PSI* 12 février 2018
4 heures

DS DE PHYSIQUE MODELISATION

Eléments de correction (hors codes).

Calculatrice autorisée

Cette épreuve est constituée de deux parties indépendantes.
Rédaction sur copies séparées:
Une copie traitant les questions marquées par un trait dans la marge (corrigée par le professeur de physique)
Une copie traitant les questions non marquées dans la marge (corrigée par le professeur d'informatique)

Problème 1 Equation de Poisson

I.1 - Établissement de l'équation

Q1. Rappeler l'équation de Maxwell-Gauss ainsi que la relation entre le champ $\vec{E}$ et le potentiel électrostatique V. En déduire l'équation de Poisson :

$$\Delta V + \frac{\rho}{\varepsilon_0} = 0 .$$

Préciser les noms et les unités usuelles de ρ et ε_0.

Q2. Citer plusieurs situations physiques en dehors de l'électrostatique pour lesquelles il existe une équation analogue.

I.2 - Équation adimensionnée pour un problème plan

On veut résoudre l'équation de Poisson dans une portion de plan $\mathcal{P}$ carrée de côté L. On pose :

$$X = x/L, Y = y/L .$$

Q3. Montrer qu'on peut écrire l'équation sous la forme suivante :

$$\frac{\partial^2 V(X,Y)}{\partial X^2} + \frac{\partial^2 V(X,Y)}{\partial Y^2} + \rho'(X,Y) = 0$$

où $\rho'(X,Y)$ sera exprimé en fonction de ρ, L et ε_0.

Comme notre problème est plan, on a : $\Delta V(x,y) = \frac{\partial^2 V}{\partial x^2}(x,y) + \frac{\partial^2 V}{\partial y^2}(x,y)$.

On a classiquement (composition des dérivations partielles) :

$$\frac{\partial}{\partial x} = \frac{\partial X}{\partial x} \times \frac{\partial}{\partial X} \text{ et } \frac{\partial}{\partial y} = \frac{\partial Y}{\partial y} \times \frac{\partial}{\partial Y}$$

Or, $X = \frac{x}{L}$ et $Y = \frac{y}{L}$. On en déduit : $\frac{\partial X}{\partial x} = \frac{\partial Y}{\partial y} = \frac{1}{L}$ puis : $\frac{\partial}{\partial x} = \frac{1}{L} \times \frac{\partial}{\partial X}$ et $\frac{\partial}{\partial y} = \frac{1}{L} \times \frac{\partial}{\partial Y}$.

En poursuivant les dérivations suivant le même principe, il vient :

$$\frac{\partial^2}{\partial x^2} = \frac{1}{L^2} \times \frac{\partial^2}{\partial X^2} \text{ et } \frac{\partial^2}{\partial y^2} = \frac{1}{L^2} \times \frac{\partial^2}{\partial Y^2}$$

On a alors :

$$\Delta V + \frac{\rho}{\varepsilon_0} = 0$$

$$\Leftrightarrow \forall (x, y) \in [0 ; L]^2, \Delta V(x, y) + \frac{\rho(x, y)}{\varepsilon_0} = 0$$

$$\Leftrightarrow \forall (x, y) \in [0 ; L]^2, \frac{\partial^2 V}{\partial x^2}(x, y) + \frac{\partial^2 V}{\partial y^2}(x, y) + \frac{\rho(x, y)}{\varepsilon_0} = 0$$

$$\Leftrightarrow \forall (X, Y) \in [0 ; 1]^2, \frac{1}{L^2} \times \frac{\partial^2 V}{\partial X^2}(X, Y) + \frac{1}{L^2} \times \frac{\partial^2 V}{\partial y^2}(X, Y) + \frac{\rho(X, Y)}{\varepsilon_0} = 0$$

$$\Leftrightarrow \forall (X, Y) \in [0 ; 1]^2, \frac{\partial^2 V}{\partial X^2}(X, Y) + \frac{\partial^2 V}{\partial y^2}(X, Y) + \frac{L^2 \times \rho(X, Y)}{\varepsilon_0} = 0$$

L'équation se récrit donc : $\frac{\partial^2 V}{\partial X^2}(X, Y) + \frac{\partial^2 V}{\partial y^2}(X, Y) + \rho'(X, Y) = 0$

avec : $\rho'(X, Y) = \frac{L^2 \times \rho(X, Y)}{\varepsilon_0}$.

I.3 - Discrétisation

Afin de résoudre numériquement l'équation de Poisson, on va utiliser un maillage de $\mathcal{P}$, de pas $h = 1/N$, et on va transformer les dérivées partielles par des différences entre les valeurs de V aux différents points du maillage (on parle aussi des *nœuds* du maillage). La **figure 1** (page suivante) représente le maillage de $\mathcal{P}$ pour $N = 5$.

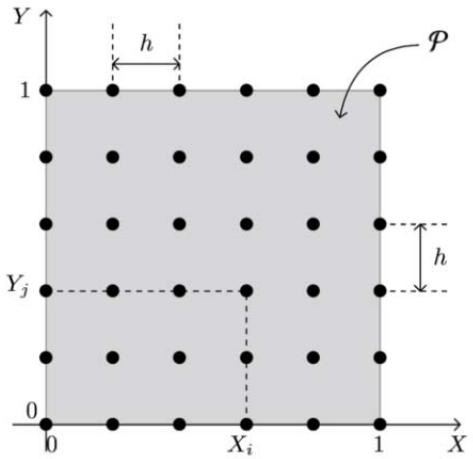

Figure 1 – Maillage de $\mathcal{P}$ pour $N = 5$

Q4. En faisant un développement limité à l'ordre 2 autour du point de coordonnées (X_i, Y_j), montrer qu'on peut exprimer la valeur de $\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}$ en ce point sous la forme suivante :

$$\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2} = \frac{V(X_i + h, Y_j) + V(X_i - h, Y_j) + V(X_i, Y_j + h) + V(X_i, Y_j - h) - 4V(X_i, Y_j)}{h^2} + O(h) .$$

On a, à l'ordre 2 :

$$V\left(X_i + h, Y_j\right) = V\left(X_i, Y_j\right) + h \times \frac{\partial V}{\partial X}\left(X_i, Y_j\right) + \frac{h^2}{2} \times \frac{\partial^2 V}{\partial X^2}\left(X_i, Y_j\right) + \mathrm{o}\left(h^2\right)$$

On peut raisonnablement supposer que le « $\mathrm{o}\left(h^2\right)$ » soit un « $\mathrm{O}\left(h^3\right)$ ». On obtient alors :

$$V\left(X_i + h, Y_j\right) = V\left(X_i, Y_j\right) + h \times \frac{\partial V}{\partial X}\left(X_i, Y_j\right) + \frac{h^2}{2} \times \frac{\partial^2 V}{\partial X^2}\left(X_i, Y_j\right) + \mathrm{O}\left(h^3\right)$$

Puis, en remplaçant h par $-h$:

$$V\left(X_i - h, Y_j\right) = V\left(X_i, Y_j\right) - h \times \frac{\partial V}{\partial X}\left(X_i, Y_j\right) + \frac{h^2}{2} \times \frac{\partial^2 V}{\partial X^2}\left(X_i, Y_j\right) + \mathrm{O}\left(h^3\right)$$

De façon similaire, en raisonnant sur la deuxième variable :

$$V\left(X_i, Y_j + h\right) = V\left(X_i, Y_j\right) + h \times \frac{\partial V}{\partial Y}\left(X_i, Y_j\right) + \frac{h^2}{2} \times \frac{\partial^2 V}{\partial Y^2}\left(X_i, Y_j\right) + \mathrm{O}\left(h^3\right)$$

$$V\left(X_i, Y_j - h\right) = V\left(X_i, Y_j\right) - h \times \frac{\partial V}{\partial Y}\left(X_i, Y_j\right) + \frac{h^2}{2} \times \frac{\partial^2 V}{\partial Y^2}\left(X_i, Y_j\right) + \mathrm{O}\left(h^3\right)$$

En additionnant ces quatre égalités membre à membre, on obtient :

$$\begin{aligned} &V\left(X_i + h, Y_j\right) + V\left(X_i - h, Y_j\right) + V\left(X_i, Y_j + h\right) + V\left(X_i, Y_j - h\right) \\ &\quad = 4V\left(X_i, Y_j\right) + h^2 \times \left(\frac{\partial^2 V}{\partial X^2}\left(X_i, Y_j\right) + \frac{\partial^2 V}{\partial Y^2}\left(X_i, Y_j\right)\right) + \mathrm{O}\left(h^3\right) \\ &\quad = 4V\left(X_i, Y_j\right) + h^2 \times \left(\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right)\left(X_i, Y_j\right) + \mathrm{O}\left(h^3\right) \end{aligned}$$

D'où :

$$\left(\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right)\left(X_i, Y_j\right) = \frac{V\left(X_i + h, Y_j\right) + V\left(X_i - h, Y_j\right) + V\left(X_i, Y_j + h\right) + V\left(X_i, Y_j - h\right) - 4V\left(X_i, Y_j\right)}{h^2} + \mathrm{O}\left(h\right)$$

On obtient bien le résultat cherché.

Q5. Comme $X_i = ih$ et $Y_j = jh$, on note désormais $V(i, j)$ le potentiel $V(X_i, Y_j)$ en un point (X_i, Y_j) du maillage. Montrer alors qu'on peut écrire l'équation de Poisson sous la forme suivante :

$$V(i + 1, j) + V(i - 1, j) + V(i, j + 1) + V(i, j - 1) - 4V(i, j) + \rho''(i, j) = 0 \qquad (1)$$

$\rho''(i, j)$ étant une fonction à définir en fonction de ρ, L, ε_0 et h.

Cette question introduit d'abord le changement de notation : $V\left(X_i, Y_j\right) = V\left(i,\ j\right)$.

D'après la question Q3, on a : $\dfrac{\partial^2 V}{\partial X^2}\left(X_i, Y_j\right) + \dfrac{\partial^2 V}{\partial y^2}\left(X_i, Y_j\right) = -\rho'\left(X_i, Y_j\right) = -\dfrac{L^2 \times \rho\left(X_i, Y_j\right)}{\varepsilon_0}$

L'égalité précédente se récrit donc :

$$\frac{V\left(X_i + h, Y_j\right) + V\left(X_i - h, Y_j\right) + V\left(X_i, Y_j + h\right) + V\left(X_i, Y_j - h\right) - 4V\left(X_i, Y_j\right)}{h^2} + \mathrm{O}\left(h\right) = -\frac{L^2 \times \rho\left(X_i, Y_j\right)}{\varepsilon_0}$$

D'où :

$$V\left(X_i + h, Y_j\right) + V\left(X_i - h, Y_j\right) + V\left(X_i, Y_j + h\right) + V\left(X_i, Y_j - h\right) - 4V\left(X_i, Y_j\right) + \frac{h^2 L^2 \times \rho\left(X_i, Y_j\right)}{\varepsilon_0} = \mathrm{O}\left(h\right)$$

En utilisant les nouvelles notations et en négligeant le terme « $\mathrm{O}\left(h\right)$ », on obtient finalement :

$$V\left(i+1,\ j\right) + V\left(i-1,\ j\right) + V\left(i,\ j+1\right) + V\left(i,\ j-1\right) - 4V\left(i,\ j\right) + \frac{h^2 L^2 \times \rho\left(i,\ j\right)}{\varepsilon_0} = 0$$

Soit :

$$V\left(i+1,\ j\right) + V\left(i-1,\ j\right) + V\left(i,\ j+1\right) + V\left(i,\ j-1\right) - 4V\left(i,\ j\right) + \rho"\left(i,\ j\right) = 0$$

avec :

$$\rho"\left(i,\ j\right) = \frac{h^2 L^2 \times \rho\left(i,\ j\right)}{\varepsilon_0}$$

I.4 - Résolution

La fonction $\rho''(i, j)$ étant connue, on montre en mathématiques que la solution de l'équation de Poisson est unique si on fixe les conditions aux limites sur la frontière $\mathcal{F}$ du domaine $\mathcal{P}$. Ces conditions sont essentiellement de deux types :

- on impose le potentiel en tout point de $\mathcal{F}$ (conditions de Dirichlet),
- on impose une condition sur les dérivées partielles de V en tout point de $\mathcal{F}$ (conditions de Neumann).

Dans ce problème, on ne va considérer que des conditions de Dirichlet.

La frontière $\mathcal{F}$ contient naturellement les points du bord de $\mathcal{P}$ (donc appartenant aux quatre côtés du carré), mais elle peut aussi contenir certains points à l'intérieur de $\mathcal{P}$ où le potentiel est fixé en raison de la présence d'électrodes.

L'ensemble des points de coordonnées (i, j) est donc composé de deux sous-ensembles :

- ceux dont le potentiel est connu, appartenant à la frontière $\mathcal{F}$,
- ceux dont le potentiel est inconnu, appartenant à $\mathcal{P}$ mais pas à $\mathcal{F}$ (donc dans $\mathcal{P}\backslash\mathcal{F}$).

Méthode de Jacobi

À partir de l'équation (1), on peut exprimer :

$$V(i,j) = \frac{1}{4}(V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j-1) + \rho''(i,j)) \,. \tag{2}$$

La résolution s'effectue alors en deux étapes.

— Initialisation

a) On fixe le potentiel des points de $\mathcal{F}$ à la valeur imposée physiquement (bords et électrodes).

b) On donne aux points de potentiel inconnu, donc appartenant à $\mathcal{P}\backslash\mathcal{F}$, une valeur arbitraire $V_0(i,j)$, en général nulle.

— Itérations

On calcule une nouvelle valeur $V_1(i,j)$ des potentiels en appliquant l'équation (2) pour tous les points de $\mathcal{P}\backslash\mathcal{F}$, tandis que $V_1(i,j) = V_0(i,j)$ pour les points de $\mathcal{F}$.

Le processus est répété jusqu'à obtenir des valeurs du potentiel quasiment stables. En notant k le nombre d'itérations, on a donc pour le point de coordonnées (i,j) n'appartenant pas à la frontière :

$$V_{k+1}(i,j) = \frac{1}{4}(V_k(i+1,j) + V_k(i-1,j) + V_k(i,j+1) + V_k(i,j-1) + \rho''(i,j)) \,. \tag{3}$$

La convergence de la méthode est vérifiée à l'aide du critère de convergence e_k, défini par :

$$e_k = \sqrt{\frac{1}{N^2}\sum_{i,j}(V_{k+1}(i,j) - V_k(i,j))^2} \,. \tag{4}$$

Le calcul sera stoppé au bout de k itérations, quand e_k deviendra inférieur à un seuil de convergence ε fixé arbitrairement.

Implémentation informatique

On va utiliser la bibliothèque `numpy` permettant une utilisation simple des tableaux de flottants à deux dimensions ; un aide-mémoire est disponible en fin de problème

Le chargement des bibliothèques classiques est assuré par les lignes suivantes :

```
# importation des bibliothèques
import numpy as np
import matplotlib.pyplot as plt
import math
```

On supposera que les tableaux `numpy` suivants, utilisés comme arguments dans les fonctions à définir dans les questions qui suivent, ont pour signification :

— `V[i,j]`, $(i,j) \in [\![0 \dots N]\!]^2$: tableau courant du potentiel en un point de $\mathcal{P}$,

— `rhos[i,j]`, $(i,j) \in [\![0 \dots N]\!]^2$: tableau contenant la densité de charge ρ'' en un point de $\mathcal{P}$,

— `frontiere[i,j]`, $(i,j) \in [\![0 \dots N]\!]^2$: tableau de booléens indiquant si le point de coordonnées (i,j) appartient ou non à $\mathcal{F}$. En particulier, tous les points du bord du domaine seront tels que `frontiere[i,j]==True`.

Q6. Écrire la fonction `nouveau_potentiel(V, rhos, frontiere, i, j)` retournant la nouvelle valeur du potentiel au point $(i,j) \in [\![0 \dots N]\!]^2$ selon l'équation (3).

Q7. Montrer que pour modifier toutes les valeurs contenues dans `V[i,j]` pendant une itération, il est nécessaire de disposer d'une copie de ce tableau.

Supposons que nous ne disposions pas d'une copie du tableau `V`.
La mise à jour de `V[i,j]` à l'itération $k+1$ requiert, en particulier, les éléments `V[i-1,j]` et `V[i,j-1]` correspondant aux valeurs correspondant à la *k*ième itération. Si, dans le code, nous faisons classiquement évoluer les indices `i` et `j` dans l'ordre croissant, alors les éléments `V[i-1,j]` et `V[i,j-1]` auront déjà été modifiés lorsque l'on devra modifier `V[i,j]`.
Il est donc indispensable de disposer d'une copie du tableau `V`, les éléments de cette copie servant à calculer les nouvelles valeurs de `V`.

On rappelle que l'attribut `shape` *permet de récupérer les dimensions d'un tableau* `numpy`.

Q8. Écrire la fonction `itere_J(V, rhos, frontiere)` modifiant la totalité du tableau `V[i,j]` lors d'une seule itération et retournant l'erreur calculée conformément à l'équation (4).

Q9. Écrire la fonction `poisson(f_iter, V, rhos, frontiere, eps)` ayant pour premier argument une fonction du même type que celle définie à la question précédente, pour dernier argument `eps` le seuil arbitraire de convergence ε et dont le rôle est de modifier le tableau des potentiels `V[i,j]` jusqu'à convergence.

I.5 - Améliorations

Méthode de Gauss-Seidel

C'est une modification de la méthode de Jacobi, pour laquelle on montre que la convergence est légèrement plus rapide. Supposons que l'on balaye le tableau des potentiels selon les indices i et j croissants : dans ces conditions, les points situés à gauche et en dessous du point courant ont déjà été calculés. On va utiliser ces nouvelles valeurs, probablement plus proches de la solution, dans la formule permettant le calcul de $V_{k+1}(i, j)$. Ceci donne l'algorithme de Gauss-Seidel :

$$V_{k+1}(i,j) = \frac{1}{4}(V_k(i+1,j) + V_{k+1}(i-1,j) + V_k(i,j+1) + V_{k+1}(i,j-1) + \rho''(i,j)) \,. \qquad (5)$$

Q10. Montrer qu'il n'est plus nécessaire de copier le tableau `V[i,j]` pour la mise à jour lors d'une itération en utilisant l'équation (5). Faut-il modifier la fonction `nouveau_potentiel` pour passer de la méthode de Jacobi à celle de Gauss-Seidel ?

Q11. Écrire la fonction `itere_GS(V, rhos, frontiere)` modifiant la totalité du tableau `V[i,j]` lors d'une seule itération et retournant l'erreur calculée conformément à l'équation (4).

Avec le nouvel algorithme, on constate que les deux éléments qui « posaient problème », `V[i-1,j]` et `V[i,j-1]`, correspondent aux valeurs obtenues à l'itération $k+1$. Le fait que ces éléments aient été mis à jour pendant l'itération $k+1$ avant la mise à jour de l'élément `V[i,j]` n'est plus un problème puisque c'est précisément des nouvelles valeurs dont nous avons besoin !

La fonction `nouveau_potentiel` ne requiert aucune modification puisque la seule chose qui importe désormais ce sont les indices, i et j, de l'élément devant être modifié. Formellement, le calcul reste le même, c'est ce que nous ferons du résultat dans la fonction appelant qui va devoir être légèrement modifié.

Méthode de Gauss-Seidel adaptative

Les méthodes de Jacobi et de Gauss-Seidel n'utilisent pas la valeur de $V_k(i, j)$ pour calculer $V_{k+1}(i, j)$. La méthode de sur-relaxation (***S**uccessive **O**ver **R**elaxation method*) consiste à calculer la nouvelle valeur d'un nœud comme une combinaison linéaire de la valeur courante et de celle donnée par le schéma de Gauss-Seidel. En introduisant le paramètre de relaxation ω, on a alors :

$$V_{k+1}(i, j) = (1-\omega)V_k(i, j) + \frac{\omega}{4}(V_k(i+1, j) + V_{k+1}(i-1, j) + V_k(i, j+1) + V_{k+1}(i, j-1) + \rho''(i, j)) \,. \quad (6)$$

L'étude mathématique de cette relation permet de montrer les résultats suivants :

- la méthode converge uniquement si $0 < \omega < 2$ et elle converge plus rapidement que la méthode de Gauss-Seidel si $1 < \omega < 2$,
- il existe une valeur optimale de ω qui permet la convergence avec un nombre d'itérations en $O(N)$ pour une valeur de ε fixée.

Pour la résolution de l'équation de Poisson envisagée dans ce problème (conditions de Dirichlet sur un maillage carré), on montre que la valeur optimale ω_{opt} est :

$$\omega_{\text{opt}} = \frac{2}{1 + \pi/N} \,. \quad (7)$$

Q12. Écrire la fonction `nouveau_potentiel_SOR(V, rhos, frontiere, i, j, omega)` retournant la nouvelle valeur du potentiel au point (i, j) selon l'équation (6).

Q13. Écrire la fonction `itere_SOR(V, rhos, frontiere)` optimale modifiant la totalité du tableau `V[i,j]` lors d'une seule itération et retournant l'erreur calculée conformément à l'équation (4).

La résolution du problème peut alors se faire par un appel de la forme

```
poisson(itere_SOR, V, rhos, frontiere, eps),
```

les tableaux carrés `V, rhos, frontiere` étant de dimensions convenables pour représenter un maillage comportant $(N + 1)^2$ nœuds.

Q14. Quelle est la complexité temporelle de l'appel précédent quand $\omega = \omega_{\text{opt}}$?
La **figure 2** représente, pour $\varepsilon = 10^{-4}$, la durée d'exécution T (en secondes) en fonction de N.
Cette courbe est-elle en accord avec la complexité temporelle attendue ?
Quelle serait la durée d'exécution pour $N = 1\,000$? Commenter.

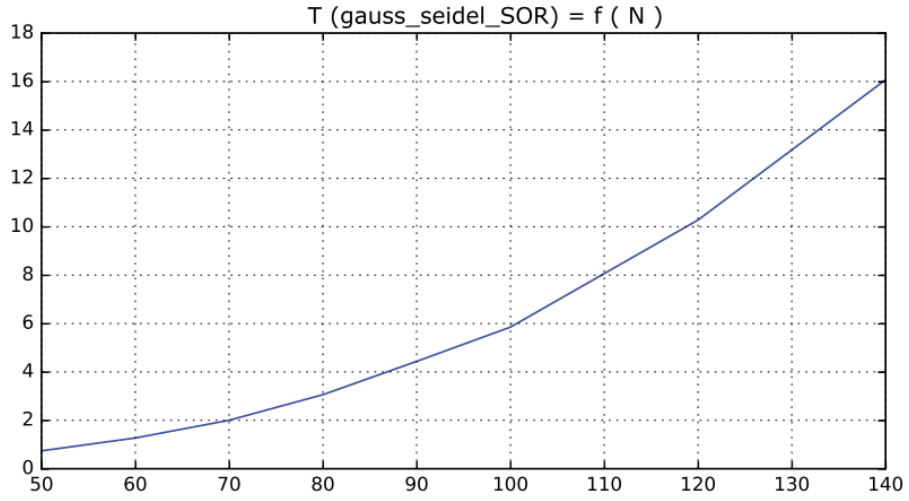

Figure 2 – Durée d'exécution (en secondes) de `poisson(itere_SOR, V, rhos, frontiere, eps)` en fonction de N

Pour la valeur ω_{opt} de ω, l'énoncé nous précise que le nombre d'itérations sera un $O(N)$ pour une valeur de ε fixée. Or, il y a dans la maillage un totale de $(N+1)^2$ points. D'où une complexité temporelle de chaque itération en $O(N^2)$ à chaque itération et donc une complexité temporelle globale en $O(N^3)$.

Par lecture graphique, nous obtenons les couples de valeurs suivants : $(70, 2)$, $(80, 3)$, $(100, 6)$, $(110, 8)$, $(120, 10)$, $(130, 13)$ et $(140, 16)$.

En divisant la seconde valeur par le cube de la première, on obtient une valeur approximativement constante égale à $5,8\times10^{-6}$.

Pour $N = 1000 = 10^3$, on aurait ainsi une durée d'exécution (en secondes) égale à :

$$5,8\times10^{-6}\times\left(10^3\right)^3 = 5,8\times10^{-6}\times10^9 = 5,8\times10^3$$

Soit environ 1 heure et 37 minutes.

Ainsi, aussi performant soit-il, l'algorithme conduit très vite à de longs temps d'exécution qui peuvent s'avérer excessifs d'un point de vue opérationnel et/ou économique.

I.6 - Détermination du champ électrique

Connaissant le potentiel `V[i,j]`, il est souvent nécessaire de calculer numériquement les composantes E_x et E_y du champ électrique au niveau des nœuds du maillage, qui sont alors conservées dans deux tableaux `Ex` et `Ey` dimensionnés correctement (ce sont donc des tableaux carrés de $(N + 1)^2$ éléments).

Q15. Expliquer rapidement comment il serait possible de définir la fonction `calc_ExEy(Ex, Ey, V, h)`, permettant, à partir du tableau `V` et du pas du maillage `h`, le remplissage des deux tableaux `Ex` et `Ey` passés en arguments.
Remarque : on ne demande pas d'écrire le code de la fonction, juste de décrire précisément les étapes de calcul, ainsi que les différents cas à considérer.

On utilise $\vec{E} = -\overrightarrow{\text{grad}}V$ qui donne, comme nous travaillons dans le plan : $E_x = -\dfrac{\partial V}{\partial x}$ et $E_y = -\dfrac{\partial V}{\partial y}$.

En reprenant les notations de la question Q5, On cherche :

$$E_x(x_i, y_j) = E_x(ih,\ jh) = E_X\left(\frac{ih}{L}, \frac{jh}{L}\right) = E_X(X_i, Y_j) = E_X(i,\ j)$$

Avec $h = \dfrac{L}{N}$.

On a :

$$V(X_i + h, Y_j) = V(X_i, Y_j) + h\frac{\partial V}{\partial X}(X_i, Y_j) + O(h) \text{ et } V(X_i - h, Y_j) = V(X_i, Y_j) - h\frac{\partial V}{\partial X}(X_i, Y_j) + O(h)$$

D'où classiquement, en soustrayant : $\dfrac{\partial V}{\partial X}(X_i, Y_j) = \dfrac{V(X_i + h, Y_j) - V(X_i - h, Y_j)}{2h} + O(1)$.

Soit, en négligeant le « $O(1)$ » : $\dfrac{\partial V}{\partial X}(X_i, Y_j) = \dfrac{V(i+1,\ j) - V(i-1,\ j)}{2h}$.

De façon similaire : $\dfrac{\partial V}{\partial Y}(X_i, Y_j) = \dfrac{V(i,\ j+1) - V(i,\ j-1)}{2h}$.

Rappelons que l'on a : $\frac{\partial}{\partial x}=\frac{\partial X}{\partial x}\times\frac{\partial}{\partial X}$ et $\frac{\partial}{\partial y}=\frac{\partial Y}{\partial y}\times\frac{\partial}{\partial Y}$.

Il vient donc :

$$E_X(i,\,j)=E_X\left(X_i,Y_j\right)=E_x\left(x_i,y_j\right)=-\frac{\partial V}{\partial x}\left(x_i,y_j\right)=-\frac{1}{L}\times\frac{\partial V}{\partial X}\left(X_i,Y_j\right)=\frac{V(i-1,\,j)-V(i+1,\,j)}{2Lh}$$

Et : $E_Y(i,\,j)=\frac{V(i,\,j-1)-V(i,\,j+1)}{2Lh}$.

Ces deux expressions ne sont bien sûr pas valables pour $i=0$ ou $j=0$, c'est-à-dire pour tout point de la frontière…

Dans le cas de la frontière, on utilise des approximations non symétriques de certaines dérivées partielles.

Par exemple, quand $i=0$ (bord gauche du carré), on utilisera : $E_X(0,\,j)=\frac{V(0,\,j)-V(1,\,j)}{Lh}$ et quand $i=N$ (bord droit du carré), on utilisera : $E_X(N,\,j)=\frac{V(N-1,\,j)-V(N,\,j)}{Lh}$.

On a des formules similaires pour les bords supérieur et inférieur.
Enfin, on traitera le cas particulier des quatre coins du carré où on doit considérer des approximations non symétriques des <u>deux</u> dérivées partielles.

A partir de ce qui précède, on remplira les tableaux `Ex` et `Ey` en distinguant :

- Le cas général : $(i,\,j)\in[\![1;N-1]\!]^2$ (le point $\left(x_i,y_j\right)$ se trouve à l'intérieur du carré) : deux boucles `for` imbriquées et les formules.
  $E_X(i,\,j)=\frac{V(i-1,\,j)-V(i+1,\,j)}{2Lh}$ et $E_Y(i,\,j)=\frac{V(i,\,j-1)-V(i,\,j+1)}{2Lh}$
- Les cas particuliers $i=0$ et $i=N$ (à chaque fois on fera varier *j* dans $[\![1;N-1]\!]$).
  $E_X(0,\,j)=\frac{V(0,\,j)-V(1,\,j)}{Lh}$ et $E_X(N,\,j)=\frac{V(N-1,\,j)-V(N,\,j)}{Lh}$
  Pour les dérivées partielles suivant *Y*, on utilise les formules du cas général.
- Les cas particuliers $j=0$ et $j=N$ (à chaque fois on fera varier *i* dans $[\![1;N-1]\!]$).
  $E_Y(i,0)=\frac{V(i,0)-V(i,1)}{Lh}$ et $E_Y(i,N)=\frac{V(i,N-1)-V(i,N)}{Lh}$
  Pour les dérivées partielles suivant *X*, on utilise les formules du cas général.
- Les quatre coins $(0,0)$, $(0,N)$, $(N,0)$ et (N,N).
  $E_X(0,0)=\frac{V(0,0)-V(1,0)}{Lh}$ et $E_Y(0,0)=\frac{V(0,0)-V(0,1)}{Lh}$
  $E_X(0,N)=\frac{V(0,N)-V(1,N)}{Lh}$ et $E_Y(0,N)=\frac{V(0,N-1)-V(0,N)}{Lh}$
  $E_X(N,0)=\frac{V(N-1,0)-V(N,0)}{Lh}$ et $E_Y(N,0)=\frac{V(N,0)-V(N,1)}{Lh}$
  $E_X(N,N)=\frac{V(N-1,N)-V(N,N)}{Lh}$ et $E_Y(N,N)=\frac{V(N,N-1)-V(N,N)}{Lh}$

Aide-mémoire numpy/matplotlib/pyplot

Importation des bibliothèques

Les bibliothèques sont importées de la façon suivante :

```
import math
import numpy as np
import matplotlib.pyplot as plt
```

Manipulation des tableaux numpy

La création d'un tableau `numpy` à deux dimensions dont toutes les valeurs sont initialisées à 0 est faite par l'instruction `np.zeros(format)`, `format` étant un doublet de la forme `(n_lignes , n_colonnes)` :

```
>> t0=np.zeros((2,3)); print(t0)
[[ 0.  0.  0.]
 [ 0.  0.  0.]]
```

Pour avoir un tableau rempli de 1, on utilise `np.ones(format)` :

```
>> t1=np.ones((2,2)); print(t1)
[[ 1.  1.]
 [ 1.  1.]]
```

On peut récupérer le format d'un tableau en demandant son attribut `shape`, ce qui retourne un doublet :

```
>> print(t0.shape); print(t1.shape)
(2, 3)
(2, 2)
```

Dans le cas d'un tableau carré, on peut donc récupérer le nombre de lignes, égal au nombre de colonnes, en accédant au premier élément du doublet :

```
>> t1.shape[0]
 2
```

On peut créer un tableau `numpy` de booléens en ajoutant le type `bool`. La valeur 0 est associée à `False`, la valeur 1 à `True` :

```
>> np.zeros((2,3), bool)
[[False False False]
 [False False False]]}
>> np.ones((2,3),bool)
[[ True True True]
 [ True True True]]
```

L'accès à un élément du tableau `a` (en lecture ou en modification) se fait par `a[i,j]`, les lignes et les colonnes étant numérotées à partir de 0 :

```
>> a=np.zeros((2,3)) ; a[0,0]=1 ; a[1,2]=2
>> a
[[ 1. 0. 0.]
 [ 0. 0. 2.]]
```

Une copie indépendante d'un tableau `a` se fait à l'aide de `np.copy(a)`

```
>> b = np.copy(a) ; b[0,0]=3 ; b[1,1]=5
>> a
[[ 1. 0. 0.]
 [ 0. 0. 2.]]
>> b
[[ 3. 0. 0.]
 [ 0. 5. 2.]]
```