Sujet : CCP Modélisation PC 2017
Corrigé partiel

1 Équation de Poisson

1.1 Établissement de l'équation

Q 1.

L'équation locale de Maxwell-Gauss donne la divergence du **champ électrique** $\overrightarrow{E}$ en fonction de la **densité de charge électrique** ρ : $\mathrm{div}\overrightarrow{E} = \frac{\rho}{\varepsilon_0}$ où ε_0 est la **permittivité diélectrique du vide**.

Le champ $\overrightarrow{E}$ et le potentiel électrostatique V sont reliés par l'équation locale de Maxwell-Faraday sans potentiel-vecteur : $\overrightarrow{E} = -\overrightarrow{\mathrm{grad}}\, V$.

Enfin, $\Delta V = \mathrm{div}(\overrightarrow{\mathrm{grad}}\, V)$, donc en calculant la divergence de $\overrightarrow{E}$ dans la seconde équation, on obtient :

$\Delta V = -\mathrm{div}\overrightarrow{E} = -\frac{\rho}{\varepsilon_0}$. D'où l'équation demandée :

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

Avec ρ en $\mathrm{C}\cdot\mathrm{m}^{-3}$, et ε_0 en $\mathrm{F}\cdot\mathrm{m}^{-1}$ ou en $\mathrm{C}\cdot\mathrm{V}^{-1}\cdot\mathrm{m}^{-1}$.

Q 2.

En gravitation newtonienne en faisant l'analogie entre $\overrightarrow{E}$ et $\overrightarrow{G} = \overrightarrow{F}/m$, entre V et $\Phi = Ep/m$, ρ et la masse volumique et enfin entre ε_0 et $-1/(4\cdot\pi\cdot\mathcal{G})$, on retrouve une équation de Poisson sous la forme suivante.

$$\Delta\Phi - 4\cdot\pi\cdot\rho\cdot\mathcal{G} = 0$$

En diffusion thermique en régime stationnaire en écrivant $\mathrm{div}\overrightarrow{j} = P$ où P est la puissance de production interne de chaleur, $\vec{j} = -\lambda\cdot\overrightarrow{\mathrm{grad}}\, T$ où λ est la conductivité thermique du milieu, on retrouve une équation de Poisson sous la forme suivante.

$$\Delta T + \frac{P}{\lambda} = 0$$

En magnétostatique où le champ est engendré exclusivement par de la matière aimantée, en l'absences de courants électriques, et que l'on cherche à déterminer l'excitation magnétique $\overrightarrow{H}$, alors $\mathrm{div}\overrightarrow{H} = \rho_m$ où ρ_m est la densité de charge magnétique, et $\overrightarrow{\mathrm{rot}} = \vec{0}$ qui implique $\overrightarrow{H} = -\overrightarrow{\mathrm{grad}}\, \phi$, on retrouve là encore une équation de Poisson sous la forme :

$$\Delta\phi + \rho_m = 0$$

Autres situations physiques : on peut encore rencontrer l'équation de Poisson dans la modélisation de la diffusion de particules dans un milieu en régime stationnaire, ainsi qu'en mécanique des milieux continus, pour un fluide incompressible en régime permanent.

1.2 Équation adimensionnée pour un problème plan

Q 3.

En procédant à l'adimensionnalisation des coordonnées, on a $x = L\cdot X$ et $X \in [0,1]$. On peut en déduire également que $\partial x = L\cdot\partial X$ et que donc $\partial^2 x = L^2\cdot\partial^2 X$. De même pour y.

De plus, la géométrie du problème permet d'utiliser un système de coordonnées cartésiennes tout en négligeant l'influence de z. On peut donc écrire le laplacien ainsi :

$$\Delta V = \frac{\partial^2 V}{\partial^2 x} + \frac{\partial^2 V}{\partial^2 y} = \frac{1}{L^2}\cdot\left[\frac{\partial^2 V}{\partial^2 X} + \frac{\partial^2 V}{\partial^2 Y}\right]$$

on peut alors poser $\rho'(X,Y) = \frac{L^2 \cdot \rho(X,Y)}{\varepsilon_0}$, et on a alors la forme adimensionnée de l'équation de Poisson pour un problème plan en coordonnées cartésiennes :

$$\frac{\partial^2 V}{\partial^2 X} + \frac{\partial^2 V}{\partial^2 Y} + \frac{L^2 \cdot \rho}{\varepsilon_0} = \Delta V(X,Y) + \rho'(X,Y) = 0$$

1.3 Discrétisation

Pour avoir un pas de h/N, le maillage est basé sur $N+1$ points ou nœuds par direction.

Q 4.

J'utilise la formule de Taylor-Young deux fois, en $(X_i + h, Y_i)$ et $(X_i - h, Y_i)$.
$V(X_i + h, Y_i) = V(X_i, Y_i) + h \cdot \frac{\partial V(X_i,Y_i)}{\partial X_i} + \frac{h^2}{2} \cdot \frac{\partial^2 V(X_i,Y_i)}{\partial X_i^2} + \mathcal{O}(h^3)$
$V(X_i - h, Y_i) = V(X_i, Y_i) - h \cdot \frac{\partial V(X_i,Y_i)}{\partial X_i} + \frac{h^2}{2} \cdot \frac{\partial^2 V(X_i,Y_i)}{\partial X_i^2} + \mathcal{O}(h^3)$
En faisant la somme de ces deux expressions, j'obtiens l'expression suivante.
$V(X_i + h, Y_i) + V(X_i - h, Y_i) = 2 \cdot V(X_i, Y_i) + 2 \cdot \frac{h^2}{2} \cdot \frac{\partial^2 V(X_i,Y_i)}{\partial X_i^2} + \mathcal{O}(h^3)$
De même, il est possible d'écrire ces expressions pour $(X_i, Y_i + h)$ et $(X_i, Y_i - h)$.
$V(X_i, Y_i + h) + V(X_i, Y_i - h) = 2 \cdot V(X_i, Y_i) + 2 \cdot \frac{h^2}{2} \cdot \frac{\partial^2 V(X_i,Y_i)}{\partial Y_i^2} + \mathcal{O}(h^3)$
Alors, sommer ces deux dernières relations aboutit à la relation suivante.
$V(X_i+h, Y_i)+V(X_i-h, Y_i)+V(X_i, Y_i+h)+V(X_i, Y_i-h) = 4 \cdot V(X_i, Y_i) + h^2 \cdot \left[\frac{\partial^2 V(X_i,Y_i)}{\partial X_i^2} + \frac{\partial^2 V(X_i,Y_i)}{\partial Y_i^2}\right] + \mathcal{O}(h^3)$
Finalement, le laplacien de $V(X_i, Y_i)$ s'écrit :

$$\frac{\partial^2 V(X_i,Y_i)}{\partial X_i^2} + \frac{\partial^2 V(X_i,Y_i)}{\partial Y_i^2} = \frac{V(X_i + h, Y_i) + V(X_i - h, Y_i) + V(X_i, Y_i + h) + V(X_i, Y_i - h) - 4 \cdot V(X_i, Y_i)}{h^2} + \mathcal{O}(h)$$

Remarque 1. *Utiliser la formule de Taylor-Young à l'ordre 3 aurait fait apparaître un terme en h^3 qui se serait simplifié lors des additions. Le résultat précédent peut donc s'écrire à l'ordre 2, c'est à dire en remplaçant le $\mathcal{O}(h)$ par $\mathcal{O}(h^2)$.*

Q 5.

L'équation précédente permet **d'approcher** le laplacien de $V(i,j)$ à l'ordre 1 par l'expression suivante.

$$\left[V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j-1) - 4 \cdot V(i,j)\right]/h$$

Si on remplace le laplacien par cette expression dans l'équation de Poisson adimensionnalisée de la question 3, et qu'on multiplie par h^2, on obtient l'approximation ci-dessous.

$$V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j-1) - 4 \cdot V(i,j) + \frac{h^2 \cdot L^2 \cdot \rho}{\varepsilon_0} \approx 0$$

On peut alors identifier $\rho''(i,j) = \frac{h^2 \cdot L^2 \cdot \rho(i,j)}{\varepsilon_0}$.

1.4 Résolution

Méthode de Jacobi

Q 6.

```
def nouveau_potentiel(V, rhos, frontiere, i, j):
    '''renvoie le nouveau potentiel au point (i,j) in [1, N - 1]^2'''
    return( (V[i + 1, j] + V[i - 1, j] + V[i, j + 1] + V[i, j - 1] + \
    rhos[i, j]) / 4 )
```

Q 7.

Avec la relation de récurrence (3) du sujet, $V_{k+1}(i,j)$ nécessite de connaître $V_k(i-1,j)$ et $V_k(i,j-1)$. Or ces deux valeurs, si on ne procède pas à une copie de V_k auront été modifiées lors du calcul de V_{k-1}. Il est donc nécessaire de disposer d'une copie de ces valeurs.

Une solution est donc de copier le tableau V_k avant de calculer les nouveaux potentiels comme le sujet le proposait.

Q 8.

```
def iter_J(V, rhos, frontiere):
    somme_ek, N = 0, 0  # initialisations du calcul de l'erreur
    taille = len(V)     # taille du tableau V
    Vk = np.copy(V)
    for i in range(taille):
        for j in range(taille):
            # parcourt de tout le tableau V
            if not frontiere[i, j]:
                # le point (i, j) n'est pas sur la frontière
                V[i, j] = nouveau_potentiel(Vk, rhos, frontiere, i, j)
                # le calcul d'erreur ne se fait que sur les points
                # hors frontière
                somme_ek += (V[i, j ] - Vk[i, j]) ** 2
                N += 1
    return(pow(somme_ek / N, 0.5))
```

Pour moi, le sujet n'était pas très clair sur le calcul de l'erreur, et sur le N qui intervient dans le calcul de l'erreur moyenne. Ce n'est sans doute pas le N défini à la figure 1 du sujet, mais $N+1$, le nombre de valeurs où l'erreur est calculée.

Je l'ai interprété en ne calculant l'erreur que sur les points hors frontière puisque aux points de la frontière, l'erreur sera identiquement nulle à chaque itération, ce qui diminue artificiellement l'erreur commise.

Remarque 2. *Elle n'était pas demandée, mais la complexité de cette fonction est donc en* $\mathcal{O}((N+1)^2)$.

Q 9.

```
def Poisson(f_iter, V, rhos, frontiere, eps):
    '''fonction qui modifie sur place V, elle ne renvoie donc rien.'''
    erreur = f_iter(V, rhos, frontiere)
    while erreur > eps:
        erreur = f_iter(V, rhos, frontiere)
```

Remarque 3. *Si on note* n_ε*, le nombre d'itérations de la boucle* `while`*, la complexité de cette fonction est alors en* $\mathcal{O}(n_\varepsilon \times (N+1)^2)$.

1.5 Améliorations

Méthode de Gauss-Seidel

Q 10.

Avec la nouvelle relation de recurrence (5), le terme $V_{k+1}(i,j)$ est calculé à partir des termes $V_{k+1}(i-1,j)$ et $V_{k+1}(i,j-1)$ qui ont été calculés aux itérations précédentes de V_{k+1}, et des termes $V_k(i+1,j)$ et $V_k(i,j+1)$ qui n'ont pas encore été modifiés.

Il n'est donc plus nécessaire de copier le tableau `V[i, j]` pour la mise à jour, ce qui épargne une opération en $\mathcal{O}((N+1)^2)$ dans l'algorithme.

Il n'est donc curieusement pas nécessaire de modifier la fonction `nouveau_potentiel()`.

Q 11.

```
def iter_GS(V, rhos, frontiere):
    somme_ek, N = 0, 0  # initialisations du calcul de l'erreur
    taille = len(V)     # taille du tableau V
    for i in range(taille):
        for j in range(taille):
            # parcourt de tout le tableau V
            if not frontiere[i, j]:
                # le point (i, j) n'est pas sur la frontière
                Vk = V[i, j]
                V[i, j] = nouveau_potentiel(V, rhos, frontiere, i, j)
                # le calcul d'erreur ne se fait que sur les points
                # hors frontière
                somme_ek += (V[i, j ] - Vk) ** 2
                N += 1
    return(pow(somme_ek / N, 0.5))
```

Pratiquement, il y avait la ligne où le tableau V était copié à supprimer et l'appel de `nouveau_potentiel()` à changer. Par contre, il faut ajouter une ligne pour mémoriser l'ancienne valeur de `V[i, j]` pour le calcul de l'erreur. On se retrouve malgré tout à copier chaque terme du tableau V...

Méthode de Gauss-Seidel adaptative

Q 12.

```
def nouveau_potentiel_SOR(V, rhos, frontiere, i, j, omega):
    '''renvoie le nouveau potentiel au point (i,j) in [1, N - 1]^2'''
    return( (1 - omega) * V[i, j] + omega * \
    (V[i + 1, j] + V[i - 1, j] + V[i, j + 1] + V[i, j - 1] + rhos[i, j]) / 4 )
```

Q 13.

```
def iter_SOR(V, rhos, frontiere):
    somme_ek, N = 0, 0  # initialisations du calcul de l'erreur
    taille = len(V)     # taille du tableau V
    omega_opt = 2 / (1 + np.pi / (taille - 1))
    for i in range(taille):
        for j in range(taille):
            # parcourt de tout le tableau V
            if not frontiere[i, j]:
                # le point (i, j) n'est pas sur la frontière
                Vk = V[i, j]    # mémorisation de V_k[i, j]
                V[i, j] = nouveau_potentiel_SOR(\
                V, rhos, frontiere, i, j, omega_opt)
                # le calcul d'erreur ne se fait que sur les points
                # hors frontière
                somme_ek += (V[i, j ] - Vk) ** 2
                N += 1
    return(pow(somme_ek / N, 0.5))
```

Q 14.

L'introduction de ω_{opt} assure un nombre d'itérations en $\mathcal{O}(N)$ pour la boucle `while` de la fonction `Poisson()`.

La complexité de cette fonction est donc en $\mathcal{O}(N \times (N+1)^2) = \mathcal{O}(N^3)$.

Une telle complexité temporelle augmente approximativement d'un facteur $2^3 = 8$ le temps d'exécution de l'algorithme lorsqu'on double la taille des données traitées, $(2 \times N)^3 = 2^3 \times N^3$.

D'après la figure 2, nous pouvons tester cette croissance exponentielle sur quelques valeurs : de $N = 50$ à $N = 100$, la durée passe de $T \approx 0,8$ s à $T \approx 6$ s, soit un accroissement de $6/0,8 = 7.5$. De même pour $N = 60$ à $N = 120$, ce facteur est approximativement de $10/1.25 = 8$, et pour $N = 70$ à $N = 140$, ce facteur est de $16/3 = 8$.

La courbe de la figure 2 est donc bien en accord avec la complexité temporelle évaluée à $\mathcal{O}(N^3)$.

Dans ce cas, la durée d'exécution pour $N = 1000 = 10 \times 100$ serait de $10^3 \times T(N = 100) = 1000 \times 6 = 6\,000$ s = 1 h 40 mn. Cette complexité ne permettra pas de simuler l'équation de Poisson sur des domaines très étendus.

1.6 Détermination du champ électrique

Q 15.

Dans la première question, nous avons rappelé que $\overrightarrow{E} = -\overrightarrow{\text{grad}}\, V$, donc en 2D et dans un système de coordonnées cartésiennes, puis selon la configuration de la question 3 :

$$\overrightarrow{E} = \left(-\frac{\partial V(x,y)}{\partial x}, -\frac{\partial V(x,y)}{\partial y}\right) = \left(-\frac{\partial V(X,Y)}{L \cdot \partial X}, -\frac{\partial V(X,Y)}{L \cdot \partial Y}\right)$$

Hors frontière : On peut approcher ces dérivées par une dérivée symétrique qui est meilleure que la simple dérivée à gauche, et comme on est hors frontière, il n'y aura pas de conflits d'indices.

$$\begin{cases} Ex(i,j) &= -\frac{V(i+1,j)-V(i-1,j)}{2 \cdot h \cdot L} \\ Ey(i,j) &= -\frac{V(i,j+1)-V(i,j-1)}{2 \cdot h \cdot L} \end{cases}$$

Aux frontières : Dans ce cas, il faut approcher E_x et E_y par des dérivées simples, à gauche ou à droite, selon le point considéré.

$$\begin{cases} Ex(0,j) &= -\frac{V(1,j)-V(0,j)}{h \cdot L} \quad \text{et} \quad Ey(i,0) &= -\frac{V(i,1)-V(i,0)}{h \cdot L} \\ Ex(N,j) &= -\frac{V(N,j)-V(N-1,j)}{h \cdot L} \quad \text{et} \quad Ey(i,N) &= -\frac{V(i,N)-V(i,N-1)}{h \cdot L} \end{cases}$$

2 Deux études de cas

Seconde partie non corrigée.