Introduction à quelques méthodes numériques

14

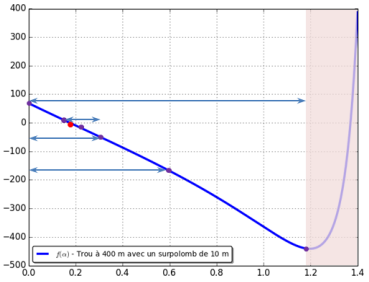
Objectif – C3-02 : Résoudre numériquement une équation ou un système d'équations

- Réécriture des équations d'un problème ;
- résolution de problèmes du type f(x) = 0 (méthodes de dichotomie et de Newton) ;
- résolution d'un système linéaire du type $A \cdot X = B$;
- résolution d'équations différentielles (schéma d'Euler explicite) ;
- intégration et dérivation numérique (schémas arrière et avant).

14.1 Equations stationnaires – Résolution de $f(x) = 0$

C3-02

14.1.1 Principe de la méthode de dichotomie



Théorème – Théorème des valeurs intermédiaires

Soit f une fonction définie et continue sur l'intervalle $[a, b]$ à valeur dans $\mathbb{R}$. Pour tout $u \in [f(a), f(b)]$, il existe au moins un réel $c \in [a, b]$ tel que $f(c) = u$.
En particulier (Théorème de Bolzano), si $f(a)$ et $f(b)$ sont de signes différents, il existe au moins un réel c tel que $f(c) = 0$.

Ainsi, pour une fonction donnée définie sur un intervalle donné, le but de l'algorithme de dichotomie va être de découper en 2 l'intervalle [a,b] en deux, afin d'y trouver la solution. Par divisions successives de l'intervalle, on convergera vers la solution.

Remarque

Tester le signe de $f(a)$ et $f(b)$.
Il existe plusieurs méthodes pour tester si $f(a)$ et $f(b)$ sont de signes différents. Si on ne se préoccupe pas de savoir la relation d'ordre entre $f(a)$ et $f(b)$, un test efficace consiste en un test du signe de $f(a) \cdot f(b)$.

14.1.2 Principe de la méthode de Newton

Définition – Développement de Taylor à l'ordre 1

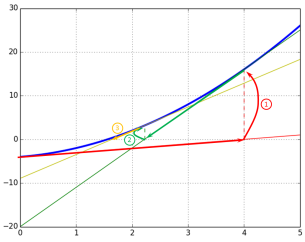
Soit f une fonction C^1 sur un intervalle I et $a \in I$. Le développement de Taylor à l'ordre 1 de f est donné par

$$f(x) = f(a) + f'(a) \cdot (x - a) + o(x - a)$$

Géométriquement, lorsqu'on néglige le reste, le développement de Taylor donne l'équation de la tangente en a. Notons $\Delta(x)$ cette équation.

L'abscisse c de l'intersection de la tangente avec l'axe des abscisses est donnée par la résolution de

$$\Delta(c) = 0 \Longleftrightarrow f(a) + f'(a) \cdot (c - a) = 0 \Longleftrightarrow c = a - \frac{f(a)}{f'(a)}$$



Évaluation de la dérivée numérique

Résultat –

En première approximation, il est possible d'approximer la dérivée en approximant la tangente à la courbe par une droite passant par deux points successifs. Dans ces conditions, pour une valeur de h suffisamment faible, on a :

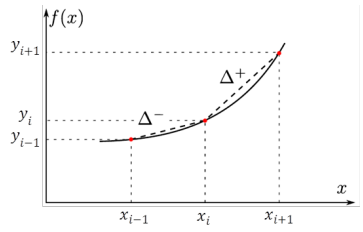
$$f'(x_0) \simeq \frac{f(x_0 + h) - f(x_0)}{h}.$$

Méthodes à un pas

Résultat –

Différence avant – Schéma d'Euler explicite
Dans ce cas, l'estimation de la dérivée au point P_i s'appuie sur le point P_{i+1} :

$$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

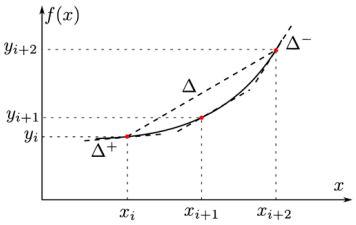


Résultat – Différence arrière – Schéma d'Euler implicite

Dans ce cas, l'estimation de la dérivée au point P_i s'appuie sur le point P_{i-1} :

$$f'(x_i) \simeq \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Méthode à deux pas



Résultat –

On peut aussi utiliser les points P_{i-1} et P_{i+1} pour estimer la dérivée en P_i :

$$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}$$

Remarques

- Lorsqu'il s'agit de dériver une fonction temporelle « en temps réel », le point suivant n'est pas encore connu donc seule la différence arrière peut être calculée.
- Le calcul de la dérivée conduit à un tableau de valeurs de dimension $n - 1$.

14.1.3 Bibliothèque Python

Il est possible de résoudre l'équation $f(x) = 0$ en utilisant les modules de la bibliothèque `scipy` :

Résolution de $\sin(x) = 0$ avec 0,5 comme valeur d'initialisation.

```
def f(x):
    return sin(x)

sol = newton(f,0.5)
print(sol)
print(f(sol))
```

Résolution du système :

$$\begin{cases} x + 10y - 3z - 5 = 0 \\ 2x - y + 2z - 2 = 0 \\ -x + y + z + 3 = 0 \end{cases}$$

```
from scipy.optimize import fsolve
# définition du système
def syst(var):
    # définition des variables
    x, y, z = var[0], var[1], var[2]
    eq1 = x +10*y-3*z-5
    eq2 = 2*x-y+2*z-2
    eq3 = -x+y+z+3
    res = [eq1, eq2, eq3]
    return res
    # Initialisation de la recherche
    # des solutions numériques
x0, y0, z0 = 0, 0, 0
sol_ini = [x0, y0, z0]
sol = fsolve(syst, sol_ini)
sol = newton(f,0.5)
print(sol)
```

14.2 Intégration numérique

Hypothèse

$f : [a, b] \longrightarrow \mathbb{R}$ est une fonction continue sur $[a, b]$. On note $I = \int_a^b f(x)\mathrm{d}x$.

14.2.1 Principe des méthodes des rectangles

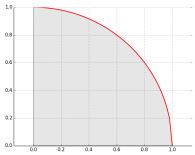
Définition – Méthode des rectangles

Dans cette méthode, la fonction à intégrer est interpolée par un polynôme de degré 0, à savoir une fonction constante. Géométriquement, l'aire sous la courbe est alors approximée par un rectangle. Plusieurs choix sont possibles.

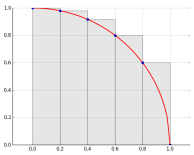
- Rectangles à gauche : $I = \int_a^b f(x)\mathrm{d}x \simeq (b - a)\, f(a)$.

- Point milieu : $I = \int_a^b f(x)\mathrm{d}x \simeq (b-a)\, f\left(\frac{a+b}{2}\right)$.
- Rectangles à droite : $I = \int_a^b f(x)\mathrm{d}x \simeq (b-a)\, f(b)$.

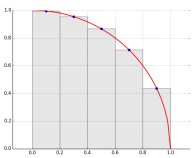
14.2.2 Interprétation graphique



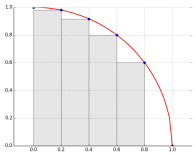
Calcul intégral



Rectangles à gauche

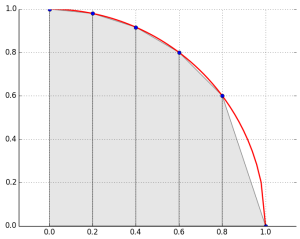


Point milieu



Rectangles à droite

14.2.3 Principe des méthodes des trapèzes



Définition –

Méthode des trapèzes Dans cette méthode, la fonction à intégrer est interpolée par un polynôme de degré 1, à savoir une fonction affine. Géométriquement, l'aire sous la courbe est alors approximée par un trapèze :

$$I = \int_a^b f(x)\mathrm{d}x \simeq (b-a)\,\frac{f(a)+f(b)}{2}$$

Notion d'erreur d'intégration

Résultat –

Dans chaque cas, on intègre f sur n subdivisions régulières de I.
Erreur sur la méthode des rectangles à gauche et à droite
Soit f fonction dérivable sur $I = [a,b]$ et dont f' est continue sur I. Soit M_1 un majorant de f' sur I. L'erreur ε commise lors de l'intégration par la méthode des rectangles à droite ou à gauche est telle que $\varepsilon \leq \frac{M_1}{2n}$.
Erreur sur la méthode des rectangles – point milieu
Si de plus f est deux fois dérivables sur $I = [a,b]$ et f'' est continue sur I, on note M_2 un majorant de f'' sur I.L'erreur ε commise lors de l'intégration par la méthode des rectangles – point milieu est telle que $\varepsilon \leq \frac{M_2}{12n^2}$.
Erreur sur la méthode des trapèzes
L'erreur commise ε est telle qu'il existe un entier M tel que $\varepsilon \leq \frac{M}{12n^2}$.

Bibliothèque Python

Il est possible d'intégrer une fonction en utilisant les modules de la bibliothèque `scipy` :

```python
from scipy.integrate import quad
from math import sin
# Définition des bornes de gauche et de droite
g,d = -1,1
def f(x):
    return sin(x)

I,erreur = quad(f,g,d)
print(I,erreur)
```

14.3 Résolution d'équations différentielles

14.3.1 Problème de Cauchy

Le problème consiste à trouver les fonctions y de $[0,T] \longrightarrow \mathbb{R}^n$ telles que

$$\begin{cases} \dfrac{\mathrm{d}y(t)}{\mathrm{d}t} = f(t,y(t)) \\ y(t_0) = y_0 \quad \text{avec } t_0 \in [0,T] \text{ et } y_0 \in \mathbb{R}^n \text{ donnés} \end{cases} \qquad \text{avec } t_0 \in [0,T] \text{ et } y_0 \in \mathbb{R}^n \text{ donnés.}$$

L'existence et l'unicité de la solution peut se démontrer en utilisant le théorème de Cauchy-Lipschitz.

Définition – Fonction lipschitzienne

f est lipschitzienne en y s'il existe un réel $k > 0$ tel que $\forall y(t) \in \mathbb{R}^n$, $\forall z(t) \in \mathbb{R}^n$, $\forall t \in [0,T]$, alors

$$||f(t,y(t)) - f(t,z(t))|| \leq k||y(t) - z(t)||$$

Théorème – Théorème de Cauchy – Lipschitz

Soit f une fonction de $[0,T] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ continue et lipschitzienne en y.
Alors, $\forall t_0 \in [0,T]$ et $\forall y_0 \in \mathbb{R}^n$, le problème de Cauchy admet une unique solution définie sur $[0,T]$.

14.3.2 Méthode d'Euler

Pour un temps de simulation compris entre t_0 et t_1, si on choisi un nombre n d'échantillons, alors le pas d'intégration est défini par $h = \dfrac{t_1 - t_0}{n}$. (On a donc $t_i = t_0 + h \cdot i$ avec $i \in [0,n]$.)

Résultat –

En intégrant l'équation du problème de Cauchy sur un intervalle $[t_i, t_{i+1}]$, on a :

$$\int_{y_i}^{y_{i+1}} \mathrm{d}y = \int_{t_i}^{t_{i+1}} f(t,y(t))\mathrm{d}t$$

$$\Leftrightarrow y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t, y(t))\mathrm{d}t \text{ et donc : } y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t, y(t))\mathrm{d}t$$

En utilisant la méthode des rectangles à gauche, $\int_{t_i}^{t_{i+1}} f(t, y(t))\mathrm{d}t \simeq h \cdot f(t_i, y(t_i))$.

14.3.3 Méthode d'Euler explicite

À l'instant i, $\dfrac{\mathrm{d}y(t)}{\mathrm{d}t}$ peut être approximé par $\dfrac{\mathrm{d}y(t_i)}{\mathrm{d}t} \simeq \dfrac{y_{i+1} - y_i}{h}$. Ainsi, $y_{i+1} = y_i + hf(t_i, y_i)$.

14.3.4 Méthode d'Euler implicite

À l'instant i, $\dfrac{\mathrm{d}y(t)}{\mathrm{d}t}$ peut être approximé par $\dfrac{\mathrm{d}y(t_i)}{\mathrm{d}t} \simeq \dfrac{y_i - y_{i-1}}{h}$. Ainsi, $y_i = y_{i-1} + hf(t_{i-1}, y_i)$ ou encore $y_{i+1} = y_i + hf(t_i, y_{i+1})$.

Bibliothèque Python

Voir exemples : `http://python.physique.free.fr/outils_math.html` ou `https://cpge.frama.io/fiches-cpge/Python/00-Outils/`

14.3.5 Exemples – Reformulation d'équations différentielles en vue de leur résolution numérique.

Équation différentielle du premier ordre à coefficients constants

On a $\omega(t) + \tau\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} = \omega_c$.

En utilisant la formulation du problème de Cauchy, on a $f(t, y(t)) = \dfrac{\mathrm{d}}{\mathrm{d}t}[\omega(t)] = \dfrac{1}{\tau}(\omega_c - \omega(t))$.

Pour la méthode d'Euler explcite, on a donc, $\omega[k+1] = \omega[k] + \dfrac{h}{\tau}(\omega_c - \omega[k])$.

Pour la méthode d'Euler implicite, on a $\omega[k+1] = \omega[k] + \dfrac{h}{\tau}(\omega_c - \omega[k+1]) \Leftrightarrow \omega[k+1]\left(1 + \dfrac{h}{\tau}\right) = \omega[k] + \dfrac{h}{\tau}\omega_c \Leftrightarrow \omega[k+1]\left(1 + \dfrac{h}{\tau}\right) = \dfrac{\tau}{1+h}\left(\omega[k] + \dfrac{h}{\tau}\omega_c\right)$.

On retrouve la même chose en discrétisant l'équation différentielle : pour un schéma d'Euler explicite, on a $\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} \simeq \dfrac{\omega_{k+1} - \omega_k}{h} \Rightarrow \omega_k + \tau\dfrac{\omega_{k+1} - \omega_k}{h} = \omega_c \Leftrightarrow \omega_{k+1} = \dfrac{h}{\tau}(\omega_c - \omega_k) + \omega_k$.

Avec le schéma d'Euler implicite : on a $\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} \simeq \dfrac{\omega_k - \omega_{k-1}}{h} \Rightarrow \omega_k + \tau\dfrac{\omega_k - \omega_{k-1}}{h} = \omega_c \Leftrightarrow \omega_k = \dfrac{h\omega_c + \tau\omega_{k-1}}{h + \tau}$.

Équation différentielle du premier ordre à coefficients non constants

On a : $\omega(t)f(t) + \tau\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t}g(t) = \omega_c h(t)$:

- Euler explicite : $\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} \simeq \dfrac{\omega_{k+1}-\omega_k}{h} \Rightarrow \omega_k f_k + \tau\dfrac{\omega_{k+1}-\omega_k}{h}g_k = \omega_c h_k \Leftrightarrow \omega_{k+1} = \dfrac{h}{\tau g_k}(\omega_c h_k - \omega_k f_k) + \omega_k$;
- Euler implicite : $\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} \simeq \dfrac{\omega_k-\omega_{k-1}}{h} \Rightarrow \omega_k f_k + \tau\dfrac{\omega_k-\omega_{k-1}}{h}g_k = \omega_c h_k \Leftrightarrow \omega_k = \dfrac{h\omega_c h_k + \tau g_k \omega_{k-1}}{f_k h + \tau g_k}$.

Équation différentielle du premier ordre

On a $\sin(\omega(t)) + \dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} = K$:

- Euler explicite : $\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} \simeq \dfrac{\omega_{k+1}-\omega_k}{h} \Rightarrow \sin\omega_k + \dfrac{\omega_{k+1}-\omega_k}{h} = K \Leftrightarrow \omega_{k+1} = h(K - \sin\omega_k) + \omega_k$;
- Euler implicite : $\dfrac{\mathrm{d}\omega(t)}{\mathrm{d}t} \simeq \dfrac{\omega_k-\omega_{k-1}}{h} \Rightarrow \sin\omega_k + \dfrac{\omega_k-\omega_{k-1}}{h} = K \Leftrightarrow h\sin\omega_k + \omega_k - \omega_{k-1} = hK$. Dans ce cas, il faut utiliser la méthode de Newton ou de dichotomie pour déterminer ω_k.

Équation différentielle du second ordre

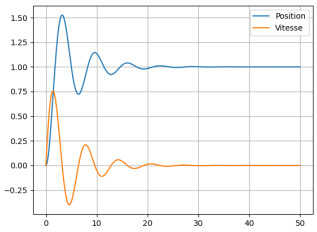
Soit l'équation $\ddot{s}(t) + \dfrac{2\xi}{\omega_0}\dot{s}(t) + \omega_0^2 s(t) = Ke(t)$ avec $s(0) = 0$ et $\dot{s}(t) = 0$.

Utilisation de la fonction de Cauchy En utilisant le problème de Cauchy, on pose :

$$Y(t) = \begin{cases} y_1(t) = s(t) \\ y_2(t) = \dot{s}(t) \end{cases} \Rightarrow \begin{cases} \dot{y_1}(t) = \dot{s}(t) = y_2(t) \\ \dot{y_2}(t) = \ddot{s}(t) \end{cases} \Rightarrow \begin{cases} \dot{y_1}(t) = y_2(t) \\ \dot{y_2}(t) = Ke(t) - \dfrac{2\xi}{\omega_0}y_2(t) - \omega_0^2 y_1(t) \end{cases}$$

En utilisant le schéma d'euler explicte, on a donc $Y[k+1] = Y[k] + h\begin{bmatrix} y_2[k] \\ Ke[k] - \dfrac{2\xi}{\omega_0}y_2[k] - \omega_0^2 y_1[k] \end{bmatrix}$

et donc $\begin{bmatrix} y_1[k+1] \\ y_2[k+1] \end{bmatrix} = \begin{bmatrix} y_1[k] \\ y_2[k] \end{bmatrix} + h\begin{bmatrix} y_2[k] \\ Ke[k] - \dfrac{2\xi}{\omega_0}y_2[k] - \omega_0^2 y_1[k] \end{bmatrix} = \begin{bmatrix} y_1[k] + hy_2[k] \\ y_2[k] + h\left(Ke[k] - \dfrac{2\xi}{\omega_0}y_2[k] - \omega_0^2 y_1[k]\right) \end{bmatrix}$




```python
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt

# Définition des conditions initiales
y0, v0 = 0,0

# Définition du système d'équation différentielle
def systeme(t,Y):
    K, xi, om0, e = 1,0.2,1,1
    y1 = Y[0]
    y2 = Y[1]

    dy1_dt = y2
    dy2_dt = K*e-2*xi/om0*y2-om0*om0*y1


    return [dy1_dt,  dy2_dt]


# Résolution
solution = solve_ivp(systeme, [0, 50], [y0, v0], max_step=0.01)

plt.plot(solution.t, solution.y[0], label="Position")
plt.plot(solution.t, solution.y[1], label="Vitesse")
plt.grid()
plt.legend()
plt.show()
```

Discrétisation de l'équation différentielle Schéma d'Euler explicite : $\dfrac{\mathrm{d}y(t)}{\mathrm{d}t} \simeq \dfrac{y_{k+1}-y_k}{h}$.

$$\text{On a donc : } \begin{cases} \dfrac{y_{1,k+1}-y_{1,k}}{h} = y_{2,k} \\ \dfrac{y_{2,k+1}-y_{2,k}}{h} + \dfrac{2\xi}{\omega_0} y_{2,k} + \omega_0^2 y_{1,k} = K e_k \end{cases}$$

$$\Leftrightarrow \begin{cases} y_{1,k+1} = h y_{2,k} + y_{1,k} \\ y_{2,k+1} = hKe_k - \dfrac{2\xi h}{\omega_0} y_{2,k} - h\omega_0^2 y_{1,k} + y_{2,k} \end{cases}$$

Avec Euler implicite $\dfrac{\mathrm{d}y(t)}{\mathrm{d}t} \simeq \dfrac{y_k - y_{k-1}}{h}$.

$$\text{On a donc : } \begin{cases} \dfrac{y_{1,k}-y_{1,k-1}}{h} = y_{2,k} \\ \dfrac{y_{2,k}-y_{2,k-1}}{h} + \dfrac{2\xi}{\omega_0} y_{2,k} + \omega_0^2 y_{1,k} = K e_k \end{cases}$$

$$\Leftrightarrow \begin{cases} y_{1,k} = h y_{2,k} + y_{1,k-1} \\ y_{2,k} = \dfrac{hKe_k + y_{2,k-1} - h\omega_0^2 y_{1,k-1}}{1 + h\dfrac{2\xi}{\omega_0} + \omega_0^2 h^2} \end{cases}$$

Suite ...

- Équation différentielle du second ordre : $\ddot{\theta}(t) + k \sin \theta(t) = 0$. On pose :

$$\begin{cases} y_0(t) = \theta(t) \\ y_1(t) = \dot{\theta}(t) \end{cases} \Rightarrow \begin{cases} \dot{y_0}(t) = \dot{\theta}(t) = y_1(t) \\ \dot{y_1}(t) = \ddot{\theta}(t) \end{cases} \Rightarrow \begin{cases} \dot{y_0}(t) = y_1(t) \\ \dot{y_1}(t) + k \sin y_0(t) = 0 \end{cases}$$

Discrétisation avec Euler explicite

$$\begin{cases} \dfrac{y_{0,k+1} - y_{0,k}}{h} = y_{1,k} \\ \dfrac{y_{1,k+1} - y_{1,k}}{h} + k \sin y_{0,k} = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} y_{0,k+1} = h y_{1,k} + y_{0,k} \\ y_{1,k+1} = y_{1,k} - kh \sin y_{0,k} \end{cases}$$

Discrétisation avec Euler implicite

$$\begin{cases} \dfrac{y_{0,k} - y_{0,k-1}}{h} = y_{1,k} \\ \dfrac{y_{1,k} - y_{1,k-1}}{h} + k \sin y_{0,k} = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} y_{0,k} = h y_{1,k} + y_{0,k-1} \\ y_{1,k} = y_{1,k-1} - kh \sin y_{0,k} \end{cases}.$$

14.4 Résolution de systèmes linéaires

14.4.1 Utilisation du pivot de Gauss

Le pivot de Gauss est une méthode permettant de résoudre les systèmes linéaires. Sur l'idée du pivot de Gauss, il est alors possible de calculer l'inverse d'une matrice (lorsqu'elle est inversible), de calculer le déterminant d'une matrice $A \in \mathcal{M}_n(\mathbb{R})$ ou de calculer le rang de $A \in \mathcal{M}_{n,p}(\mathbb{R})$.

Outre la compréhension et la mise en œuvre de l'algorithme, deux problèmes numérique peuvent se poser :

- les erreurs d'arrondis peuvent provoquer des erreurs importantes selon la méthode choisie ;
- des comparaisons d'un réel à zéro peuvent aussi engendrer des erreurs numériques.

Remarque

Les résultats du cours sont écrits avec des coefficients réels mais restent vrais avec des coefficients complexes.

Définitions

Définition – Système linéaire

On dit d'un système qu'il est linéaire s'il est de la forme :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p = b_1 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{np}x_p = b_n \end{cases}$$

On note $n, p \in \mathbb{N}^*$ (n équations et p inconnues), $\forall i \in \{1, \ldots, n\}$, $\forall j \in \{1, \ldots, p\}$, $a_{ij} \in \mathbb{R}$. i désigne l'indice de la ligne et j l'indice de la colonne.

$b_1, ..., b_n \in \mathbb{R}$ est appelé second membre.

Définition – Système homogène

On dit que le système est homogène su $b_1 = b_2 = ... = b_n = 0$.

Exemple –

Exemple de système linéaire (S) et son système homogène associé (S_0) :

$$(S)\begin{cases} x + y - 2z + 4t = 5 \\ 2x + 2y - 3z + t = 3 \\ 3x + 3y - 4z - 2t = 1 \end{cases} \qquad (S_0)\begin{cases} x + y - 2z + 4t = 0 \\ 2x + 2y - 3z + t = 0 \\ 3x + 3y - 4z - 2t = 0 \end{cases}$$

Exemple de système non linéaire :

$$\begin{cases} x + \cos \mathbf{y} + \mathbf{xy} = 2 \\ x - \mathbf{y^2} = 4 \end{cases}$$

Définition – Système linéaire

Une solution d'un système linéaire est un p-uplet de réels, c'est-à-dire un élément de $(x_1, ..., x_p)$ qui vérifie les n équations.

Définition – Système compatible

Si un système a au moins une solution, il est dit compatible (incompatible sinon).

Opérations élémentaires

Définition –

Les opérations élémentaires sont les suivantes :

- opération de transvection (*Gaussian Elimination*) :
  - $(L_i) \leftarrow (L_i) + \lambda(L_j)$ où $i, j \in \{1, ..., n\}, i \neq j, \lambda \in \mathbb{R}$;
  - $(L_i) \leftarrow \alpha(L_i)$ où $i \in \{1, ..., n\}, \alpha \in \mathbb{R}^*$;
- opération d'échange de lignes :
  - $(L_i) \leftrightarrow (L_j)$ où $i, j \in \{1, ..., n\}$.

Remarque

Pour éviter les fractions on peut également utiliser $(L_i) \leftarrow \alpha(L_i) + \lambda L_j$ où $i, j \in \{1, ..., n\}, \alpha \neq 0, \lambda \in \mathbb{R}$ qui est une combinaison des deux premiers items.

Proposition 14.4.1 *L'utilisation des opérations élémentaires sur un système ne change pas l'ensemble de ses solutions. Autrement dit, elles donnent des systèmes équivalents au premier.*

Notation matricielle

Remarque

Les opérations élémentaires sur les lignes du système n'opèrent que sur les coefficients.

Définition –

Au système défini précédemment, on associe la matrice de ses coefficients :

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1(p-1)} & a_{1p} \\ & & a_{ij} & & \\ a_{n1} & a_{n2} & \dots & a_{n(p-1)} & a_{np} \end{pmatrix} \in \mathcal{M}_{n,p}(\mathbb{R})$$

$\mathcal{M}_{n,p}(\mathbb{R})$ désigne l'ensemble des matrices à coefficients réels à n lignes et p colonnes.

Définition –

On note $B = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$ la matrice de second membre et $(A|B)$ la matrice A augmentée de B.

Définition –

Si deux matrices M et M' diffèrent d'un nombre fini d'opération sur les lignes, on dit qu'elles sont équivalentes en lignes et on note $M \underset{L}{\sim} M'$.

Remarque

Sous forme matricielle le système s'écrit $AX = B$ avec X le vecteur inconnu.

14.4.2 Pivot de Gauss

Définition – Pivot d'une ligne

On appelle pivot d'une ligne le premier nombre non nul de cette ligne.

Algorithme du pivot

Premier cas, chacun des coefficients de la première colonne est nul. En conséquence, on note $M = \left(\begin{array}{c|c} 0 & \\ \vdots & M \\ 0 & \end{array}\right)$

Dans un second cas, la première colonne contient au moins un nombre non nul.

Quitte à effectuer un changement de ligne, on se ramène au cas où $a_{11} \neq 0$: $\left(\begin{array}{c|ccc} a_{11} & \alpha & \alpha & \alpha \\ \hline \alpha & & & \\ \alpha & & \alpha & \\ \alpha & & & \end{array}\right)$ Les α représentent des nombres possiblement nuls.

On fait apparaître des zéros sous a_{11} avec des opérations de la forme $(L_2) \leftarrow (L_2) - \frac{a_{21}}{a_{11}}(L_1)$
$$\left(\begin{array}{c|ccc} a_{11} & \alpha & \alpha & \alpha \\ \hline 0 & & & \\ \vdots & & M' & \\ 0 & & & \end{array}\right)$$

On effectue le pivot à nouveau sur la matrice M'.

En généralisant, lorsqu'on en est à la ligne i, les lignes $i+1$ à n subissent la transvection suivante : $\forall k \in [i+1, n] \quad L_k \leftarrow L_k - \frac{a_{k,i}}{a_{i,i}} L_i$

Remarques

Le nombre de colonnes diminue à chaque pivot.
L'algorithme se termine en un nombre fini d'étapes.

Exemple –

Soit le système suivant à résoudre ainsi que la matrice augmentée qui lui est associée : $\begin{cases} x+y-2z+4t=5 \\ 2x+2y-3z+2t=3 \\ 3x+3y-4z-2t=1 \\ 1x+2y+3z+3t=-1 \end{cases}$ $\left(\begin{array}{cccc|c} 1 & 1 & -2 & 4 & 5 \\ 2 & 2 & -3 & 2 & 3 \\ 3 & 3 & -4 & -2 & 1 \\ 1 & 2 & 3 & 3 & -1 \end{array}\right)$

$$\begin{cases} (L_2) \leftarrow (L_2) - 2(L_1) \\ (L_3) \leftarrow (L_3) - 3(L_1) \\ (L_4) \leftarrow (L_4) - (L_1) \end{cases} \Longrightarrow \left(\begin{array}{cccc|c} 1 & 1 & -2 & 4 & 5 \\ 0 & 0 & 1 & -7 & -6 \\ 0 & 0 & 2 & -14 & -14 \\ 0 & 1 & 5 & -1 & -6 \end{array}\right)$$

$$(L_2) \leftrightarrow (L_4) \Longrightarrow \left(\begin{array}{cccc|c} 1 & 1 & -2 & 4 & 5 \\ 0 & 1 & 5 & -1 & -6 \\ 0 & 0 & 2 & -14 & -14 \\ 0 & 0 & 1 & -6 & -7 \end{array}\right)$$

$$(L_4) \leftarrow (L_4) - \frac{1}{2}(L_3) \Longrightarrow \left(\begin{array}{cccc|c} 1 & 1 & -2 & 4 & 5 \\ 0 & 1 & -1 & -7 & -6 \\ 0 & 0 & 2 & -14 & -14 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

Remarque

Numériquement, le codage du 0 posant des problèmes informatiques, il est difficile de s'assurer qu'un pivot est non nul. On utilisera donc la méthode du **pivot partiel**. Le pivot utilisé ne sera donc pas le premier nombre non nul d'une ligne, mais le plus grand élément d'une colonne (en valeur absolue). Si le pivot n'est pas sur la première ligne, on effectuera les échanges de lignes nécessaires. On s'affranchit ainsi de comparaisons à 0.

Matrice échelonnée

A la fin du pivot, on obtient une matrice de la **forme** : $\left(\begin{array}{cccc|c} \alpha & \beta & \beta & \beta & \beta \\ 0 & \alpha & \beta & \beta & \beta \\ 0 & 0 & \alpha & \beta & \beta \\ 0 & 0 & 0 & \alpha & \beta \end{array}\right)$ avec α non nul et β réel quelconque.

Définition – Matrice échelonnée

On dit d'une matrice qu'elle est échelonnée en ligne si :

1. une ligne est nulle, les suivantes le sont aussi ;
2. L_1, ..., L_r désignent les lignes non nulles, $j(L_1)$, ..., $j(L_r)$ désignent la position des pivots dans ces lignes et $j(L_1) < ... < j(L_r)$.

Exemple –

Les matrices suivantes ne sont pas échelonnées :

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 2 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Proposition 14.4.2 *Le pivot de Gauss permet d'obtenir une matrice échelonnée en lignes.*

Définition –

Le rang d'une matrice échelonnée désigne le nombre de ses pivots (ce qui correspond aussi au nombre de lignes non nulles).

Matrices échelonnées réduites

On peut poursuivre le pivot en :

1. divisant les lignes non nulles par leur pivot, les pivot deviennent alors 1 ;
2. faisant apparaître des zéros au dessus des pivots.

Proposition 14.4.3 *Toute matrice est équivalente en ligne à une et une seule matrice échelonnée réduite. En conséquence, on peut définir le rang d'une matrice quelconque comme le rang de sa matrice échelonnée réduite en ligne. On peut définir (le nombre) le rang d'un système comme le rang de la matrice de ses coefficients* (A).

Remarque

Deux matrices équivalentes en ligne ont le même rang.

14.4.3 Implémentation de l'algorithme

Choix d'un type de données

On verra ultérieurement que la bibliothèque numpy permet de gérer plus aisément le calcul matriciel. Dans un premier temps, le système à résoudre sera mis sous forme matricielle afin que les valeurs puissent être stockées dans un tableau. On cherchera donc à résoudre le système suivant :

$$A \cdot X = B$$

La matrice $A \in \mathcal{M}_{n,n}(\mathbb{R})$ sera stockée dans un tableau de n lignes et n colonnes. Le vecteur B, second membre de l'équation, sera stocké dans un tableau de n lignes et une colonne.

Fonctions élémentaires

Recherche du pivot Dans le cadre d'une résolution numérique, on utilise un pivot partiel.

```
def recherche_pivot(A,i):
    n = len(A) # le nombre de lignes
    j = i # la ligne du maximum provisoire
    for k in range(i+1, n):
        if abs(A[k][i]) > abs(A[j][i]):
            j = k # un nouveau maximum provisoire
    return j
```

On considère qu'on est à la ligne i. À la ligne i, les éléments compris entre les colonnes 0 et $i-1$ sont théoriquement nuls. Initialement, le plus grand élément de la ligne est a priori en position i.

On cherche alors le plus grand élément, en valeur absolu de la ligne $i+1$à la ligne n (n exclus en Python).

On renvoie enfin l'indice du pivot.

Échange de lignes L'échange de lignes peut être nécessaire pour réordonner les lignes de la matrice si le pivot de la ligne considéré est plus petit que le pivot d'une des lignes suivantes.

```
def echange_lignes(A,i,j):
    # Li <-->Lj
    A[i][:],A[j][:]=A[j][:],A[i][:]
```

Remarque

En python, le passage des objets se faisant par référence, il n'est pas nécessaire de retourner une liste.

```
def transvection_ligne(A, i, j, mu):
    # L_i <- L_i + mu.L_j
    nc = len(A[0]) # le nombre de colonnes
    for k in range(nc):
        A[i][k] = A[i][k] + mu * A[j][k]
```

Phase de remontée : résolution du système Une fois toutes les transvections réalisées, on est en présence d'une matrice échelonnée T. Le système a donc été mis sous la forme $T \cdot X = Y$. On va donc résoudre le système en partant «du bas».

```
def remontee(A,B):
    n=len(A)
    X = [0.] * n
    for i in range(n-1, -1, -1):
        somme=0
        for j in range (i+1,n):
            somme=somme+A[i][j]*X[j]
        X[i]=(B[i][0]-somme)/A[i][i]
        print(X[i])
    return X
```

Algorithme complet

```python
def resolution(AA, BB):
    """Résolution de AA.X=BB; AA doit etre inversible"""
    A, B = AA.copy(), BB.copy()
    n = len(A)
    assert len(A[0]) == n
    # Mise sous forme triangulaire
    for i in range(n):
        j = recherche_pivot(A, i)
        if j > i:
            echange_lignes(A, i, j)
            echange_lignes(B, i, j)
        for k in range(i+1, n):
            mu = - A[k][i] / A[i][i]
            transvection_ligne(A, k, i, mu)
            transvection_ligne(B, k, i, mu)
    # Phase de remontée
    return remontee(A,B)
```

14.4.4 Résolution de systèmes linéaires avec Numpy

Il est possible d'utiliser le bibliothèque Numpy pour résoudre le système $A \cdot X = B$. Pour cela, les matrice A et B peuvent être mises sous la forme d'une liste de liste.

```python
import numpy as np
A = [[1,1,-2,4],
     [2,2,-3,2],
     [3,3,-4,-2],
     [1,2,3,3]]
B = [[5],[3],[1],[-1]]

np.linalg.solve(A,B)
    array([[-3.80000000e+01],
           [ 2.90000000e+01],
           [-7.00000000e+00],
           [-3.62672855e-15]])
```