Épreuve d'informatique et modélisation des systèmes physiques

Durée 2h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

AVERTISSEMENT

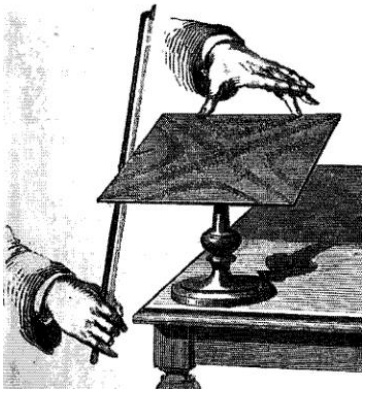
La présentation, la lisibilité, l'orthographe, la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies. En particulier, les résultats non justifiés ne seront pas pris en compte. Les candidats sont invités à encadrer les résultats de leurs calculs.

En dernière partie du sujet, vous trouverez un aide-mémoire *Python* et les consignes générales à respecter pour les codes exigés.

Tout code fourni en réponse à une question doit être commenté de façon claire. Les commentaires sont à placer dans le code, ou éventuellement dans la brève introduction qui le précède.

FIGURES DE CHLADNI

Introduction historique



En 1787, le physicien et musicien, Ernst Florence Friedrich Chladni de Wittenberg, alors qu'il procédait à de nombreuses expériences, fit une intéressante découverte. Il constat que lorsqu'il excitait une plaque de métal avec l'archet de son violon, il la faisait vibrer et il pouvait produire des sons de distinctes tonalités selon l'endroit où il touchait la plaque. La plaque métallique était fixée en son centre, et il eut l'idée d'y disperser de la poussière. Lors des phases vibratoires, pour chaque tonalité, la poussière s'arrangeait et se distribuait selon les lignes nodales des vibrations, de magnifiques figures géométriques apparaissaient alors. Expérimentateur chevronné, il prit soin de cataloguer et recenser ces différentes figures, après s'être assuré de leur caractère reproductif.

Ces figures, désormais dénommées figures de Chladni, attirèrent l'attention de nombreuses personnalités de l'époque. Scientifiques et puissants se pessaient aux nombreuses démonstrations du musicien pour admirer le phénomène. Sa compréhension échappait toutefois à l'entendement des contemporains de Chladni.

La première à proposer une explication de ces obsqervations fut la mathématicienne Sophie Germain dans une correspondance privée à partir de 1811. Elle publia quelques années plus tard un résumé des ces études qui constitua le premier modèle mathématique pour la déformpation d'une plaque sous une contrainte de force extérieure. Son travail, « recherche sur la théorie des surfaces élastiques » sera couronné par l'Académie des sciences en 1816. Lagrange et Poisson corrigèrent et améliorèrent ce premier modèle, mais il fallut attendre l'intervention de Kirschoff pour aboutir à une modélisation permettant d'approcher de façon satisfaisante le comportement d'une plaque. Il en profita pour résoudre le problème des figures de Chladni sur une plaque circulaire, où la complexité des solutions est moindre.

Nous nous proposons dans ce problème de reprendre en partie la démarche d'analyse du problème et de mettre en œuvre la conception du code permettant une résolution numérique.

La figure 1 ci-contre affiche quelques figures de Chladni qu'il est possible d'obtenir en excitant la plaque métallique de différentes manières.

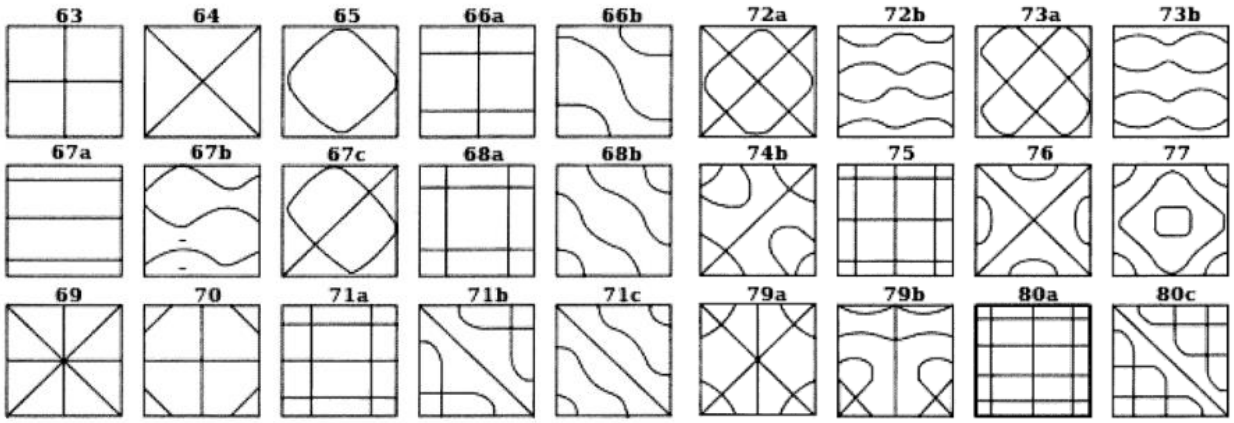


FIGURE 1 – Figures de Chladni

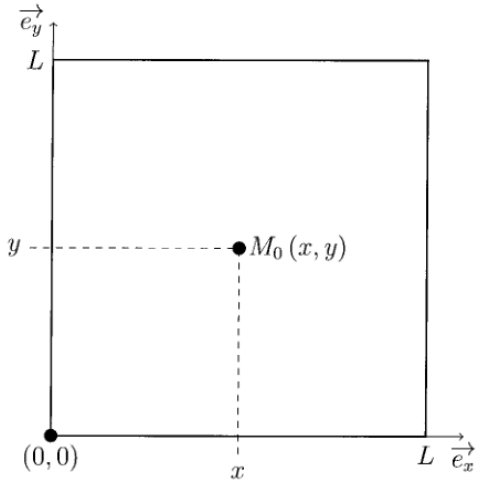
Introduction au modèle physique

Nous considérons une plaque métallique carrée de côté L, elle est d'épaisseur e, de module de Young E et sa masse volumique vaut ρ.

Pour suivre les éventuelles déformations de la plaque, nous utilisons un repère cartésien $\mathcal{R}(O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$. Au repos, la plaque se situe dans un plan horizontal de cote $z = 0$ et chaque point de la plaque au repos est repéré par $M_0(x, y)$.

Sous l'action d'une sollicitation extérieure, ou d'une contrainte, la plaque se déforme, chacun des points initialement en $M_0(x, y)$ est alors décrit par le point $M(x', y', z, t)$. M est une fonction du champ initial de position M_0 et du temps t.

Nous supposons que si les sollicitations sont modérées, nous aurons :



Repérage d'un point sur la plaque

$$\forall t \quad x' = x \quad \text{et} \quad y' = y$$

Dans le cadre de cette approximation, le point M est à tout instant à la verticale de M_0 et sa seule variable d'espace indéterminée est son altitude z. Nous pouvons, dans ce cas, considérer que la position de M est décrite par la fonction :

$$z = f(x, y, t)$$

z est l'amplitude de la vibration, et $f(x, y, t)$ est la fonction d'onde de vibration.

Une analyse du problème utilisant une approche de déformation élastique, permet de considérer que la fonction d'onde f vérifie :

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{1}{c^2} \cdot \frac{\partial^2 f}{\partial t^2} \qquad (1)$$

Où $c > 0$ est la célérité de l'onde.

1. Quel est le nom de cette équation d'onde ? Quelles sont ses principales propriétés ?
2. Pensez-vous que cette équation soit le reflet d'un modèle physique autorisant de la dissipation d'énergie ? Justifiez votre réponse.
3. Par analyse dimensionnelle, proposez une expression de c cohérente avec les paramètres physiques de la plaque.
4. Application numérique : masse volumique $\rho = 2700\ \text{kg.m}^{-3}$, épaisseur $e = 1\ \text{mm}$, module de Young $E = 69\ \text{GPa}$ et longueur du côté de la plaque $L = 25{,}0\ \text{cm}$. Calculer un ordre de grandeur de c.

Soit $f_L(x, y)$ une solution de l'équation (1) pour une plaque de côté L, $f_L(x, y)$. Nous considérons, dans cette seule question, une plaque faite du même matériau, possédant la même épaisseur, et soumise aux mêmes types de conditions aux limites, mais de côté $L' = \gamma L$.

5. Déterminer, parmi (α, β), la valeur du coefficient α qui permet à la fonction $g(x, y, t) = f_L(\alpha x, \alpha y, \beta t)$ d'être solution de l'équation (1) pour cette nouvelle plaque.

Modes de vibration

En règle générale, les vibrations d'une plaque sont complexes et se décomposent en superposition de modes. Chacun de ces modes correspond à une vibration monochromatique (d'une seule fréquence).

6. Pour résoudre l'équation (1) en $z(x, y, t)$, nous allons commencer par rechercher des solutions à variables séparables, c'est-à-dire des solutions de forme :

$$z = f(x, y, t) = u(x, y) \cdot h(t)$$

   (a) En déduire les équations différentielles que doivent vérifier les fonctions $u(x, y)$ et $h(t)$.

(b) Pourquoi les solutions acceptables pour $h(t)$ sont-elles seulement sinusoïdales ?

(c) Les solutions $h(t)$ sont mises sous la forme $h(t) = h_0 \exp(i\omega t)$. Exprimez alors l'équation vérifiée par $u(x, y)$.

Simulation numérique temporelle

L'équation différentielle que doit vérifier la fonction temporelle peut s'écrire (après adimensionnement) sous la forme :

$$\frac{d^2h}{dt^2}(t) = -h(t) \qquad (2)$$

C'est une équation différentielle linéaire d'ordre deux classique appelée équation harmonique. Nous l'utiliserons pour tester la pertinence de certaines méthodes numériques.

7. **Méthode d'Euler**

(a) Rappeler, en quelques lignes concises, le principe de la méthode d'Euler pour résoudre une équation différentielle.

(b) Mettre l'équation précédente sous la forme d'un système différentiel linéaire du premier ordre et le présenter sous forme matricielle en caractérisant la matrice A et en exploitant le vecteur $X(t)$ tel que :

$$\frac{dX}{dt}(t) = A\,X(t) \qquad \text{avec} \quad X(t) = \begin{pmatrix} h(t) \\ h'(t) \end{pmatrix}$$

(c) À partir de cette forme matricielle, introduire le pas de discrétisation temporel τ et mettre en œuvre la méthode d'Euler pour obtenir une équation aux différences correspondant au système initial. Vous pourrez noter de façon réduite $X(t_n) = X[n]$.

8. **Méthode d'Euler à droite**

Pour obtenir $X(t + \tau)$, nous pouvons procéder de la façon suivante :

$$X(t_n + \tau) = X(t_n) + \int_{t_n}^{t_n+\tau} \frac{dX(t)}{dt}\, dt \simeq X(t_n) + \tau \frac{dX(t_n + \tau)}{dt}$$

C'est la méthode d'Euler dite *à droite*, ou *implicite*, la version classique étant dite *à gauche*, ou *explicite*.

(a) Exprimer l'équation liant $X[n + 1]$ à $X[n]$.

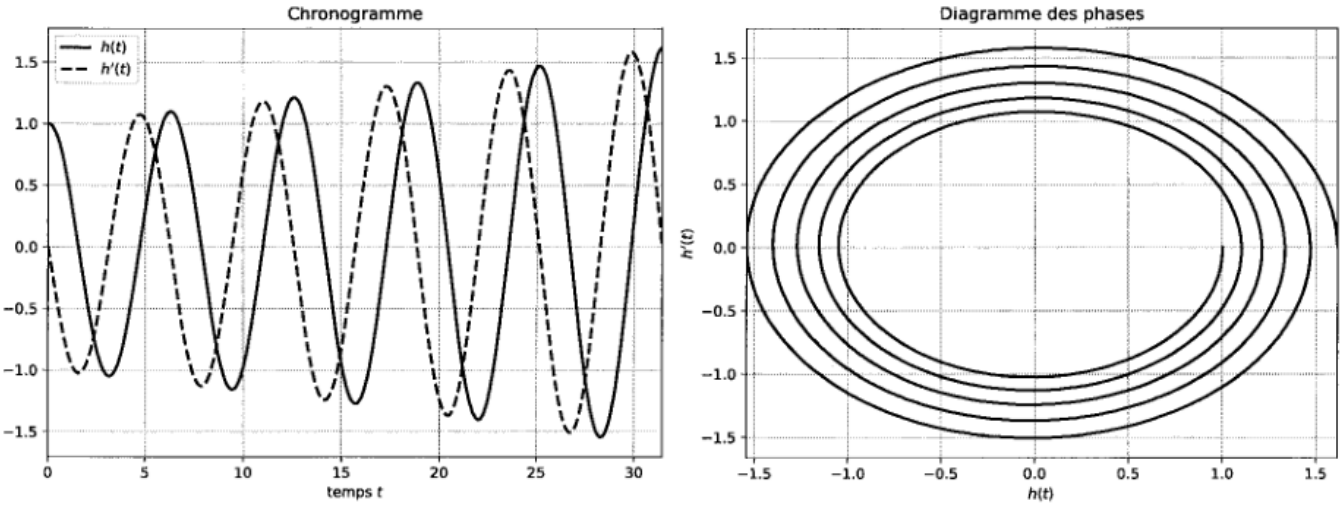
(b) Le code ci-après correspond à l'implémentation de la méthode d'Euler à gauche. Le modifier pour implémenter la méthode d'Euler à droite.

```
# Constantes paramétriques
npoints=2**10 # index maximal des tableaux
tporte=5*2*np.pi # durée d'affichage
X0=[1.,0] # conditions initiales

# Initialisation des tableaux
t=np.linspace(0,tporte,npoints+1) # temps
tau=t[1]-t[0]
X=np.zeros((2,npoints+1),dtype=float) # h et h'

# Algorithme d'Euler
X[:,0]=X0 #initialisation
M=np.array([[1,tau],[-tau,1]]) # matrice de l'equation
for i in range(npoints): # Calcul iteratif
        X[:,i+1]=M.dot(X[:,i])
```

(c) La méthode d'Euler à droite est-elle plus pertinente que celle à gauche ?

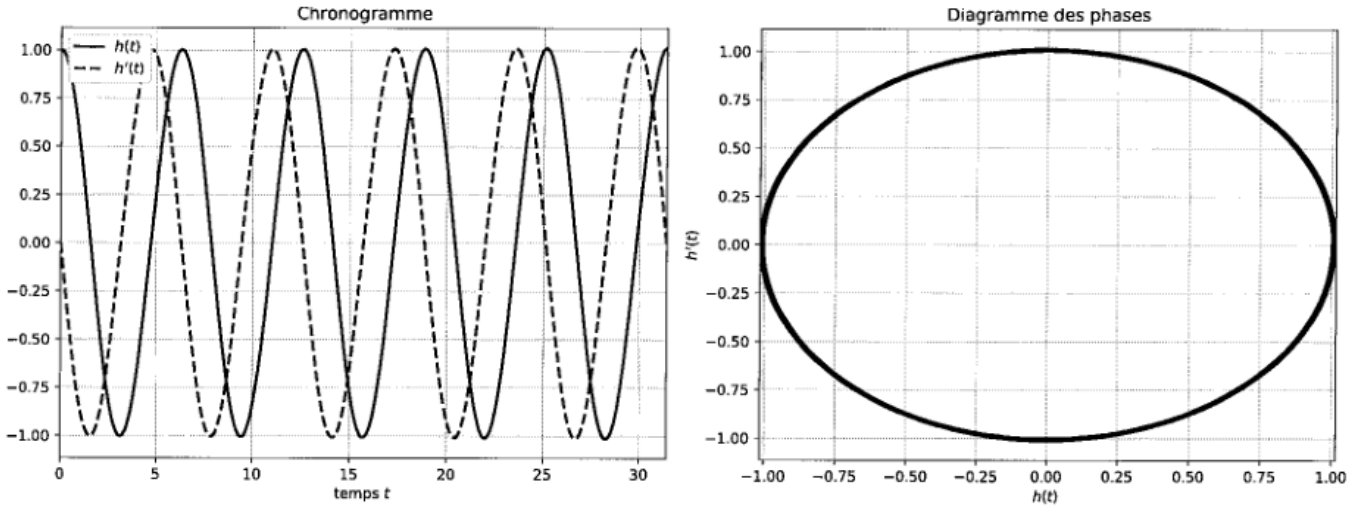
9. **Représentations graphiques**

(a) À partir du code donné à la question précédente, mettre en place les instructions pour obtenir l'affichage correspondant aux figures présentées ci-après. Vous veillerez à la présence des légendes des courbes, des axes et des titres.



(b) Commentez les courbes et corrélez leur allure aux propriétés de la méthode d'Euler et à l'équation (2).

(c) Sans modifier le principe de l'algorithme utilisé, nous pouvons néanmoins obtenir les courbes ci-après. Quelle est, selon vous, la modification apportée au code précédent ? Quel en est l'inconvénient ?



10. **Méthode de Runge-Kutta d'ordre deux** (aucune connaissance préalable n'est nécessaire).

$X(t+\tau)$ s'obtient exactement à partir de $X(t)$ avec la classique relation intégrale :

$$X(t_n+\tau) \simeq X(t_n) + \int_{t_n}^{t_n+\tau} \frac{dX(t)}{dt}\, dt$$

Nous pouvons obtenir une version approchée de cette relation en utilisant l'approximation :

$$X(t_n+\tau) \simeq X(t_n) + \frac{\tau}{2}\left(\frac{dX(t_n+\tau)}{dt} + \frac{dX(t_n)}{dt}\right)$$

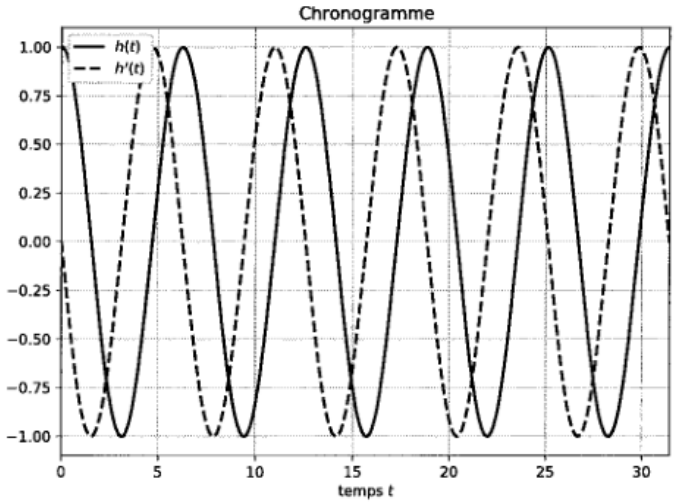
(a) Quel est le nom de la méthode employée pour approximer l'intégrale ? Qualitativement, pourquoi est-ce préférable aux méthodes précédentes ?

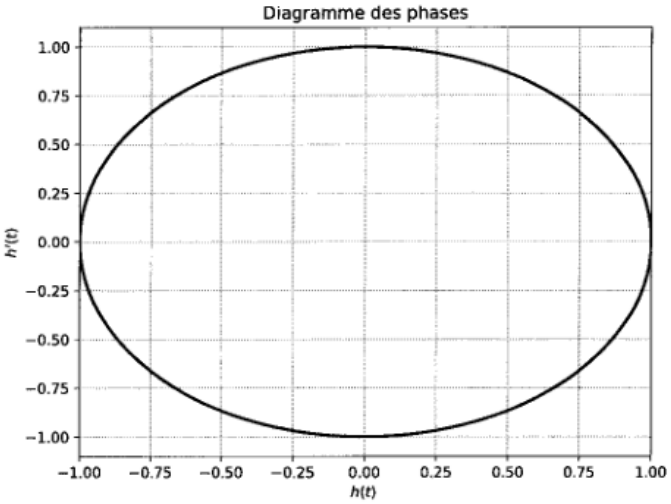
(b) L'analyse est faite sur la durée $T = N\tau$.
La valeur de la dérivée de X au temps $t_n + \tau$ demande la mise en œuvre d'une méthode de calcul implicite ; nous l'éviterons en estimant la valeur de X au temps $t_n + \tau$ par la méthode d'Euler : c'est le principe de la méthode de Runge-Kutta. L'algorithme de calcul est alors le suivant :

$$K_1 = \frac{dX(t_n)}{dt} = A \cdot X[n] \qquad K_2 = A \cdot (X[n] + \tau K_1) \qquad X[n+1] = X[n] + \frac{\tau}{2}(K_1 + K_2)$$

Reprenez les codes écrit pour la méthode d'Euler et adaptez les pour établir la méthode de Runge-Kutta.

(c) La simulation numérique dans des conditions identiques à la méthode d'Euler nous fournit les graphes ci-après. Ils semblent correspondre aux attendus usuels. Peut-on conclure à l'adéquation de la méthode de Runge-Kutta d'ordre 2 ? Argumentez votre réponse.





11. **Méthode « stable »**

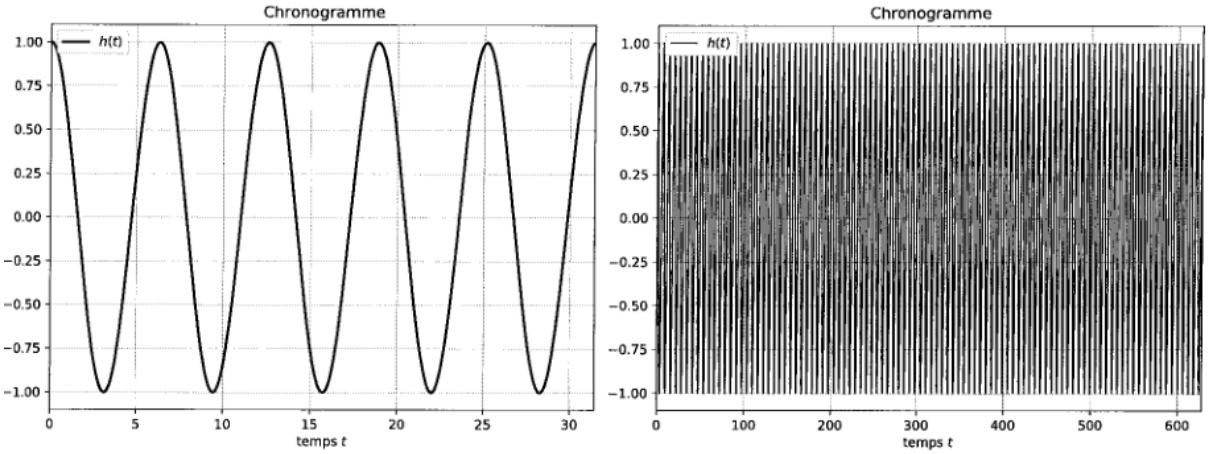
Nous implémentons désormais le code ci-après :

```
# Constantes paramétriques
npoints=2**10 # index maximal des tableaux
tporte=100*2*np.pi # durée d'affichage
X0=[1.,0] # conditions initiales

# Initialisation des tableaux
t=np.linspace(0,tporte,npoints+1) # temps
tau=t[1]-t[0]
X=np.zeros((2,npoints+1),dtype=float) # h et h'

# Algorithme XXX
X[:,0]=X0 #initialisation
A=np.array([[0,1],[-1,-tau]]) # matrice de couplage
M=np.identity(2)+tau*A # matrice d'évolution
for i in range(npoints): # Calcul itératif
        X[:,i+1]=M.dot(X[:,i])
```

Il permet d'obtenir les courbes de la figure ci-dessous.

(a) À partir du code déployé, donnez les équations itératives permettant la mise en œuvre de la simulation.

(b) Quel est l'ordre de cette méthode ? Pourquoi est-elle stable ?



Discrétisation spatiale du problème

L'équation correspondant à un mode propre de pulsation ω peut s'écrire, pour $u(x, y)$ sous la forme :

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\lambda\, u$$

Où $\lambda > 0$ est la valeur propre associée au mode étudié.

Nous nous proposons de résoudre numériquement cette équation en utilisant une discrétisation et la méthode des différences finies. Pour cela, nous introduisons un maillage de la plaque étudiée, de pas régulier, uniforme $d = L/N$, où N représente le nombre de segments du maillage.

12. **Équation adimensionnée**

Nous allons effectuer le changement de variable ci-après pour faciliter le processus de discrétisation. Nous posons, pour une plaque de côté L :

$$X = \frac{x}{L} N \qquad Y = \frac{y}{L}\, N \qquad U(X, Y) = u(x, y)$$

L'équation vérifiée par $U(X, Y)$ se mettra sous la forme :

$$\Delta U = \frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2} = -\lambda'\ U \qquad (4)$$

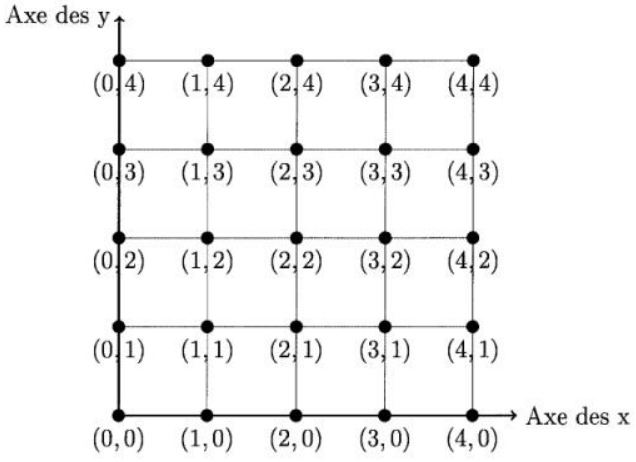
Déterminer l'expression de λ' en fonction de la pulsation ω. Que devient le pas de discrétisation initiale dans ce nouveau jeu de variables ?

Nous allons chercher à résoudre numériquement cette équation différentielle linéaire en la discrétisant avec le pas régulier unité. Nous découpons les axes X et Y en sous-intervalles délimités par des ensembles de points définis de la manière suivante :

$$\forall i \in [0, N] \qquad X_i = i$$

$$\forall j \in [0, N] \qquad Y_j = j$$

La plaque sera représentée par un ensemble de points, distribués comme sur le schéma ci-contre (si $N > 4$).

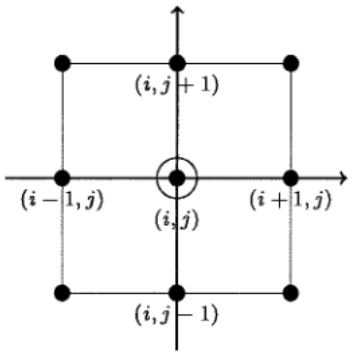


Nous noterons, selon la convention suivante, la valeur approchée de $U(X, Y)$ correspondant au point de coordonnées (X_i, Y_i) :

$$U(X_i, Y_i) \simeq U_{i,j}$$

13. **Discrétisation autour d'un point central**

Nous étudions la situation d'un point du maillage loin des bords. Nous supposerons que ce dernier, de coordonnées (X_i, Y_i), possède des huit plus proches voisins comme indiqué sur la figure ci-contre.



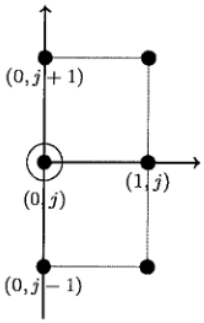
(a) Exprimer la relation liant $U(X_{i-1}, Y_j)$ à $U(X_i, Y_j)$ par un développement limité à l'ordre 2.

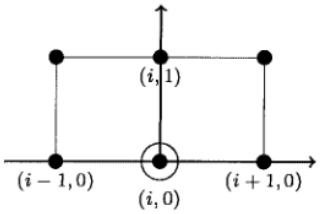
(b) Faire de même avec tous les points cardinaux du maillage, et démontrer la formule approchée suivante :

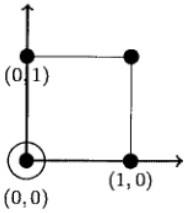
$$\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \simeq U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}$$

14. **Discrétisation autour des bords**

Les bords de la plaque doivent être considérés différemment selon les conditions aux limites que nous désirons imposer. Il nous faut générer les configurations types suivantes que nous désignerons par les termes de *côté vertical*, *côté horizontale*, et *coins*.







Les conditions aux limites classiques sont de deux types : soit les bords sont considérés comme fixes, soit ils sont considérés comme libres. Dans notes cas, nous allons considérer que les points de la périphérie doivent être traités comme des points centraux, en associant aux points manquants de la distribution des valeurs nulles.

(a) En déduire la formule approchée de (4) pour le côté vertical (X_0, Y_j)

(b) En déduire la formula approchée de (4) pour le côté horizontal (X_i, Y_0)

(c) En déduire la formule approchée de (4) pour le coin (X_0, Y_0)

(d) Quelles sont les conditions aux limites dans ce cas ? Justifier votre réponse.

Dans la suite du sujet, on guide le candidat dans la description matricielle de l'équation (4). On montre que la version discrétisée de (4) est une équation matricielle de la forme $MF = \lambda' F$, où M est une matrice dont on peut déterminer les coefficients à partir des questions 13 et 14, et F un vecteur représentant l'intégralité des N^2 positions des points de la plaque. On peut alors utiliser la méthode de la puissance itérée pour identifier les vecteurs propres de M, qui sont les solutions spatiales recherchées.

ANNEXE : RAPPEL PYTHON

En accord avec les exigences officielles, les fonctions et procédures spécifiques de la bibliothèque Numpy, dont nous avons l'usage, sont rappelées dans la présente annexe.

Recommandations :
Les fonctions attendues dans le sujet n'utiliseront pas la récursivité.
Elles ne doivent disposer dans leur code d'implémentation que d'un seul *return*.
Les éventuelles procédures n'en auront, bien sûr, aucun.

Librairies utilisées

Nous partons du principe que tous les programmes demandés sont précédés des appels suivants :

```
# -*-coding:Utf-8 -*

#Librairies utilisées et alias
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rd
```

Les codes produits s'inspireront de ces quelques lignes pour l'exploitation des alias facilitant l'usage des fonctions et des procédures des librairies appelées.

Usage courant de Numpy

La librairie *Numpy* privilégie l'usage du type tableau. Ces derniers seront ici essentiellement des tableaux à une dimension ou deux dimensions.

À l'occasion, les premiers seront parfois désignés comme des vecteurs, et les seconds comme des matrices. Nous n'utiliserons toutefois pas le type spécifique python *Matrix*.

```
# définir un tableau nul de taille nxp
A=np.zeros(shape=(n,p))

# définir un tableau identité de taille nxn
B=np.eye(n)

# définir un tableau rempli de 1 uniformément nxp
C=np.ones(shape=(n,p))

# obtenir les dimensions d'un tableau A : nxp
n,p=A.shape # usage de l'attribut shape

# obtenir le nombre d'élements d'un tableau A
nb_elements=A.size # usage de l'attribut size
```

Les tableaux se prêtent à diverses opérations arithmétiques. Par défaut, chaque opération est effectuée sur l'ensemble des cellules contenues dans le tableau.

Attention : le produit de base est un produit terme à terme de chaque élément du tableau. Il ne correspond pas au produit matriciel dont certains ont coutume.

```
## Opérations élémentaires sur les tableaux

# Addition d'un scalaire j à chaque terme du tableau
j=1
D=A+j

# multiplication par un scalaire k
k=2
D=k*D

# addition, soustration, produit terme à terme
AddBC=B+C
SubBC=B-C
ProdBC=B*C
```

Pour effectuer un produit matriciel, il faut utiliser la méthode *dot*.

Nous rappelons à cette occasion que l'indexation d'un tableau 2D à $N \times N$ éléments couvre la distribution $(i,j) \in [0,1,\ldots,N-1]$.

```
## Opérations "matricielles" courantes

# produit matriciel B x C
PmatBC=B.dot(C)

# Tranposée du tableau C
C.T

# Lecture de la cellule [i,j] du tableau C à 2 dimensions
i,j=1,2
r=C[i,j]
```

Manipulation et copies de tableaux

Pour assurer la, copie complète d'uni tableau, il faut utiliser la méthode copy. Le simple égal est, à l'instar des listes de base, l'occasion de définir des alias

```
# Alias
F = C # F et C référencent le même tableau

## Copie d'un tableau
F=C.copy() # F référence une copie du tableau C
```

La librairie permet le re-dimensionnement d'un tableau avec la fonction *reshape*. Il faut bien sûr que la forme visée ait le même nombre d'éléments que la forme initiale.

```
## Reformage

# un tableau peut être réorganisé par la méthode reshape
E=np.ones(9) # vecteur 1D de neuf 1
print(E)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.]

F=E.reshape(3,3) # reforme E en tableau 2D 3x3
print(F)
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]

G=F.flatten() # renvoie le tableau sous forme 1D
print(G)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

Numpy permet d'assembler les vecteurs et les matrices, de les concaténer en utilisant la fonction correspondante *concatenate*.

Par défaut l'assemblage se fait selon la 1ère dimension (les lignes, donc assemblage vertical). L'option axis=1 assemble "horizontalement".

```
a=np.arange(4).reshape(2,2)
        array([[0, 1],
               [2, 3]])

b=4+np.arange(4).reshape(2,2)
        array([[4, 5],
               [6, 7]])

np.concatenate((a,b))
        array([[0, 1],
               [2, 3],
               [4, 5],
               [6, 7]])
```

```
np.concatenate((a,b),axis=0)
        array([[0, 1],
               [2, 3],
               [4, 5],
               [6, 7]])

np.concatenate((a,b),axis=1)
        array([[0, 1, 4, 5],
               [2, 3, 6, 7]])
```

Les plus coutumiers d'entre vous pourront utiliser les techniques de *slicing* et de masquage qui ne seront pas rappelées ici. Si la librairie favorise le traitement en masse, nous acceptons néanmoins tous les codes utilisant de simples boucles.