i :

$$s_{|I_i} = p_i \in P^3(I_i)$$



Sur chaque intervalle I_i, la spline vérifie

$$s_{|I_i}(x) = p_i(x) = y_i'' \frac{(x - x_{i-1})^3}{6h} - y_{i-1}'' \frac{(x - x_i)^3}{6h} + \left(y_i - \frac{h^2}{6} y_i''\right) \frac{(x - x_{i-1})}{h} - \left(y_{i-1} - \frac{h^2}{6} y_{i-1}''\right) \frac{(x - x_i)}{h}$$

où les coefficients y_i'' s'obtiennent par la résolution du système linéaire suivant

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & 4 & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0'' \\ y_1'' \\ \vdots \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{pmatrix} = \begin{pmatrix} s''(a) \\ 6\left[\frac{y_2 - y_1}{h} - \frac{y_1 - y_0}{h}\right]/h \\ \vdots \\ \vdots \\ 6\left[\frac{y_n - y_{n-1}}{h} - \frac{y_{n-1} - y_{n-2}}{h}\right]/h \\ s''(b) \end{pmatrix}$$

Continuité de la dérivée seconde de s

La dérivée seconde du polynôme p_i'' est un polynôme de degré 1, qu'on exprime sous la forme

$$p_i''(x) = y_{i-1}'' \frac{(x - x_i)}{(x_{i-1} - x_i)} + y_i'' \frac{(x - x_{i-1})}{(x_i - x_{i-1})},$$

où les coefficients $y_i'' = s''(x_i)$ sont pour l'instant indéterminés à l'exception de $y_0'' = s''(a)$ et $y_n'' = s''(b)$.

En intégrant 2 fois, on obtient

$$p_i(x) = y_i'' \frac{(x - x_{i-1})^3}{6h} - y_{i-1}'' \frac{(x - x_i)^3}{6h} + q_i(x),$$

où $q_i(x) \in P_1(I_i)$.
Dans la suite, on recherche q_i sous la forme

$$q_i(x) = \alpha_i \frac{(x - x_{i-1})}{h} - \beta_i \frac{(x - x_i)}{h}.$$

Conditions d'interpolation $s(x_i) = y_i$

On rappelle que

$$p_i(x) = y_i'' \frac{(x - x_{i-1})^3}{6h} - y_{i-1}'' \frac{(x - x_i)^3}{6h} + \alpha_i \frac{(x - x_{i-1})}{h} - \beta_i \frac{(x - x_i)}{h}.$$

Alors

$$\begin{cases} p_i(x_i) & = y_i \\ p_i(x_{i-1}) & = y_{i-1} \end{cases} \Longrightarrow \begin{cases} \alpha_i & = y_i - \frac{h^2}{6} y_i'' \\ \beta_i & = y_{i-1} - \frac{h^2}{6} y_{i-1}'' \end{cases}$$

Et

$$\begin{aligned} p_i(x) = y_i'' \frac{(x - x_{i-1})^3}{6h} & - y_{i-1}'' \frac{(x - x_i)^3}{6h} + \left(y_i - \frac{h^2}{6} y_i'' \right) \frac{(x - x_{i-1})}{h} \\ & - \left(y_{i-1} - \frac{h^2}{6} y_{i-1}'' \right) \frac{(x - x_i)}{h} \end{aligned}$$

Continuité de la dérivée première de s

Avec

$$\begin{cases} p_i'(x) & = y_i'' \frac{(x-x_{i-1})^2}{2h} - y_{i-1}'' \frac{(x-x_i)^2}{2h} + \left(\frac{(y_i-y_{i-1})}{h} + \frac{h}{6}(y_{i-1}'' - y_i'') \right) \\ p_{i+1}'(x) & = y_{i+1}'' \frac{(x-x_i)^2}{2h} - y_i'' \frac{(x-x_{i+1})^2}{2h} + \left(\frac{(y_{i+1}-y_i)}{h} + \frac{h}{6}(y_i'' - y_{i+1}'') \right) \end{cases}$$

Et

$$\begin{cases} p_i'(x_i) & = y_i'' \frac{h}{2} + \left(\frac{(y_i-y_{i-1})}{h} + \frac{h}{6}(y_{i-1}'' - y_i'') \right) \\ p_{i+1}'(x_i) & = -y_i'' \frac{h}{2} + \left(\frac{(y_{i+1}-y_i)}{h} + \frac{h}{6}(y_i'' - y_{i+1}'') \right) \end{cases}$$

L'égalité $p_i'(x_i) = p_{i+1}'(x_i)$ implique que

$$y_{i-1}'' + 4y_i'' + y_{i+1}'' = 6 \left[\frac{\frac{y_{i+1}-y_i}{h} - \frac{y_i-y_{i-1}}{h}}{h} \right]$$

En conclusion,

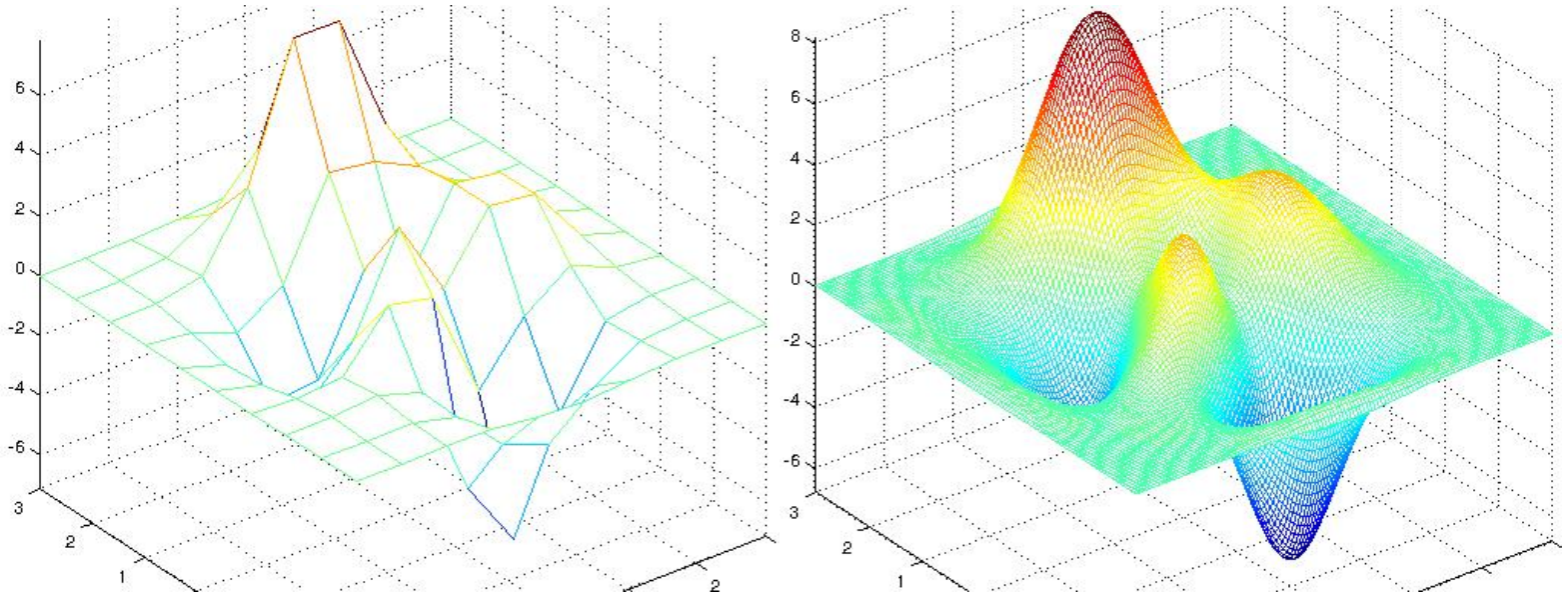
les coefficients y_i'' s'obtiennent donc en résolvant le système linéaire suivant

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & 4 & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0'' \\ y_1'' \\ \vdots \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{pmatrix} = \begin{pmatrix} s''(a) \\ 6\left[\frac{y_2-y_1}{h} - \frac{y_1-y_0}{h}\right]/h \\ \vdots \\ \vdots \\ 6\left[\frac{y_n-y_{n-1}}{h} - \frac{y_{n-1}-y_{n-2}}{h}\right]/h \\ s''(b) \end{pmatrix}$$

Et sur chaque intervalle I_i, la spline vérifie

$$\begin{aligned} s_{|I_i}(x) = y_i'' \frac{(x-x_{i-1})^3}{6h} &- y_{i-1}'' \frac{(x-x_i)^3}{6h} + \left(y_i - \frac{h^2}{6} y_i''\right) \frac{(x-x_{i-1})}{h} \\ &- \left(y_{i-1} - \frac{h^2}{6} y_{i-1}''\right) \frac{(x-x_i)}{h} \end{aligned}$$

Exemple d'interpolation spline 2d



Théorème (Erreur d'interpolation)

Soit $f \in C^4[a,b]$ *avec* $a = x_0 < x_1 < x_2 ... < x_n = b$. *Alors*

$$\max_{x\in[a,b]} \left|f(x) - s(x)\right| \leq \frac{1}{2} h^4 \max_{x\in[a,b]} \left|f^{(4)}(x)\right|$$

Théorème (Minimisation d'énergie)

Soit $f \in C^2[a,b]$ telle que $f(x_i) = y_i$. Alors la spline cubique naturelle s vérifie

$$\int_a^b (s''(x))^2 dx \leq \int_a^b (f''(x))^2 dx$$

Démonstration : Montrer que

$$\int_a^b \left(f''(x)s''(x) - s''(x)^2\right) dx = 0,$$

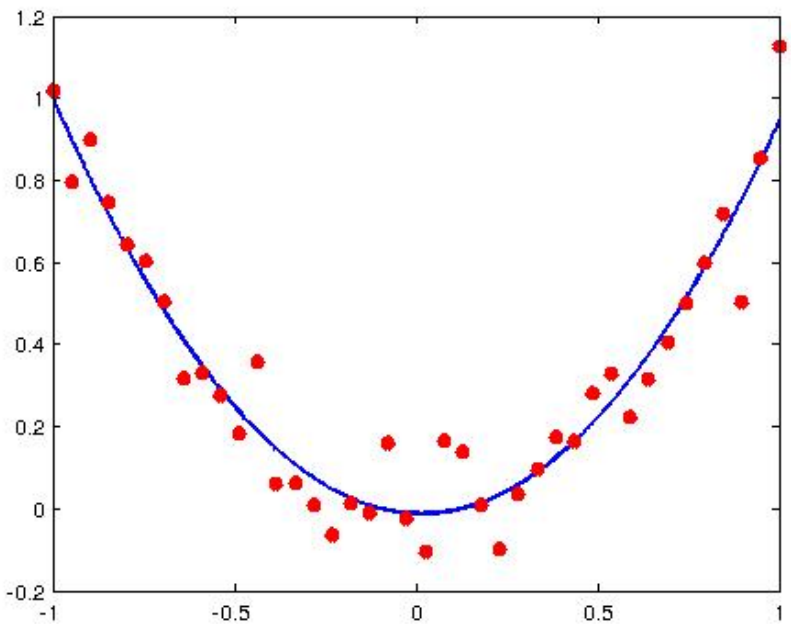
puis en déduire le résultat.

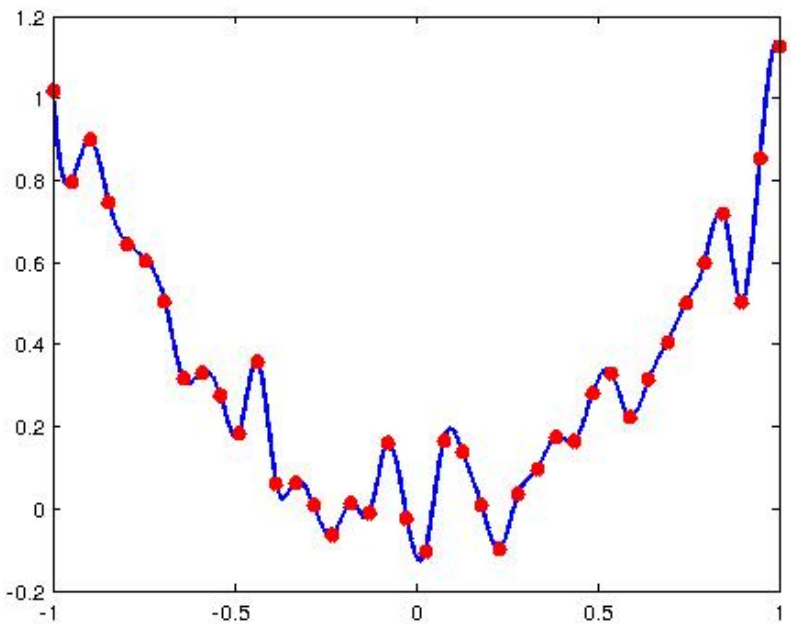
1. Interpolation polynomiale
2. Interpolation polynomiale par morceaux
3. Approximation au sens des moindres carrés

Approximation au sens des moindres carrés

Motivation :

Lorsque les données sont bruitées, l'interpolation de f aux noeuds $\{(x_i, y_i)\}_{i=0:n}$ n'est pas efficace. Il est alors préférable de rechercher une fonction dans un espace vectoriel plus petit qui minimise une erreur L^2 par rapport aux attaches au données.





Approximation au sens des moindres carrés

Plus précisément, l'idée consiste à introduire un ensemble de fonctions linéairement indépendantes

$$\left\{g_0, g_1, ...g_p\right\}$$

et parmi l'ensemble des combinaisons linéaires des g_k, on recherche la fonction $g = \sum_{k=0}^{p} a_k g_k$ qui minimise l'énergie

$$\phi(a_0, a_1..., a_p) = \sum_{i=0}^{n} \left| y_i - \sum_{k=0}^{p} a_k g_k(x_i) \right|^2 .$$

Régression linéaire

Pour la régression linéaire, on utilise $g_0 = 1$ et $g_1 = x$ et on minimise

$$\phi(a_0, a_1) = \sum_{i=0}^{n} \left| y_i - (a_0 + a_1 x_i) \right|^2 .$$

Le minimum de ϕ est alors atteint si $\nabla\phi = 0$:

$$\begin{cases} \partial_{a_0}\phi(a_0, a_1) & = -2 \sum_{i=0}^{n} (y_i - (a_0 + a_1 x_i)) = 0 \\ \partial_{a_1}\phi(a_0, a_1) & = -2 \sum_{i=0}^{n} x_i (y_i - (a_0 + a_1 x_i)) = 0, \end{cases}$$

où encore

$$\begin{aligned} (n+1)a_0 \quad & + \left(\sum_{i=0}^{n} x_i\right) a_1 \quad = \sum_{i=0}^{n} y_i \\ \left(\sum_{i=0}^{n} x_i\right) a_0 \quad & + \left(\sum_{i=0}^{n} x_i^2\right) a_1 \quad = \sum_{i=0}^{n} x_i y_i \end{aligned}$$

Régression linéaire

Les coefficients a_0 et a_1 s'identifient donc à

$$\begin{cases} a_0 &= \frac{1}{D}\left(\sum_{i=0}^{n} y_i \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i \sum_{i=0}^{n} x_i y_i\right) \\ a_1 &= \frac{1}{D}\left((n+1)\sum_{i=0}^{n} x_i y_i - \sum_{i=0}^{n} x_i \sum_{i=0}^{n} y_i\right), \end{cases}$$

où $D = (n+1)\sum_{i=0}^{n} x_i^2 - (\sum_{i=0}^{n} x_i)^2$.

Définition du gradient d'une fonctionnelle

- Soit ϕ une fonctionnelle de $\mathbb{R}^p \to \mathbb{R}$.
- La différentielle de ϕ en x dans la direction y est noté $D\phi(x)(y)$

$$D\phi(x)(y) = \lim_{t \to 0} \frac{\phi(x + ty) - \phi(x)}{t}.$$

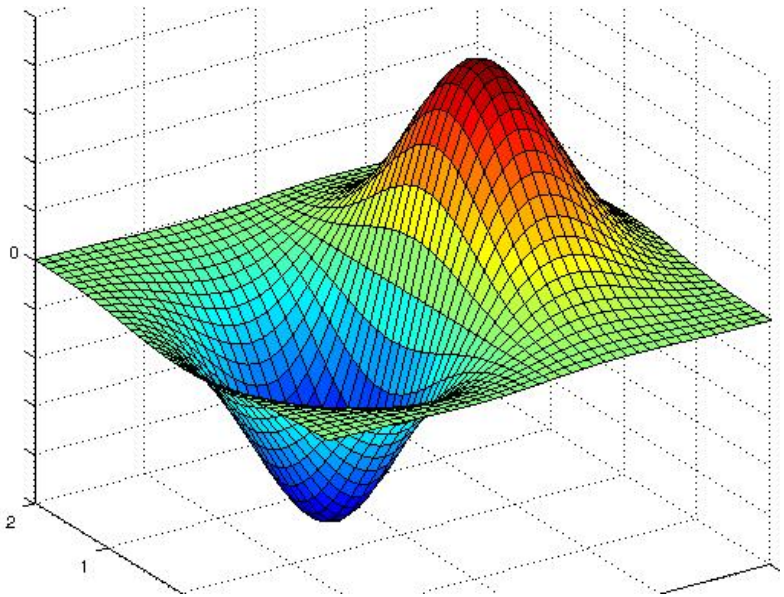
- Le gradient de ϕ en x par rapport à un produit scalaire $< ., . >$ donné est alors défini par

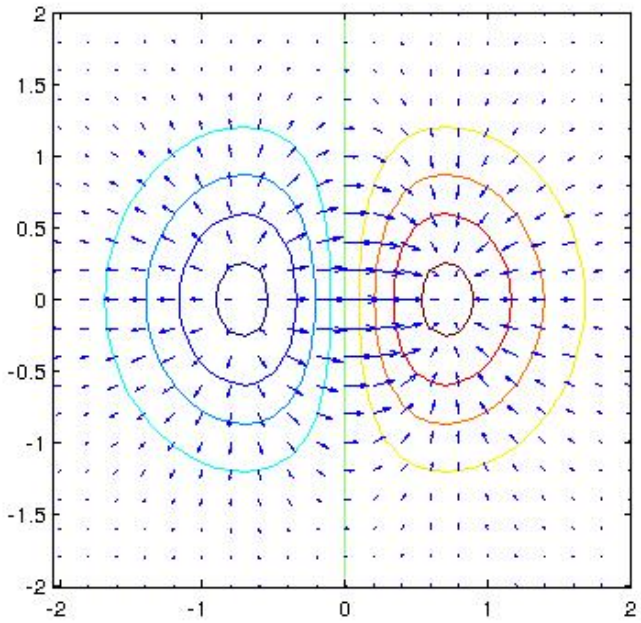
$$< \nabla\phi(x), y >= D\phi(x)(y), \quad \forall y \in \mathbb{R}^p.$$

- Dans le cas du produit scalaire canonique $< x, y >= \sum_{i=0}^{p} x_i y_i$,

$$(\nabla\phi(x))_i = \partial_{x_i}\phi(x).$$

Définition du gradient d'une fonctionnelle





Exercice

Calculer le gradient des fonctionnelles suivantes

$$\begin{cases} \phi_1(x) = < x, b > \\ \phi_2(x) = < Ax, b > \\ \phi_3(x) = \frac{1}{2} < Ax, Ax > \end{cases}$$

Moindres carrés, cas général

- On introduit $p+1$ fonctions $\{g_i\}_{i=0:p}$ et on recherche le minimum de l'énergie

$$\phi(a_0, a_1 ..., a_p) = \sum_{i=0}^{n} \left| y_i - \sum_{k=0}^{p} a_k g_k(x_i) \right|^2 .$$

- La fonctionnelle ϕ s'exprime sous la forme

$$\phi(a) =< y - Ba, y - Ba >,$$

avec

$$a = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{pmatrix}, B = \begin{pmatrix} g_0(x_0) & g_1(x_0) & \cdots & g_p(x_0) \\ g_0(x_1) & g_1(x_1) & \cdots & g_p(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ g_0(x_n) & g_1(x_n) & \cdots & g_p(x_n) \end{pmatrix}, y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix},$$

Moindres carrés, cas général

Le gradient de ϕ s'identifie à

$$\nabla\phi(a) = 2\left(B^t Ba - B^t y\right).$$

Les coefficients a sont donc solution du système linéaire suivant

$$B^t Ba = B^t y.$$

Exercice

Ecrire la matrice B dans le cas d'approximation polynomiale : $\left\{g_k = x^k\right\}_{k=0:p}$.

Exercice

Retrouver l'expression de a_0 *et* a_1 *dans le cas d'une régression linéaire.*

Chapitre 3 : Intégration numérique

...

Objectif : Proposer des méthodes numériques pour calculer des intégrales :

$$\mathcal{I}(f) = \int_a^b f(x)dx.$$

Motivations : Exemple, évaluer des fonctions qui ne s'expriment pas à l'aide de fonctions usuelles

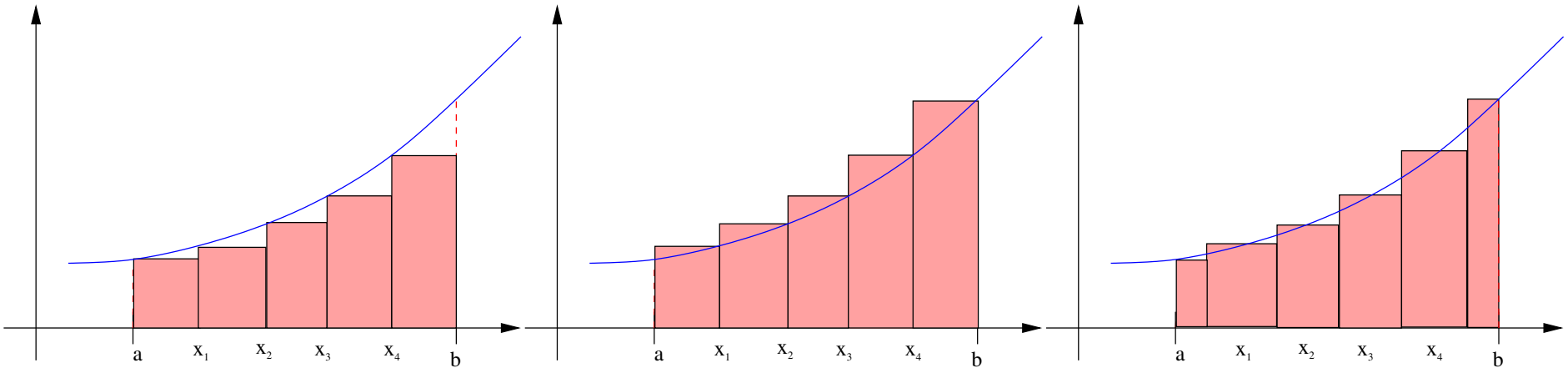
$$\phi(t) = \int_0^t e^{-\pi x^2} dt, \quad ln(t) = \int_1^t 1/x dx.$$

Point de vue : Introduire une discrétisation de l'intervalle $[a, b]$, $a = x_0 < x_1 < ... < x_n = b$ et regarder des approximations de $\mathcal{I}(f)$ de la forme

$$\mathcal{I}_n(f) = \sum_{i=0}^{n} \alpha_i f(x_i).$$

Il s'agit de bien choisir les coefficients α_i pour obtenir une bonne approximation de $\mathcal{I}(f)$!

Première exemple : méthode des rectangles



Quelle est la meilleure approximation de l'intégrale $\mathcal{I}(f)$?

$$\begin{cases} \mathcal{I}_n(f) &= \frac{(b-a)}{n} \sum_{i=0}^{n-1} f(x_i), \\ \mathcal{I}_n(f) &= \frac{(b-a)}{n} \sum_{i=1}^{n} f(x_i) \\ \mathcal{I}_n(f) &= \frac{(b-a)}{n} \left(f(a)/2 + \sum_{i=1}^{n-1} f(x_i) + f(b)/2 \right) \end{cases}$$

1 Formules de Newton-Cotes

2 Formules de Newton-Cotes composite

Principe des formules de Newton-Cotes : intégrer des interpolations polynomiales de f.

- L'interpolation polynomiale de Lagrange s'écrit

$$f(x) \simeq p_n(x) = \sum_{k=0}^{n} f(x_i)\mathcal{L}_i^n(x),$$

- Une approximation de $\mathcal{I}(f)$ s'obtient avec

$$\mathcal{I}_n(f) = \int_a^b p_n(x)dx = \sum_{i=0}^{n} \left(\int_a^b \mathcal{L}_i^n(x)dx \right) f(x_i),$$

- Les coefficients α_i s'identifient aux intégrales

$$\alpha_i = \int_a^b \mathcal{L}_i^n(x)dx = \int_a^b \prod_{j\neq i} \frac{(x - x_j)}{(x_i - x_j)}dx$$

- Dans le cas d'une discrétisation uniforme, on a de plus

$$\alpha_i = \int_a^b \prod_{j=0,j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)}dx = h\int_0^n \prod_{j=0,j\neq i}^{n} \frac{(t - j)}{i - j}dt,$$

avec $h = (b - a)/n$.

Formules du trapèze

Avec $n = 1$ et $x_0 = a$, $x_1 = b$, la formule d'approximation $\mathcal{I}_1$ s'identifie à

$$\mathcal{I}_1(f) = \frac{b-a}{2}\left[f(a) + f(b)\right]$$

Exercice

Montrer cette formule

Exercice

Calculer une approximation de $\ln(2)$

Théorème

Soit $f \in C^2([a,b])$, *alors il existe* $\eta \in [a,b]$ *tel que*

$$\mathcal{I}(f) - \mathcal{I}_1(f) = -\frac{(b-a)^3}{12} f''(\eta).$$

La démonstration de ce théorème repose essentiellement sur la formule d'erreur d'interpolation polynomiale et le théorème de la moyenne :

Théorème (Théorème de la moyenne)

Soit g une fonction continue et h une fonction de signe constant sur $[a, b]$. Alors il existe $\eta \in [a, b]$ tel que

$$\int_a^b g(x)h(x)dx = g(\eta) \int_a^b h(x)dx.$$

Exercice

Démontrer le théorème de la moyenne.

Démonstration :

On rappelle que pour tout $x \in [a,b]$, il existe $\xi_x \in [a,b]$ tel que

$$f(x) - p_1(x) = \frac{1}{2} f''(\xi_x)(x-a)(x-b).$$

Alors

$$\mathcal{I}(f) - \mathcal{I}_1(f) = \frac{1}{2} \int_a^b f''(\xi_x)(x-a)(x-b)dx.$$

On applique le théorème de la moyenne avec $g(x) = f''(\xi_x)$, et $h(x) = (x-a)(x-b)$: on en déduit qu'il existe $\eta \in [a,b]$ tel que

$$\mathcal{I}(f) - \mathcal{I}_1(f) = \frac{1}{2} f''(\eta) \int_a^b (x-a)(x-b)dx.$$

Au final, avec

$$\int_a^b (x-a)(x-b)dx = -\frac{1}{6}(b-a)^3,$$

et

$$\mathcal{I}(f) - \mathcal{I}_1(f) = -\frac{(b-a)^3}{12} f''(\eta)$$

Formule de Simpson

Avec $n = 2$ et $x_0 = a$, $x_1 = (a+b)/2$, $x_2 = b$, la formule d'approximation $\mathcal{I}_2$ s'identifie à

$$\mathcal{I}_2(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Exercice

Montrer cette formule

Exercice

Calculer une approximation de $\ln(2)$

Théorème

Soit $f \in C^4([a,b])$, alors il existe $\eta \in [a,b]$ tel que

$$\mathcal{I}(f) - I_2(f) = -(b-a)^5 \frac{1}{2880} f^{(4)}(\eta).$$

Intuition de la preuve :

1) Erreur d'interpolation polynomiale

$$f(x) - p_2(x) = \frac{1}{3!} f^{(3)}(\xi_x)(x-a)(x-(a+b)/2)(x-b)$$

2) Théorème de la moyenne sur $[a,(a+b)/2]$ et $[(a+b)/2,b]$:

$$\mathcal{I}(x) - \mathcal{I}_2(f) = \frac{1}{4}\left(\frac{b-a}{2}\right)^4 \left(f^{(3)}(\eta_1) - f^{(3)}(\eta_2)\right)$$

avec $\eta_1 \in [a,(a+b)/2]$ et $\eta_2 \in [(a+b)/2,b]$

3) Théorème de la moyenne

$$f^{(3)}(\eta_1) - f^{(3)}(\eta_2) = (\eta_2 - \eta_1) f^{(4)}(\eta)$$

1. Formules de Newton-Cotes

2. Formules de Newton-Cotes composite

Formules de Newton-Cotes composites

Motivation : Augmenter la précision des intégrales sans augmenter le degré des polynômes (phénomène de Runge) !.

$\Longrightarrow$ Utiliser des approximations polynomiales par morceaux.

On utilise une discrétisation de l'intervalle $[a, b]$, $a = x_0 < x_1 < ... < x_n = b$ pour découper l'intégrale $\mathcal{I}(f)$

$$\int_a^b f(x)dx = \sum_{i=1:n} \left(\int_{x_{i-1}}^{x_i} f(x)dx \right)$$

puis on applique les méthodes précédentes sur chacune des nouvelles intégrales.

Formule des trapèzes composites

Avec $h_i = |x_i - x_{i-1}|$, on a

$$\int_{x_{i-1}}^{x_i} f(x)dx \simeq \frac{h_i}{2}\left(f(x_{i-1} + f(x_i)\right).$$

et on pose

$$I_1^c(f,n) = \sum_{i=1}^{n} \frac{h_i}{2}\left(f(x_{i-1}) + f(x_i)\right).$$

Dans le cas uniforme

$$I_1^c(f,h) = \frac{h}{2}\left(f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)\right)$$

Théorème

Si $f \in C^2([a,b])$ *alors*

$$\left|\mathcal{I}(f) - I_1^c(f,h)\right| \leq (b-a)\frac{h^2}{12}\max_{x\in[a,b]}|f''(x)|$$

Formule Simpson composites

Avec $h_i = |x_i - x_{i-1}|$, on a

$$\int_{x_{i-1}}^{x_i} f(x)dx \simeq \frac{h_i}{6}\left(f(x_{i-1} + 4f\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_i)\right).$$

et on pose

$$I_2^c(f,h) = \sum_{i=1}^{n} \frac{h_i}{6}\left(f(x_{i-1}) + 4f\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_i)\right).$$

Dans le cas uniforme

$$I_2^c(f,n) = \frac{h}{6}\left(f(a) + 2\sum_{i=1}^{n-1} f(x_i) + 4\sum_{i=1}^{n} f\left(\frac{x_{i-1}+x_i}{2}\right) + f(b)\right).$$

Théorème

Si $f \in C^4([a,b])$ *alors*

$$\left|\mathcal{I}(f) - I_2^c(f,h)\right| \leq (b-a)\frac{h^4}{2880}\max_{x\in[a,b]}|f^{(4)}(x)|$$

Estimation d'erreur a posteriori

Motivation : On souhaite déterminer une estimation numérique de l'erreur commise entre $\mathcal{I}(f)$ et $\mathcal{I}_2^c(f,h)$ pour un pas de discrétisation h.

Cas de la Formule Simpson composites :

- On suppose que l'erreur entre $\mathcal{I}(f)$ et $\mathcal{I}_2^c(f,h)$ est de la forme

$$E(f,h) = \mathcal{I}(f) - \mathcal{I}_2^c(f,h) \simeq \omega_f h^4$$

  pour h suffisamment petit
- En remarquant que

$$E(f,h/2) = \mathcal{I}(f) - \mathcal{I}_2^c(f,h/2) \simeq \omega_f \left(\frac{h}{2}\right)^4,$$

  une approximation de $E(f,h)$ s'obtient avec

$$E(f,h) \simeq \frac{\mathcal{I}_2^c(f,h/2) - \mathcal{I}_2^c(f,h)}{1-(1/2)^4}$$

Estimation d'erreur a posteriori

Algorithme (Simpson composites)

Donnée : *Précision* $\epsilon > 0$,

$E = 1$, $h = 1$, $I = \mathcal{I}_2^c(f, h)$

While $E \geq \epsilon$

$I_p = I$,
$h = h/2$,
$I = \mathcal{I}_2^c(f, h)$
$E = (I - I_p)/(1 - 1/2^4)$

End

Chapitre 4 : Résolution de systèmes linéaires

...

Ce chapitre s'intéresse à différentes méthodes numériques pour la résolution de systèmes linéaires

$$Ax = b.$$

Motivations : La plupart des méthodes numériques conduisent à la résolution d'un système linéaire qui représente en général, la tache la plus coûteuse d'un point de vue algorithmique !

2 types de méthodes :

- Méthodes directes : factorisation de la matrice A et résolution exacte du système linéaire.
- Méthodes itératives : résolution approchée du système linéaire

Coût algorithmique : (à l'exception de matrices particulières) $O(n^3)$

Cas simples

Matrices diagonales

$$\begin{pmatrix} a_{1,1} & & 0 \\ & \ddots & \\ 0 & & a_{n,n} \end{pmatrix}\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \Longrightarrow \begin{cases} x_1 = b_1/a_{1,1} \\ x_i = b_i/a_{i,i} \\ x_n = b_n/a_{n,n} \end{cases}$$

Coût algorithmique -> $O(n)$

Matrices triangulaires

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 0 & a_{2,2} & & a_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \Longrightarrow \begin{cases} x_n = \frac{b_n}{a_n} \\ x_{n-1} = \frac{b_{n-1} - a_{n-1,n} x_n}{a_{n-1,n-1}} \\ x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{k,j} x_j}{a_{k,k}} \end{cases}$$

Coût algorithmique -> $O(n^2)$

1. Méthodes directes
   - Décomposition LU
   - Décomposition Cholesky
2. Conditionnement et erreur pour la résolution de systèmes linéaires
3. Méthodes itératives classiques
   - Algorithme du point fixe
   - Méthode du gradient

Décomposition LU : idée générale

La décomposition LU consiste à factoriser A sous la forme

$$PA = LU,$$

avec

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{2,1} & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ l_{n,1} & \cdots & l_{n-1,n} & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & & u_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & u_{n,n} \end{pmatrix}$$

et P, une matrice de permutation (en particulier $P^t = P^{-1}$).

Motivation : La résolution du système $Ax = b$ s'effectue en 2 étapes :

$$LUx = P^t b \Longrightarrow \begin{cases} Ly = P^t b \\ Ux = y \end{cases}$$

Décomposition LU sans pivot

Hypothèse : Supposons que toutes les sous matrices principales de A sont inversibles :

$$det(A_k) \neq 0, \quad \text{avec} \quad A_k = \begin{pmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & & \vdots \\ a_{k,1} & \cdots & u_{k,k} \end{pmatrix}, \quad \forall k = 1 : n,$$

alors, il existe une telle décomposition LU avec $P = I_d$, c'est à dire

$$A = LU.$$

Exemple en dimension 3 :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & u_{3,3} \end{pmatrix} \quad \text{et} \quad \begin{cases} a_{1,1} = det(A_1) \neq 0 \\ a_{1,1}a_{2,2} - a_{1,2}a_{2,1} = det(A_2) \neq 0 \end{cases}$$

Avec

$$G_1 = \begin{pmatrix} 1 & 0 & 0 \\ -q_{2,1} & 1 & 0 \\ -q_{3,1} & 0 & 1 \end{pmatrix}, \quad \text{avec} \quad \begin{cases} q_{2,1} = \frac{a_{2,1}}{a_{11}} \\ q_{3,1} = \frac{a_{3,1}}{a_{11}} \end{cases}$$

on a

$$A^{(1)} = G_1 A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} \\ 0 & a_{3,2}^{(1)} & a_{3,3}^{(1)} \end{pmatrix}, \quad \text{avec} \quad \begin{cases} a_{2,2}^{(1)} = a_{2,2} - a_{1,2} q_{2,1} \\ a_{2,3}^{(1)} = a_{2,3} - a_{1,3} q_{2,1} \\ a_{3,2}^{(1)} = a_{3,2} - a_{1,2} q_{3,1} \\ a_{3,3}^{(1)} = a_{3,3} - a_{1,2} q_{3,1} \end{cases}$$

On recommence avec $A^{(1)}$:

$$G_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -q_{3,2} & 1 \end{pmatrix}, \quad \text{avec} \quad q_{3,2} = \frac{a_{3,2}^{(1)}}{a_{22}^{(1)}}$$

$$G_2 G_1 A = A^{(2)} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} \end{pmatrix}, \quad \text{où} \quad a_{3,3}^{(2)} = a_{3,3}^{(1)} - q_{3,2} a_{2,3}^{(1)},$$

Au final

$$A = (G_2 G_1)^{-1} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} \end{pmatrix} = L U,$$

où $L = (G_2 G_1)^{-1} = G_1^{-1} G_2^{-1}$ et donc

$$L = \begin{pmatrix} 1 & 0 & 0 \\ q_{2,1} & 1 & 0 \\ q_{3,1} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & q_{3,2} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ q_{2,1} & 1 & 0 \\ q_{3,1} & q_{3,2} & 1 \end{pmatrix}$$

Algorithme (Décomposition LU sans Pivot)

For $k = 1 : n-1$
  For $i = k+1 : n$
    $q_{i,k} = a_{i,k}^{(k-1)} / a_{k,k}^{(k-1)}$
    For $j = k+1 : n$
      $a_{i,j}^{(k)} = a_{i,j}^{(k-1)} - q_{i,k} a_{k,j}^{(k-1)}$
    End
  End
End

Exercice

Calculer le coût de cet algorithme

Exemple avec

$$A = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 11 \end{pmatrix}, \quad \text{et} \quad b = \begin{pmatrix} 18 \\ 24 \\ 32 \end{pmatrix}$$

On pose $q_{2,1} = a_{2,1}/a_{1,1} = 2$ et $q_{3,1} = a_{3,1}/a_{1,1} = 3$

$$\left(\begin{array}{ccc|c} 1 & 4 & 7 & 18 \\ 2 & 5 & 8 & 24 \\ 3 & 6 & 11 & 32 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} 1 & 4 & 7 & 18 \\ 2 & -3 & -6 & -12 \\ 3 & -6 & -10 & -22 \end{array}\right).$$

On pose $q_{3,2} = a^{(1)}_{3,2}/a^{(1)}_{2,2} = (-6)/(-3) = 2$.

$$\left(\begin{array}{ccc|c} 1 & 4 & 7 & 18 \\ 2 & -3 & -6 & -12 \\ 3 & -6 & -10 & -22 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} 1 & 4 & 7 & 18 \\ 2 & -3 & -6 & -12 \\ 3 & 2 & 2 & 2 \end{array}\right).$$

Au final,

$$A = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & 0 & 2 \end{pmatrix}$$

La solution du système s'obtient en résolvant le système

$$\begin{pmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 18 \\ -12 \\ 2 \end{pmatrix} \Longrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$$

En effet,

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 18 \\ -12 \\ 2 \end{pmatrix} = \begin{pmatrix} 18 \\ 24 \\ 32 \end{pmatrix}$$

Algorithme *LU* avec pivot

Motivation : La décomposition LU est instable lorsque les coefficients $a_{i+1,i+1}^{(i)}$ deviennent trop petits et ne s'applique plus si l'hypothèse des sous-matrices principales inversibles n'est pas vérifiée.

En pratique, il est plus efficace d'utiliser une décomposition *LU* avec pivot

$$PA = LU.$$

Exemple avec

$$A = \begin{pmatrix} 0 & 2 & 2 \\ 4 & 2 & 4 \\ 2 & 4 & 4 \end{pmatrix}, \quad \text{et} \quad b = \begin{pmatrix} 4 \\ 10 \\ 10 \end{pmatrix}$$

On commence par choisir le pivot, le plus grand élément de la première colonne puis on procède comme pour l'algorithme LU sans pivot

$$\left(\begin{array}{ccc|c} 0 & 2 & 2 & 4 \\ 4 & 2 & 4 & 10 \\ 2 & 4 & 4 & 10 \end{array}\right) \longrightarrow P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow \left(\begin{array}{ccc|c} 4 & 2 & 4 & 10 \\ 0 & 2 & 2 & 4 \\ 2 & 4 & 4 & 10 \end{array}\right)$$

Puis

$$\left(\begin{array}{ccc|c} 4 & 2 & 4 & 10 \\ \hline 0 & 2 & 2 & 4 \\ 1/2 & 3 & 2 & 5 \end{array}\right) \longrightarrow P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \longrightarrow \left(\begin{array}{ccc|c} 4 & 2 & 4 & 10 \\ \hline 1/2 & 3 & 2 & 5 \\ 0 & 2 & 2 & 4 \end{array}\right)$$

$$\left(\begin{array}{c|cc|c} 4 & 2 & 4 & 10 \\ \hline 1/2 & 3 & 2 & 5 \\ 0 & 2 & 2 & 4 \end{array}\right) \longrightarrow \left(\begin{array}{c|cc|c} 4 & 2 & 4 & 10 \\ \hline 1/2 & 3 & 2 & 5 \\ 0 & 2/3 & 2/3 & 2/3 \end{array}\right).$$

Alors,

$$\begin{cases} x_3 = 1 \\ x_2 = (5-2)/3 = 1 \\ x_1 = (10-4-2)/4 = 1 \end{cases}$$

et

$$P = P_2 P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix},$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 0 & 3/2 & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} 4 & 2 & 4 \\ 0 & 3 & 2 \\ 0 & 0 & 2/3 \end{pmatrix},$$

Exercice

Résoudre le système linéaire suivant en utilisant la décomposition LU avec pivot

$$A = \begin{pmatrix} 3 & 1 & 6 \\ 2 & 1 & 3 \\ 1 & 1 & 1 \end{pmatrix}, \quad et \quad b = \begin{pmatrix} 2 \\ 4 \\ 7 \end{pmatrix}.$$

1. Méthodes directes
   - Décomposition LU
   - Décomposition Cholesky
2. Conditionnement et erreur pour la résolution de systèmes linéaires
3. Méthodes itératives classiques
   - Algorithme du point fixe
   - Méthode du gradient

Décomposition de Cholesky

La décomposition de Cholesky consiste à factoriser A sous la forme $A = LL^t$, où

$$L = \begin{pmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & & \vdots \\ \vdots & & \ddots & 0 \\ l_{n,1} & \cdots & l_{n-1,n} & l_{n,n} \end{pmatrix}$$

Motivation : Comme pour la décomposition LU, la résolution du système $Ax = b$ s'effectue en 2 étapes :

$$LL^t\, x = b \Longrightarrow \begin{cases} Ly = b \\ L^t x = y \end{cases},$$

L'avantage par rapport à la factorisation LU est

- de n'avoir qu'une seule matrice à stocker
- de présenter un algorithme 2 fois plus rapide !

Definition

Une matrice A est définie positive si

$$< x, Ax >\geq 0 \quad \text{et} \quad < x, Ax >= 0 \Longrightarrow x = 0.$$

Un exemple de telles matrices sont les matrices à diagonale strictement dominante !

Théorème

Si A est une matrice symétrique définie positive, alors il existe une unique matrice L triangulaire inférieure avec coefficients diagonaux positifs telle que

$$A = LL^t.$$

Exemple en dimension 2 :

Soit $A = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$ une matrice définie positive.

En particulier, l'hypothèse de positivité implique que

$$\begin{cases} a =< e_1, Ae_1 > \ > 0 \\ ab - c^2 = det(A) > 0 \end{cases}$$

Avec

$$L = \begin{pmatrix} l_{1,1} & 0 \\ l_{2,1} & l_{2,2} \end{pmatrix} \quad \text{et} \quad LL^t = \begin{pmatrix} l_{1,1}^2 & l_{1,1}l_{2,1} \\ l_{2,1}l_{1,1} & l_{1,1}^2 + l_{2,2}^2 \end{pmatrix}.$$

On en déduit que

$$\begin{cases} l_{1,1} &= \sqrt{a} \\ l_{2,1} &= \frac{c}{\sqrt{a}} \\ l_{2,2} &= \sqrt{b - \frac{c^2}{a}} \end{cases}$$

Cas général

$$\begin{pmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & & \vdots \\ \vdots & & \ddots & 0 \\ l_{n,1} & \cdots & l_{n-1,n} & l_{n,n} \end{pmatrix} \begin{pmatrix} l_{1,1} & l_{2,1} & \cdots & l_{n,1} \\ 0 & l_{2,2} & & \vdots \\ \vdots & & \ddots & l_{n-1,n} \\ 0 & \cdots & 0 & l_{n,n} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{1,2} & a_{2,2} & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ a_{1,n} & \cdots & a_{n-1,n} & a_{n,n} \end{pmatrix}$$

La première ligne montre que $a_{1,i} = l_{11} l_{i,1}$ et

$$\begin{cases} l_{1,1} = \sqrt{a_{1,1}} \\ l_{i,1} = \frac{a_{1,i}}{l_{1,1}}, \quad \text{pour} \quad i > 1 \end{cases}$$

La première colonne de L est maintenant connue.

La deuxième ligne montre que $a_{2,i} = \ell_{2,1}\ell_{i,1} + \ell_{2,2}\ell_{2,i}$ pour $i \geq 2$ et

$$\begin{cases} \ell_{2,2} = \sqrt{a_{2,2} - \ell_{2,1}^2} \\ \ell_{i,2} = \frac{a_{2,i} - \ell_{2,1}\ell_{i,1}}{\ell_{2,2}}, \quad \text{pour} \quad i > 2 \end{cases}$$

Les deux premières colonnes de L sont maintenant connues.

Cas général

$$\begin{pmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & & \vdots \\ \vdots & & \ddots & 0 \\ l_{n,1} & \cdots & l_{n-1,n} & l_{n,n} \end{pmatrix} \begin{pmatrix} l_{1,1} & l_{2,1} & \cdots & l_{n,1} \\ 0 & l_{2,2} & & \vdots \\ \vdots & & \ddots & l_{n-1,n} \\ 0 & \cdots & 0 & l_{n,n} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{1,2} & a_{2,2} & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ a_{1,n} & \cdots & a_{n-1,n} & a_{n,n} \end{pmatrix}$$

A la k^{ieme} ligne : les $k-1$ premières colonnes de L sont supposées connues, alors

$$a_{k,i} = \sum_{j=1}^{k} l_{k,j} l_{i,j}, \quad \text{pour} \quad i \geq k$$

et

$$\begin{cases} \ell_{k,k} = \sqrt{a_{k,k} - \sum_{j=1}^{k-1} \ell_{k,l}^2} \\ \ell_{i,k} = \left(a_{k,i} - \sum_{j=1}^{k-1} \ell_{k,j} \ell_{i,j}\right) / \ell_{k,k} \end{cases}$$

Les k premières colonnes de L sont connues.

Algorithme (Décomposition de Cholesky)

For $k = 1 : n$

$\ell_{k,k} = \sqrt{a_{k,k} - \sum_{j=1}^{k-1} \ell_{k,l}^2}$

For $i = k + 1 : n$

$\ell_{i,k} = \left(a_{k,i} - \sum_{j=1}^{k-1} \ell_{k,j}\ell_{i,j}\right) / \ell_{k,k}$

End

End

Exercice

Calculer le coût algorithmique de la décomposition de Cholesky

Exercice

Appliquer l'algorithme de Cholesky pour factoriser la matrice

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 4 \end{pmatrix}$$

1. Méthodes directes
   - Décomposition LU
   - Décomposition Cholesky
2. Conditionnement et erreur pour la résolution de systèmes linéaires
3. Méthodes itératives classiques
   - Algorithme du point fixe
   - Méthode du gradient

Conditionnement d'une matrice

Motivation : Il s'agit ici de comprendre comment une erreur δA sur la matrice A et une erreur δb sur le vecteur b influencent la solution x du système linéaire

$$Ax = b.$$

En particulier, en notant $x + \delta_x$ la solution de

$$(A + \delta A)(x + \delta_x) = b + \delta_b,$$

on souhaite contrôler l'erreur relative

$$\frac{\|\delta x\|}{\|x\|}$$

Norme vectorielle

Definition

$\|.\|$ est une norme sur $\mathbb{R}^n$ si $\forall (x, y) \in (\mathbb{R}^n)^2$ et $\alpha \in \mathbb{R}$,

- Positivité

$$\|x\| \geq 0 \quad \text{et} \quad \|x\| = 0 \Leftrightarrow x = 0$$

- Linéarité

$$\|\alpha x\| = |\alpha| \|x\|$$

- Inégalité triangulaire

$$\|x + y\| \leq \|x\| + \|y\|$$

Exemple

$$\begin{cases} \|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} \\ \|x\|_1 = \sum_{i=1}^n |x_i| \\ \|x\|_\infty = \max_{i=1:n} \{|x_i|\} \end{cases}$$

Norme matricielle induites

Definition

Soit $\|.\|$ une norme sur $\mathbb{R}^n$. On note $|||.|||$ la norme induite sur l'espace vectoriel des matrices carrées de taille $(n \times n)$ définie par

$$|||A||| = \max_{\|x\| \leq 1} \|Ax\|$$

Exercice

Montrer que $|||.|||$ *est bien une norme sur l'espace des matrices carrées* $(n \times n)$

Exemples :

$$\begin{cases} |||A|||_2 = \sqrt{\lambda_{\max}} \\ |||A|||_1 = \max_j \sum_{i=1}^n |a_{ij}| \\ |||A|||_\infty = \max_i \sum_{j=1}^n |a_{ij}|, \end{cases}$$

où $\lambda_{\max}$ est la plus grande valeur propre de $A^t A$.

Conditionnement

Definition

Soit $A \in \mathbb{R}^{n\times n}$ une matrice inversible. Le conditionnement de A par rapport à une norme donnnée $\|.\|$ est défini par

$$cond(A) = |||A||| \; |||A^{-1}|||$$

Exercice

Calculer le conditionnement par rapport à la norme $\|.\|_\infty$ de la matrice

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}$$

Théorème

Soit x et $x + \delta x$ les solutions des systèmes linéaires

$$\begin{cases} Ax = b \\ A(x + \delta x) = b + \delta b \end{cases}$$

Alors

$$\frac{\|\delta x\|}{\|x\|} \leq cond(A) \frac{\|\delta b\|}{\|b\|}$$

Démonstration : Il suffit de remarquer que

$$Ax = b \Longrightarrow \|b\| \leq |||A||| \, \|x\| \Longrightarrow \frac{1}{\|x\|} \leq \frac{|||A|||}{\|b\|}$$

et

$$A\delta x = \delta b \Longrightarrow \|\delta_x\| \leq |||A^{-1}||| \, \|\delta_b\|$$

Théorème

Soit x et $x + \delta x$ les solutions des systèmes linéaires

$$\begin{cases} Ax = b \\ (A + \delta A)(x + \delta x) = b + \delta b \end{cases}$$

Alors

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{cond(A)}{(1 - |||A||| \, |||\delta A|||)} \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right)$$

Démonstration : Avec $\delta x = \delta x_b + \delta x_A$, où

$$A\delta x_b = \delta b,$$

il ressort que

$$A\delta x_b = \delta b \Longrightarrow \|\delta x_b\| \leq |||A^{-1}||| \, \|\delta b\|$$

et

$$(A + \delta A)(x + \delta x_b + \delta x_A) = b + \delta b \Longrightarrow \delta x_A = -A^{-1}\delta A(x + \delta x)$$
$$\Longrightarrow \|\delta x_A\| \leq |||A^{-1}||| \, |||\delta A||| \, (\|x\| + \|\delta_x\|)$$

On en déduit que

$$\|\delta x\| \leq \|\delta x_A\| + \|\delta x_b\| \leq |||A^{-1}||| \, \|\delta_b\| + |||A^{-1}||| \, |||\delta A||| \, (\|x\| + \|\delta_x\|)$$

Enfin avec

$$Ax = b \Longrightarrow \|b\| \leq |||A||| \, \|x\| \Longrightarrow \frac{1}{\|x\|} \leq \frac{|||A|||}{\|b\|}$$

on en déduit que

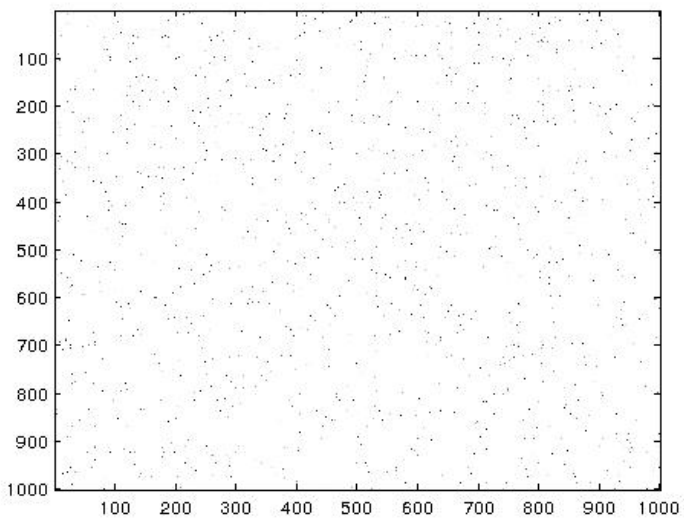
$$\|\delta x\| \leq \frac{1}{1 - |||A^{-1}||| \, |||\delta A|||} \left(|||A^{-1}||| \, \|\delta_b\| + |||A^{-1}||| \, |||\delta A||| \, \|x\| \right)$$

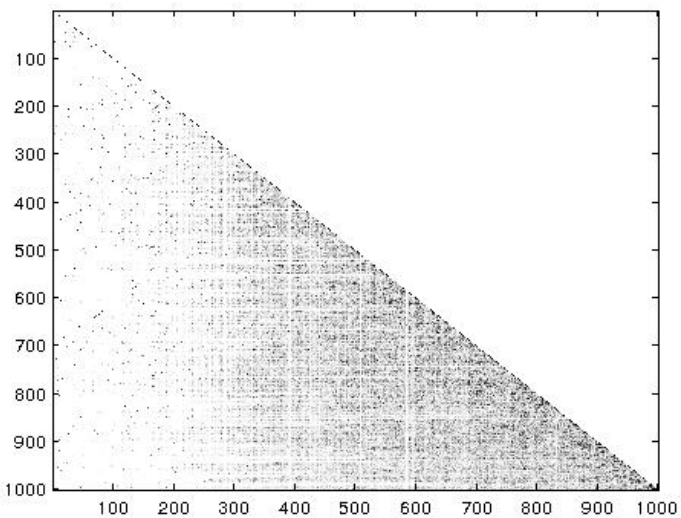
1. Méthodes directes
   - Décomposition LU
   - Décomposition Cholesky
2. Conditionnement et erreur pour la résolution de systèmes linéaires
3. Méthodes itératives classiques
   - Algorithme du point fixe
   - Méthode du gradient

Méthode itératives classiques

Motivations :

- Le coût de la résolution des méthodes directes est en $O(n^3)$!
- Dans de nombreuses situations, les matrices A sont creuses (peu des coefficients non nuls) et la décomposition LU perd cette propriété $\rightarrow$ problème de place mémoire !





1. Méthodes directes
   - Décomposition LU
   - Décomposition Cholesky
2. Conditionnement et erreur pour la résolution de systèmes linéaires
3. Méthodes itératives classiques
   - Algorithme du point fixe
   - Méthode du gradient

Algorithme du point fixe

Le principe est de rechercher la solution x^* du système linéaire

$$Ax = b,$$

comme le point fixe d'une application $\phi : \mathbb{R}^n \to \mathbb{R}^n$:

$$\phi(x^*) = x^*.$$

L'algorithme du point fixe construit alors la suite x_k définie par

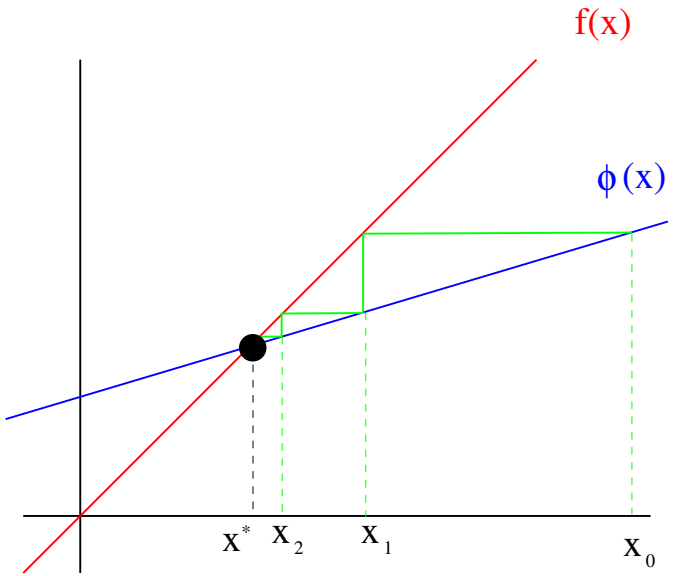
$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = \phi(x_k) \end{cases}$$

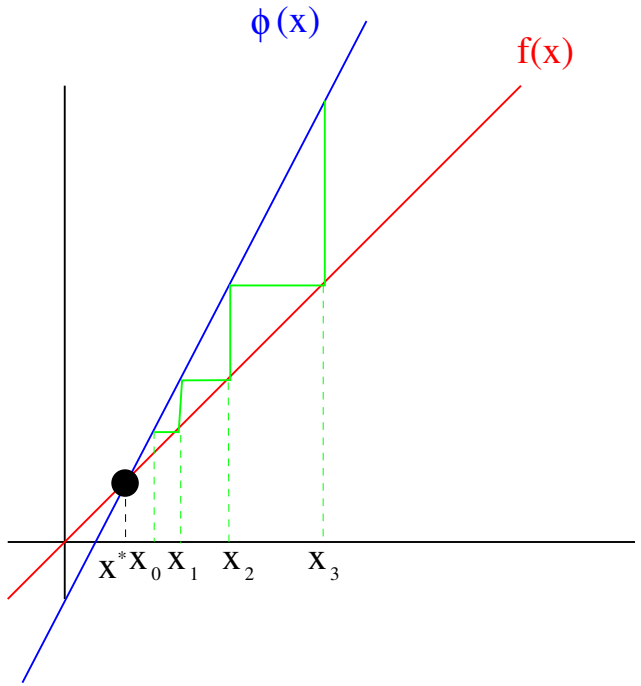
en espérant qu'avec de bonnes propriétés sur ϕ,

$$\lim_{k \to \infty} x_k = x^*$$

Exemple en dimension 1

Avec des fonctions ϕ de la forme $\phi(x) = ax - b$.





une condition nécessaire et suffisante pour que $x_k \to x^*$ est

$$|a| < 1$$

Contraction et existence de point fixe

Definition

Une application $\phi : \mathbb{R}^n \to \mathbb{R}^n$ est une contraction pour la norme $\|.\|$ s'il existe $\lambda < 1$ tel que

$$\|\phi(x) - \phi(y)\| \leq \lambda \|x - y\|$$

Théorème

Si ϕ est une contraction, alors ϕ admet un unique point fixe noté x^*.

Démonstration :

- Existence : admise (suite de Cauchy)
- Unicité : si $x_1 = \phi(x_1)$ et $x_2 = \phi(x_2)$ alors

$$\begin{aligned} \|x_1 - x_2\| &= \|\phi(x_1) - \phi(x_2)\| \\ &\leq \lambda \|x_1 - x_2\| \\ &< \|x_1 - x_2\| \quad \text{si} \quad x_1 \neq x_2 \end{aligned}$$

Convergence de l'algorithme du point fixe

Théorème

Soit $\phi : \mathbb{R}^n \to \mathbb{R}^n$ une contraction et $x_0 \in \mathbb{R}^n$. Alors la suite x_k définie

$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = \phi(x_k) \quad \forall k > 0 \end{cases}$$

converge vers l'unique point fixe de x^* et

1. *Estimation a posteriori*

$$\|x^* - x_k\| \leq \frac{1}{1-\lambda}\|x_k - x_{k+1}\|$$

2. *Estimation a priori*

$$\|x^* - x_k\| \leq \frac{\lambda^k}{1-\lambda}\|x_1 - x_0\|$$

Démonstration :

1. Il suffit de remarquer que

$$\begin{aligned} \|x^* - x_k\| &\leq \|x^* - x_{k+1} + x_{k+1} - x_k\| \leq \|x^* - x_{k+1}\| + \|x_{k+1} - x_k\| \\ &\leq \|\phi(x^*) - \phi(x_k)\| + \|x_{k+1} - x_k\| \\ &\leq \lambda\|x^* - x_k\| + \|x_{k+1} - x_k\| \end{aligned}$$

2. Avec l'inégalité (1), on a

$$\begin{aligned} \|x^* - x_k\| &\leq \frac{1}{1-\lambda}\|x_{k+1} - x_k\| \leq \frac{1}{1-\lambda}\|\phi(x_k) - \phi(x_{k-1})\| \\ &\leq \frac{\lambda}{1-\lambda}\|x_k - x_{k-1}\| \leq \frac{\lambda^k}{1-\lambda}\|x_1 - x_0\| \end{aligned}$$

Algorithme avec une précision ϵ

Algorithme (Méthode du Point fixe)

Donnée : $x_0 \in \mathbb{R}^n$, $\epsilon > 0$

$x_1 = \phi(x_0)$, $k = 1$

While $\|x_k - x_{k-1}\| \leq c_\epsilon$

$x_{k+1} = \phi(x_k)$

End

Exercice

Comment choisir c_ϵ pour que l'algorithme s'arrête avec une précision de ϵ sur x^* ?

Application pour la résolution de système linéaire

On s'intéresse à des applications ϕ de la forme

$$\phi(x) = M^{-1}(Nx + b).$$

où $A = M - N$.

Exercice

Montrer que si x^* est un point fixe de ϕ alors

$$Ax^* = b$$

L'application ϕ est une contraction si

$$|||M^{-1}N||| < 1.$$

On remarque de plus que $x_{k+1} = \phi(x_k)$ est solution de système linéaire

$$Mx_{k+1} = Nx_k + b.$$

Idée : choisir M proche de A et facile à inverser !

Méthode de Jacobi

La méthode de Jacobi utilise

$$M = D, \quad N = -L - U.$$

Exercice

Montrer que

$$(x_{k+1})_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{i,j} (x_k)_j \right)$$

Théorème

Si A est à diagonale strictement dominante, alors la méthode de Jacobi converge

Démonstration : Avec $M^{-1}N = -D^{-1}(L + U)$, il ressort que

$$\|M^{-1}N\|_\infty = max_j \left(\sum_{i \neq j} \left| \frac{a_{i,j}}{a_{ii}} \right| \right) < 1$$

Méthode de Gauss-Seidel

La méthode de Gauss-Seidel utilise

$$M = D + L, \quad N = -U.$$

Exercice

Montrer que

$$(x_{k+1})_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}(x_{k+1})_j - \sum_{j=i+1}^{n} a_{ij}(x_k)_j\right) \tag{1}$$

Théorème

Si A est symétrique définie positive, alors la méthode de Gauss-Seidel converge

Démonstration : Admise (regarder la norme $|||M^{-1}N|||_2$ ...)

1. Méthodes directes
   - Décomposition LU
   - Décomposition Cholesky
2. Conditionnement et erreur pour la résolution de systèmes linéaires
3. Méthodes itératives classiques
   - Algorithme du point fixe
   - Méthode du gradient

Méthode du gradient

Hypothèse : A symétrique définie positive.

Le principe est de regarder la solution x^* du système linéaire

$$Ax = b$$

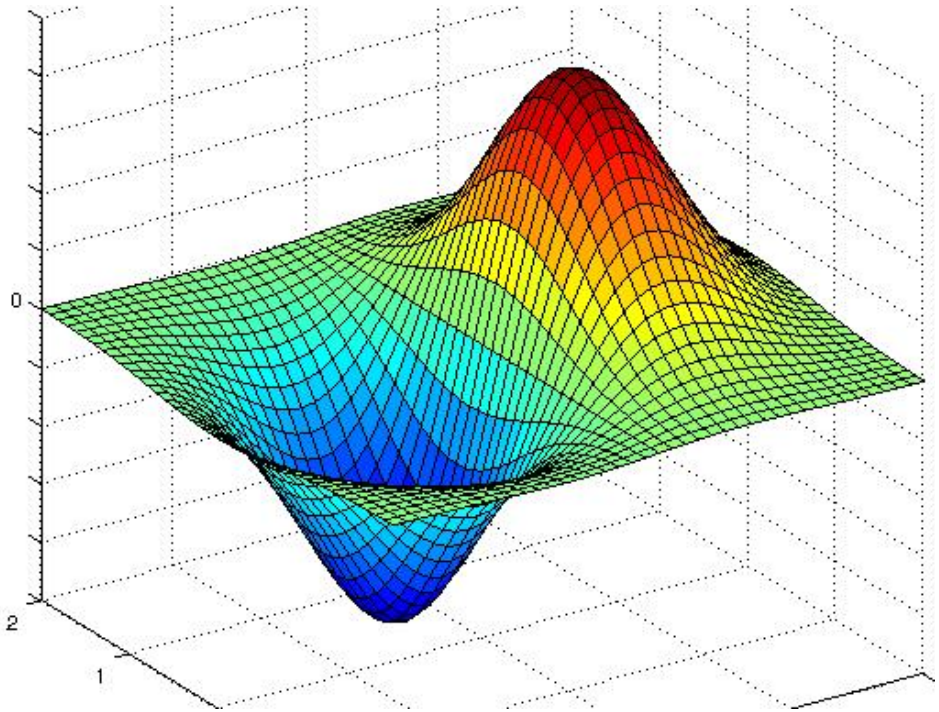
comme le minimum de l'énergie

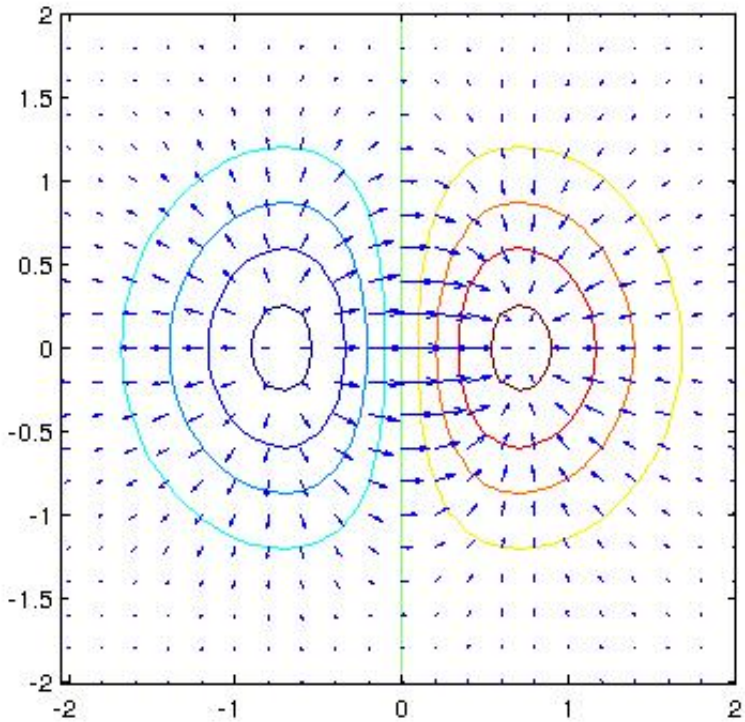
$$J(x) = \frac{1}{2} < Ax, x > - < b, x > .$$

On construit alors une suite $\{x_k\}$ qui minimise J de la forme

$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = x_k + \alpha_k d_k \quad \forall k \geq 1, \end{cases}$$

où d_k est une direction de descente et α_k un pas de descente.


0
2
1


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2

Choix de d_k :

La direction d_k est l'opposé du gradient de J :

$$d_k = -\nabla J(x_k) = -(Ax_k - b) = b - Ax_k$$

Choix de α_k :

On choisit α_k comme le pas qui minimise J dans la direction d_k :

$$\tilde{J}(\alpha) = J(x_k + \alpha d_k).$$

Avec

$$\begin{aligned} \tilde{J}'(\alpha) &= < \nabla J(x_k + \alpha d_k), d_k > \\ &= < A(x_k + \alpha d_k) - b, d_k > \\ &= \alpha < Ad_k, d_k > - < d_k, d_k >, \end{aligned}$$

on en déduit que

$$\tilde{J}'(\alpha_k) = 0 \Longleftrightarrow \alpha_k = \frac{< d_k, d_k >}{< Ad_k, d_k >}$$

Algorithme (Méthode du gradient)

Donnée : $x_0 \in \mathbb{R}^n$, $\epsilon > 0$

While $\|Ax_k - b\| \leq \epsilon \|\|A^{-1}\|\|$

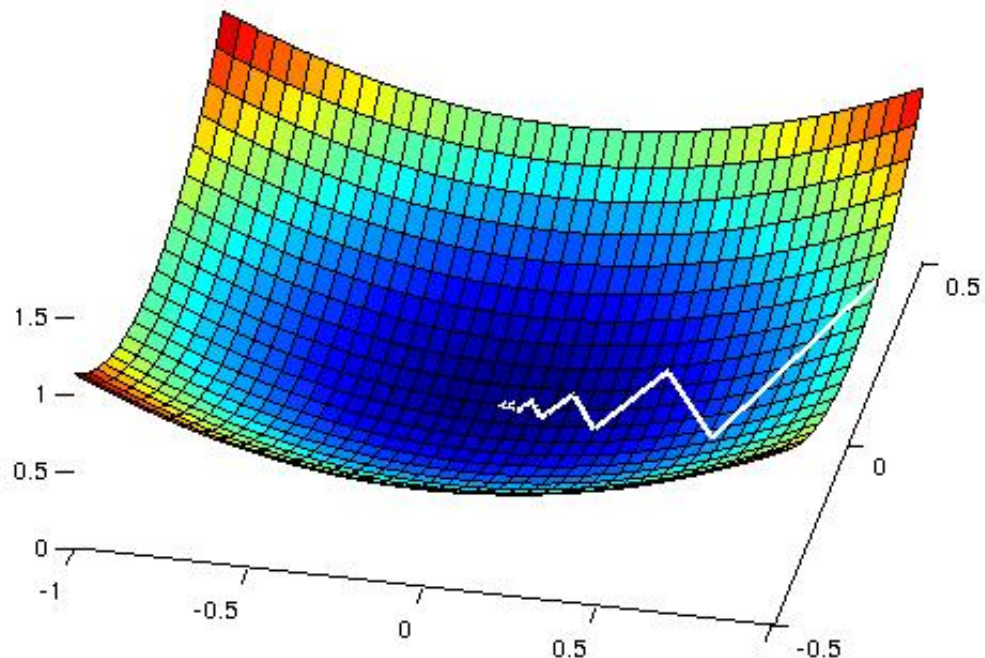
$d_k = b - Ax_k$

$\alpha_k =< d_k, d_k > / < Ad_k, dk >$

$x_{k+1} = x_k + \alpha_k d_k$

End

Exemple avec $A = (1,0; 0,5)$, $b = (0,0)'$ et $x_0 = (1, 0.2)'$



Algorithme (Méthode du gradient conjugué)

Donnée : $x_0 \in \mathbb{R}^n$, $\epsilon > 0$

While $\|Ax_k - b\| \leq \epsilon \|\|A^{-1}\|\|$

$r_k = b - Ax_k$

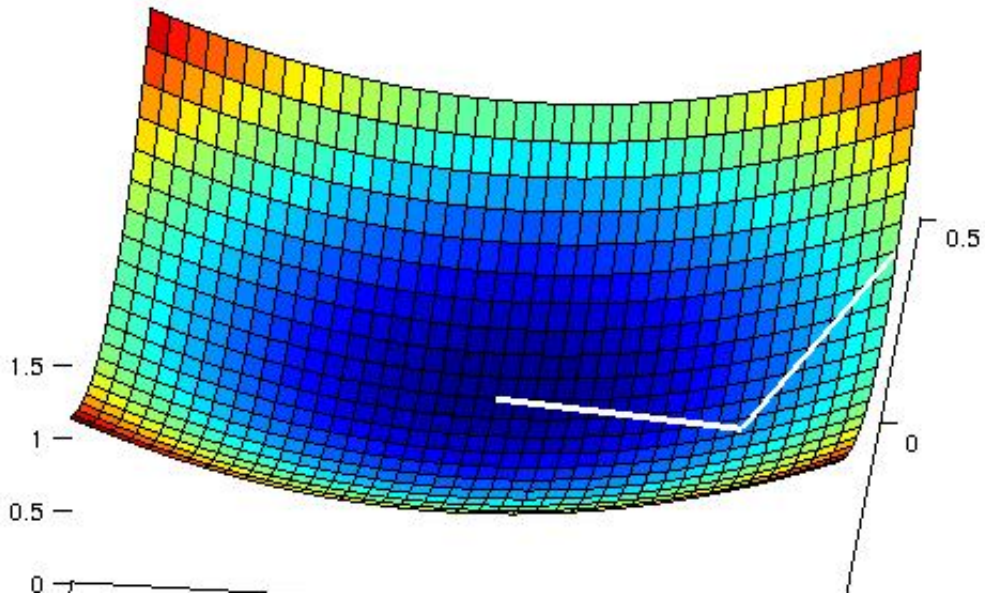
$d_k = r_k - \sum_{i<k} \frac{<d_i, Ar_k>}{<d_i, Ad_i>} d_i$

$\alpha_k = < d_k, r_k > / < Ad_k, d_k >$

$x_{k+1} = x_k + \alpha_k d_k$

End

Exemple avec $A = (1,0; 0,5)$, $b = (0,0)'$ et $x_0 = (1, 0.2)'$



Chapitre 5 : Résolution d'équations non-linéaires

...

Ce chapitre s'intéresse aux méthodes numériques pour la résolution d'équations non linéaires de la forme

$$f(x) = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{pmatrix} = 0$$

Exemples :

- Calcul de la racine carré $\sqrt{a}$: Premier algorithme, Babyloniens
- Méthode des Moindres carrés non linéaires : Trouver la fonction de la forme

  $$g(\alpha, \beta, x) = \alpha x^{\beta}$$

  qui minimise l'énergie

  $$\phi(\alpha, \beta) = \sum_{i=1}^{n} \left(y_i - \alpha x_i^{\beta}\right)^2$$

  La solution s'obtient en résolvant

  $$\nabla \phi = 0.$$

1. Méthode de la Dichotomie
2. Méthode du point fixe
3. Méthode de Newton

Méthode de la Dichotomie

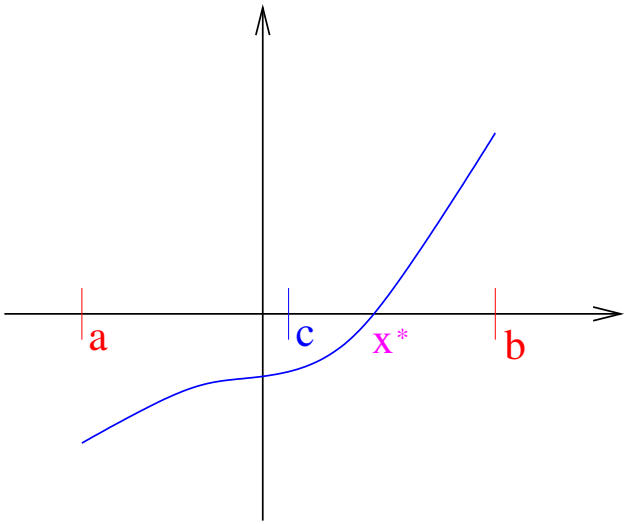
Hypothèses

Soit $f : [a, b] \to \mathbb{R}$, continue telle que $f(a)f(b) < 0$

Principe

On regarde la fonction f au milieu de [a,b], $c = (a + b)/2$,

$$\begin{cases} \text{Si } f(a)f(c) < 0 & \Longrightarrow \exists x^* \in [a, c] \\ \text{Si } f(a)f(c) > 0 & \Longrightarrow \exists x^* \in [c, b] \end{cases}$$



Algorithme (Méthode de la dichotomie)

Donnée : $a_a = a$, $b_0 = b$, $x_0 = (a+b)/2$ *et* $\epsilon > 0$

For $k = 0 : N_\epsilon$

If $f(x_k)f(a_k) < 0$

$a_{k+1} = a_k$

$b_{k+1} = x_k$

else

$a_{k+1} = x_k$

$b_{k+1} = b_k$

end

$x_{k+1} = (a_{k+1} + b_{k+1})/2$

End

Exercice

Avec $l_k = b_k - a_k$, montrer que

$$l_k = \frac{b-a}{2^k}.$$

En déduire que le nombre d'itérations N_ϵ nécessaire à l'algorithme pour obtenir une précision de ϵ sur la solution vérifie

$$N_\epsilon \geq \frac{\ln((b-a)/\epsilon)}{ln(2)}$$

Exercice

Application pour l'estimation de $\sqrt{2}$ à une précision de 10^{-2} avec $f(x) = x^2 - 2$.

Limite de l'approche de la Dichotomie

- Cas pathologiques !
- Vitesse de convergence linéaire.
- Extension en dimension supérieure ?

1. Méthode de la Dichotomie
2. Méthode du point fixe
3. Méthode de Newton

Méthode du point fixe

Principe :

Transformer le problème $f(x) = 0$ en un problème équivalent de la forme $g(x) = x$.

Exemple pour le calcul d'une racine carré $\sqrt{a}$:

Cette racine peut s'obtenir en résolvant $f(x) = x^2 - a = 0$, ou encore en regardant les points fixes des fonctions g suivantes

$$\begin{cases} g_1(x) = a/x \\ g_2(x) = x^2 + x - a \\ g_3(x) = (1 - 1/m)x + a/(mx), \end{cases}$$

Question : Comment choisir une fonction g optimale ?

Méthode du point fixe

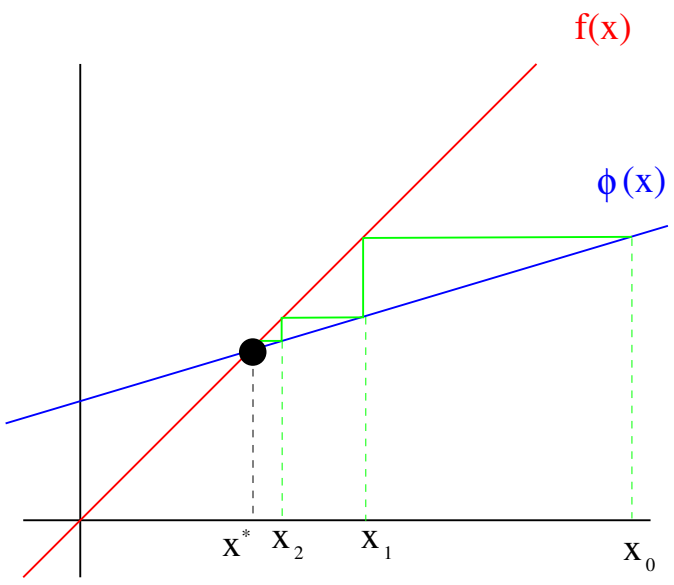
Algorithme (Méthode du Point fixe)

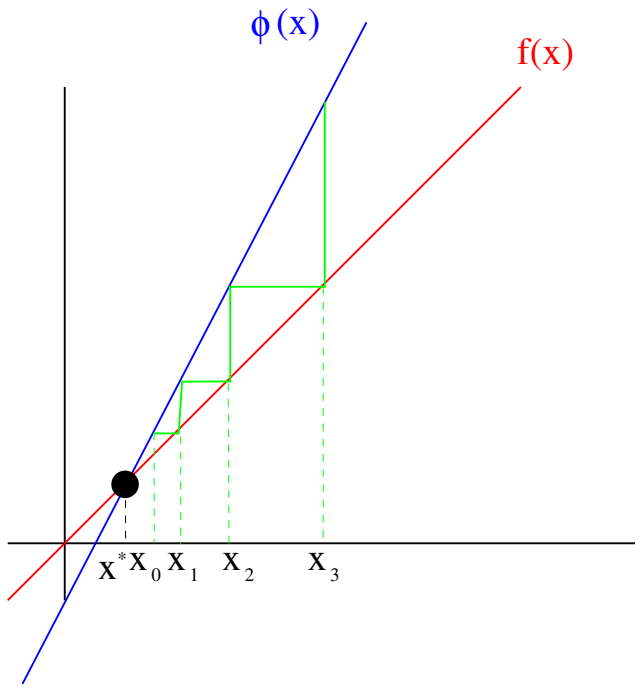
Donnée : $x_0 \in \mathbb{R}^n$, $\epsilon > 0$

for $k = 0 : N_\epsilon$

$$x_{k+1} = g(x_k)$$

End





Théorème (Version globale en dimension 1)

Soit $g : [a, b] \to [a, b]$ *de régularité* $C^1([a, b])$ *telle que*

$$|g'(x)| \leq \lambda < 1, \quad \forall x \in [a, b].$$

Alors g possède un unique point fixe $x^* \in [a, b]$ *et* $x_k \to x^*$ *lorsque* $k \to \infty$. *Plus précisément,*

1. *Estimation a posteriori*

$$\|x^* - x_k\| \leq \frac{1}{1-\lambda}\|x_k - x_{k+1}\|$$

2. *Estimation a priori*

$$\|x^* - x_k\| \leq \frac{\lambda^k}{1-\lambda}\|x_1 - x_0\|$$

Démonstration :

Existence : Avec $h(x) = g(x) - x$, remarquer que $h(a)h(b) \leq 0$

Unicité : Avec $x_1 = g(x_1)$ et $x_2 = g(x_2)$, remarquer que

$$|x_1 - x_2| = \left|g(x_1) - g(x_2)\right| \leq \int_{x_1}^{x_2} |g'(x)|dx \leq \lambda|x_1 - x_2|$$

Vitesse de convergence : voir cours résolution systèmes linéaires.

Exercice

Montrer que le nombre d'itérations N_ϵ permettant à l'algorithme d'obtenir un précision de l'ordre de ϵ sur la solution x^* du problème vérifie

$$N_\epsilon \geq \frac{ln\left(\frac{(1-\lambda)\epsilon}{\|x_1 - x_0\|}\right)}{ln(\lambda)}$$

Théorème (Version locale en dimension 1)

Soit x^* un point fixe d'une fonction g de régularité C^1 au voisinage de x^* telle que

$$|g'(x^*)| < 1.$$

Alors il existe un réel $\alpha > 0$ tel que $\forall x_0 \in [x^* - \alpha, x^* + \alpha]$, l'algorithme du point fixe converge vers le point fixe x^*

Démonstration :

Appliquer le théorème global sur l'intervalle $I_\alpha = [x^* - \alpha, x^* + \alpha]$ avec α suffisamment petit.

Cas de plusieurs points fixes

Definition

Un point fixe x^* d'une fonction g est dit

$$\begin{cases} \text{attractif} & \text{si} \quad |g_1'(x^*)| < 1 \\ \text{répulsif} & \text{si} \quad |g_1'(x^*)| > 1 \end{cases}$$

Definition

Le bassin d'attraction d'un point fixe x^* est l'ensemble des valeurs initiales x_0 pour lesquelles x_k tend vers x^* lorsque k tend vers l'infini

Exercice

Etudier le bassin d'attraction des points fixes de

$$g(x) = x + (x^2 - 1)x.$$

Vitesse de convergence

Le taux de convergence est donné par $|g'(x^*)|$.

$$\begin{cases} \text{si } |g'(x^*)| > 1 & \Longrightarrow \quad \text{l'algorithme diverge} \\ \text{si } |g'(x^*)| < 1 \text{ et } g'(x^*) \neq 0 & \Longrightarrow \quad \text{convergence linéaire} \\ \text{si } |g'(x^*)| = 0 & \Longrightarrow \quad \text{convergence quadratique} \end{cases}$$

Exercice (Cas où $f(x) = x^2 - a$)

Etudier la convergence de l'algorithme du point fixe dans le cas des fonctions suivantes

$$\begin{cases} g_1(x) = a/x \\ g_2(x) = x^2 + x - a \\ g_3(x) = (1 - 1/m)x + a/(mx) \end{cases}$$

Déterminer une approximation de $\sqrt{2}$.

Méthode du point fixe en dimension supérieure

Théorème (Version globale)

On introduit

$$B(x,R) = \{y \in \mathbb{R}^n \,;\, \|y-x\| \leq R\},$$

Soit $g : B(\overline{x},R) \to B(\overline{x},R)$ *est une contraction, i.e.* $\exists \lambda < 1$ *tel que*

$$\|g(x)-g(y)\| \leq \lambda \|x-y\|, \quad \forall (x,y) \in B(x,R).$$

Alors g admet un point fixe $x^* \in B(\overline{x},R)$ *et* $x_k \to x^*$

Théorème (Version locale)

Soit $g : \mathbb{R}^n \to \mathbb{R}^n$ *de régularité* C^1 *au voisinage d'un point fixe* x^* *de g. Si* $\|Dg(x^*)\| < 1$*, alors il existe un réel* $\alpha > 0$ *tel que* $\forall x_0 \in B(x^*,\alpha)$*, l'algorithme du point fixe converge vers* x^**.*

Exercice

- *Déterminer les solutions de*

$$F(X) = \begin{pmatrix} x_1^2 + x_2^2 - 2 \\ x_1^2 - x_2^2 \end{pmatrix} = 0.$$

- *Montrer que ces solutions s'obtiennent aussi comme les points fixes de l'application*

$$G(X) = \begin{pmatrix} (1 - 1/m)x_1 + \frac{2 - x_2^2}{m x_1} \\ (1 - 1/m)x_2 + \frac{x_1^2}{m x_2} \end{pmatrix}$$

- *Etudier la convergence de l'algorithme du point fixe dans le cas où* $m = 4$.

1. Méthode de la Dichotomie
2. Méthode du point fixe
3. Méthode de Newton

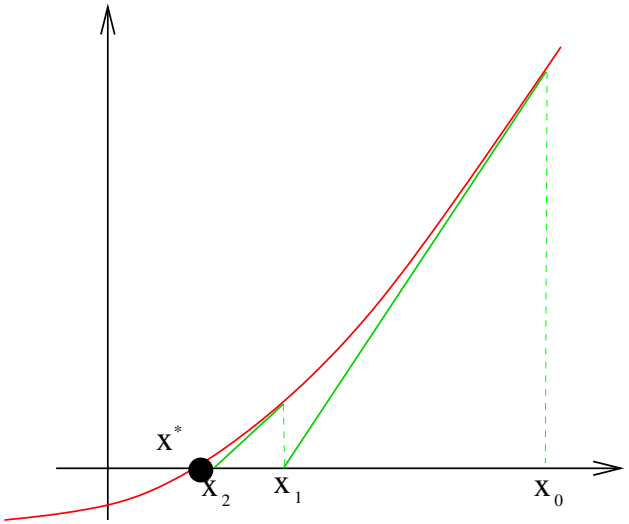
Méthode de Newton : cas dimension 1

Principe

Approcher la fonction f par sa tangente en $(x_0, f(x_0))$:

1) $p(x) = f'(x_0)(x - x_0) + f(x_0)$
2) Résoudre $p(x) = 0$:

$$\Longrightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$



Algorithme (Méthode du Point fixe)

Donnée : $x_0 \in \mathbb{R}^n$, $\epsilon > 0$, $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

While $\|x_{k+1} - x_k\| \leq \epsilon/2$,

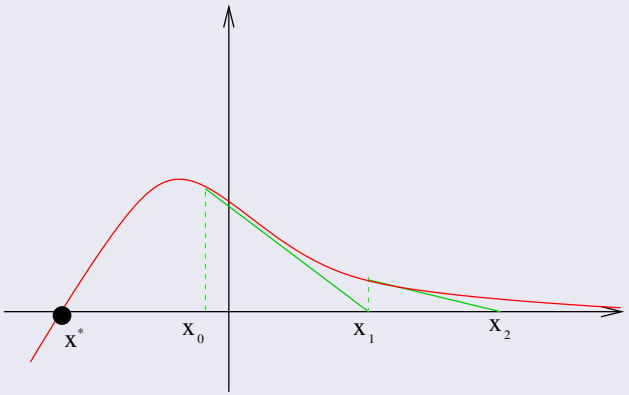
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

End

Méthode de Newton : cas dimension 1

Remarque

Choisir x_0 suffisamment proche de x^*



Remarque

Il s'agit d'une méthode de point fixe avec

$$g(x) = x - \frac{f(x)}{f'(x)}$$

En effet, si $f'(x^*) \neq 0$ alors "$g(x) = x \iff f(x) = 0$".

Théorème (Cas de zéro simple)

Soit $f \in C^2([a,b])$ *et* $x^* \in [a,b]$ *tel que*

$$\begin{cases} f(x^*) = 0 \\ f'(x^*) \neq 0, \end{cases}$$

Alors il existe $\alpha > 0$ *tel que* $\forall x_0 \in I_\alpha = [x^* - \alpha, x^* + \alpha]$, *l'algorithme de Newton converge vers* x^*.
Sa vitesse de convergence est de plus quadratique :

$$|x_{k+1} - x^*| \leq \frac{M}{2}|x_k - x^*|^2, \quad \forall k \in \mathbb{N}$$

où $M = \dfrac{\sup_{x \in I_\alpha} \{|f''(x)|\}}{\inf_{x \in I_\alpha} \{|f'(x)|\}}$.

Démonstration :

Idée : appliquer le théorème de convergence du point fixe local à la fonction $g(x) = x - \frac{f(x)}{f'(x)}$.

- Régularité de g : f étant C^2 et $f'(x^*) \neq 0$, il existe un voisinage $I_\alpha = [x^* - \alpha, x^* + \alpha]$ tel que $g \in C^1(I_\alpha)$.
- Valeur de $g'(x^*)$: en remarquant que

$$g'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2},$$

on en déduit que $|g'(x^*)| = 0$.

- Vitesse de convergence : La formule de Taylor montre que

$$0 = f(x^*) = f(x_k) - f'(x_k)(x^* - x_k) + \frac{1}{2}f''(\xi)(x^* - x_k)^2, \quad \text{avec } \xi \in [x_k, x^*]$$

et

$$x_{k+1} - x^* = (x_k - x^*) - \frac{f(x_k)}{f'(x_k)} = \frac{1}{2}\frac{f''(\xi)}{f'(x_k)}(x^* - x_k)^2$$

Exercice

Ecrire l'algorithme de Newton dans le cas de la fonction $f(x) = x^2 - 2$ et calculer une approximation de $\sqrt{2}$.

Exercice

Ecrire l'algorithme de Newton dans le cas de la fonction $f(x) = exp(x) - 1$. Vérifier que la vitesse de convergence au voisinage de zéro est bien quadratique.

Exercice

Même exercice pour la fonction $f(x) = sin(x)^2$. La vitesse de convergence au voisinage de zéro est-elle quadratique ? Pourquoi ?

Méthode de Newton : Cas de racines multiples

Hypothèses : $f(x) = (x - x^*)^m h(x)$ avec $m > 1$ et h suffisamment régulière.

Exercice

Montrer que l'algorithme de Newton converge mais que sa vitesse de convergence est seulement linéaire.

Exercice

En appliquant l'algorithme de Newton à la fonction $\tilde{f}(x) = f(x)^{1/m}$, montrer que la variante de l'algorithme de Newton

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)},$$

admet cette fois-ci une vitesse de convergence quadratique. Application dans le cas de la fonction $f(x) = sin(x)^2$.

Algorithme de Newton, cas dimension supérieure

Hypothèse : Soit $f : \mathbb{R}^n \to \mathbb{R}^n$ de régularité C^2.
Principe : Le développement de Taylor montre que

$$f(x) = f(x_0) + Df(x_0)(x - x_0) + O(\|x - x_0\|^2)$$

où

$$Df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Algorithme (Algorithme de Newton)

Donnée : $x_0 \in \mathbb{R}^n$, $\epsilon > 0$, $x_1 = x_0 - Df^{-1}(x_0)f(x_0)$
While $\|x_{k+1} - x_k\| \leq \epsilon/2$,
$x_{k+1} = x_k - Df^{-1}(x_k)f(x_k)$
End

Théorème

Soit $f : \mathbb{R}^d \to \mathbb{R}^d$ de régularité C^2 et x^* un point fixe de f tel que $Df(x^*)$ soit inversible. Alors il existe un réel $\alpha > 0$ tel que $\forall x_0 \in B(x^*, \alpha)$, l'algorithme de Newton converge vers x^*. Sa vitesse de convergence est de plus quadratique.

Exercice

Déterminer les solutions de

$$F(X) = \begin{pmatrix} x_1^2 + x_2^2 - 2 \\ x_1^2 - x_2^2 \end{pmatrix} = 0,$$

puis calculer 2 itérations de l'algorithme de Newton en partant de

$$X_0 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Exemple : Méthode des Moindres carrés non linéaires

On cherche la fonction de la forme

$$g(\alpha, \beta, x) = \alpha x^{\beta}$$

qui minimise l'énergie

$$\phi(\alpha, \beta) = \sum_{i=1}^{n} \left(y_i - \alpha x_i^{\beta}\right)^2 .$$

Exercice

- *Montrer que les coefficients optimaux* (α, β) *vérifient*

$$F(\alpha, \beta) = \begin{pmatrix} \alpha \sum_{i=1}^{n} (x_i^{\beta})^2 - \sum_{i=1}^{n} y_i x_i^{\beta} \\ \alpha^2 \sum_{i=1}^{n} ln(x_i)(x_i^{\beta})^2 - \alpha \sum_{i=1}^{n} y_i x_i^{\beta} ln(x_i) \end{pmatrix} = 0,$$

- *Expliciter l'algorithme de Newton pour déterminer les coefficients* (α, β).

Chapitre 6 : Méthodes numériques pour les équations différentielles ordinaires (EDO)

...

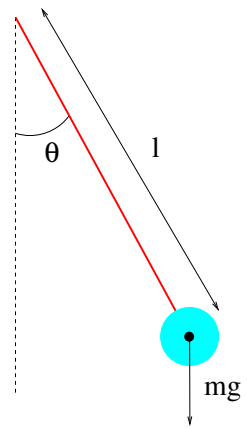
Ce chapitre s'intéresse aux méthodes numériques pour la résolution d'équations différentielles ordinaires (EDO)

$$(1) \quad \begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \in \mathbb{R}^n, \end{cases}$$

où $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ est supposée suffisamment régulière.

Exemple: Le pendule

$$\begin{cases} ml\theta'' = -\alpha\ell\theta' - mgsin(\theta) \\ \theta(0) = \theta_0, \text{ et } \theta'(0) = \theta'_0, \end{cases}$$



Le problème peut se réécrire sous la forme (1) avec

$$y = \begin{pmatrix} \theta \\ \theta' \end{pmatrix}, \quad f(t, y) = \begin{pmatrix} y_2 \\ -\frac{\alpha}{m} y_2 - \frac{g}{\ell} sin(y_1) \end{pmatrix}, \quad \text{et} \quad y(0) = \begin{pmatrix} \theta_0 \\ \theta'_0 \end{pmatrix},$$

Existence + unicité de solution

Soit

$$(1) \quad \begin{cases} y'(t) = f(t, y(t)), & \forall t \in [a, b] \\ y(t_0) = y_0 \in \mathbb{R}^n, & \text{avec } t_0 \in [a, b] \end{cases}$$

Théorème

Hypothèses : *On suppose que* $f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n$ *est continue, localement Lipschitzienne par rapport à la deuxième variable, i.e,* $\forall (u, v) \in (\mathbb{R}^2)^2$ *et* $\forall t \in [a, b]$, *il existe un réel* $L > 0$ *tel que*

$$\|f(t, u) - f(t, v)\| \leq L\|u - v\|.$$

Alors (1) *admet une unique solution* C^1 *sur* $I_\alpha = [t_0 - \alpha, t_0 + \alpha]$.

Exercice

Exemple avec

$$\begin{cases} y'(t) = \sqrt{|y(t)|}, & \forall t \in \mathbb{R} \\ y(0) = 0 \end{cases}$$

La fonction $f(t,u) = \sqrt{|u|}$ n'est pas lipschitzienne par rapport à u ! Montrer que cette équation admet plusieurs solutions !

Exercice

Exemple avec

$$\begin{cases} y'(t) = y(t)^2, & \forall t \in \mathbb{R} \\ y(0) = y_0 > 0 \end{cases}$$

La fonction f associée est bien continue et localement lipschitz. Montrer que ce problème admet pour solution

$$y(t) = \frac{1}{y_0^{-1} + t_0 - t}, \quad \textit{pour} \quad t < t_0 + y_0^{-1}.$$

Discrétisation de l'équation

On suppose que le problème

$$(1) \quad \begin{cases} y'(t) = f(t, y(t)), & \forall t \in [a, b] \\ y(t_0) = y_0 \in \mathbb{R}^n, & \text{avec } t_0 \in [a, b] \end{cases}$$

admet une solution y suffisamment régulière sur $[t_0, t_0 + T]$.

- Discrétisation de l'intervalle $[t_0, t_0 + T]$ avec N points :
  Avec $h = \frac{T}{N}$, on pose $t_n = t_0 + hn$ pour tout $n = 0 : N$.
- Approximation numérique de y aux noeuds t_n

$$y_n \simeq y(t_n).$$

- Schéma numérique de la forme

$$y_{n+1} = y_n + h\phi(t_n, y_n).$$

Convergence d'une méthode numérique

Définition (Consistance et ordre d'une méthode numérique)

Avec

$$\tau_{n+1}(h) = \frac{y(t_{n+1}) - y(t_n)}{h} - \phi(t_n, y(t_n)),$$

La méthode numérique est dite consistante si $\lim_{h\to 0} \tau_{n+1}(h) = 0$.
Elle est de plus d'ordre p si

$$\|\tau_{n+1}(h)\| \leq Ch^p.$$

Définition (Stabilité)

La stabilité d'un schéma numérique consiste à contrôler, pour un pas de temps δ_t *fixé, l'erreur*

$$e_n = y(t_n) - y_n,$$

au cours des itérations.

1 Méthodes d'Euler explicite-implicite

2 Méthodes d'ordre supérieur

Méthode d'Euler explicite

Idée :

$$y(t_1) - y(t_0) = \int_{t_0}^{t_1} y'(t)dt = \int_{t_0}^{t_1} f(t, y(t)dt \simeq hf(t_0, y(t_0))$$

Algorithme (Euler explicite)

Donnée : $y_0 \in \mathbb{R}^n$,

for $k = 0 : N - 1$

$$y_{n+1} = y_n + h\, f(t_n, y_n)$$

End

Euler explicite : Consistance et ordre de la méthode

On remarque que

$$\begin{aligned}\tau_{n+1}(h) &= \frac{1}{h}\left[y(t_{n+1}) - y(t_n) - hf(t_n, y(t_n))\right] \\ &= \frac{1}{h}\left[y(t_n + h) - y(t_n) - hy'(t_n)\right] \\ &= \frac{1}{h}h^2\frac{1}{2}y''(\eta) \quad \text{avec } \eta \in [t_n, t_{n+1}],\end{aligned}$$

Avec $M = \sup_{t\in[t_0,t_0+T]}\{|y''(t)|\}$, on en déduit que

Théorème (Consistance méthode Euler explicite)

$$|\tau_{n+1}(h)| \leq \frac{M}{2}h.$$

La méthode d'Euler explicite est donc consistante et d'ordre 1.

Euler explicite : Stabilité

Avec $e_n = y(t_n) - y_n$, on montre que

$$\begin{aligned}
e_{n+1} &= y(t_{n+1}) - y_{n+1} \\
&= y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt - (y_n + hf(t_n, y_n)) \\
&= e_n + \int_{t_n}^{t_{n+1}} f(t, y(t))dt - hf(t_n, y_n) \\
&= e_n + \left[\int_{t_n}^{t_{n+1}} f(t, y(t))dt - hf(t_n, y(t_n))\right] + h\left[f(t_n, y(t_n)) - f(t_n, y_n)\right] \\
&= e_n + h\tau_{n+1}(h) + h\left[f(t_n, y(t_n)) - f(t_n, y_n)\right]
\end{aligned}$$

On rappelle que f est L-localement Lipschitzienne et donc

$$|e_{n+1}| \leq (1 + Lh)|e_n| + \frac{h^2}{2}M.$$

Euler explicite : Stabilité

Au final,

$$\begin{aligned}
|e_{n+1}| &\leq (1+Lh)|e_n| + \frac{h^2}{2}M \\
&\leq (1+Lh)^2|e_{n-1}| + \frac{h^2}{2}M(1+(1+h\,L)) \\
&\leq (1+Lh)^n|e_0| + \frac{h^2}{2}M\sum_{k=0}^{n-1}(1+h\,L)^k \\
&\leq exp(Lt_n)|e_0| + \frac{h}{2}M\frac{1}{L}\left[exp(t_nL)-1\right]
\end{aligned}$$

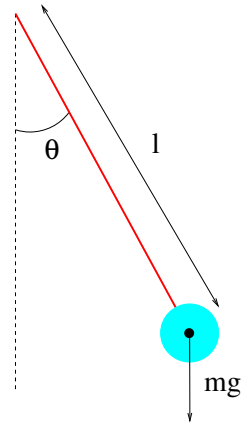
On en déduit le théorème suivant

Théorème (Stabilité algorithme Euler explicite)

$$\|y(t_n)-y_n\| \leq exp(LT)\|y(t_0)-y_0\| + \frac{Mh}{2}T\ exp(LT)$$

Exemple: Le pendule

$$\begin{cases} ml\theta'' = -\alpha\ell\theta' - mgsin(\theta) \\ \theta(0) = \theta_0, \text{ et } \theta'(0) = \theta'_0, \end{cases}$$



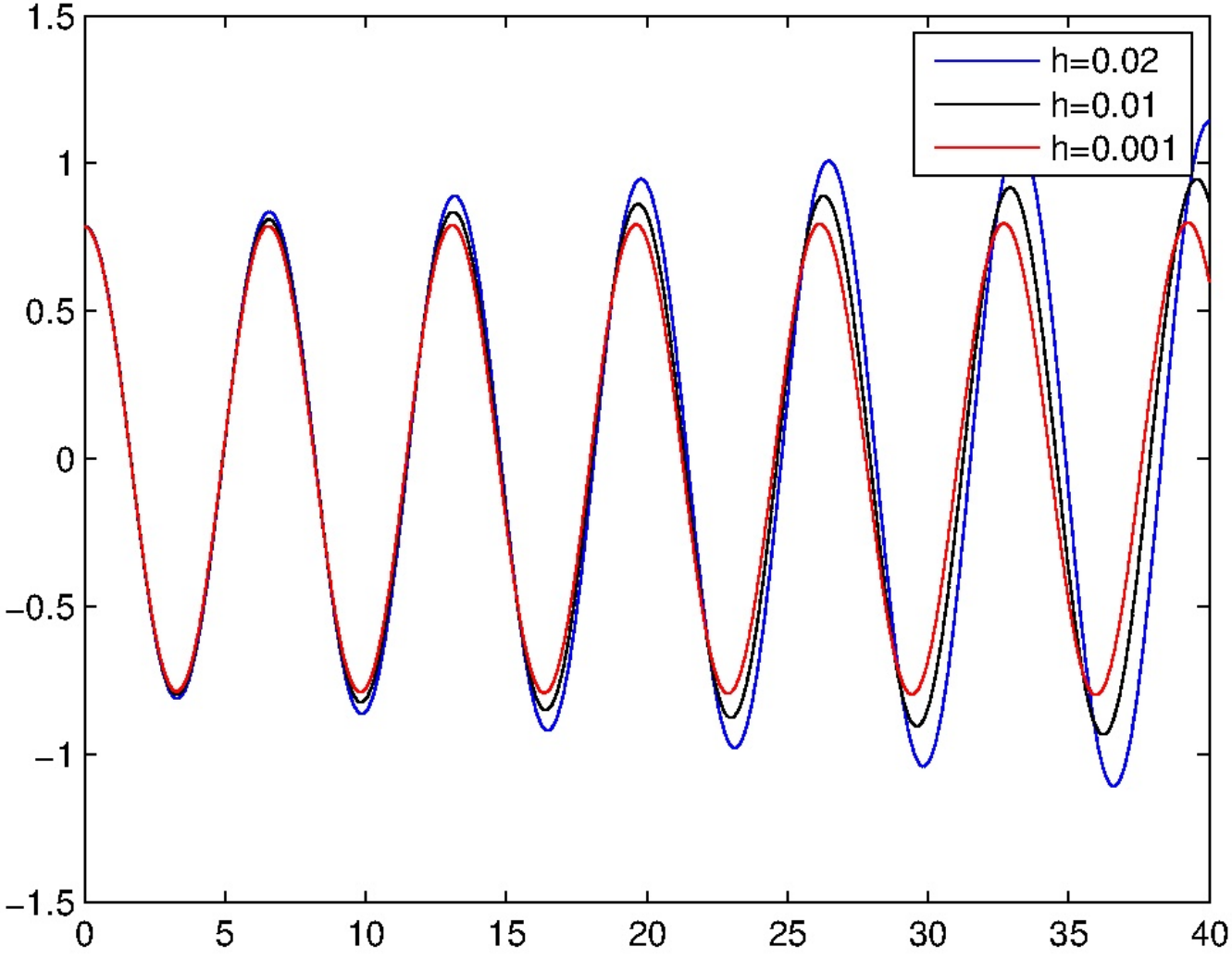
Avec $y = \begin{pmatrix} \theta \\ \theta' \end{pmatrix}$, cette équation se réécrit sous la forme

$$y'(t) = f(t, y) = \begin{pmatrix} y_2 \\ -\frac{\alpha}{m} y_2 - \frac{g}{\ell} sin(y_1) \end{pmatrix},$$

Exercice

Écrire l'algorithme d'Euler explicite dans le cas particulier de l'équation du pendule.

Le pendule sans frottement : simulation numérique avec la méthode d'Euler explicite



Méthode d'Euler implicite

Idée :

$$y(t_1) - y(t_0) = \int_{t_0}^{t_1} y'(t)dt = \int_{t_0}^{t_1} f(t, y(t)dt \simeq hf(t_1, y(t_1))$$

Algorithme (Euler explicite)

Donnée : $y_0 \in \mathbb{R}^n$,

for $k = 0 : N - 1$

Déterminer y_{n+1}, *solution de*

$$F(y) = y - h\, f(t_{n+1}, y) - y_n = 0$$

End

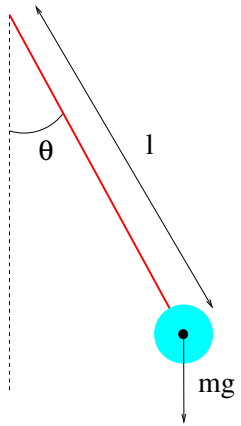
Exercice

Montrer que le schéma d'Euler explicite est bien consistant d'ordre 1

Équation du pendule

Exemple: Le pendule

$$\begin{cases} ml\theta'' = -\alpha\ell\theta' - mgsin(\theta) \\ \theta(0) = \theta_0, \text{ et } \theta'(0) = \theta'_0, \end{cases}$$



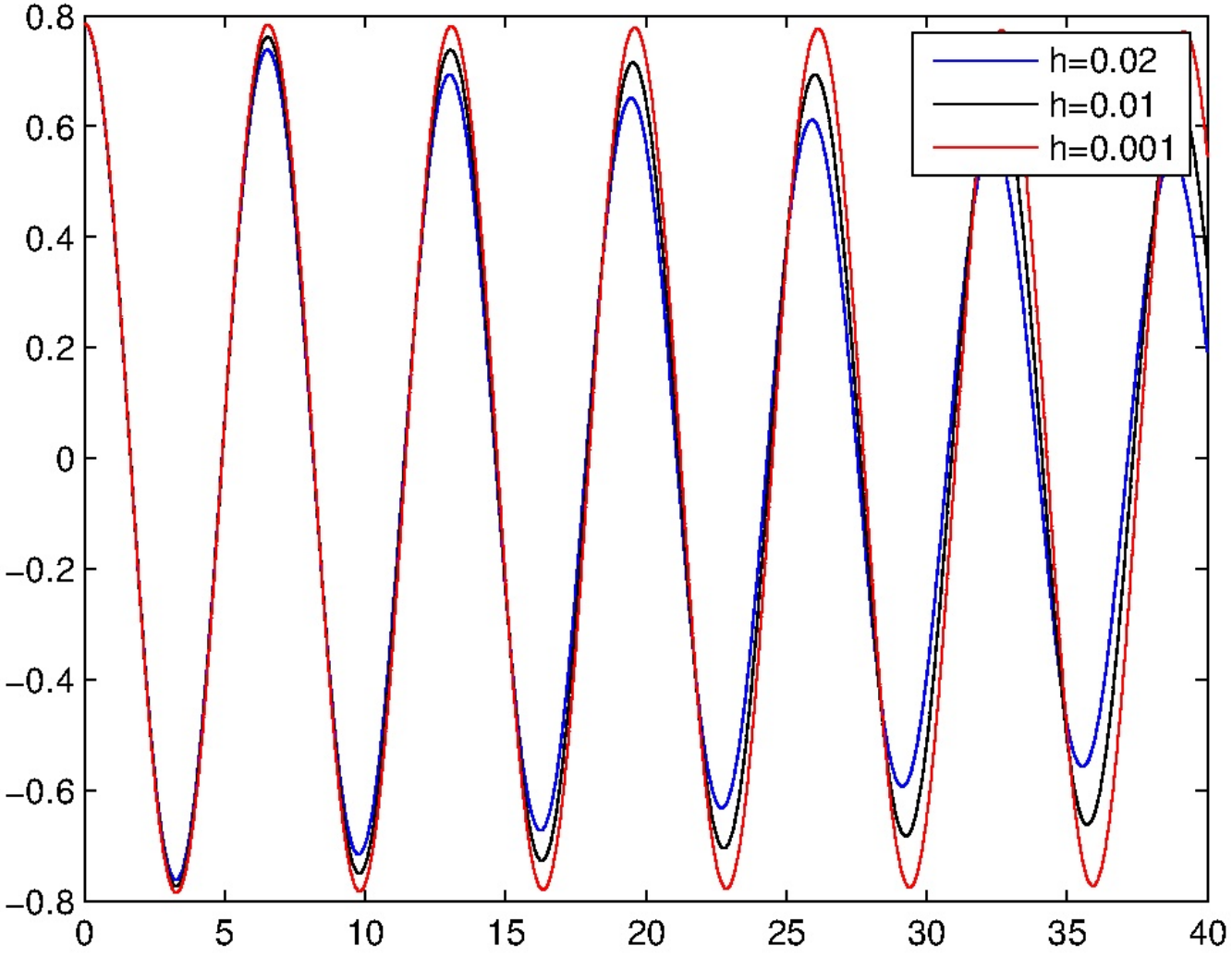
Avec $y = \begin{pmatrix} \theta \\ \theta' \end{pmatrix}$, cette équation se réécrit sous la forme

$$y'(t) = f(t, y) = \begin{pmatrix} y_2 \\ -\frac{\alpha}{m} y_2 - \frac{g}{\ell} sin(y_1) \end{pmatrix},$$

Exercice

Écrire l'algorithme d'Euler implicite dans le cas particulier de l'équation du pendule. On utilisera l'algorithme de Newton pour évaluer y_{n+1} en fonction de y_n.

Le pendule sans frottement : simulation numérique avec la méthode d'Euler implicite



Comparaison Euler Explicite-implicite sur un système raide

On s'intéresse au problème suivant

$$\begin{cases} y'(t) + Ly(t) = 0, \quad \forall t \in \mathbb{R}^+ \\ y(0) = y_0 \in \mathbb{R}^n \end{cases}$$

- Calculer la solution exacte de ce problème
- Déterminer l'approximation numérique y_n par la méthode d'Euler Explicite.
- Déterminer l'approximation numérique y_n par la méthode d'Euler implicite.

1 Méthodes d'Euler explicite-implicite

2 Méthodes d'ordre supérieur

Schémas numériques d'ordre supérieur à 1 pas

On suppose que le problème

$$(1) \quad \begin{cases} y'(t) = f(t, y(t)), & \forall t \in [a, b] \\ y(t_0) = y_0 \in \mathbb{R}^n, & \text{avec } t_0 \in [a, b] \end{cases}$$

admet une solution y suffisamment régulière sur $[t_0, t_0 + T]$.

- Discrétisation de l'intervalle $[t_0, t_0 + T]$ avec N points :
  Avec $h = \frac{T}{N}$, on pose $t_n = t_0 + hn$ pour tout $n = 0 : N$.
- Approximation numérique de y aux noeuds t_n

$$y_n \simeq y(t_n).$$

- Schéma numérique de la forme

$$y_{n+1} = y_n + h\phi(t_n, y_n).$$

Objectif : Trouver des schémas numériques d'ordre supérieur à 1.

Méthode de Taylor

Idée :

$$\begin{aligned} y(t_{n+1}) &= y(t_n + h) = y(t_n) + y'(t_n)h + y''(t_n)\frac{h^2}{2} + O(h^3) \\ &= y(t_n) + f(t_n, y(t_n))h + \frac{d}{dt}\left[f(t, y(t))\right]_{|t=t_n}\frac{h^2}{2} + O(h^3), \end{aligned}$$

où

$$\begin{aligned} \frac{d}{dt}\left[f(t, y(t))\right] &= \partial_t f(t, y(t)) + \partial_y f(t, y(t))y'(t) \\ &= \partial_t f(t, y(t)) + \partial_y f(t, y(t))f(t, y(t)) \end{aligned}$$

On en déduit que

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + f(t_n, y(t_n))h + \frac{h^2}{2}\partial_t f(t_n, y(t_n)) \\ &\quad + \frac{h^2}{2}\partial_y f(t_n, y(t_n))f(t_n, y(t_n)) + O(h^3) \end{aligned}$$

Méthode de Taylor

Algorithme

Donnée : $y_0 \in \mathbb{R}^n$,

For $k = 0 : N - 1$

$$y_{n+1} = y_n + f(t_n, y_n))h + \frac{h^2}{2}\left[\partial_t f(t_n, y_n) + \partial_y f(t_n, y_n) f(t_n, y_n)\right]$$

End

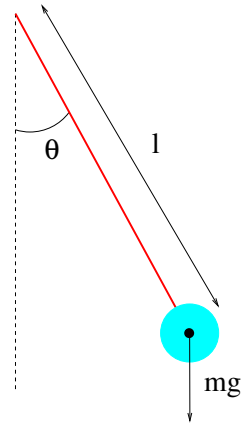
- Consistance : Ordre 2 : $|\tau_{n+1}| \leq Ch^2$
- Extension dimension supérieure :
  Utiliser les dérivées partielles d'ordre supérieur

$$\partial^2_{tt} f, \quad \partial^2_{ty} f, \quad \text{et} \quad \partial^2_{yy} f \quad \ldots$$

- Difficultés : Pas de connaissance systématique de ces dérivées partielles !

Exemple: Le pendule

$$\begin{cases} ml\theta'' = -\alpha\ell\theta' - mgsin(\theta) \\ \theta(0) = \theta_0, \text{ et } \theta'(0) = \theta'_0, \end{cases}$$



Avec $y = \begin{pmatrix} \theta \\ \theta' \end{pmatrix}$, cette équation se réécrit sous la forme

$$y'(t) = f(t, y) = \begin{pmatrix} y_2 \\ -\frac{\alpha}{m} y_2 - \frac{g}{\ell} sin(y_1) \end{pmatrix},$$

Exercice

Écrire l'algorithme de Taylor d'ordre 2 dans le cas particulier de l'équation du pendule.

Méthode de Runge Kutta d'ordre 2

Idée : On souhaite déterminer des coefficients (a_1, a_2, a_3, a_4) tels que

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + a_1 h\, f(t_n, y(t_n)) \\ &\quad + a_2 h\, f(t_n + a_3 h, y(t_n) + a_4 h) + O(h^3) \end{aligned}$$

En remarquant que

$$f(t_n + a_3 h, y(t_n) + a_4 h) = f(t_n, y(t_n)) + a_3 h \partial_t f(t_n, y(t_n)) + a_4 h \partial_y f(t_n, y(t_n))$$

On en déduit que

$$\begin{aligned} A &= y(t_n) + a_1 h\, f(t_n, y(t_n)) + a_2 h\, f(t_n + a_3 h, y(t_n) + a_4 h) \\ &= y(t_n) + (a_1 + a_2) h\, f(t_n, y(t_n)) + a_2 a_3 h^2 \partial_t f(t_n, y(t_n)) \\ &\quad + a_2 a_4 h^2 \partial_y f(t_n, y(t_n)) + O(h^3) \end{aligned}$$

On rappelle que le développement de Taylor montre

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + f(t_n, y(t_n))h + \frac{h^2}{2}\partial_t f(t_n, y(t_n)) \\ &\quad + \frac{h^2}{2}\partial_y f(t_n, y(t_n)) f(t_n, y(t_n)) + O(h^3). \end{aligned}$$

Les coefficients (a_1, a_2, a_3, a_4) doivent donc satisfaire les conditions suivantes :

$$\begin{cases} a_1 + a_2 &= 1 \\ a_2 a_3 &= 1/2 \\ a_2 a_4 &= f(t_n, y(t_n))/2 \end{cases}$$

Méthode d'Euler modifiée (RK2)

Choix des coefficients (a_1, a_2, a_3, a_4) :

$a_1 = a_2 = \frac{1}{2}, \quad a_3 = 1, \quad$ et $\quad a_4 = f(t_n, y(t_n))$

Algorithme (Runge Kutta 2)

Donnée : $y_0 \in \mathbb{R}^n$,

for $k = 0 : N - 1$

$k_1 = y_n + hf(t_n, y_n)$

$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_n + h, k_1)\right)$

End

Lien avec la méthode des trapèzes

$$
\begin{aligned}
y(t_{n+1}) - y(t_n) &= \int_{t_n}^{t_{n+1}} f(t, y(t))dt \\
&= \frac{h}{2}\left[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\right] + O(h^2) \\
&= \frac{h}{2}\left[f(t_n, y(t_n)) + f(t_{n+1}, y_n + hf(t_n, y(t_n)))\right] + O(h^2)
\end{aligned}
$$

Méthode du point du milieu

Choix des coefficients (a_1, a_2, a_3, a_4) :

$a_1 = 0, \quad a_2 = 1, \quad a_3 = 1/2, \quad$ et $\quad a_4 = f(t_n, y(t_n))/2$

Algorithme (Méthode du point du milieu)

Donnée : $y_0 \in \mathbb{R}^n$,

for $k = 0 : N - 1$

$k_1 = y_n + h/2\, f(t_n, y_n)$

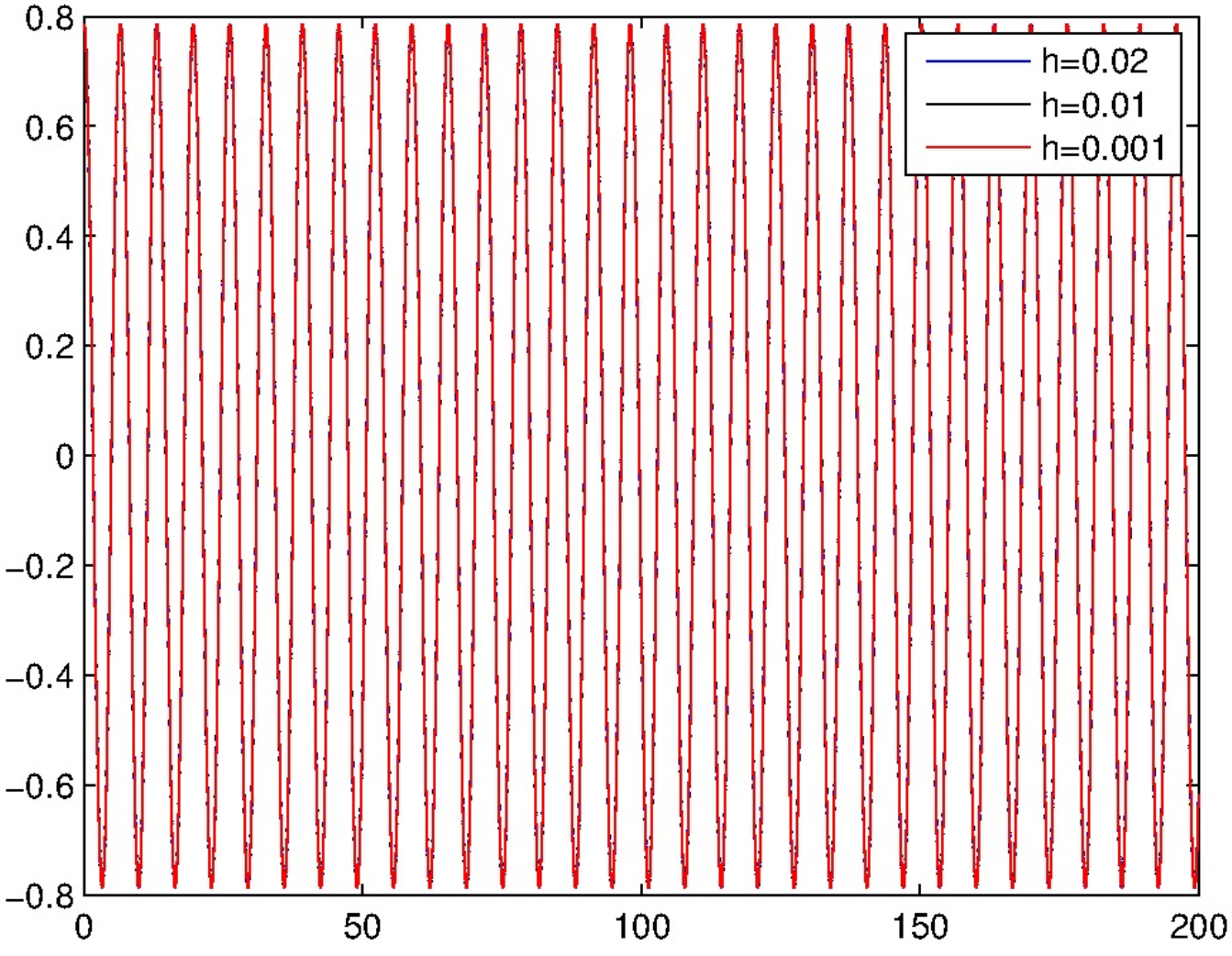
$y_{n+1} = y_n + hf(t_n + h/2, k_1)$

End

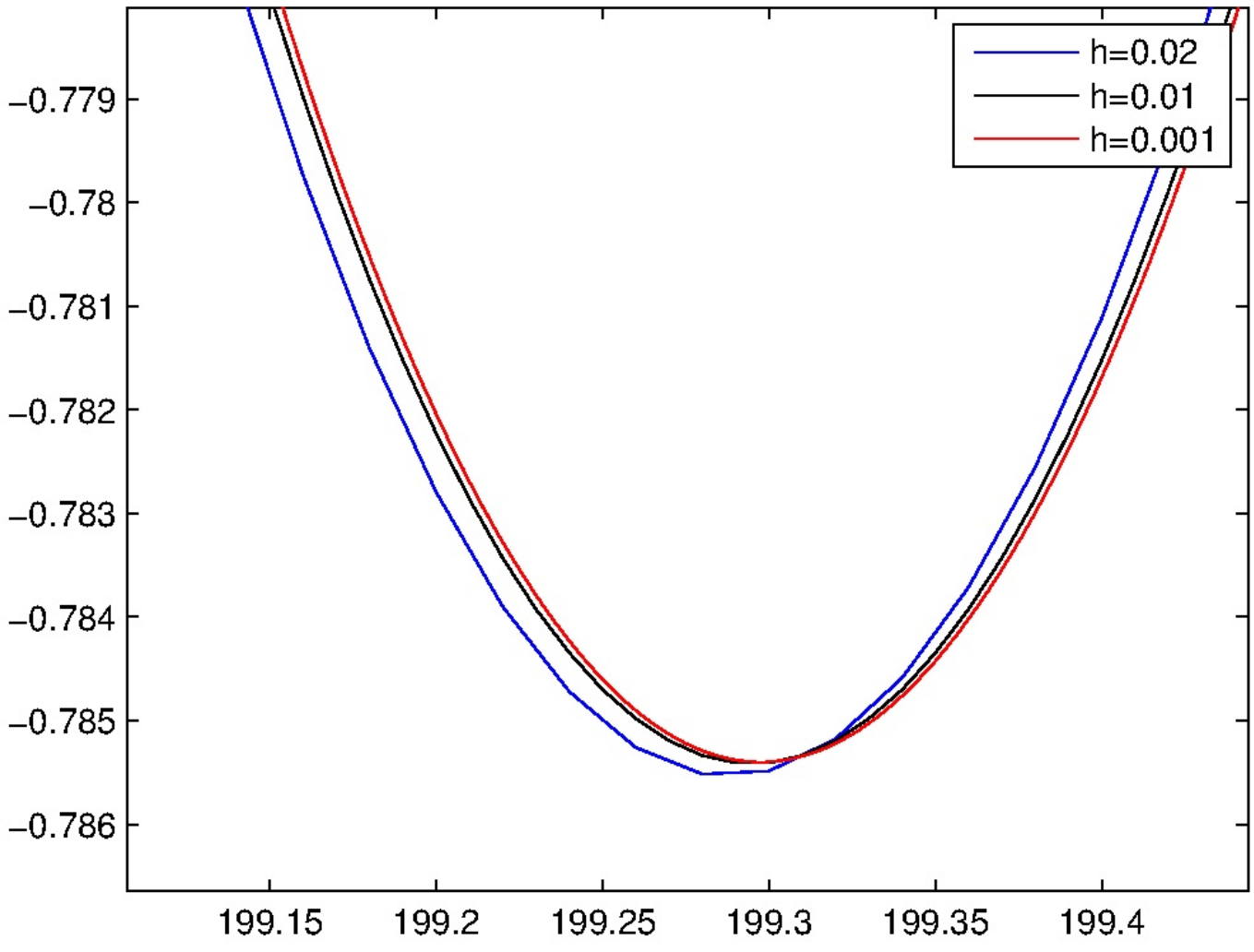
Lien avec la méthode des rectangles

$$
\begin{aligned}
y(t_{n+1}) - y(t_n) &= \int_{t_n}^{t_{n+1}} f(t, y(t))dt = hf(t_n + h/2, y(t_n + h/2)) + O(h^2) \\
&= h\, f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2} f(t_n, y(t_n))\right) + O(h^2)
\end{aligned}
$$

Le pendule sans frottement : simulation numérique avec la méthode RK2



Le pendule sans frottement : Exemple simulation numérique avec la méthode RK2



Méthode Runge Kutta d'ordre 4 (RK4)

Même idée avec des développements de Taylor d'ordre supérieur :

Algorithme (Méthode RK4)

Donnée : $y_0 \in \mathbb{R}^n$,

for $k = 0 : N - 1$

$$
\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + h/2, y_n + h/2k_1) \\
k_3 &= f(t_n + h/2, y_n + h/2k_2) \\
k_3 &= f(t_n + h, y_n + hk_3) \\
y_{n+1} &= y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

End

- Méthode d'ordre 4, très utilisée en pratique ...
- Voir le lien avec l'intégration de Simpson !

Méthodes d'ordre supérieur multi-pas : formule D'Adams-Bashforth

Idée : Utiliser les estimations de $y_n, y_{n-1}, y_{n-2} ... y_{n-k}$ pour obtenir une approximation de y_{n+1}.

Exemple avec k=1

Interpolation linéaire de $y'(t) = f(t, y(t))$ au noeuds t_n et t_{n-1}. Avec $f_n = f(t_n, y_n)$, alors

$$y'(t) = f(t, y(t)) \simeq f_{n-1} \frac{t - t_n}{h} + f_n \frac{t - t_{n-1}}{h},$$

et finalement,

$$\begin{aligned} y(t_{n-1}) - y(t_n) &= \int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t, y(t))dt \\ &\simeq \int_{t_n}^{t_{n+1}} f_{n-1} \frac{t - t_n}{h} + f_n \frac{t - t_{n-1}}{h} dt \end{aligned}$$

Méthode d'ordre supérieur multi-pas : formule D'Adams-Bashforth

Exercice

Montrer que

$$y(t_{n-1}) - y(t_n) \simeq \frac{h}{2}\left[3f_n - f_{n-1}\right]$$

Algorithme (Méthode d'Adams-Bashforth d'ordre 2)

Donnée : $y_0 \in \mathbb{R}^n$,

for $k = 0 : N - 1$

$f_n = f(t_n, y_n)$

$y_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1})$

End

- Extension ordre supérieur // explicite-implicite !

Utilisation d'un pas adaptatif

Motivation : L'utilisation d'un pas de temps h uniforme est très coûteux dans certain cas, et il devient plus efficace d'utiliser un pas de temps h_n adaptatif, choisi plus petit dans les zones où y évolue fortement !

Rappel :

Schéma numérique : $y_{n+1} = y_n + h_n \phi(t_n, y_n)$

Consistance : $\tau_n(h_n) \simeq \omega_n h_n^p + O(h_n^{p+1})$

Consistance : $\max_n |e_n| \leq C(T) \sum_n h_n |\tau_n(h_n)|$

Estimation de ω_n à l'instant t_n

Avec

$$\begin{aligned} y^h_{n+1} &= y_n + h\phi(t_n, y_n) \\ y^{h/2}_{n+1} &= y^{h/2}_{n+1/2} + \frac{h}{2}\phi(t_n, y^{h/2}_{n+1/2}), \text{ où } y^{h/2}_{n+1/2} = y_n + h\phi(t_n, y_n) \end{aligned}$$

alors

$$\omega_n \simeq \frac{y^{h/2}_{n+1} - y^h_{n+1}}{h^{p+1}(1 - 2^{-p})}.$$

Déterminer h_n afin de satisfaire $\max_n |e_n| \leq \epsilon$

Il suffit d'imposer

$$C(T)|\tau_n| \leq \epsilon / T.$$

En effet

$$\max_n |e_n| \leq C(T) \sum_n h_n |\tau_n(h_n)| \leq \sum h_n \frac{\epsilon}{T} \leq \epsilon$$

Alors

$$|\tau_n| \leq \frac{\epsilon}{TC(T)} \Longleftrightarrow h_n \lesssim \left(\frac{\epsilon}{TC(T)|\omega_n|}\right)^{1/p}$$

Chapitre 7 : Méthodes numériques pour les équations aux dérivées partielles (EDP)

...

Ce chapitre s'intéresse aux méthodes des différences finies pour la résolution d'équations aux dérivées partielles. On regardera en particulier le cas des équations suivantes :

Equation de Poisson :

$$-\Delta u(x) = f(x)$$

Equation de la chaleur :

$$\partial_t u(x,t) = \Delta u$$

Equation de transport :

$$\partial_t u(x,t) + v.\nabla u = 0$$

Equation des ondes :

$$\partial_{tt}^2 u(x,t) = c^2 \Delta u$$

Méthodes des différences finies

Approximation des dérivées d'ordre 1 :

Schéma centré :

$$u'(x) = \frac{u(x+h) - u(x-h)}{2h} + O(h^2)$$

Schéma décentré à gauche :

$$u'(x) = \frac{u(x) - u(x-h)}{h} + O(h)$$

Schéma décentré à droite :

$$u'(x) = \frac{u(x+h) - u(x)}{h} + O(h)$$

Approximation des dérivées d'ordre 2 :

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x+h)}{h^2} + O(h^2)$$

Exercice

Montrer les approximations précédentes en explicitant les constantes dans les $O(h)$ et $O(h^2)$

1. Exemple d'un problème aux limites : l'équation de Poisson
2. Exemple d'un problème d'évolution : l'équation de la chaleur
3. Exemple d'un problème d'évolution : l'équation de transport
4. Exemple d'un problème d'évolution : l'équation des ondes

Equation de Poisson

On s'intéresse à la résolution numérique de l'équation

$$(P_1) \quad \begin{cases} -u''(x) = f(x) \quad \text{pour} \quad x \in [0,1] \\ u(0) = g_0, \quad u(1) = g_1 \end{cases}$$

Discrétisation de l'intervalle $[0,1]$ avec $N+2$ points :
On pose $x_i = i\,h$ pour $i \in [0, N+1]$ et où $h = \frac{1}{N+1}$.

Approximation numérique de u et f aux noeuds x_i

$$\begin{cases} u_i \simeq u(x_i), \quad u_0 = g_0 \text{ et } u_{N+1} = g_1 \\ f_i = f(x_i) \end{cases}$$

Approximation de l'EDP :

$$-\frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = f_i, \quad \forall i = 1 : N$$

Approximation de l'équation de Poisson

Avec

$$u_h = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}, b_h = \begin{pmatrix} f_1 + g_0/h^2 \\ f_2 \\ \vdots \\ f_{N-1} \\ f_N + g1/h^2 \end{pmatrix}, A_h = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}$$

La solution u_h s'obtient en résolvant le système linéaire

$$A_h u_h = b_h.$$

Propriété

La matrice A_h est symétrique définie positive.

Preuve Montrer que $< Z, A_h Z >= \frac{z_1^1 + (z_1 - z_2)^2 + \cdots + (z_{N-1} - z_N)^2 + z_N^2}{h^2}$

Propriétés de la matrice A_h

Exercice

Montrer que les vecteurs $\{v^n\}_{n=1:N}$ *dont les composantes sont définies par* $v_i^n = sin(n\pi x_i)$, *sont des vecteurs propres de* A_h *associés respectivement aux valeurs propres*

$$\lambda_h^n = \frac{2}{h^2}\left(1 - cos(n\pi h)\right).$$

En déduire que pour N suffisamment grand, les valeurs propres λ_n *de* A_h *sont comprises entre*

$$\lambda_h^1 \simeq \pi^2 \text{ et } \lambda_h^N \simeq \frac{4}{h^2},$$

et que

$$\|A_h\|_2 \simeq \frac{4}{h^2} \quad et \quad \|A_h^{-1}\|_2 \simeq \frac{1}{\pi^2}.$$

On admet aussi par la suite que

$$\|A_h^{-1}\|_\infty \leq \frac{1}{8}.$$

Consistance de la méthode des différences finies

Soit u la solution du problème (P_1) et $\overline{u_h}$ le vecteur défini par $(\overline{u_h})_i = u(x_i)$.

Définition

Le schéma numérique est dit consistant si

$$\Pi_h(u) = A_h\overline{u_h} - b_h \to 0 \quad \textit{quand } h \to 0.$$

En particulier, le schéma est d'ordre p pour une norme $\|.\|$ *donnée si*

$$\|\Pi_h(u)\| \leq Ch^p.$$

Propriété

Supposons que la solution u de (P_1) *soit de régularité* $C^4([0,1])$. *Alors le schéma est consistant d'ordre 2 pour la norme* $\|.\|_\infty$, *i.e.*

$$\|\Pi_h(u)\|_\infty \leq \frac{h^2}{12} \sup_{x\in[0,1]} \left\{|u^{(4)}(x)|\right\}$$

Convergence de la méthode des différences finies

Démonstration : utiliser un développement de Taylor de u :

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + 2\frac{h^2}{4!}u^{(4)}(\eta_x), \text{ avec } \eta_x \in (x-h, x+h).$$

Théorème

Supposons que la solution u de (P_1) soit de régularité $C^4([0,1])$. Alors, le schéma est convergent d'ordre 2 pour la norme $\|.\|_\infty$, i.e.

$$\|u_h - \overline{u}_h\|_\infty \leq \frac{h^2}{96} \sup_{x\in[0,1]} \left\{|u^{(4)}(x)|\right\}$$

Démonstration :

Utiliser la propriété précédente et $\|A_h^{-1}\|_\infty \leq \frac{1}{8}$

1. Exemple d'un problème aux limites : l'équation de Poisson
2. Exemple d'un problème d'évolution : l'équation de la chaleur
3. Exemple d'un problème d'évolution : l'équation de transport
4. Exemple d'un problème d'évolution : l'équation des ondes

Equation de la chaleur

On s'intéresse à la résolution numérique de l'équation de la chaleur

$$(P_2) \quad \begin{cases} \partial_t u(x,t) = \Delta u(x,t) + f(x,t), & \text{pour} \quad (x,t) \in [0,T] \times [0,1] \\ u(x,0) = u_0, \quad \text{pour } x \in [0,1] \\ u(0,t) = u(1,t) = 0, \quad \text{pour } t \in [0,T] \end{cases}$$

- Discrétisation de l'intervalle $[0,1]$ avec $N+2$ points :
  On pose $x_i = i\,h$ pour $i \in [0 : N+1]$ et où $h = \frac{1}{N+1}$.
- Discrétisation de l'intervalle $[0,T]$ avec $M+1$ points :
  On pose $t_n = n\,\delta_t$ pour $n \in [0 : M]$ et où $\delta_t = \frac{1}{M}$.
- Approximation numérique de u et f aux noeuds (x_i, t_n)

$$u_i^n \simeq u(x_i,t_n), \quad f_i^n = f(x_i,t_n), \quad u_0^n = u_{M+1}^n = 0, \text{ et } u_i^0 = u_0(x_i)$$

Méthode d'Euler explicite

Approximation de l'EDP :

$$\frac{u_i^{n+1} - u_i^n}{\delta_t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} + f_i^n,$$

$\forall (i,n) \in [1:N] \times [0, M-1]$.

Schéma numérique :

$$u_h^{n+1} = (I_d - \delta_t A_h) u_h^n + \delta_t f_h^n,$$

où

$$u_h^n = (u_1^n, u_2^n, \cdots, u_N^n)^t, \text{ et } f_h^n = (f_1^n, f_2^n, \cdots, f_N^n)^t,$$

Méthode d'Euler explicite

On note $\overline{u}_h^n$ le vecteur défini par $(\overline{u}_h^n)_i = u(x_i, t_n)$ où u est la solution du problème (P_2).

Propriété

Supposons que la solution u de (P_2) *soit de régularité* $C^2([0,T], C^4([0,1]))$. *Alors le schéma est consistant d'ordre* 2 *en espace et d'ordre* 1 *en temps pour la norme* $\|.\|_\infty$, *i.e.*

$$\|\Pi_h^n(u)\|_\infty \leq C\left(\delta_t + h^2\right),$$

où

$$\Pi_h^n(u) = (\overline{u}_h^{n+1} - \overline{u}_h^{n+1})/\delta_t + \delta_t A_h \overline{u}_h^n - f_h^n$$

Démonstration : Utiliser un développement de Taylor.

Propriété

Le schéma d'Euler explicite est L^2-stable sous la condition

$$\delta_t \leq \frac{1}{2} h^2.$$

Démonstration : Regarder les valeurs propres de la matrice $(I_d - \delta_t A_h)$.

Méthode d'Euler implicite

Approximation de l'EDP :

$$\frac{u_i^{n+1} - u_i^n}{\delta_t} = \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{h^2} + f_i^{n+1}$$

$\forall (i, n) \in [1 : N] \times [0, M-1]$.

Schéma numérique :

Le vecteur u_h^{n+1} est donc solution du système linéaire

$$(I_d + \delta_t A_h) u_h^{n+1} = u_h^n + \delta_t f_h^{n+1}$$

Exercice

On suppose que la solution u de (P_2) est suffisamment régulière. Montrer que le schéma d'Euler implicite est consistant d'ordre 2 en espace et d'ordre 1 en temps et qu'il est inconditionnellement L^2-stable

Schéma de Crank-Nicolson

Approximation de l'EDP :

$$\frac{u_i^{n+1} - u_i^n}{\delta_t} = \left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{h^2} + f_i^{n+1} \right) /2 + \left(\frac{u_{i+1}^{n} - 2u_i^{n} + u_{i-1}^{n}}{h^2} + f_i^{n} \right) /2$$

Schéma numérique :

Le vecteur u_h^{n+1} est donc solution du système linéaire

$$(I_d + \delta_t A_h/2) u_h^{n+1} = (I_d - \delta_t A_h/2) u_h^n + \delta_t \left(f_h^n + f_h^{n+1} \right) /2$$

Exercice

On suppose que la solution u de (P_2) est suffisamment régulière. Montrer que le schéma d'Euler implicite est consistant d'ordre 2 en espace et en temps et qu'il est inconditionnellement L^2-stable

1. Exemple d'un problème aux limites : l'équation de Poisson
2. Exemple d'un problème d'évolution : l'équation de la chaleur
3. Exemple d'un problème d'évolution : l'équation de transport
4. Exemple d'un problème d'évolution : l'équation des ondes

Equation de transport

On s'intéresse à la résolution numérique de l'équation de transport

$$(P_3) \quad \begin{cases} \partial_t u(x,t) + c\partial_x u(x,t) = 0 \\ u(x,0) = u_0 \end{cases}$$

qui a pour solution

$$u(x,t) = u_0(x - ct).$$

Cette équation vérifie en particulier des principes de stabilité qu'on souhaiterait conserver numériquement !

- Stabilité L^∞ :

$$c_0 \leq u_0 \leq c_1 \Longrightarrow c_0 \leq u(x,t) \leq c_1$$

- Stabilité L^2

$$\|u(x,t)\|_2 \leq \|u_0\|_2$$

Schéma explicite différences finies centré

Schéma numérique :

$$\frac{u_i^{n+1} - u_i^n}{\delta_t} + c\frac{u_{i+1}^n - u_{i-1}^n}{2h} = 0 \quad \text{avec} \quad u_0^i = u_0(x_i)$$

Ce schéma est inconditionnellement instable et s'avère inutilisable. Il ne vérifie aucun des principes de stabilité précédents !

Exercice

Avec $u_0(x) = e^{ipx}$, calculer la solution exacte de $(P3)$ et la solution numérique u_h^n. Remarques ?

Schéma explicite décentré en amont

Schéma numérique (cas $c > 0$) :

$$\frac{u_i^{n+1} - u_i^n}{\delta_t} + c\frac{u_i^n - u_{i-1}^n}{h} = 0 \quad \text{avec} \quad u_0^i = u_0(x_i)$$

Propriété

Ce schéma est L^∞-stable sous la condition de Courant-Friedrichs-Levy (CFL)

$$c\delta_t \leq h.$$

Démonstration : Montrer que sous la CFL, u_i^{n+1} est une combinaison convexe des $\left\{u_i^n\right\}_{i\in\mathbb{Z}}$.

Exercice

Si $u_0 \in C^2(\mathbb{R})$, montrer que le schéma numérique est consistant à l'ordre 1 en temps et en espace!

1. Exemple d'un problème aux limites : l'équation de Poisson
2. Exemple d'un problème d'évolution : l'équation de la chaleur
3. Exemple d'un problème d'évolution : l'équation de transport
4. Exemple d'un problème d'évolution : l'équation des ondes

Equation des ondes

- On s'intéresse à la résolution numérique de l'équation des ondes

$$(P_4) \quad \begin{cases} \partial_{tt} u(x,t) = c^2 \Delta u(x,t) \\ u(x,0) = u_0(x) \\ \partial_t u(x,0) = u_0'(x) \end{cases}$$

- Formule de d'Alembert

$$u(x,t) = \frac{1}{2}\left[u_0(x-ct) + u_0(x+ct)\right] + \frac{1}{2c}\int_{x-ct}^{x+ct} u_0'(y)dy$$

- Conservation de l'énergie

$$\mathcal{E}(t) = \frac{1}{2}\int \left((\partial_t u(x,t))^2 + c^2(\partial_x u(x,t))^2\right) dxdt$$

Approche directe explicite

Approximation de l'EDP :

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\delta_t^2} = c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}, \quad \text{et} \quad \begin{cases} u_i^0 & = u_0(x_i) \\ u_i^1 & = u_0(x_i) + \delta_t u_0'(x_i) \end{cases}$$

Schéma numérique :

$$u_h^{n+1} = \left(2I_d - c^2\delta_t^2 A_h\right) u_h^n - u_h^{n-1}.$$

Propriété

On suppose que la solution u de (P_4) est suffisamment régulière. Alors, le schéma est consistant à l'ordre 2 en temps et en espace. Ce schéma est de plus stable sous la condition CFL

$$c\delta_t \leq \frac{h}{2}.$$

Approche Euler implicite

Idée : Réécrire le problème (P_4) sous la forme d'un système d'EDP d'ordre 1 :

$$\partial_t Y(x,t) = \begin{pmatrix} 0 & I_d \\ c^2\Delta & 0 \end{pmatrix} Y(x,t), \quad \text{avec} \quad Y(t) = \begin{pmatrix} u(x,t) \\ \partial_t u(x,t) \end{pmatrix}$$

Euler implicite : Y_h^{n+1} solution de

$$B_h Y_h^{n+1} = Y_h^n \quad \text{avec} \quad B_h = \begin{pmatrix} I_d & -\delta_t I_d \\ +\delta_t c^2 A_h & I_d \end{pmatrix}$$

Propriété

Ce schéma est d'ordre 1 en temps et d'ordre 2 en espace . Il est de plus inconditionnellement stable

Schéma de Crank-Nicolson

Exercice

Montrer que l'approche Crank-Nicolson conduit au schéma consistant d'ordre 2,

$$C_h Y_h^{n+1} = D_h Y_h^n$$

avec

$$C_h = \begin{pmatrix} I_d & -\delta_t I_d/2 \\ +\delta_t c^2 A_h/2 & I_d \end{pmatrix} \quad et \quad D_h = \begin{pmatrix} I_d & +\delta_t I_d/2 \\ -\delta_t c^2 A_h/2 & I_d \end{pmatrix}$$