Factorisation LU

Pour simplifier la présentation de l'algorithme, on ne va pas tenir compte d'éventuelles permutations, ni de l'initialisation des $l_{ii} = 1$.
Note : la commande `lu()` de `Scilab` produit une matrice de permutations, *cf.* `help lu`.

Factorisation LU :

$L = I_n, U = O$
pour $i = 1$ **à** $n - 1$
  pour $j = i + 1$ **à** n
    $l_{ji} = a_{ji}/a_{ii}$
  pour $j = i$ **à** n
    $u_{ij} = a_{ij}$
  pour $j = i + 1$ **à** n
    pour $k = i + 1$ **à** n
      $a_{jk} = a_{jk} - l_{ji} \cdot u_{ik}$
$u_{nn} = a_{nn}$

Quel est le nombre de multiplications+additions nécessaires ?

?

Résolution de $Ax = b$

Appliquons la factorisation LU à la résolution du système.

Factoriser A en $A = LU$.
Résoudre un système triangulaire inférieur : $L\tilde{x} = b$.
Résoudre un système triangulaire supérieur : $Ux = \tilde{x}$
Ainsi $x = U^{-1}(L^{-1}b)$

Coût : Résolution de deux systèmes triangulaires et décomposition LU.

$$2\,O(n^2) + O(\frac{2}{3}n^3) = O(\frac{2}{3}n^3).$$

Note : on a simplifié en ne tenant pas compte des permutations, nécessaires dans le cas général.

Applications de la factorisation LU

• Comment calculer $\det(A)$? combien coûte l'évaluation de $\det(A)$?

À partir de la décomposition LU on obtient immédiatement

$$\det(A) = \prod_{i=1}^{n} u_{ii} \, .$$

Le coût de calcul du déterminant est donc liée à la décomposition LU, c-à-d $O(\frac{2}{3}n^3)$!

• La factorisation LU permet de résoudre plusieurs systèmes $Ax = b_r$, où b_r peut varier.

Applications de la factorisation LU

Exemple : Pour déterminer A^{-1} il suffit de résoudre les n systèmes $Ax_i = e_i$, $1 \leq i \leq n$.

Si e_i =`[zeros(1,i-1), 1, zeros(1,n-i)]'` ,
alors x_i est la i-ème colonne de A^{-1}.

Coût de l'inversion : $O(\frac{2}{3}n^3) + 2nO(n^2) = O(\frac{8}{3}n^3)$.

Conclusion :
• En pratique, pour résoudre $Ax = b$, on ne calculera jamais A^{-1} !
• En *Scilab* les commandes `det()` et `inv()` utilisent une décomposition LU, avec pivot partiel :
fonction `degetrf.f` de la librairie `Lapack`
• Il existe beaucoup d'autres méthodes de résolution numériques d'un système d'équations linéaires : pour des matrices symétriques, diagonales par blocs, creuses, . . . , on trouve souvent des algorithmes adaptées et plus efficaces que la méthode générale de factorisation LU.

Erreurs lors de la résolution de systèmes linéaires

On considère $A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$ avec $\det(A) = +1$.

On pose

$$\delta A = \begin{pmatrix} -0.002553422381 & 0.004227714089 & -0.001061727529 & 0.000629732580 \\ 0.004227714089 & -0.006999847167 & 0.001757907531 & -0.001042651353 \\ -0.001061727529 & 0.001757907531 & -0.000441472337 & 0.000261846383 \\ 0.000629732580 & -0.001042651353 & 0.000261846383 & -0.000155306511 \end{pmatrix}$$

Alors $\det(A + \delta A) = 3.797 \cdot 10^{-14}$.

Des erreurs relatives de un millième rendent la matrice A presque singulière !

La matrice A est **mal conditionnée** :

elle n'est pas "loin" d'une matrice non inversible.

Erreurs lors de la résolution de systèmes linéaires

La matrice inverse $A^{-1} = \begin{pmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{pmatrix}$

Si on résout le système $Ax = b$ avec

$b_1 = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$, on obtient comme solution $x_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

$b_2 = \begin{pmatrix} 32.5 \\ 22.5 \\ 33.5 \\ 30.5 \end{pmatrix}$, on obtient comme solution $x_2 = \begin{pmatrix} 42 \\ -67 \\ 18.5 \\ -9.5 \end{pmatrix}$

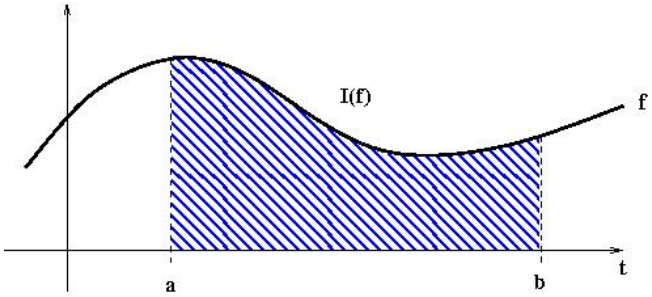
Donc pour une matrice mal conditionnée, les solutions du système linéaire varient beaucoup, même pour de petites variations de b !

Intégration numérique

Soit $f : [a, b] \to \mathbb{R}$, une fonction continue. On cherche à calculer une valeur approchée de l'intégrale

$$I(f) = \int_a^b f(t)\, dt.$$

Ceci est utile si l'on ne connaît pas d'expression analytique (simple) d'une primitive de f ou si celle ci est trop compliquée à déterminer et que l'on a uniquement besoin de la valeur numérique de $I(f)$.



Intégration numérique : approximation

On considère une subdivision de l'intervalle $[a, b]$:

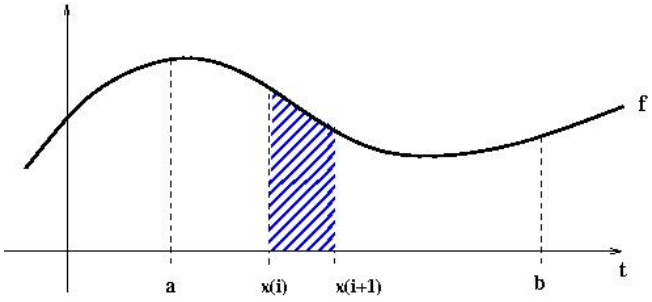
$$a = x_0 < x_1 < \cdots < x_N = b\ .$$

Alors,

$$I(f) = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(t)\, dt\ .$$

Il faut calculer au moyen de formules approchées les quantités

$$\int_{x_i}^{x_{i+1}} f(t)\, dt\ .$$



Intégration numérique

Soit $I_{x_i,x_{i+1}}(f)$ l'approximation de $\int_{x_i}^{x_{i+1}} f(t)\,dt$,
on obtient la **formule de quadrature** :

$$I_N(f) = \sum_{i=0}^{N-1} I_{x_i,x_{i+1}}(f)\,.$$

C'est une approximation de l'aire comprise entre le graphe de f et le segment $[a, b]$.

Qualité de l'approximation :
Une méthode d'intégration numérique est dite **d'ordre** k si elle est exacte pour les polynômes de degré inférieur ou égal à k, et si elle est fausse pour au moins un polynôme de degré $k+1$.
Si f est en plus $k+1$ fois continûment dérivable, on montre que

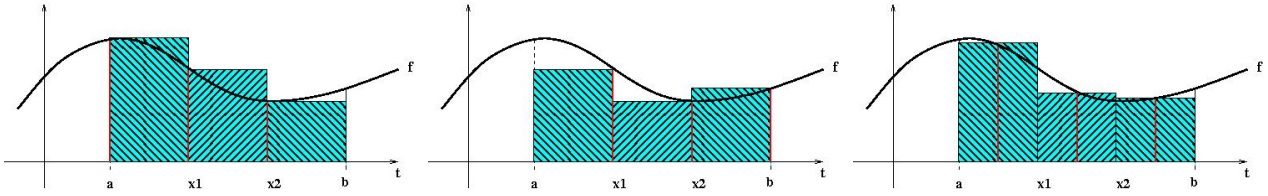
$$|I(f) - I_N(f)| = O\left(h^{k+1}\right) \qquad \text{où } h = \max_{0\le i\le N-1}(x_{i+1} - x_i)\,.$$

Méthodes de quadrature à un point

Soit $\theta_i \in [x_i, x_{i+1}]$, on fait l'approximation suivante :

$$\int_{x_i}^{x_{i+1}} f(t)dt \approx (x_{i+1} - x_i)f(\theta_i).$$

- Si $\theta_i = x_i$, on obtient la méthode des *rectangles à gauche*;
- si $\theta_i = x_{i+1}$, on obtient la méthode des *rectangles à droite*;
- si $\theta_i = \dfrac{x_i + x_{i+1}}{2}$, on obtient la méthode du *point milieu*.



Méthodes de quadrature à un point

- La méthode des *rectangles à gauche* est d'ordre 0 et s'écrit :

$$I_N^G(f) = \sum_{i=0}^{N-1} (x_{i+1} - x_i) f(x_i) \; ;$$

- la méthode des *rectangles à droite* est d'ordre 0 et s'écrit :

$$I_N^D(f) = \sum_{i=0}^{N-1} (x_{i+1} - x_i) f(x_{i+1}) = \sum_{i=1}^{N} (x_i - x_{i-1}) f(x_i) \; ;$$

- la méthode du *point milieu* est d'ordre 1 et s'écrit :

$$I_N^M(f) = \sum_{i=0}^{N-1} (x_{i+1} - x_i) f\left(\frac{x_i + x_{i+1}}{2}\right) \; .$$

Méthodes de quadrature à un point

Exemple :
Si la fonction `f` est définie
et si le vecteur `x` contient la subdivision de l'intervalle $[a, b]$,
c.-à-d. `x = [ x(1), x(2), ... , x(N+1)]`,
avec `x(1)`=a et `x(N+1)`=b, alors

```
IG = sum( (x(2:$)-x(1:$-1)) .* f(  x(1:$-1) ) ) ;

ID = sum( (x(2:$)-x(1:$-1)) .* f(  x(2:$)   ) ) ;

IM = sum( (x(2:$)-x(1:$-1)) .* f(  (x(1:$-1)+x(2:$))/2 ) )
```

Méthodes de quadrature à deux points

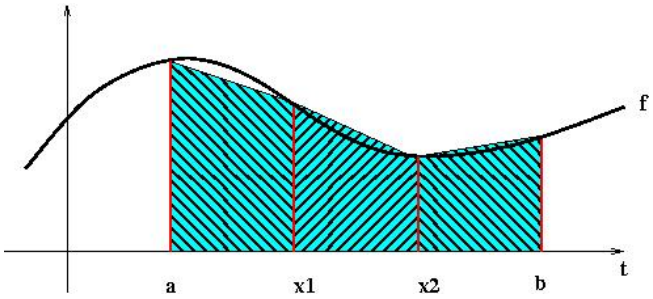
Pour $\alpha \in [0,1]$,

$$\int_{x_i}^{x_{i+1}} f(t)dt \approx (x_{i+1} - x_i)\left(\alpha f(x_{i+1}) + (1-\alpha) f(x_i)\right)$$

Pour $\alpha = \frac{1}{2}$, on obtient la *méthode du trapèze* :

$$I_N^T(f) = \sum_{i=0}^{N-1} (x_{i+1} - x_i) \frac{f(x_i) + f(x_{i+1})}{2}$$

C'est une méthode d'ordre 1.



Méthode de Simpson

C'est une méthode de quadrature à trois points, basée sur l'approximation :

$$\int_{x_i}^{x_{i+1}} f(t)dt \approx \frac{1}{6}(x_{i+1} - x_i)\left(f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right).$$

Ainsi

$$I_N^S(f) = \frac{1}{6}\sum_{i=0}^{N-1} (x_{i+1} - x_i)\left(f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right).$$

La méthode de Simpson (1710-1761) est d'ordre 3.

Remarques

- On considère souvent une subdivision régulière de l'intervalle $[a, b]$:

$$x_i = a + i\frac{b-a}{N}, \quad i = 0, \ldots, N\ .$$

  Dans ce cas, le pas est constant et vaut $\frac{1}{N}$.

- Dans `Scilab` diverses fonctions sont prévues pour l'intégration numérique :

  1. à partir des points `(x(i),y(i))`
     pour les fonctions `inttrap()` et `intsplin()` ;
  2. à partir d'une fonction f et de l'intervalle $[a, b]$
     pour les fonctions `intg()` et `integrate()`.