TP 1
Résolution de systèmes linéaires

À la fin de la séance: envoyer vos programmes (format .py) et graphiques (format pdf) à l'adresse ~~vincent.ballenegger@univ-fcomte.fr~~ *(spécifier PAN2:TP 1 comme titre de message).*

david.cornu@utinam.cnrs.fr

1. Méthode d'élimination de Gauss

A. **Echauffement: calcul matriciel**

On considère le système d'équations $A\vec{x} = \vec{b}$, où $A = \begin{bmatrix} 4 & 8 & 12 \\ 3 & 8 & 13 \\ 2 & 9 & 18 \end{bmatrix}$ et $\vec{b} = \begin{bmatrix} 4 \\ 5 \\ 11 \end{bmatrix}$

1) Définir la matrice A ~~dans la console ipython de Spyder.~~ Avec l'outil de votre choix
Indication: créer un tableau numpy via une conversion liste → tableau.
2) Calculer le déterminant de A (utiliser le module *numpy.linalg*).
3) Calculer la matrice inverse A^{-1}.
4) Définir le vecteur $\vec{b}$ (sous forme d'un tableau *unidimensionel* contenant les composantes du vecteur), puis calculer la solution du système via la formule $\vec{x} = A^{-1}\vec{b}$.
5) Dans la console, définir les deux tableaux `bv = b.reshape(3,1)` et `bh = b.reshape(1,3)` et les afficher. Calculer ensuite `b @ b`, `bh @ bv`, `bv @ bh` et `np.outer(b,b)`.

B. **Programmation de la méthode de Gauss**

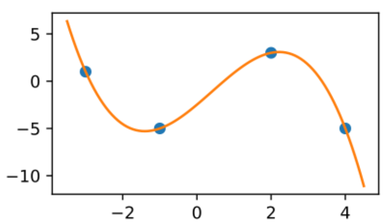
1) Dans un programme Python, définir A comme la matrice augmentée (A|b) correspondant au système défini précédemment.
2) Que retournent les 3 expressions `A.shape`, `A.shape[0]` et `A.shape[1]` ?
3) Définir une fonction `PivotGauss(A)` qui résout le système donné en argument par la méthode d'élimination de Gauss (*cf cours,* ~~*diapo 13*~~). La fonction doit retourner la liste des solutions sous forme d'un tableau, c'est-à-dire le vecteur $\vec{x}$.
Conseils:
- Définir n comme A.shape[0].
- Utiliser des sous-tableaux pour effectuer de manière compacte des opérations sur des lignes ou colonnes.
- Pour augmenter la lisibilité du programme, définir un vecteur $\vec{b}$ comme une vue sur la dernière colonne de la matrice augmentée (b est un sous-tableau).
4) Tester votre fonction `PivotGauss(A)` en l'appliquant au système défini au point 1).

2. Polynôme interpolant

On cherche un polynôme (de degré 3) qui passe par les points

x	-3	-1	2	4
y	1	-5	3	-5

1) Ecrire le système linéaire associé et le résoudre. Numériquement
2) Afficher sur un même graphique le polynôme et les points d'interpolation (utiliser matplotlib). →Graphique2.pdf



3. Résolution de l'équation de Poisson en 1D

La discrétisation de l'équation de Poisson

$$\Delta V(\vec{r}) \;=\; -\frac{1}{4\pi\varepsilon_0}\rho(\vec{r})$$

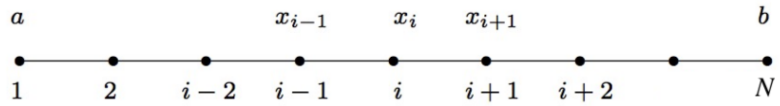
conduit à un système d'équations linéaires pour les valeurs de la fonction $V(\vec{r})$ aux points de la grille. Considérons le cas à 1 dimension. L'équation de Poisson est alors une équation différentielle de la forme

$$V''(x) = -f(x)$$

où $f(x) \;=\; \rho(x)/4\pi\varepsilon_0$. Cherchons à résoudre numériquement cette équation, pour x dans l'intervalle $[-1, 1]$, dans le cas où la distribution de charge est $f(x) = 6x$. L'équation étant d'ordre 2, il faut imposer 2 conditions aux limites sur la fonction $V(x)$ pour que la solution soit unique. Le potentiel électrostatique est ainsi supposé fixé aux points $x = -1$ et $x = 1$:

$$V(-1) = -6 \qquad \text{et} \qquad V(1) = 6.$$

Nous cherchons à résoudre l'équation de Poisson numériquement. Discrétisons alors l'intervalle d'intérêt $[a, b] = [-1, 1]$ en N points équidistants x_i:



On a $x_1 = a$, $x_N = b$ et on définit $h \;=\; (b-a)/(N-1)$ comme la distance entre les points. Dénotons, à partir de maintenant, le potentiel par $u(x)$ plutôt que $V(x)$. $u_i = u(x_i)$ représente le potentiel au point i de la grille.

1) Montrer que $f''(x) \approx \frac{f(x-h)-2f(x)+f(x+h)}{h^2}$ pour toute fonction $f(x)$.
   Indication: utiliser la formule des différences finies pour la dérivée: $f'(x) \approx \frac{f\left(x+\frac{h}{2}\right)-f\left(x-\frac{h}{2}\right)}{h}$
2) En déduire que $u''(x_i) \approx \frac{u_{i-1}-2u_i+u_{i+1}}{h^2}$.
   Lorsqu'on applique l'équation de Poisson à chaque point *à l'intérieur* de l'intervalle, on obtient ainsi le système d'équations
   $u_{i-1} - 2u_i + u_{i+1} = h^2 f_i$ pour i = 2, 3, …, N–1.
3) Justifier que la matrice associée au système d'équations que satisfait le vecteur $\vec{u} = [u_1 \quad u_2 \quad \ldots \quad u_N]^{\mathrm{T}}$ est

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

   Que vaut le membre de droite du système linéaire $A\vec{u} \;=\; \vec{b}$? Combien de lignes/colonnes la matrice contient-elle?

5) Définir la matrice A, le vecteur b et la matrice augmentée C=(A|b) dans un programme Python que vous nommerez `Ex3.py`.
   Conseil: utiliser `np.hstack((T1, T2))` *pour accoler horizontalement deux tableaux T1 et T2 (matrice ou vecteur). Pour convertir le vecteur b en un tableau bi-dimensionel de 100 lignes et 1 colonne, utiliser* `b.reshape(100,1)`.
4) Déterminer le potentiel en tous les points de la grille en résolvant le système $A\vec{u} \;=\; \vec{b}$. Prendre N=100 comme taille de grille. *Indication: utiliser la fonction PivotGauss() ou la fonction numpy.linalg.solve().*
5) Tracer, avec matplotlib, un graphique du potentiel $V(x)$.
6) Déterminer la solution exacte du problème en intégrant analytiquement l'équation de Poisson. L'ajouter au graphique. Légender et enregistrer le graphique au format pdf (→ graphique3a.pdf).
7) Résoudre à nouveau l'équation, mais pour la densité de charge $f(x) \;=\; \pi e^x\,[\pi e^x \cos(\pi e^x) \;+\; \sin(\pi e^x)]$ et les conditions aux limites $u(\pm 1) = \cos\,(\pi e^{\pm 1})$. Tracer la solution numérique et la comparer à la solution exacte ($u(x) \;=\; \cos\,(\pi e^x)$).
8) Calculer l'erreur quadratique moyenne de la solution numérique.
9) Mesurer le temps de calculs avec la commande ~~`%timeit PivotGauss(…)` dans la console ipython.~~
10) Comparer les temps de calculs pour n=500 et n=1000. Commenter les résultats.

```
Import time
...
start = time.time()
... some calculation ...
end = time.time()
...
print "elapsed time : ", (end-start)
```