CPES Scientifique

Informatique

TP 11

Résolution de systèmes linéaires

1 Principe de l'algorithme

L'algorithme procède en deux grandes étapes :

1) La mise sous forme triangulaire (en utilisant le pivot de Gauss).

2) La résolution de systèmes triangulaires (appellée « *phase de remontée* »).

Il est pratique de travailler avec une représentation matricielle des systèmes d'équations. Un système

$$\begin{pmatrix} a_{1,1}x_1 & + & \dots & + & a_{1,n}x_n & = & b_1 \\ \vdots & & & & \vdots & & \vdots \\ a_{n,1}x_1 & + & \dots & + & a_{n,n}x_n & = & b_n \end{pmatrix}$$

sera représenté à l'aide des matrices

$$A = \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \dots & a_{n,n} \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

1.1 Mise sous forme triangulaire

C'est la partie la plus difficile de l'algorithme.

Il est pratique de commencer par les transvections. L'action sur le membre droit des équations est simple à programmer (mais il ne faut pas l'oublier !). Concernant les membres gauches :

Question 1.1. *Écrire une fonction «* `trans_mat` *» prennant en arguments une matrice «* `a` *», deux indices «* `i` *» et «* `j` *» ainsi qu'une valeur «* `v` *», et telle que si* `a` *est une matrice carrée et* `i` *et* `j` *sont deux indices de lignes de* `a`*, alors «* `trans_mat(a,i,j,v)` *» réalise la transvection*

$$L_{\texttt{i}} \leftarrow L_{\texttt{i}} + \texttt{v} \cdot L_{\texttt{j}}$$

où $L_{\texttt{i}}$ *désigne la ligne d'indice* `i` *de* `a`*.*

Question 1.2. *Écrire une fonction «* `echange` *» prennant en arguments une matrice «* `a` *», deux indices «* `i` *» et «* `j` *» et telle que si* `a` *est une matrice carrée et* `i` *et* `j` *sont deux indices de lignes de* `a`*, alors «* `echange(a,i,j)` *» correspond à la matrice dans laquelle les lignes* `i` *et* `j` *de la matrice* `a` *sont échangées.*

Pour le choix du pivot, on adoptera la stratégie suivante, qui fonctionne relativement bien en pratique vis-à-vis des problèmes de précision avec les nombres flottants : pour chaque inconnue x_i, on va choisir comme pivot la valeur « `a[k][i]` » qui est maximale en valeur absolue.

Question 1.3. *Écrire une fonction Python «* `rech_pivot` *» qui prend en argument une matrice carrée «* `a` *» et un indice de colonne «* `i` *» (représentant l'inconnue* $x_{\texttt{i}}$*) et qui renvoie un indice de ligne* $k \geq$ `i` *tel que la valeur absolue de* `a`$[k]$[`i`] *est maximale parmi les valeurs absolues des* `a`$[j]$[`i`] *pour* $j \geq$ `i`*.*

2 Écriture Python de l'algorithme

L'architecture généralle de l'algorithme peut être basée sur le patron suivant :

```
def res_lin(a_0,b_0):
    a = copie_matrice(a_0)
    b = copie_liste(b_0)

    # Mise sous forme triangulaire
    for i in range(n):
        # On s'interesse a l'inconnue x_i
        j = rech_pivot(a,i)
        # On echange les lignes
        echange(a,i,j)

        # Realisation des transvections, pour chaque ligne j > i
        # On n'oublie pas les membres droits !
        ....

    # Phase de remontee
    res = remontee(a,b)

    # Renvoi du resultat
    return res
```

Question 2.1. *Compléter le patron ci-dessus. Tester cette fonction sur des exemples.*

3 Quelques exemples de systèmes

$$\begin{pmatrix} 2x & + & 2y & - & 3z & = & 2 \\ & & y & - & 6z & = & -3 \\ & & & + & z & = & 4 \end{pmatrix}$$

$$\begin{pmatrix} 2x & + & 2y & - & 3z & = & 2 \\ -2x & - & y & - & 3z & = & -5 \\ 6x & + & 4y & + & 4z & = & 16 \end{pmatrix}$$

$$\begin{pmatrix} x & + & \frac{1}{4}y & + & z & = & 0 \\ x & + & \frac{1}{3}y & + & 2z & = & 0 \\ & & y & + & 12z & = & 1 \end{pmatrix}$$

Pour ces trois systèmes, commencez par les résoudre sur papier, et testez ensuite votre programme dessu. Pour comprendre ce qu'il se passe avec le troisième système, il peut être intéressant de comparer les valeurs Python de $1/12$ et de $1/3 - 1/4$ (faire bien attention à ce que les nombres soient des flottants, par exemple en écrivant `1/12.` au lieu de `1/12` etc).