

Résolution d'un système linéaire (méthode de Gauss)

Nous cherchons à résoudre un système linéaire de n équations à n inconnues :

$$\begin{array}{cccccc} c_{0,0}x_0 + & c_{0,1}x_1 + & \dots & c_{0,n-1}x_{n-1} & = & b_0 \\ c_{1,0}x_0 + & c_{1,1}x_1 + & \dots & c_{1,n-1}x_{n-1} & = & b_1 \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{n-1,0}x_0 + & c_{n-1,1}x_1 + & \dots & c_{n-1,n-1}x_{n-1} & = & b_{n-1} \end{array}$$

La méthode de Gauss consiste à faire une suite d'opérations sur le système d'équations qui, sans changer ses solutions, le transforment en un système triangulaire, dont la résolution est facile. Ces opérations sont :

- permuter deux équations ;
- ajouter à une équation une autre équation multipliée par un coefficient non nul.

REPRESENTATION. Nous représenterons le système par la matrice $n \times (n + 1)$

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} & a_{0,n} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} & a_{1,n} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \dots & a_{n-1,n-1} & a_{n-1,n} \end{bmatrix} \text{ donnée par } \begin{cases} a_{i,j} = c_{i,j} \text{ pour } 0 \leq i < n \text{ et } 0 \leq j < n \\ a_{i,n} = b_i \text{ pour } 0 \leq i < n \end{cases}$$

A est donc la matrice obtenue en ajoutant à la matrice des coefficients du système une dernière colonne formée des seconds membres des équations. Les opérations sur les équations indiquées ci-dessus se traduisent par des opérations correspondantes sur les lignes de A .

DESCRIPTION DE LA PREMIERE ETAPE (attention, ce n'est pas l'explication de la première étape qui va le plus vous aider à trouver le programme ; fiez-vous plutôt à l'explication de l'étape générale) :

- 0.1 Rechercher, parmi les coefficients $a_{0,0} a_{1,0} \dots a_{n-1,0}$ de la première colonne, celui dont la valeur absolue est la plus grande ; soit m l'indice de sa ligne. Le coefficient $a_{m,0}$ est appelé le *pivot* de l'étape 0.
- 0.2 Si $a_{m,0}$ est à peu près nul (la question des valeurs « à peu près nulles » est expliquée plus loin) arrêter tout : la résolution du système est impossible.
- 0.3 Sinon, permuter les lignes d'indices 0 et m .
- 0.4 Remplacer chacune des lignes d'indices 1, ... $n-1$ par le résultat de l'addition d'elle-même et de la première ligne multipliée par ce qu'il faut pour que le premier coefficient de la ligne résultante soit 0.

Ces transformations fournissent une nouvelle matrice, correspondant à un système ayant les mêmes solutions que le système initial, et dont la première colonne est faite d'un coefficient non nul sous lequel on ne trouve que des zéros :

$$\begin{bmatrix} a'_{0,0} & a'_{0,1} & \dots & a'_{0,n-1} & a'_{0,n} \\ 0 & a'_{1,1} & \dots & a'_{1,n-1} & a'_{1,n} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a'_{n-1,1} & \dots & a'_{n-1,n-1} & a'_{n-1,n} \end{bmatrix}$$

On peut alors recommencer ces opérations sur le sous-système obtenu en « oubliant » la première ligne et la première colonne du système original. Et ainsi de suite. On aboutit à la :

DESCRIPTION DE LA k^{EME} ETAPE (La première étape, expliquée ci-dessus, correspondait à $k = 0$). Au début de la k^{eme} étape, la matrice a la forme suivante :

$$\begin{bmatrix} a'_{0,0} & a'_{0,1} & \cdots & a'_{0,k} & a'_{0,k+1} & \cdots & a'_{0,n-1} & a'_{0,n} \\ 0 & a'_{1,1} & \cdots & a'_{1,k} & a'_{1,k+1} & \cdots & a'_{1,n-1} & a'_{1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{k,k} & a'_{k,k+1} & \cdots & a'_{k,n-1} & a'_{k,n} \\ 0 & 0 & \cdots & a'_{k+1,k} & a'_{k+1,k+1} & \cdots & a'_{k+1,n-1} & a'_{k+1,n} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{n-1,k} & a'_{n-1,k+1} & \cdots & a'_{n-1,n-1} & a'_{n-1,n} \end{bmatrix}$$

Les opérations à faire à la $k^{\text{ème}}$ étape sont :

- k.1 Rechercher, parmi les coefficients $a'_{k,k}, a'_{k+1,k}, \dots, a'_{n-1,k}$ de la colonne d'indice k celui dont la valeur absolue est la plus grande ; soit m l'indice de sa ligne. Le coefficient $a'_{m,k}$ est appelé le *pivot* de la $k^{\text{ème}}$ étape.
- k.2 Si $a'_{m,k}$ est à peu près nul, arrêter tout. La résolution est impossible.
- k.3 Sinon, permuter les lignes k et m .
- k.4 Remplacer chacune des lignes d'indices $k+1, \dots, n-1$ par le résultat de l'addition d'elle-même et de la ligne de rang k multipliée par ce qu'il faut pour que le coefficient de rang k de la ligne résultante soit 0.

LA FIN DES ETAPES. Si la méthode a réussi à effectuer avec succès n étapes ($k = 0, \dots, n-1$), alors la matrice a pris la forme suivante :

$$\begin{bmatrix} a''_{0,0} & a''_{0,1} & \cdots & a''_{0,k} & a''_{0,k+1} & \cdots & a''_{0,n-1} & a''_{0,n} \\ 0 & a''_{1,1} & \cdots & a''_{1,k} & a''_{1,k+1} & \cdots & a''_{1,n-1} & a''_{1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a''_{k,k} & a''_{k,k+1} & \cdots & a''_{k,n-1} & a''_{k,n} \\ 0 & 0 & \cdots & 0 & a''_{k+1,k+1} & \cdots & a''_{k+1,n-1} & a''_{k+1,n} \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & a''_{n-1,n-1} & a''_{n-1,n} \end{bmatrix}$$

qui correspond à un système, ayant les mêmes solutions que le système initial, de la forme suivante, dite *triangulaire* :

$$\begin{aligned} a''_{0,0}x_0 + a''_{0,1}x_1 + \cdots + a''_{0,k}x_k + \cdots + a''_{0,n-2}x_{n-2} + a''_{0,n-1}x_{n-1} &= a''_{0,n} \\ a''_{1,1}x_1 + \cdots + a''_{1,k}x_k + \cdots + a''_{1,n-2}x_{n-2} + a''_{1,n-1}x_{n-1} &= a''_{1,n} \\ &\vdots \\ &\vdots \\ a''_{k,k}x_k + \cdots + a''_{k,n-2}x_{n-2} + a''_{k,n-1}x_{n-1} &= a''_{k,n} \\ &\vdots \\ &\vdots \\ a''_{n-2,n-2}x_{n-2} + a''_{n-2,n-1}x_{n-1} &= a''_{n-2,n} \\ a''_{n-1,n-1}x_{n-1} &= a''_{n-1,n} \end{aligned}$$

où $a''_{0,0}, a''_{1,1}, \dots, a''_{n-1,n-1}$ sont tous non nuls. La solution de ce système est facile à obtenir : de la dernière équation on tire la valeur de x_{n-1} ; connaissant cette valeur, de l'avant-dernière équation on peut sortir x_{n-2} , et ainsi de suite jusqu'à la première équation, qui fournit x_0 .

FORMULES. Notons « $\alpha \leftarrow \beta$ » pour « remplacer α par β ». La transformation décrite au k.4 s'écrit :

pour $i = k+1, \dots, n-1$

$$\text{pour } j = k+1, \dots, n \quad a_{i,j} \leftarrow a_{i,j} - \frac{a_{i,k}}{a_{k,k}} a_{k,j}$$

Le système étant devenu triangulaire, l'obtention des solutions peut s'écrire

pour $i = n-1, n-2, \dots, 0$

$$x_i \leftarrow \frac{a_{i,n} - \sum_{j=i+1}^{n-1} a_{i,j}x_j}{a_{i,i}}$$

UN EXEMPLE. Système donné :

$$2x_1 + 4x_2 + 4x_3 = 22$$

$$x_1 + 3x_2 = 7$$

$$1.5x_1 + x_3 = 4.5$$

il correspond à la matrice

$$\begin{array}{cccc} 2 & 4 & 4 & 22 \\ 1 & 3 & 0 & 7 \\ 1.5 & 0 & 1 & 4.5 \end{array}$$

Première étape : à l'aide de la première ligne, on place des zéros au début des autres :

$$\begin{array}{cccc} 2 & 4 & 4 & 22 \\ 0 & 1 & -2 & -4 \\ 0 & -3 & -2 & -12 \end{array}$$

Première partie de la deuxième étape : permutation des lignes d'indice 1 et 2, car la valeur absolue de -3 est supérieure à celle de 1 :

$$\begin{array}{cccc} 2 & 4 & 4 & 22 \\ 0 & -3 & -2 & -12 \\ 0 & 1 & -2 & -4 \end{array}$$

Fin de la deuxième étape :

$$\begin{array}{cccc} 2 & 4 & 4 & 22 \\ 0 & -3 & -2 & -12 \\ 0 & 0 & -2.6667 & -8 \end{array}$$

Y a-t-il une troisième étape ? Si votre programme est bien conçu, il y aura intérêt à ce que cette dernière étape ait lieu. Elle ne fera aucune transformation, mais elle vérifiera que le dernier pivot n'est pas à peu près nul (cette vérification est indispensable).

Résolution. A partir du système triangulaire représenté par la matrice ci-dessus, on obtient *dans l'ordre* $x_3 = 3$, puis $x_2 = 2$ et finalement $x_1 = 1$.

A PROPOS DES *PIVOTS*. Dans les calculs précédents on fait figurer au dénominateur les éléments diagonaux de la matrice, qu'on appelle les *pivots*. Il faut donc que ces derniers soient non nuls. Or même pour une matrice de déterminant non nul on peut tomber sur un pivot $m_{k,k}$ nul.

Pour résoudre ce problème on utilise la technique du *pivot maximal* : à chaque étape on recherche, parmi les coefficients qui — au prix d'une permutation de lignes — sont susceptibles de jouer le rôle de pivot, celui dont la valeur absolue est la plus grande, puis on permute les lignes concernées :

- si l'élément qui a la plus grande valeur absolue est nul, c'est que tous les « candidats pivots » sont nuls. La théorie prouve alors que le système n'a pas une solution unique ;
- sinon, la permutation faite assure que le pivot utilisé comme dénominateur est non nul ;
- enfin, choisir comme dénominateur l'élément de plus grande valeur absolue augmente la précision des calculs

A PROPOS DES VALEURS « A PEU PRES NULLES ». Dans une explication strictement mathématique de la méthode de Gauss, dans la description de l'étape *k.2* on dirait :

« ... si $a_{m,k}$ est nul, arrêter le traitement : la résolution est impossible ... »

ce qui suggérerait la traduction en C suivante

```
if (a[m][k] == 0)
    traitement de la situation « résolution impossible »
```

Il faut savoir que, par suite des erreurs dans les calculs avec des nombres flottants, qui sont toujours approchés (la valeur de $1/3$ n'est pas 0.333333), l'égalité à zéro d'un nombre flottant obtenu par un calcul n'est en général que le fruit du hasard. Pour obtenir des résultats qui signifient quelque chose, il faut remplacer les comparaisons à zéro par des conditions comme :

```
if (fabs(a[m][k]) < EPSILON)
    traitement de la situation « résolution impossible »
```

où EPSILON est un seuil en dessous duquel nous tenons une valeur pour équivalente à zéro. Par exemple, notre programme pourrait comporter la définition

```
#define EPSILON 1.0E-06
```

ORGANISATION DU PROGRAMME. On attachera une grande importance à écrire un programme ayant une structure *claire*. Notamment, les opérations suivantes apparaîtront nettement séparées :

- acquisition, au clavier, des coefficients du système et du second membre.
N.B. Dans une « version de travail » du programme, pour gagner du temps, vous pouvez remplacer l'acquisition au clavier par une initialisation de la matrice avec des valeurs constantes figées dans le programme (cf. initialisation de tableaux) ;
- triangularisation de la matrice. Cette opération est une boucle, composée à son tour de trois parties nettement séparées :
 - recherche du pivot maximal ;
 - test de ce pivot (la résolution n'est elle pas impossible ?) ;
 - permutation éventuelle de deux lignes ;
 - réduction des lignes en dessous de la ligne *k*.
- résolution du système triangulaire
- affichage de la solution

DEUXIEME VERSION DU PROGRAMME. Lorsque la première version de votre programme sera au point, vous envisagerez l'écriture d'une deuxième version rendue plus claire par l'introduction de certaines fonctions :

- deux fonctions, `void lecture_systeme(void)`, `void affichage_solution(void)`, permettant de séparer de manière encore plus évidente les parties lecture/écriture de la partie traitement proprement dit ;
- une fonction `void permutation(int k, int m)` effectuant la permutations des lignes *k* et *m* de la matrice ;
- une fonction `void reduction(int j, int k, double u)` qui remplace la ligne *j* par l'addition de la ligne *j* et de *u* fois la ligne *k* ;
- une fonction `int pivotmax(int k)` qui calcule et renvoie l'indice de la ligne correspondant au pivot maximum de l'étape *k*.

ainsi que toute autre fonction ou procédure dont vous pourriez avoir l'idée, à la condition qu'elle aille dans le sens de la clarté et de la simplification de votre programme.

