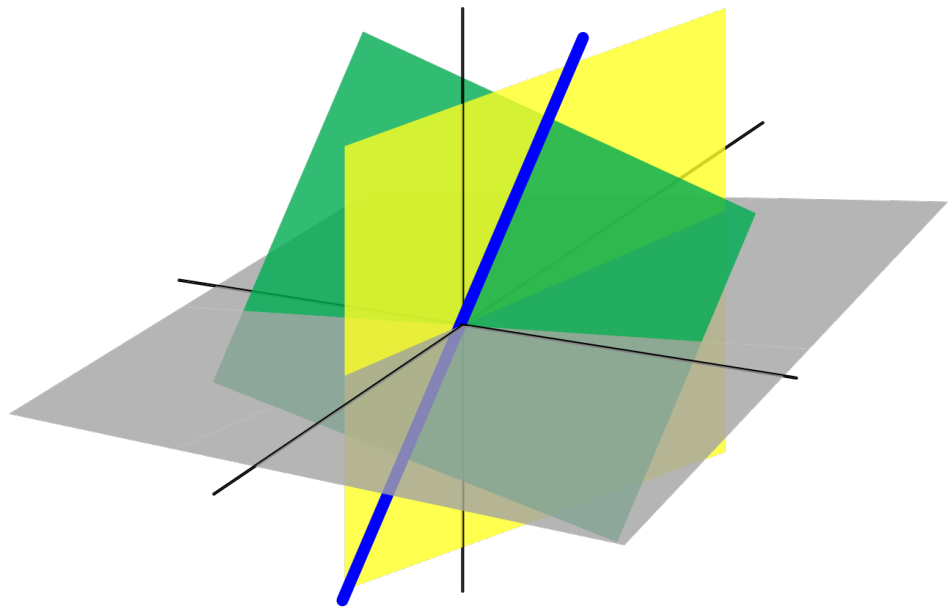
Algèbre Linéaire

Compléxité algorithmique de LU

Jérôme Scherer



La méthode de Gauss

Lorsqu'on échelonne et réduit une matrice c'est la première partie, échelonner, qui est la plus coûteuse en temps. Supposons pour simplifier que la matrice est carrée de taille $n \times n$. On va compter le nombre d'opérations nécessaires "en général", c'est-à-dire qu'on supposera que les pivots sont $a_{11}, a_{22}, \ldots, a_{nn}$.

1. Pour obtenir 0 en position $(2,1)$ il faut multiplier chaque a_{1i} par a_{21}/a_{11} pour $2 \leq i \leq n$ puis soustraire à a_{2i} ce nombre. Cela fait $2(n-1)$ opérations.
2. De même pour chacune des lignes $3, \ldots, n$. En tout ce sont $2(n-1)^2$ opérations.

Echelonner

1. Pour obtenir des 0 sous a_{11} il faut $2(n-1)^2$ opérations.
2. Pour obtenir des 0 sous a_{22} il faut ensuite $2(n-2)^2$ opérations.
3. De même pour les autres pivots: $2(n-3)^2, \ldots, 4$ et enfin 1 opération.

Conclusion

Il faut 2 fois $(n-1)^2 + (n-2)^2 + \cdots + 2^2 + 1^1$ opérations:

$$\boxed{\frac{2n(n-1)(2n-1)}{6} \text{ opérations pour échelonner}}$$

L'ordre de grandeur est $2n^3/3$.

Pour échelonner une matrice augmentée il faut $\dfrac{2n(n+1)(2n+1)}{6}$ opérations.

Réduire

Pour réduire la matrice l'ordre de grandeur est moindre. Nous avons un système de n équations à n inconnues à résoudre.

Pour calculer les effets des opérations élémentaires sur les coefficients inhomogènes, il faut

1. 1 multiplication et $(n-1)$ soustractions avec le pivot a_{nn},
2. 1 multiplication et $(n-2)$ soustractions avec le pivot $a_{n-1,n-1}$,
3. et ainsi de suite.

Conclusion

Il faut 2 fois $n + (n-1) + \cdots + 2 + 1$ opérations:

$$\boxed{\frac{n(n+1)}{2} \text{ opérations pour réduire}}$$

L'ordre de grandeur est $\frac{n^2}{2}$

Coût algorithmique de Gauss

L'ordre de grandeur du nombre d'opérations nécessaires pour résoudre un système de n équations à n inconnues, est

$$\boxed{\frac{2n^3}{3} \text{ opérations}}$$

Par conséquent si on doit résoudre n systèmes de n équations à n inconnues, par exemple pour inverser une matrice de taille $n \times n$, il faut

$$\boxed{\frac{2n^4}{3} \text{ opérations}}$$

Pour $n = 1000$ on devrait donc faire $2/3 \cdot 10^{12}$ opérations.

Coût algorithmique avec LU

En revanche si l'on calcule une fois pour toutes la factorisation LU, l'ordre de grandeur du nombre d'opérations nécessaires pour résoudre un système de n équations à n inconnues est

1. de l'ordre de $\frac{2n^3}{3}$ pour obtenir la factorisation,
2. de l'ordre de $\frac{n^2}{2}$ pour résoudre chaque système $Ly = b$,
3. de l'ordre de $\frac{n^2}{2}$ pour résoudre chaque système $Ux = y$.

Par conséquent il faut

$$\boxed{\frac{2n^3}{3} + n^3 = \frac{5n^3}{3} \text{ opérations}}$$

Pour $n = 1000$ cela représente seulement $5/3 \cdot 10^9$, c'est 400 fois moins!