INITIATION
ET
TRAVAUX PRATIQUES
MATLAB

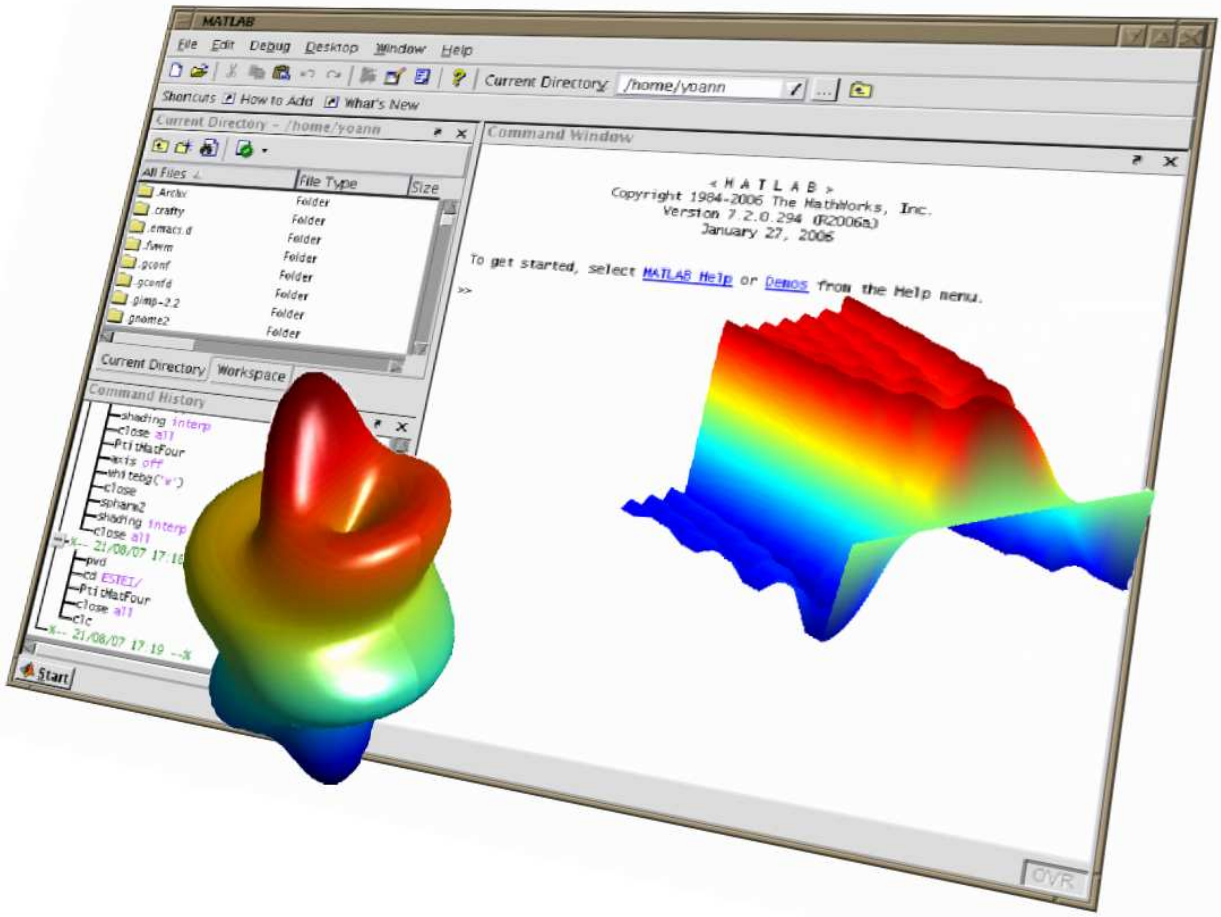


Table des matières

Initiation Matlab

Présentation et introduction à Matlab

Matlab, acronyme de "**MAT**rix **LAB**oratory", est un logiciel conçu pour fournir un environnement de calcul numérique de haut niveau. Il est particulièrement performant pour le calcul matriciel, et dispose de grandes capacités graphiques pour, par exemple, la visualisation d'objets mathématiques complexes.

Son fonctionnement repose sur un langage de programmation interprété qui permet un développement très rapide. En contre-partie, pour des applications nécessitant des performances plus élevées en temps de calcul, un langage compilé, comme le C++ ou le fortran, est plus adapté.

Sous sa forme "graphique", Matlab dispose d'une interface comprenant l'environnement Matlab à proprement parler, d'où les commandes Matlab peuvent être directement exécutées, ainsi que d'un environnement graphique, pouvant comprendre plusieurs fenêtres : liste des variables en cours d'utilisation, historique des commandes exécutées, ..., et divers menus plus ou moins habituels, "File", ("New", "Open",...), "Configuration", "Help",...

Toutes les commandes des différents menus ont leur alternative en "ligne de commande" dans l'environnement propre à Matlab, la réciproque étant bien évidemment fausse.

Dans ce paragraphe d'introduction à Matlab, on ne s'interessera qu'à l'environnement propre à Matlab, les commandes et syntaxes de base d'instructions Matlab.

Remarque : Matlab peut-être lancé sans son environnement graphique complet à l'aide de la commande *matlab -nodesktop*.

Commandes Matlab

Les commandes peuvent se taper directement suite au prompt de Matlab. L'opération est alors immédiatement effectuée et le résultat retourné. Si la commande se termine par un point virgule, la commande est effectuée, mais le résultat obtenu n'est pas retourné.

L'aide

`help` *func*	affiche l'aide concernant la fonction *func*. Voir *help help*...
`helpdesk`	version graphique et naviguable (html) de l'aide
`lookfor` *mot_cle*	lance une recherche sur toutes les fonctions Matlab, et retourne toutes les fonctions dont l'aide contient le mot clé *mot_cle*
`demo`	Matlab contient de nombreuses démo. de ses capacités. Taper *demo*, et naviguer...

Commandes générales

`cd`	change/affiche le repertoire courant
`which`	affiche le chemin complet d'une fonction Matlab
`path`	variable Matlab contenant la liste des repertoires connus, dans lesquels Matlab recherche une fonction lors de son appel
`addpath`	permet d'ajouter un chemin dans le path
`who`	liste des variables de l'espace de travail
`whos`	idem avec tailles en mémoire
`clear` *var*	supprime la variable *var* de l'espace de travail
`clear all`	supprime toutes les variables
`close all`	ferme tous les graphiques

Opérations usuelles

`=`	affectation d'une valeur à une variable (ex. $a = 2$)
`+,-,*,/`	opérations usuelles sur les variables ou valeurs numériques
`1i`	nombre complexe : $(1i)^2 = -1$
`abs, angle, real, imag`	opérations usuelles sur les nombres complexes

Définition et opérations sur les vecteurs et matrices

Définition de matrices et vecteurs :

`[ , , ; , , ]`	définition manuelle d'une matrice (ex. $A = [1, 2, 3; 4, 5, 6]$)
`deb:pas:fin`	définition d'un vecteur régulier balayant l'intervalle $[deb, fin]$ avec le pas pas (ex. $A = 1 : 1 : 6$) ; par défaut le pas est égal à 1 s'il est omis (ex. $A = 1 : 6$)
`linspace(deb,fin,N)`	définition d'un vecteur balayant l'intervalle $[deb; fin]$ avec N valeurs régulièrement espacées
`zeros, ones, eye, rand, randn, vand, magic...`	matrices particulières (cf. *help...*)

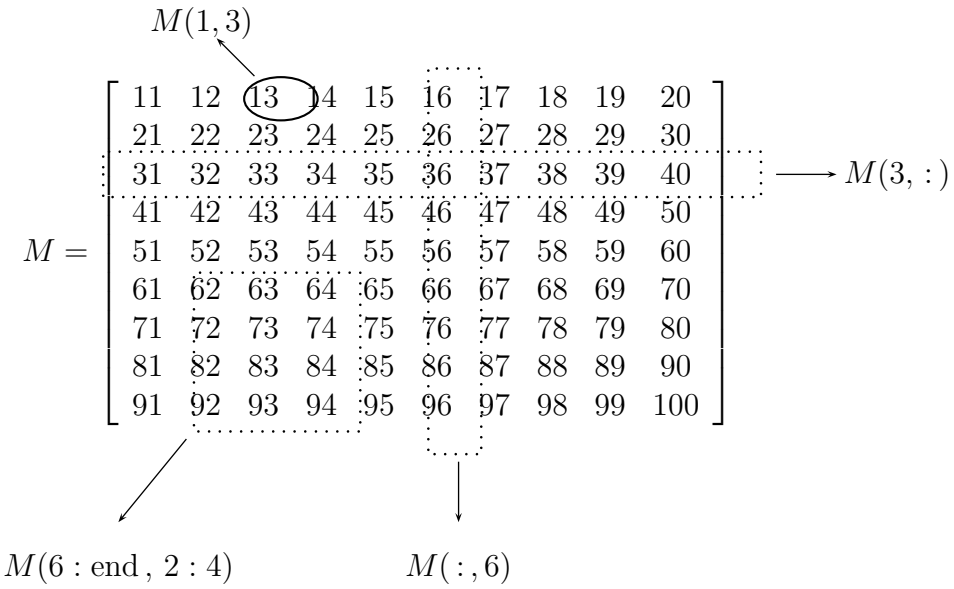
Les opérations usuelles `+,-,*,/`,... agissent indifféremment sur les réels, complexes, ou matrices (à conditions que les dimensions de celles ci le permettent).

D'autres opérations sont également disponibles :

`.* , ./ , .^ ,...`	(les opérateurs usuels précédés d'un point) opérations sur les matrices effectuées **terme à terme** (ex. calculer $A = [1 : 6] .* [7 : 12]$)
`length`	longueur d'un vecteur
`size`	dimension d'une matrice
`<, <=, >,>=, ==`	comparaison des éléments de deux matrices, terme à terme
`sum, mean,...`	somme, moyenne,..., des éléments d'un vecteur
`sin, cos, exp, log,...`	d'une façon générale toutes les fonctions usuelles s'appliquent à des matrices **terme à terme** (ex. $\log([1, 2, 3]) = [\log 1, \log 2, \log 3]$)
`find`	recherche les éléments non nuls dans une matrice
`nnz`	compte le nombre d'éléments non nuls dans une matrice

Extraction des éléments d'une matrice

`M(i,j)`	élément de la matrice M situé sur la ligne i et la colonne j
`V(end)`	dernier élément du vecteur V
`M(5:9,3)`	les éléments de la matrice M situés de la ligne 5 à 9, et sur la colonne 3
`M(:,j)`	toutes les lignes de la matrice M, colonne j
`M(i,:)`	toutes les colonnes de la matrice M, ligne i
`M(1:5,1:3)`	les éléments de la matrice M situés sur les lignes 1 à 5, et sur les colonnes 1 à 3



Fonctions graphiques

`figure`	crée une nouvelle figure ; *figure(n)* (ré)initialise la figure n° *n*
`plot`	tracé d'un ensemble de points (ex. `plot([0:0.1:2*pi],sin([0:0.1:2*pi]))`)
`subplot`	partionne la figure courante en plusieurs sous-graphiques
`imagesc`	affiche sur un graphique le contenu d'une matrice, la valeur des éléments étant représentée par une échelle de couleur
`axis`	permet de sélection manuellement l'échelle utilisée
`title, xlabel, ylabel, legend`	permet d'ajouter un titre général, un titre sur les axes et une légende à une figure
`grid on / off`	affiche une grille sur le graphique courant
`hold on /off`	permet de superposer des graphiques sur une même figure
`plot3, semilogx, semilogy, loglog, mesh, surf`,...	autres fonctions graphiques (voir l'aide...)

Scripts et fonctions Matlab

L'ensemble des commandes ci-dessus peuvent être tapées directement dans l'environnement Matlab, le résultat étant alors immédiatement affiché (suivant la complexité du calcul requis).

Il est aussi possible de regrouper ces commandes dans un fichier "texte" comportant un ensemble de commandes à effectuer. Ce fichier doit être enegistré avec l'extension ".m" (ex. *script1.m*), et peut être exécuté directement sous Matlab, si le répertoire dans lequel il est enregistré est soit le répertoire courant (cf. `cd`, ou `pwd`), soit présent dans la variable *path* de Matlab. Un tel fichier peut-être créé en utilisant l'éditeur de Matlab (commande `edit`), soit en utilisant un quelconque éditeur : vi, emacs, notepad, gedit, ...

Deux types de fichiers de commandes sont à distinguer :

- les **scripts** : ils permettent simplement de regrouper un ensemble de commandes Matlab. A l'appel du script, les commandes sont exécutées séquentiellement.
- les **fonctions** : de même que les scripts, elles permettent de regrouper un ensemble de commandes effectuant une tâche globale. Les fonctions prennent de plus un ensemble d'arguments en entrée et délivrent un ensemble de valeurs de sortie.

  Après l'exécution d'une fonction, l'ensemble des variables autres que celles utilisées en entrée et en sortie de la fonction sont effacées (variables locales).

La structure générale d'une fonction est la suivante :

```
function [y1 y2 y3]=nom_func(x1,x2,x3,x4)
% Commentaires qui seront affichés lors
% de l'appel ''help nom_func''
Corps de la fonction
```

où les x_i sont les variables passées en entrée de la fonction, tandis que les y_i sont des valeurs de sortie de la fonction, i.e. les variables que l'on récupère une fois l'execution de la fonction achevée.

Une telle fonction s'appelle ensuite par la commande

```
[y1,y2,y3]=nom_func(x1,x2,x3,x4);
```

ou, si on ne souhaite pas affecter les variables de sortie :

```
nom_func(x1,x2,x3,x4);
```

Commandes structurées d'une fonction ou script

Les structures usuelles dans tout langage de programmation sont disponible sous Matlab, par exemple :
`for...end` , `if...else...end` , `while...end` , `switch...case...end` , ...

L'aide (par exemple `help for`) permet de retrouver facilement la syntaxe de ces structures.

TP 1
Exercices d'initiation à Matlab

Exercice 1 : Initiation à MATLAB - Vecteurs et courbes

a) Définir la variable $x = \frac{\pi}{4}$, et calculer $y_1 = \sin(x)$ et $y_2 = \cos(x)$, puis $z = tan(x)$ à partir de y_1 et y_2.

b) Définir la variable $x = [\frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}]$, et calculer $y_1 = \sin(x)$ et $y_2 = \cos(x)$.

Calculer alors $\tan(x)$ en utilisant exclusivement les vecteurs y_1 et y_2 précédents.

c) Définir la variable $x = [0 : 0.1 : 2\pi]$. Combien y a-t-il de valeurs dans ce vecteur ? Afficher la courbe du sinus.
Faire varier le pas. Qu'affiche exactement la commande `plot` ?
(`plot, size, length`).

Exercice 2 : Initiation à MATLAB - Manipulation de matrices

a) Définir le vecteur $V = [0\ 1\ 2\ 3\ \cdots\ 49\ 50]$. Quelle est la taille de ce vecteur ?
Définir le vecteur W contenant les cinq premiers éléments de V, et le vecteur X contenant les cinq premiers et les cinq derniers éléments.
Définir ensuite le vecteur $Z = [0\ 2\ 4\ \cdots\ 48\ 50]$ à partir de V.

b) Définir la matrice $M = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 \end{bmatrix}$

Extraire de cette matrice la matrice $N = \begin{bmatrix} 1 & 2 \\ 11 & 12 \\ 21 & 22 \end{bmatrix}$, la matrice $P = \begin{bmatrix} 8 & 9 & 10 \\ 18 & 19 & 20 \\ 28 & 29 & 30 \end{bmatrix}$, puis la matrice $Q = \begin{bmatrix} 3 & 7 \\ 23 & 27 \end{bmatrix}$.

Extraire de la matrice M la matrice R obtenue en prenant dans la matrice M une colonne sur 2.

c) Définir les matrices $M = [2\ 4\ 6\ 8\ \cdots\ 100]$ et $N = [-1\ -3\ -5\ \cdots\ -99]$, puis le vecteur $P = [-1\ \ 2\ -3\ \ 4\ -5\ \ 8\ \cdots\ -99\ \ 100]$.

d) Définir une matrice M aléatoire à trois lignes et sept colonnes. Combien de nombres dans cette matrice sont plus grand que $0,5$? que $0,8$? Ou sont-ils situés ?
(`rand, nnz, find, sum`)

Construire alors la matrice P obtenue à partir de la matrice M en remplaçant tous les nombres de M inférieurs à $0,4$ par 0, et ceux supérieurs à 0.4 par 1.

Construire de même la matrice Q obtenue à partir de la matrice M en remplaçant tous les nombres de M inférieurs à 0.5 par -3 et tous les nombres supérieurs à 0.5 par 14.

e) Créer un vecteur contenant N valeurs binaires (0 ou 1) tel que 10 % de ces valeurs soient des 1.

Exercice 2 : Initiation à MATLAB, fonctions graphiques

Représenter sur une figure à 4 cadrans, les fonctions sinusoïde, exponentielle, logarithme et tangente. (`subplot`)

Exercice 3 : Etude de la fonction $Sig = exp(-A.t + j.2\pi f_0.t)$

a) Créer le vecteur t=[0 : 0.1 : 100] ; quel est le nombre de points ? Quelle est la place utilisée en mémoire ? (`size, length, whos`)

b) Tracer la fonction demandée en choisissant A et f_0 de façon à observer une dizaine de cycles et une attenuation d'environ 90%. (`plot, real, imag, abs`)

c) Représenter sur une figure 4 cadrans, la partie réelle, la partie imaginaire, le module et la phase de cette courbe.
Mettre les titres et les légendes de chaque graphique. (`subplot, title, legend`)

d) Représenter cette fonction complexe sous la forme d'une trajectoire 3d à l'aide de la fonction `plot3`.

e) Utiliser la fonction graphique rotate3d, pour retrouver les courbes 2D de (c) à partir de la représentation 3D.

Exercice 4 : Initiation à la programmation sous MATLAB

a) Quel est votre répertoire courant ? Créer un nouveau répertoire `Initiation_Matlab` et se placer dedans.

b) Reprendre le problème de l'exercice 3. Ecrire un script de commande qui fixe au départ les valeurs de A et f_0, puis calcule la fonction et enfin représente en 3D la courbe voulue.

c) Taper 'clear all' puis who. Il n'y a plus de variables locales. Taper le nom de votre fichier de commande : quelles sont les variables reconnues dans l'espace de travail ?

Exercice 7 : Création d'une fonction

a) Créer, à partir de votre fichier de commande, une fonction qui trace la courbe 3D de l'exercice 3, en fonction des variables A et f_0 passées en paramètres et qui retourne les valeurs prises par la fonction Sig.

b) Ajouter un "flag" passé à la fonction qui permet de choisir ou non la visualisation de la courbe (`nargin`)

Exercice 8 : Recherche d'un élément dans un vecteur

Ecrire une fonction `Trouve` qui prend en argument un vecteur v et un nombre x, et qui retourne 1 si x est un élément du vecteur v, et 0 sinon.

Deux versions de cette fonction peuvent-être implémentées, une à l'aide d'une boucle `for`, et d'un test `if` approprié, l'autre directement avec une comparaison globale `==` (et, par exemple, `find` ou `nnz`).

Exercice 9 : Matrices et systèmes linéaires

a) Ecrire une fonction, n'utilisant aucune boucle (for, while, ...) qui prend comme paramètre un entier n et qui construit la matrice suivante (fonctions `eye`, `diag`) :

$$\begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{n} & 2 & \frac{n-1}{n} & 0 & 0 & 0 & \\ 0 & \frac{2}{n} & 3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & n-1 & \frac{2}{n} & 0 \\ 0 & 0 & 0 & \cdots & \frac{n-1}{n} & n & \frac{1}{n} \\ 0 & 0 & 0 & \cdots & 0 & 1 & n+1 \end{bmatrix}$$

b) Avec Matlab, on peut résoudre tout type de système linéaire en l'écrivant préalablement sous forme matricielle.

(i) La système linéaire, d'inconnues x, y et z suivant

$$\begin{cases} 6x & + & y & - & 5z & = & 10 \\ 2x & + & 2y & + & 3z & = & 11 \\ 4x & - & 9y & + & 7z & = & 12 \end{cases}$$

s'écrit sous forme matricielle $AX = b$, où $X = [x\ y\ z]^T$ est le vecteur inconnu.

Le vecteur X se calcule alors suivant : $AX = b \iff X = A^{-1}b$, ou encore, avec Matlab, $X = A \setminus b$ (voir `help slash`). Résoudre le système linéaire précédent.

(ii) Résoudre numériquement le système :

$$\begin{cases} x & + & 2y & + & 3z & + & 4t & = & 1 \\ 2x & + & 3y & + & 4z & + & t & = & -2 \\ -2x & + & 4y & - & 5z & + & 2t & = & 0 \\ 8x & + & y & - & z & + & 3t & = & 1 \end{cases}$$

(iii) Résoudre numériquement le système suivant (moindres carrés) :

$$\begin{cases} x & + & 3y & = & 5 \\ -2x & + & 6y & = & 7 \\ 3x & - & 4y & = & 6 \\ 6x & - & 13y & = & -3 \end{cases}$$

TP 2

Signaux numériques et transformée de Fourier discrète

1. Echantillonnage d'un signal

On considère le signal x défini par $x(t) = 2\cos(2\pi f_0 t + \varphi_0)$, d'amplitude 2 et de fréquence fondamentale $f_0 = 20$ kHz.

Générer dans un vecteur `sig` $N = 1000$ points du signal x, échantillonné au rythme de :

- un échantillon toutes les 50 microsecondes,
- un échantillon toutes les 10 microsecondes,
- un échantillon toutes les 1 microsecondes,
- un échantillon toutes les 0,1 microsecondes,

Représenter graphiquement (sur quatre figure distinctes ou sur une figure avec quatre cadrants) ces quatre signaux numériques, avec une échelle des temps correcte, et en indiquant sur chaque courbe la fréquence d'échantillonnage et le nombre de points.

2. Transformée de Fourier de signaux sinusoïdaux

On considère les fonctions définies pour $t \in [0; 0.1]$ par

$$\begin{aligned} x_1(t) &= \cos(2\pi\, f_1\, t + p_1) \\ x_2(t) &= \cos(2\pi\, f_2\, t + p_2) \\ x_3(t) &= \cos(2\pi\, f_3\, t + p_3) \\ x_4(t) &= x_1(t) + x_2(t) + x_3(t) \end{aligned}$$

où, $f_1 = 50$ Hz, $f_2 = 75$ Hz, $f_3 = 125$ Hz, $p_1 = \dfrac{\pi}{2}$, $p_2 = \dfrac{\pi}{3}$, et $p_3 = 0$.

On échantillonne ces signaux à la fréquence de $f_e = 20$ kHz, soit un pas $T_e = \dfrac{1}{f_e} = 0.5$ ms.

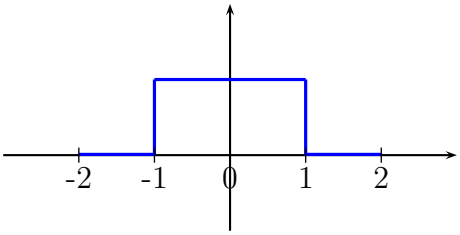
Tracer les fonctions x_i et leur transformée de Fourier $\widehat{x_i}$ (fonction `fft`).

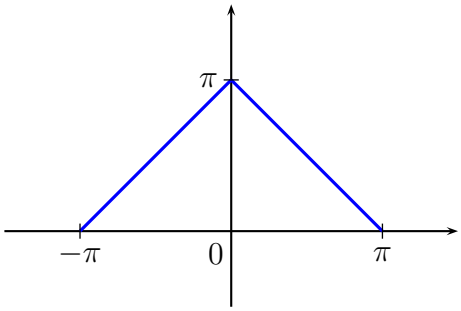
Calculer et tracer ensuite les transformées de Fourier inverses des $\widehat{x_i}$ (fonction `ifft`), et les comparer avec les fonctions de départ.

3. Transformée de Fourier de signaux carrés et triangulaires

On s'intéresse dans ce TP à la transformée de Fourier de fonctions (ou signaux).
On considère les deux fonctions périodiques f_1 et f_2 définies par :

$$f_1(x) = \begin{cases} 0, \text{ si, } -2 < x < -1 \\ 1, \text{ si } -1 < x < 1 \\ 0, \text{ si } 1 < x < 2 \end{cases}, \quad f_2(x) = \begin{cases} x + \pi, \text{ si, } -\pi \leq x \leq 0 \\ -x + \pi, \text{ si, } 0 \leq x \leq \pi \end{cases}$$





1) Définir un vecteur `x` avec N valeurs et variant de -2 à 2 pour la fonction f_1 et de $_\pi$ à π pour f_2.
   Coder et représenter alors graphiquement ces deux fonctions f_1 et f_2.
2) Calculer la transformée de Fourier de ces deux fonctions, puis vérifier, en y appliquant la transformée de Fourier inverse que l'on retrouve bien les fonctions originales (aucune perte ou erreur n'est engendrée par transformation de Fourier).
3) Représenter sur un même graphique, et pour chaque fonction, la fonction originale et la reconstruction à partir de la transformée de Fourier (i.e. la transformée inverse) obtenue en ne considérant que le mode (ou harmonique) fondamental, les deux premiers, les trois premiers...

TP 3

Réseaux et synthèse de diagramme d'antennes

Pour des antennes de communication, on cherche à optimiser le ratio $\frac{P_U}{P_T}$, où P_U est la puissance reçue par l'utilisateur, et P_T est la puissance totale émise par l'antenne.

On cherche, en d'autres termes, à ce que la plus grande partie de l'énergie fournie à l'antenne soit acheminée jusqu'à l'utilisateur distant.

La directivité de ces antennes est donc un critère déterminant.

Le contexte actuel du développement massif des réseaux mobiles (Wi-Fi et GSM principalement) rend par ailleurs inadapté le développement d'antennes à haute directivité "statique".

Comme les utilisateurs sont en mouvement et peuvent donc se trouver a priori dans n'importe quelle direction vis-à-vis de l'antenne, on a recours à des antennes rayonnant de façon homogène dans toutes les directions de l'espace (rayonnement isotrope). L'objectif initial est dans ce cas loin d'être rempli.

Une stratégie pour y remédier consiste à utiliser plusieurs antennes, assemblées en un réseau d'antennes, dans le but de pouvoir focaliser le signal, et donc la puissance émise, sur l'utilisateur. Le système est alors d'autant plus efficace que les interférences entre systèmes différents s'en trouve par la même occasion limitées.

On peut ainsi imaginer des systèmes avec des diagrammes d'émission à plusieurs lobes principaux, un par utilisateur communiquant, et capable de plus de suivre ces utilisateurs dans leur mouvement.

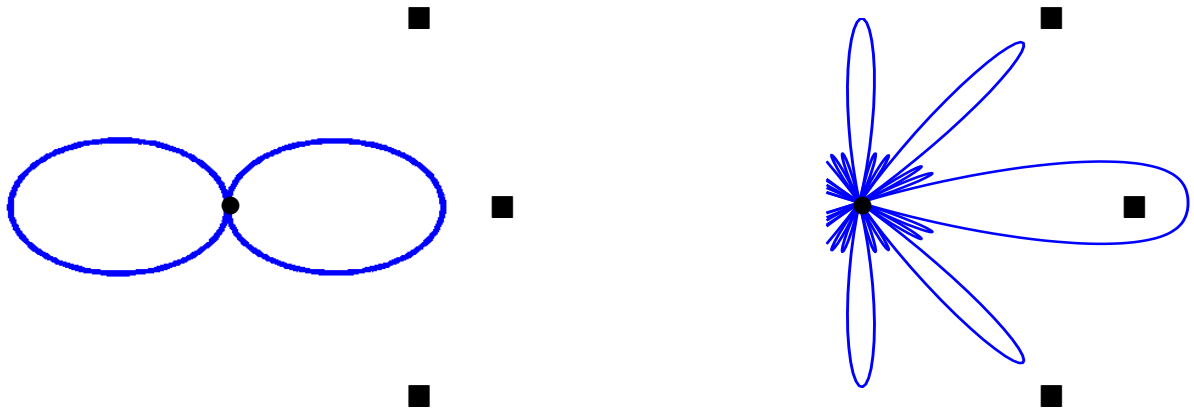


Diagramme de rayonnement d'une antenne dipolaire (gauche) et d'un réseau de 4 antennes (droite).
Le rayonnement du réseau est adapté au nombre et la position des utilisateurs (marqués par ■), qui se répartissent donc plus efficacement la puissance émise.

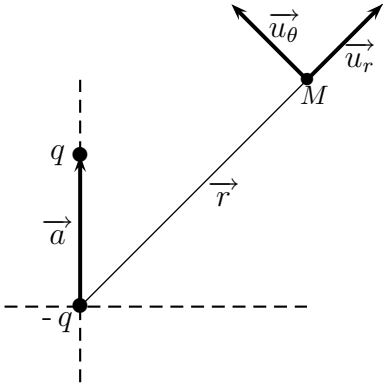
1. Rayonnement d'un dipole

Le potentiel électrostatique créé en un point M à la distance r ($\overrightarrow{OM} = \overrightarrow{r}$) par deux charges $+q$ et $-q$ distante de $\overrightarrow{a}$ (avec $a \ll r$) est donné par

$$V(\overrightarrow{r}) = \frac{1}{4\pi\varepsilon_0}\frac{\overrightarrow{p}\cdot\overrightarrow{u_r}}{r^2}$$

où, $\overrightarrow{p} = q\overrightarrow{a}$ est le moment du dipole, et ainsi, le champ électrique crée par le dipole est donc,

$$\overrightarrow{E}(\overrightarrow{r}) = -\nabla\cdot V(\overrightarrow{r}) = \frac{3\overrightarrow{u_r}(\overrightarrow{p}\cdot\overrightarrow{u_r}) - \overrightarrow{p}}{4\pi\varepsilon_0 r^3} = \frac{2p\cos\theta}{4\pi\varepsilon_0 r^3}\overrightarrow{u_r} + \frac{p\sin\theta}{4\pi\varepsilon_0 r^3}\overrightarrow{u_\theta}$$



Exercice 1 Représenter sur une figure à deux cadrans, les composantes E_r et E_θ du champ électrique du dipole.

(`polar` pour la représentation en coordonnées polaires)

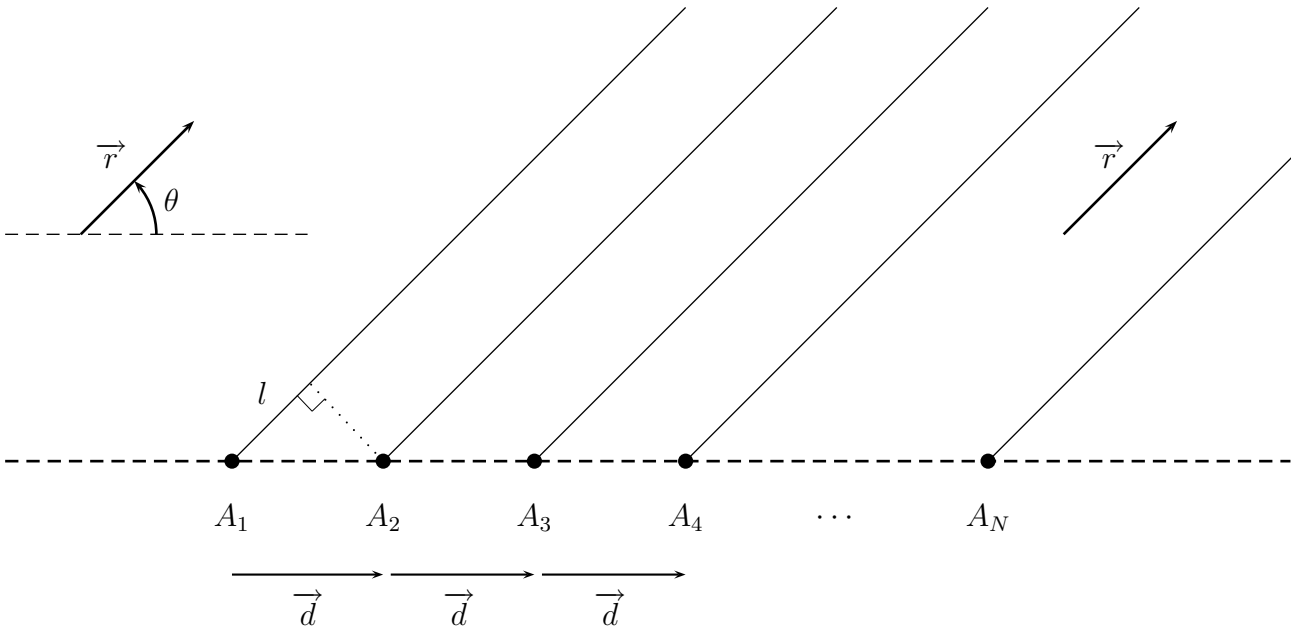
2. Réseau linéaire d'antennes

On dispose de N antennes identiques, alignées et séparées par une distance d.

Ces antennes sont alimentées à la fréquence f, sachant que l'on peut de plus imposer un déphasage φ d'une antenne à la suivante.

On s'intéresse au champ rayonné lointain dans la direction $\overrightarrow{r}$, et à une distance $r \gg d$.

Le champ rayonné par l'antenne située en A_2 a une avance de phase de kl, où k est le nombre d'onde ($k = \dfrac{2\pi}{\lambda}$ où λ est la longueur d'onde),



Les paramètres du réseau sont :

- N : nombre d'antennes
- $\overrightarrow{d}$: vecteur entre chaque antenne ($d = \|\overrightarrow{d}\|$ est la distance entre chaque antenne)
- $\alpha_n = \rho_n\, e^{j\varphi_n}$ est le poids (alimentation) de l'antenne n ; ρ_n est le gain et φ_n le déphasage imposés à l'antenne n.

Si $\overrightarrow{E}_1$ est le champ de l'antenne 1, le champ émis par la $n^{\text{ième}}$ antenne est : $\overrightarrow{E}_n = \alpha_n \overrightarrow{E}_1$.

- θ : angle d'observation

La synthèse d'un diagramme de rayonnement consiste à déterminer les excitations complexes α_p pour que le champ total émis par le réseau réponde à des spécifications précises (réunies dans un gabarit).

Le champ total rayonné par le réseau est la somme vectorielle de chaque champ émis individuellement par les N antennes :

$$\overrightarrow{E} = \sum_{n=1}^{N} \alpha_n \overrightarrow{E}_n = \overrightarrow{E}_1 \sum_{n=1}^{N} \alpha_n e^{j(n-1)kl} \quad \text{, avec } l = d\cos\theta$$

soit, si on suppose dans un premier temps que $\alpha_n = e^{-j(n-1)\varphi}$, c'est-à-dire qu'on impose simplement un déphasage constant φ entre chaque antenne, alors le champ total devient :

$$\overrightarrow{E} = \overrightarrow{E}_1 \sum_{n=1}^{N} e^{-j(n-1)\varphi} e^{j(n-1)kl} = \overrightarrow{E}_1 \sum_{n=0}^{N-1} e^{-jn\varphi} e^{jnkl} = \overrightarrow{E}_1 \sum_{n=0}^{N-1} e^{jn(-\varphi+kl)}$$

qui est la somme des termes d'une suite géométrique, soit aussi :

$$\overrightarrow{E} = N\overrightarrow{E_1}\, e^{j\frac{\phi}{2}} \frac{\sin\left(\dfrac{N\phi}{2}\right)}{N\sin\left(\dfrac{\phi}{2}\right)} \quad \text{avec,} \quad \begin{aligned} \phi &= k\,\overrightarrow{d}\cdot\overrightarrow{r} - \varphi \\ &= kd\cos\theta - \varphi \end{aligned}$$

On a ainsi, $\|\overrightarrow{E}\| = N\|\overrightarrow{E_0}\|\, F$, où F est le facteur (ou diagramme) du réseau :

$$F(\theta, d, \varphi) = \frac{\sin\left(\dfrac{N\phi}{2}\right)}{N\sin\left(\dfrac{\phi}{2}\right)}$$

Exercice 2 On considère un réseau d'antennes fonctionnant à la fréquence $f = 3.10^8$ Hz.

1. L'écart entre chaque antenne est fixé ici à $d = \dfrac{\lambda}{2}$; et le déphasage φ est nul entre elles.

   Tracer différents graphiques représentant le diagramme du réseau $F(\theta, d, \varphi)$ pour différentes valeurs de N.

   Indiquer comment varie la largeur du lobe principal en fonction du nombre d'antennes N du réseau.

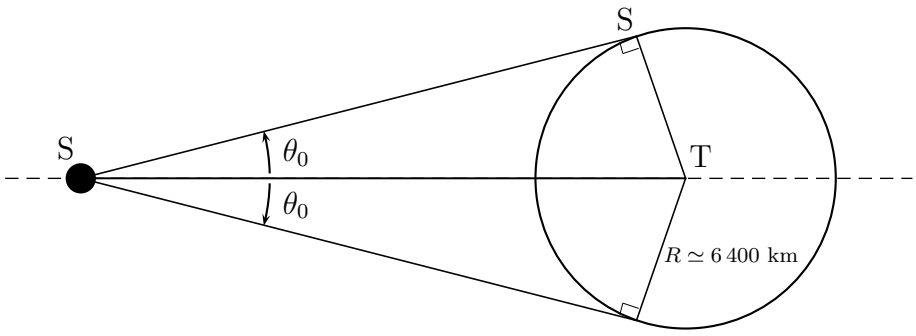
2. On considère maintenant un réseau de $N = 10$ antennes alimentées en phase, espacées de $d = \dfrac{\lambda}{2}$.

   Quel déphasage φ doit-on appliquer entre chaque antenne pour que la direction du lobe principal soit $60\,°$?

3. On considère un réseau de 10 antennes alimentées en phase ($\varphi = 0$).

   Quelle est l'influence de la distance d entre chaque antenne ?

Exercice 3 On considère un satellite situé à une distance de $D = 20\,000$ km de la Terre (satellite GPS).



On souhaiterait émettre à partir de ce satellite principalement en direction de la Terre, c'est-à-dire focaliser la puissance émise dans $[-\theta_0; \theta_0]$.

L'angle θ_0 se détermine directement par la relation trigonométrique : $\sin(\theta_0) = \dfrac{R}{R+D}$.

On souhaite déterminer les coefficients α_1, α_2, ..., α_N qui permettent d'avoir un diagramme de rayonnement respectant au mieux le gabarit suivant :

$$\begin{cases} F(\theta) = 1 \ \text{ pour } \theta \in [-\theta_0; \theta_0] \\ F(\theta) = 0 \ \text{ pour } \theta \in \left[-\dfrac{\pi}{2}; -\theta_0 \right[\ \cup \ \left] \theta_0; \dfrac{\pi}{2} \right] \end{cases}$$

On peut chercher les coefficients α_n en interpolant "au mieux" le gabarit idéal souhaité.

On échantillonne les angles $\theta \in [-\frac{\pi}{2}; \frac{\pi}{2}]$ en Q valeurs $\theta_n \in [-\frac{\pi}{2} : \delta_\theta : \frac{\pi}{2}]$, et de même le gabarit souhaité :

$$g(\theta_n) = g_n = \begin{cases} 1 \ \text{ si } \theta_n \in [-\theta_0; \theta_0] \\ 0 \ \text{ si } \theta_n \in \left[-\frac{\pi}{2}; -\theta_0 \right[\ \cup \ \left] \theta_0; \frac{\pi}{2} \right] \end{cases}$$

Si on définit les vecteurs $X = [\alpha_1 \ \alpha_2 \ \cdots \ \alpha_N]^T$ et $G = [g_1 \ g_2 \ \cdots \ g_Q]^T$, et la matrice $(Q \times N)$ par

$$M = \begin{bmatrix} 1 & e^{j\varphi_1} & e^{j\varphi_1} & \cdots & e^{j(N-1)\varphi_1} \\ 1 & e^{j\varphi_2} & e^{j\varphi_2} & \cdots & e^{j(N-1)\varphi_2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & e^{j\varphi_q} & e^{j\varphi_q} & \cdots & e^{j(N-1)\varphi_q} \end{bmatrix} \quad \text{où } \varphi_n = kd\cos(\theta_n)$$

alors le système à résoudre s'écrit : $MX = G$, et se "résout" à l'aide de Matlab par $X = M \setminus B$.

Une fois les coefficients α_n déterminés, le diagramme de rayonnement du réseau est donné par

$$S(\theta) = \left| \sum_{n=1}^{N} \alpha_n e^{jn\varphi} \right|^2 \quad \text{avec } \varphi = kd\cos(\theta)$$

On choisit un réseau de $N = 10$ antennes, et $Q = 100$ points d'interpolation.

Les antennes du réseau fonctionnent à la fréquence $f = 3.10^8$ Hz et sont distantes de $d = \dfrac{\lambda}{2}$.

1) Déterminer l'angle θ_0.

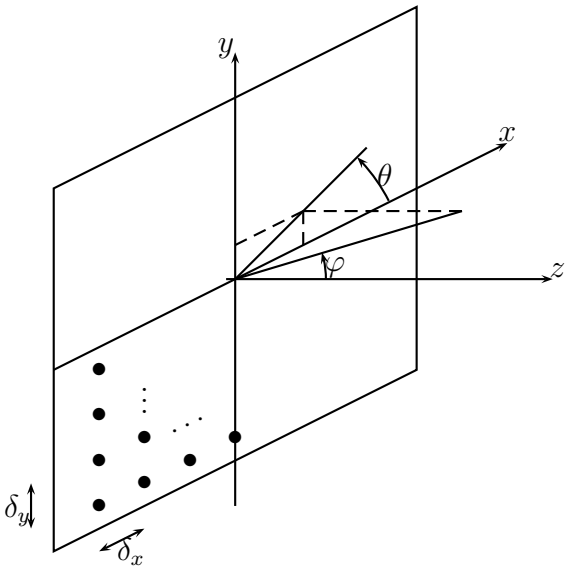
2) Définir la matrice M.

3) Résoudre le système au sens des moindres carrés ($X = M \setminus B$, voir `help slash` et TP1, Ex. 9).

4) Tracer le diagramme de rayonnement $S(\theta)$ obtenu, ainsi que la valeur des alimentations α_n correspondantes.

3. Réseau bidimensionnel d'antennes

Le cas des réseau bidimensionnels se traîte d'une manière similaire.



Si on impose un déphasage constant horizontalement φ_x et verticalement φ_y, alors on obtient

$$\overrightarrow{E} = NM\overrightarrow{E_0}e^{j\frac{\phi_x+\phi_y}{2}}\frac{\sin\left(\frac{N\phi_x}{2}\right)}{N\sin\left(\frac{\phi_x}{2}\right)}\frac{\sin\left(\frac{M\phi_y}{2}\right)}{M\sin\left(\frac{\phi_y}{2}\right)}$$

avec,
$$\begin{cases} \phi_x &= k\overrightarrow{\delta_x}\cdot\overrightarrow{r} - \varphi_x = k\delta_x\cos\theta\sin\varphi - \varphi_x \\ \phi_y &= k\overrightarrow{\delta_y}\cdot\overrightarrow{r} - \varphi_y = k\delta_y\cos\theta\sin\varphi - \varphi_y \end{cases}$$

La formation en réseau permet ainsi de mieux focaliser la puissance globale émise.

Remarque : Dans tous les développements précédents, les interférences entre les antennes ont été négligées : chaque antenne a été considérée comme rayonnant un champ indépendamment des autres antennes.
En fait, le champ rayonné par une antenne quelconque induit des courants dans les antennes voisines, modifiant ainsi d'une certaine façon son alimentation réelle.

Plus particulièrement, on peut utiliser un réseau d'antennes en n'alimentant qu'un seul de ses éléments, celui-ci alimentant les éléments voisins par induction, qui eux-mêmes alimentent leurs voisins ...
La phase de chaque élément est dans ce cas uniquement fixé par la géométrie du réseau (distance entre les éléments dans le cas d'un réseau linéaire).

Cette solution est d'ailleurs spécifiquement mise en œuvre dans les antennes de type Yagi-Uda (ou antenne-râteau) qui reposent sur le principe d'induction mutuelle entre les éléments proches.

Une autre stratégie peut consister à éviter les interactions parasites entre antennes, en utilisant des antennes dont le diagramme de rayonnement est nul dans la direction des éléments voisins. C'est le cas par exemple d'un réseau linéaire d'antennes dipolaires disposées convenablement (voir partie I, rayonnement d'un dipole).

TP 4
Méthode de Monté Carlo

Approximation de π

La méthode de Monté Carlo est une méthode probabiliste de calcul d'intégrale. Nous allons nous en servir dans ce TP pour approximer le nombre π (aire d'un cercle de rayon 1).

On considère un disque de rayon 1 centré à l'origine, inscrit dans un carré de côté 2.

La méthode de Monté Carlo consiste à tirer N points du plan, aléatoirement (fonction Matlab `rand`), situés dans le carré.

En probabilité, le ratio du nombre de points situés dans le disque sur le nombre total de points tend, lorsque N devient grand vers le ratio aire du disque sur aire du carré, soit $\pi/4$.

1- A quelle condition un point de coordonnées $(x; y)$ est-il situé dans le carré ? dans le disque ?

2- Ecrire une fonction Matlab *MonteCarlo* qui prend un seul paramètre N en argument, et qui, après avoir générer N points aléatoirement dans le carré, retourne le ratio souhaité (i.e. nombre de points dans le disque divisé par le nombre de points total dans la carré).

Vérifier que l'on obtient bien une approximation de π.

3- Ecrire un script Matlab qui trace en fonction du nombre total de points N, la valeur approchée de π trouvée.
Tracer l'erreur, en fonction de N, entre l'approximation calculée par la méthode de Monté Carlo et la valeur exacte de π (donnée par Matlab).
Commenter.

Calcul de l'intégrale d'une fonction quelconque

Plus généralement, la méthode de Monté Carlo est une méthode stochastique de calcul d'intégrale.

On cherche à déterminer la valeur de l'intégrale $I = \int_0^1 f(x)\,dx$, où f est une fonction connue mais dont l'intégrale peut-être très difficilement calculable.

On pourra prendre pour commencer la fonction simple $f(x) = x^2$, pour laquelle on calcule facilement $I = \dfrac{1}{3}$.

Dans la suite, on note $M = \sup_{x\in[0,1]} f(x)$ ($M = 1$ pour la fonction carré).

Ecrire une fonction Matlab *Montecarlo_Int* qui fournit, en fonction du nombre de points N souhaités, le ration N/n où n est le nombre de points, parmi les N, qui sont sous la courbe représentative de f.

Que vaut la limite de ce ratio lorsque N devient grand ?

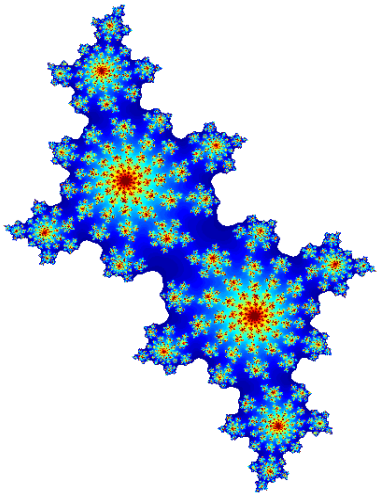
$$\lim_{N\to\infty} \frac{N}{n} = \ \ldots$$

Calculer alors une valeur approchée de I.

Utiliser cette méthode pour déterminer numériquement, puis représenter graphiquement, la fonction d'erreur :

$$\mathrm{erf}(x) = \int_0^x e^{-t^2}\,dt$$

TP 5
Fractales de Mandelbrot, Julia et Sierpinski



Les fractales sont des courbes ou surfaces de forme irrégulière ou morcelée qui se crée en suivant des règles déterministes ou stochastiques.
Le terme "fractale" est un néologisme créé par Benoît Mandelbrot en 1974 à partir de la racine latine fractus, qui signifie brisé, irrégulier.

Ces objets fractals, initialement objets purement mathématiques, ont trouvé depuis de nombreux champs d'applications.
Pour en citer quelques-uns :

- **En informatique.** Les fractales (plus précisément dans ce cadre fractales IFS pour *Systèmes de Fonctions Itérées*) permettent de décrire des paysages naturels, des nuages ...avec une grande fidélité. On repère dans l'image les parties qui présentent une certaines similarité, c'est-à-dire des parties qui peuvent se dédurie les unes des autres par des transformations géométriques (changement d'échelle, rotation). En ne stockant que ces informations, la taille de l'image peut-être nettement réduite.
  La compression fractale est très utilisée dans les animations en "temps réel" : images de synthèse, jeux vidéos ....
- **En biologie.** De nombreux exemples de géométrie fractale peuvent être trouvés dans la nature. Chez les végétaux (d'où l'idée d'ailleurs de compression d'image fractale), voir le choux Romanesco pour s'en convaincre ! Chez l'homme (ou autres animaux) dont les organes peuvent être modélisés efficacement en utilisant des géométries de type fractale.
- **En physique.** Les proprités d'auto-similarité (répétition à toutes les échelles de même motifs) des fractales ont aussi donné des idées aux physiciens : à grande échelle, l'Univers semble avoir une organisation fractale. La structure de notre galaxie se retrouve à l'échelle supérieure des amas de galaxie, mais aussi dans d'autres structures stellaires telles que les étoiles, nuages de gaz...

On se propose dans ce TP de générer trois types de géométrie fractales, celle de Mandelbrot (initiateur même des fractales), de Julia et enfin le triangle (ou éponge) de Sierpinski.

Fractale de Mandelbrot

Pour chaque point du plan $M(x;y)$, on note $z = x + iy$, et on définit la suite de nombres complexes (u_n) par $\begin{cases} u_0 = 0 \\ u_{n+1} = u_n^2 + z \end{cases}$.

Pour certains points M, donc certaines valeurs de x et y, la suite (u_n) ainsi définie diverge. L'ensemble de Mandelbrot est justement l'ensemble des points M tels que la suite (u_n) ne diverge pas, i.e. $\lim\limits_{n\to\infty} |u_n| \neq +\infty$.

Travail à réaliser. Nous allons balayer le plan à la recherche des points appartenant à l'ensemble de Mandelbrot. On fera varier les valeurs de x et y (deux boucles `for`) de -2 à 2 par pas respectivement de δ_x et δ_y.

Pour chaque position du point M, donc chaque valeur de x et y, calculer jusqu'à $Nmax$ termes de la suite (u_n) : u_1, u_2, u_3, …, u_{Nmax}. Après le calcul de chaque terme, verifier si $|u_n|$ est plus grand que la valeur *Seuil*; si c'est le cas, on considèrera que la suite diverge, et on pourra se passer du calcul des derniers termes de la suite.

Stocker alors dans une matrice *Mat* le nombre de termes calculés pour cette suite (la valeur $Nmax$ si la suite est toujours restée en deçà du seuil, ou une valeur n inférieur dans le cas contraire).

Représenter graphiquement le contenu de la matrice *Mat* (fonction `imagesc`).

On pourra prendre les valeurs numériques suivantes : $\delta_x = \delta_y = 0.01$, $Nmax = 30$, $Seuil = 2$

Fractale de Julia

Soit c un nombre complexe quelconque : $c \in \mathbb{C}$, et $M(x;y)$ un point du plan. On note $z = x+iy$.

On définit alors la suite de nombres complexes (u_n) par $\begin{cases} u_0 = z \\ u_{n+1} = u_n^2 + c \end{cases}$

De même que pour la suite de Mandelbrot, cette suite peut diverger ou non. L'ensemble de Julia est justement l'ensemble des points M du plan, donc des valeurs de z, tels que la suite ne diverge pas, i.e. $\lim\limits_{n\to\infty} |u_n| \neq +\infty$.

Travail à réaliser. Le principe est le même que pour l'ensemble de Mandelbrot, seul le calcul des termes successifs de la suite diffère quelque peu.

On pourra prendre les mêmes valeurs numériques pour précédemment, avec de plus, pour le paramètre c les valeurs suivantes : $c = -1$, $c = -1 + 0.2312i$, $c = 0.112 - 0.64i$.

Triangle de Sierpinski

La construction du triangle de Sierpinski peut se faire de manière géométrique.

Soit trois points $A(0;0)$, $B(1;0)$ et $C(0;1)$. L'algorithme permettant d'obtenir le triangle de Sierpinski est le suivant :

1. Chosisir un point P au hasard à l'intérieur du triangle ABC (à l'aide de la fonction `rand`).
2. Choisir un des trois sommets A, B ou C au hasard. On nommera S ce point par la suite.
3. Calculer le point M milieu du segment $[PS]$.
4. Redéfinir le point P par $P = M$. Stocker dans une matrice X l'abscisse de P, et dans une matrice Y l'ordonnée de P.
5. Réitérer N fois à partir de l'étape 2 ($N = 20\,000$ par exemple).

Finalement, afficher l'ensemble des points obtenus.

TP 6
Compression de données

Un message, ou texte, numérique est une suite de nombres binaires (de 0 et 1). Cette suite de nombres provient, par exemple, du codage binaire d'un texte alphanumérique (en français par exemple...). Chaque caractère du texte, les lettres de l'alphabet (y compris les caractères spéciaux : espaces, virgules,...) sont remplacés par un nombre à b chiffres constitué de 0 et 1.

En général, on utilise un codage de chaque lettre sur 8 bits, i.e. une séquence de 8 chiffres binaires. On peut ainsi coder différemment $2^8 = 256$ caractères : c'est le code ASCII.

Ainsi un texte alphanumérique composé de N caractères est représenté par $N \times 8$ bits, soit N octets. Le but de ce TP est de mettre en œuvre une méthode de compression de données, i.e. une méthode de réduction du nombre d'octets utilisés pour coder un message.

Il arrive fréquement qu'un texte donné présente des répétitions : des caractères utilisés bien sûr (seulement 26 lettres composent notre alphabet...), mais aussi répétitions de mots complets, voir séquences de mots...

C'est cette observation qui est à la base des méthodes de compression : au lieu de coder un message par une suite de nombres binaires directement issue du codage de chaque caractère successif constituant le message, on code le message suivant deux listes : la liste des mots utilisés (bien sûr sans répétition cette fois), ou dictionnaire, et la liste des positions de chaque mots du dictionnaire dans le message original.

Sous réserve d'un nombre assez important de répétitions des mots du dictionnaire utilisé, la taille de ces deux listes peut-être bien moindre que celle du codage direct du message.

1- Ecrire une fonction Matlab générant un message de N mots codés sur 8 bits (soit un message de $8N$ bits, ou N octets, au total), n'utilisant (ou répétant) que n mots d'un dictionnaire.
On pourra commencer par créer le dictionnaire comme étant une matrice (ou tableau) à n lignes (une pour chaque mot) et 8 colonnes ; chaque ligne de ce tableau étant une suite aléatoire de 8 chiffres binaires.
Ensuite, le message peut se construire en choisissant aléatoirement des mots dans ce dictionnaire, que l'on concaténera.

Ce message étant construit, on considère par la suite que ce dictionnaire est inconnu ; seul le message de N mots est donné.

2- Ecrire une fonction Matlab *Compress* qui permet de compresser ce message.
Le principe de cette fonction est le suivant. On part du premier mot du message (premier octet), on l'ajoute à au dictionnaire (liste des mots du message, sans répétition), on recherche dans le message les autres occurences de ce mot, et on enregistre dans un tableau la position dans ce message de ces occurences.
On réitère ensuite le procédé à partir du mot suivant dans le message qui n'est pas encore dans le dictionnaire.
On crée ainsi deux tableaux : le premier est le dictionnaire, le deuxième contient la position de chacun des mots du dictionnaire dans le message

On pourra construire une fonction intermédiaire retournant le $n^{\text{ème}}$ mot du message.

3- Ecrire une fonction Matlab *DeCompress* permettant, à partir des deux tableaux précédants de reconstruire le message, et s'assurer ainsi que l'on a bien un codage équivalent du message original.

4- Calculer le taux de compression obtenu, i.e. le ratio taille des deux tableaux "compressés" sur taille du message initial.

TP 7
Simulation de la diffusion thermique dans un matériau

L'objectif de ce TP est la simulation de la diffusion thermique dans un milieu.

On suppose ce milieu 1-dimensionel (un câble par exemple), pour simplifier l'étude. On décrit ainsi géométriquement ce milieu par le segment $[0, L]$.

Le problème est le suivant : on impose une température fixée et constante aux deux extémités, T_0 en $x = 0$ et T_L en $x = L$, du milieu supposé à température nulle initialement, et éventuellement la présence d'une source de chaleur décrite par une fonction $f(x)$ indépendante du temps, ou une fonction $f(x, t)$ dépendante du temps.

Dans ces conditions, on peut chercher à répondre aux questions,

- Quel est le profil de température final dans le milieu (i.e. quelle est la température $T(x)$ pour $x \in [0, L]$ pour un temps assez grand) ?
- Comment évolue le profil de température en fonction du temps, depuis la température nulle imposée initialement jusqu'au profil final calculé précédemment ?
- Comment détecter l'éventuelle présence d'une source de chaleur à l'intérieur du milieu ?

Modélisation du problème On note $T(x, t)$ la température dans le milieu à l'abscisse x et à l'instant t, $x \in [0; L]$ et $t \in [0; +\infty[$.

La diffusion thermique dans un milieu est régie par l'équation dite "de la chaleur" :

$$\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} = f(x, t), \tag{1}$$

où $f(x, t)$, éventuellement $f(x, t) = f(x)$ est indépendante du temps, est la source de chaleur fournie au milieu au cours du temps.

1. Approximation de la température à l'équilibre

On suppose dans ce paragraphe que la source de chaleur est indépendante du temps : $f(x, t) = f(x)$. On cherche alors à résoudre numériquement l'équation de la chaleur (1) en régime stationnaire (à l'équilibre) ; cela revient à considérer que la température ne varie plus en fonction du temps, et que, par conséquent, $\frac{\partial T}{\partial t} = 0$: la fonction température n'est plus qu'une fonction d'une seule variable $T(x)$. Dans ce cas l'équation de la chaleur (1) est une équation différentielle du second ordre à coefficient constant :

$$T''(x) = f(x)\,.$$

Selon la fonction source f que l'on impose, on sait résoudre théoriquement cette équation. Le but de la simulation est néanmoins double :

- s'assurer que le code programmé fonctionne correctement, ce qui est possible dans ce cas car on connaît la solution attendue ;
- calculer numériquement la solution de cette équation même lorsque le second membre f ne permet pas de le faire théoriquement.

Discrétisation On discrétise le segment $[0, L]$ par N_x segments de longueur identique $\delta_x = L/N_x$: $[0; L] \to [0 : \delta_x : L]$. On a ainsi échantillonné le segment $[0; L]$ en $N_x + 1$ points $x_n = n\delta_x$.

On cherche alors évaluer la température à chacun de ces points. On note pour cela :

$$T_n = T(n\delta_x)\,.$$

La formule de Taylor à l'ordre 2 nous donne, dans la limite où le pas δ_x est supposé assez petit,

$$T_{n+1} = T((n+1)\delta_x) \sim T(n\delta_x) + \delta_x \frac{dT}{dx}(n\delta_x) + \frac{\delta_x^2}{2}\frac{d^2T}{dx^2}(n\delta_x)\,,$$

soit

$$T_{n+1} \sim T_n + \delta_x \frac{dT_n}{dx} + \frac{\delta_x^2}{2}\frac{d^2T_n}{dx^2}\,.$$

De même, on a

$$T_{n-1} \sim T_n - \delta_x \frac{dT_n}{dx} + \frac{\delta_x^2}{2}\frac{d^2T_n}{dx^2}\,,$$

puis, après addition de ces deux relations,

$$\frac{d^2T}{dx^2} \sim \frac{T_{n+1} + T_{n-1} - 2T_n}{\delta_x^2}$$

Si on pose alors $f_n = f(n\delta_x)$, on obtient alors la relation de récurrence :

$$T_{n+1} = \delta_x f_n + 2T_n - T_{n-1} \tag{2}$$

On pourra prendre comme données initiales :

$T_1 = \alpha$ (une température en degré) et, $T_2 = T_1$ (pas de vitesse intiale de propagation)

On remarque qu'avec ce modèle statique, on ne peut pas imposer une température aux deux extrémités à la fois du milieu : la relation de récurrence (2) permet en effet de calculer toutes les températures de proche en proche à partir de T_0 et T_1.

2. Diffusion thermique - Approximation de l'évolution temporelle du profil de température

On cherche maintenant à étudier l'évolution du profil de température au cours du temps : le terme $\frac{\partial T}{\partial t}$ n'est plus nul et la température est fonction à la fois de l'abscisse x et du temps t, $T(x,t)$.

La discrétisation en x se fait comme précédemment en $N_x + 1$ points, $[0 : L] \;\rightarrow\; [0 : \delta_x : L]$.

On effectue la même opération sur les temps. On découpe l'intervalle temporel d'étude $[0; \Delta_t]$ en $N_t + 1$ points : $[0 : \Delta_t] \;\rightarrow\; [0 : \delta_t : \Delta_t]$.

On note alors, $T_{i,j} = T(i\delta_x, j\delta_t)$, et de même pour la source de chaleur : $f_{i,j} = f(i\delta_x, j\delta_t)$. La discrétisation de la dérivée seconde par rapport à x peut se faire identiquement à celle utilisée précédemment.

De la même façon, la dérivée par rapport au temps peut-être approchée par

$$\frac{\partial T}{\partial t} \sim \frac{T_{i+1,j} - T_{i,j}}{\delta_t}.$$

On obtient alors la nouvelle relation de récurrence :

$$T_{i,n+1} = T_{i,n} + \delta_t f_{i,n} + \frac{\delta_t}{\delta_x^2}\left(T_{i+1,n} + T_{i-1,n} - 2T_{i,n}\right) \tag{3}$$

à laquelle il faut ajouter les conditions en $x = 0$ et $x = L$:

$$T(1,:) = \alpha \;\text{ et }\; , T(end,:) = \beta$$

où α est donc la température en $x = 0$ et β celle en $x = L$.

Application numérique On pourra prendre les valeurs numériques (pour commencer...) suivantes :

$L = 1$, $\delta_x = 0.1$, $\delta_t = 0.0001$, $\alpha = 0$, $\beta = 100$,

$f(x) = \ldots$ au choix : toute fonction définie sur $[0; L]$.

TP 8

Simulation monodimensionnelle de la propagation d'une onde

La propagation d'une onde, acoustique ou électromagnétique, est régie par l'équation de Helmohltz, pour $t > 0$ et $x \in D$, le domaine géométrique étudié :

$$\frac{\partial^2 f}{\partial t^2}(x,t) - c^2 \frac{\partial^2 f}{\partial x^2}(x,t) = 0\,, \tag{4}$$

où c désigne la célérité de l'onde dans le milieu considéré ($c = 3.10^8$ m.s^{-1} pour la lumière dans le vide, par exemple), et la fonction f désigne l'amplitude de l'onde au point x et à l'instant t.

Cette équation n'est pas suffisante pour décrire complètement la propagation de l'onde ; il faut la compléter de conditions initiales (état et vitesse de l'onde à l'instant initial $t = 0$), ainsi que des conditions au bord (comportement de l'onde lorsqu'elle rencontre les limites physiques du domaine étudié).

Ces conditions peuvent se formuler suivant :

$$\left\{ \begin{array}{ll} f(x,0) & = h(x)\ ,\ \forall x \in D \\ \dfrac{\partial f}{\partial t}(x,0) & = \tilde{h}(x)\ ,\ \forall x \in D \\ f(x,t) & = g(t)\ ,\ \forall t > 0 \quad \text{et, } \forall x \in \partial D\,, \end{array} \right. \tag{5}$$

les fonctions h, $\tilde{h}$ et g étant des données du problème.

Le but de cette partie est de résoudre numériquement sous Matlab la propagation d'une onde, i.e. de résoudre numériquement le jeu d'équations (4) et (5), dans le cas simple 1-D, c'est-à-dire lorsque le domaine D est un intervalle : $D =]a, b[$.

La cas 1-D correspond par exemple à la simulation du mouvement d'une corde (de piano ou de guitare par exemple), ou de l'amplitude d'un champ électrique ou magnétique en polarisation TE ou TM.

Discrétisation de l'équation de Helmohltz (4)

On cherche une solution approchée de (4) en 1-D : le domaine géométrique dans lequel l'onde peut se propager est le segment $[0\,, L]$. On s'interesse alors à l'onde $f(x,t)$ pour $x \in [0\,, L]$ et $t \in [0\,, T]$.

Discrétisation de (4) : Pour approximer l'onde solution de (4), nous allons utilisé la méthode numérique dite des différences finies.

On se donne tout d'abord pour cela une "grille" régulière de calcul, c'est-à-dire une discrétisation de l'ensemble $[0\,, L] \times [0\,, T]$: soit les deux suites

$$\{x_k\} = \{k\,\delta_x\}_{k=0\ldots L/\delta_x}\ ,\ \text{et, } \{t_k\} = \{k\,\delta_t\}_{k=0\ldots T/\delta_t}\,,$$

où δ_x et δ_t désignent respectivement le pas de la grille en x (espace) et en t (temps).

Cette grille étant donnée, on cherche à calculer les valeurs de f aux noeuds de la grille. On notera

$$f_{i,j} := f(x_i, t_j)\,.$$

La deuxième étape est l'approximation des dérivées partielles de f à l'aide des éléments $f_{i,j}$. Cela s'effectue à l'aide de la formule de Taylor, par exemple pour la dérivée selon x :

$$f(x+\delta_x,t) = f(x,t) + \delta_x \frac{\partial f}{\partial x}(x,t) + \frac{\delta_x^2}{2}\frac{\partial^2 f}{\partial x^2}(x,t) + O(\delta_x^3)\,,$$

ou, en d'autres termes

$$f_{i+1,j} = f_{i,j} + \delta_x \frac{\partial f_{i,j}}{\partial x} + \frac{\delta_x^2}{2}\frac{\partial^2 f_{i,j}}{\partial x^2} + O(\delta_x^3)\,.$$

On peut de cette manière approximer les dérivées secondes de f par rapport à x et t (en considérant aussi par exemple le développement de Taylor de $f(x-\delta_x,t)$ et en sommant les relations).

On obtient, tous calculs fait :

$$\frac{\partial^2 f_{i,j}}{\partial t^2} \sim \frac{1}{\delta_t^2}\left[f_{i,j+1} + f_{i,j-1} - 2f_{i,j}\right]\,,$$

et,

$$\frac{\partial^2 f_{i,j}}{\partial x^2} \sim \frac{1}{\delta_x^2}\left[f_{i+1,j} + f_{i-1,j} - 2f_{i,j}\right]\,.$$

En combinant ces deux expressions avec l'équation de Helmohltz (4), on obtient la relation de récurrence :

$$f_{i,j+1} = -f_{i,j-1} + \gamma f_{i+1,j} + \gamma f_{i-1,j} - 2(1-\gamma)f_{i,j}\,, \tag{6}$$

où $\gamma = c^2\delta_t^2/\delta_x^2$.

Il ne reste plus alors qu'à initialiser la suite $f_{i,j}$ au moyen des équations (5).

Pour commencer, on pourra prendre $g = 0$. Cette condition modélise une paroi infiniment dure ; l'onde y sera totalement réfléchie.

La fonction $h(x)$ modélise l'état initial de l'onde, à l'instant $t = 0$. On pourra utiliser la fonction h, infiniment régulière :

$$h(x) = \begin{cases} A\exp\left(-\dfrac{1}{|(x-x_0)^2 - r^2|}\right)\ , \text{ pour } |x-x_0| \le d \\ 0\ ,\ \text{ailleurs}\,, \end{cases}$$

où $A = \exp(1/r^2)$ permet de normaliser h, et où $x_0 \in [0,L]$ est le centre de la "cloche" $h(x)$ et $r > 0$ sa largeur.

La fonction $\tilde{h}$ modélise l'"impulsion", ou la vitesse, initiale donnée à l'onde. On pourra choisir pour commencer $\tilde{h} = 0$.

Travail à réaliser

Initialisation On prendra (pour commencer) les valeurs numériques suivantes : $c = 1\text{m.s}^{-1}$, $\delta_x = \delta_t = 10^{-3}\text{m}$, soit $\gamma = 1$, et $L = 1\text{m}$, $T = 1\text{s}$.

On définit ainsi la grille de calcul : $(x_i,t_j) \in [0:\delta_x:L]\times[0:\delta_t:T]$.

La solution $f(x,t)$, approximée aux point (x_i,t_j), sera alors stockée dans une matrice rectangulaire $F_{i,j}$.

On utilisera la fonction $h(x)$ définie précédemment pour définir l'état initial de l'onde, avec $x_0 = L/2$ et $r = L/5$, tandis que l'on imposera $\tilde{h}(x) = 0$ (vitesse initiale nulle).

L'initialisation s'implémente alors simplement suivant :

$$F(:,1) = h(:)\ ,\ \text{et, } F(:,2) = F(:,1) + \delta_t h_t(:)\,,$$

où $h_t(x)$ est la vitesse initiale de l'onde (on pourra prendre pour commencer $h_t = 0$).

La condition au bord : $f(x=0,t) = f(x=L,t)\,,\forall t$ se traduit quant à elle par :

$$F(1,:) = 0\ ,\ \text{et, } F(end,:) = 0\,.$$

Récurrence Il s'agit ici d'implémenter la relation de récurrence (6). Le calcul peut se faire à l'aide de deux boucles imbriquées : à chaque pas de temps (i.e. pour chaque j), on calcul tout les $F_{i,j}$ récursivement.

Visualisation du résultat Le comportement complet de l'onde est contenu dans la matrice F : chaque colonne contient l'amplitude de l'onde à un instant donné, chaque ligne correspondant à une abscisse.

On peut simplement visualiser la propagation de l'onde en utilisant la commande *imagesc(F)*.

L'amplitude de l'onde au $n^{\text{ième}}$ pas de temps s'affiche alors simplement par la commande *plot(*$[0 : \delta_x : L], F(:, n)$*)*.

Comme il s'agit d'un problème d'évolution, il peut aussi être interessant de représenter ces tracés successivement. On pourra, par exemple, regrouper ces graphiques successifs dans une video à l'aide des commandes *getframe*, et *avifile*.

Suite de l'étude... On peut reprendre l'étude précédente, et modifier les conditions initiales et la condition au bord.

On pourra, par exemple, s'interesser au cas ou deux ondes existent initialement, et qui vont donner lieu à des phénomènes d'interférences (constructives ou destructives).

La condition au bord pourra elle aussi être modifiée en la condition dite de Neumann :

$$\frac{\partial f}{\partial x}(x,t) = 0 \ , \ \text{pour } x = 0 \ \text{ et } x = L \, .$$

TP 9

Simulation bidimensionnelle de la propagation d'une onde

Ce TP fait suite au précédent et propose de simuler la propagation d'une onde dans une cavité, dans l'espace libre, puis sa diffraction à travers une fente.

Cette propagation est régie par l'équation des ondes : si $u(x, y, t)$ désigne l'amplitude de l'onde au point d'espace de coordonnées (x, y) et à l'instant t, alors u est solution de l'équation

$$\frac{\partial^2 u}{\partial t^2} - c^2 \Delta u = s(x, y, t) \tag{1}$$

où, $\Delta u = \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}$ désigne le laplacien de u, c la célérité de l'onde dans le milieu, et $s(x, y, t)$ la source.

On cherche une solution approchée de l'équation (1) en 2-D : le domaine géométrique dans lequel l'onde peut se propager est le rectangle $[0\,, Lx] \times [0\,, Ly]$. On s'interesse alors à l'amplitude de l'onde $u(x, y, t)$ pour $x \in [0\,, L_x]$, $y \in [0\,, L_y]$ et $t \in [0\,, T]$.

Afin de traiter numériquement cette équation, le domaine est découpé en une grille de calcul :

$$[0\,, L_x] \to [0 : \delta_x : L_x]\ , \quad [0\,, L_y] \to [0 : \delta_y : L_x]\ , \quad [0\,, T] \to [0 : \delta_t : T].$$

On notera par la suite n_x, n_y et n_t les nombres de points utilisés pour les discrétisations suivant respectivement l'axe des x, des y, et des temps t.

Discrétisation de l'équation des ondes De même que dans le TP précédent, on pose $u^k_{i,j} = u(x_i, y_j, t_k)$, les x_i, y_j et t_k étant des points de notre grille de calcul.

L'utilisation des formules de Taylor (cf. TP précédent) nous permet alors de formuler l'équation des ondes (1) sous forme discrète selon la relation de récurrence :

$$u^{k+1}_{i,j} = 2u^k_{i,j} - u^{k-1}_{i,j} + \gamma_x \left[u^k_{i-1,j} + u^k_{i+1,j} - 2u^k_{i,j}\right] + \gamma_y \left[u^k_{i,j-1} + u^k_{i,j+1} - 2u^k_{i,j}\right] \tag{2}$$

où, $\gamma_x = \dfrac{c^2\delta_t^2}{\delta_x^2}$, et $\gamma_y = \dfrac{c^2\delta_t^2}{\delta_y^2}$

Initialisation Pour initialiser la relation de récurrence précédente, on suppose que le milieu est au repos : la vitesse intiale de l'onde est nulle.

Ceci se traduit par $u^2_{i,j} = u^1_{i,j}$, et donc dans la relation de récurrence précédente par :

$$u^2_{i,j} = u^1_{i,j} + \frac{\gamma_x}{2}\left[u^1_{i-1,j} + u^1_{i+1,j} - 2u^1_{i,j}\right] + \frac{\gamma_y}{2}\left[u^1_{i,j-1} + u^1_{i,j+1} - 2u^1_{i,j}\right] \tag{3}$$

Source La source est une source sinusoïdale d'amplitude 1 et de fréquence $\nu = 2$ Hz, située en $(L_x/3, L_y/2)$.

A chaque itération temporelle, l'amplitude de l'onde doit donc être modifiée suivant :

$$u^{k+1}_{s_x,s_y} = u^k_{s_x,s_y} + \sin(2\pi\nu k\delta_t) \tag{4}$$

où, $s_x = n_x/3$ et $s_y = n_y/2$ désignent les indices de la position $(L_x/3, L_y/2)$ de la source dans notre grille de calcul.

Réflections parasites contre les bords du domaine

Le domaine de calcul $[0\,,L_x] \times [0\,,L_y]$ a été introduit ici artificiellement : l'équation des ondes est valable dans tout l'espace (espace libre) ; néanmoins les calculs numériques requièrent une délimitation de ce domaine.

On observe alors dans le résultat de nos calculs des réflexions parasites le long des bords de notre domaine. Ces réflexions, quoique expliquables physiquement, n'ont pas ici lieu d'être, la prapagation étant supposée se faire en espace libre (et donc infini).

Pour pallier cet inconvénient, on peut introduire des conditions dites absorbantes au bord de notre domaine (CLA, pour Conditions aux Limites Absorbantes) , de manière à éliminer ces réflexions parasites.

Ces conditions absorbantes peuvent être simulées de la manière suivante :

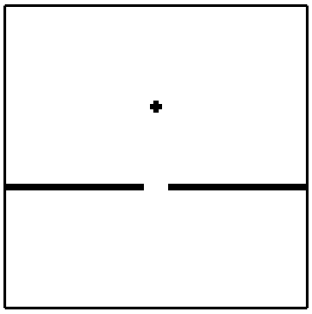
$$\begin{array}{rclcrcl} u(1,:,k+1) & = & u(2,:,k) & & u(:,1,k+1) & = & u(:,2,k) \\ u(n_x,:,k+1) & = & u(n_x-1,:,k) & & u(:,n_y,k+1) & = & u(:,n_y-1,k) \end{array} \tag{5}$$

Présence d'un obstacle

On peut simuler la présence d'un obstacle en imposant $u^k_{i,j} = 0$ pour tout k (tout instant), et pour des indices i et j correspond à la position de l'obstacle.

Diffraction par une fente simple

On simule la présence d'une fente simple par un "masque" :



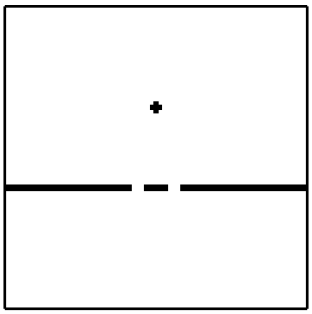
On impose donc,

$$u(I_x\,,\,I_y\,,:) = 0,$$

avec I_x et I_y des indices correspondants à la position de la fente dans la grille de calcul.

Diffraction par une double fente : interférences

Le principe est le même que pour la modélisation de la présence d'une fente simple.
La fente double est introduite par l'intermédiaire d'un "masque" :



$$u(I_x\,,\,I_y\,,:) = 0,$$

avec I_x et I_y des indices correspondants à la position de la fente dans la grille de calcul.

Travail à réaliser Le but du programme est de calculer les valeurs $u(i,j,k)$ pour $1 \leq i \leq n_x$, $1 \leq j \leq n_y$, et $1 \leq k \leq n_t$.

Pour cela, on commencera par initialiser les valeurs de la matrice u (`zeros`), puis grâce à l'initialisation (3).

Ensuite, on implémentera la discrétisation de l'équation des ondes (2), que l'on complétera pour prendre en compte la présence de la source sinusoïdale donnée par la relation (4).

Ce programme ainsi contistué simule la propagation d'une onde dans une cavité. On pourra en particulier observer les phénomènes de réflection sur les parois, et les phénomènes d'interférence qui s'en suivent.

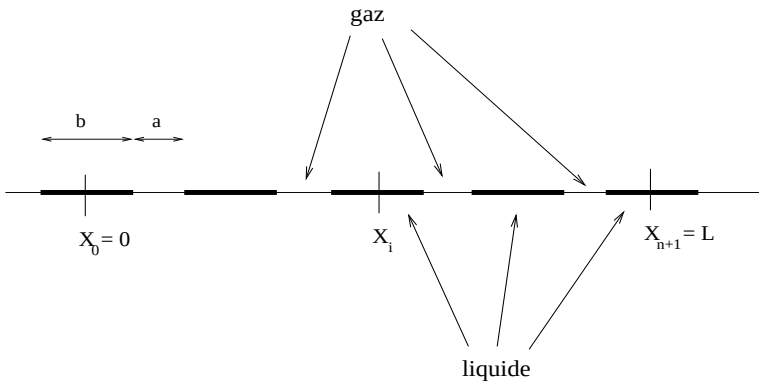
On complétera ensuite ce modèle et ces simulations en intégrant les conditions aux limites absorbantes données par les relations (5) : on obtient une simulation de la propagation en espace libre. Les bords du domaine sont "transparents" vis-à-vis des ondes, et les interférences parasites ont été supprimées.

Enfin, on pourra ajouter une fente simple ou double et ainsi observer des phénomènes de diffraction d'onde par une fente, et les phénomènes d'interférences.

Valeurs numériques On prendra (pour commencer) les valeurs numériques suivantes : $c = 10\text{m.s}^{-1}$, $L_x = L_y = 100$ m, $\delta_x = L_x/200$, $\delta_y = L_y/200$, $\delta_t = \frac{\sqrt{\delta_x^2+\delta_y^2}}{2c}$, $\nu = 2$ Hz, $T = 10$ s.

TP 10
Propagation d'ondes
Exemple de contrôle non-destructif

Modèle physique. On cherche à modéliser la propagation d'ondes dans un mélange liquide-gaz. On suppose que le liquide est incompressible, et que le gaz, compressible, se trouve sous la forme de petites bulles qui contiennent le même nombre de molécules. On s'intéressera ici à une modélisation mono dimensionnelle de cette situation : on considère sur le segment $[0, L]$ une alternance de gaz rt de liquide, comme illustré sur la figure ci-dessous.



On note ρ la masse linéique du liquide, de telle sorte que la masse d'un "segment" S_i centré en x_i est ρb. Le gaz piégé entre deux segments successifs est supposé obéir à la loi des gaz parfaits à température constante, c'et-à-dire

$$\text{(pression)} \times \text{(volume)} = \text{Constante} \times \text{(nombre de molécules de gaz)}$$

soit,

$$PV = kN\,,$$

où le "volume" représente en fait une longueur dans le cadre mono dimensionnel. On suppose les deux extrémités fixes (x_0 et x_{n+1} positionnés en 0 et L, respectivement), et on prend un nombre de molécules de gaz dans chaque poche constant égal à N. On considérera que N s'écrit $\beta(a+b)$ où β est le nombre de molécules de gaz par unité de longueur du mélange. Ce système liquide/gaz admet un état d'équilibre, représenté schématiquement sur la figure précédente, pour lequel toutes les bulles ont même taille b. Les positions des centres des segments fluides sont, dans ce cas, en notant $h = a + b$,

$$X_0 = 0,\ X_1 = h,\ \cdots,\ X_i = ih,\ \cdots,\ X_{n+1} = (n+1)h = L\,.$$

Equation d'évolution. On note $x_i = X_i + u_i$ la position du centre de S_i ($i^{\text{ème}}$ segment), de telle sorte que l'état d'équilibre correspond à $u_i = 0$ pour tout i. Le bilan F_i des forces s'exerçant sur un élément de liquide centré en x_i est la somme des pressions exercées à droite et à gauche par le gaz, c'est-à-dire, d'après le modèle précédent,

$$F_i = k\beta h\left(\frac{1}{a + u_i - u_{i-1}} - \frac{1}{a + u_{i+1} - u_i}\right)\,.$$

La masse de S_i étant ρb, la relation fondamentale de la dynamique s'écrit donc

$$\rho b u_i'' = F_i\,.$$

On note $\mathbf{u} = (\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u_n})$ le vecteur des inconnues, $v_i = u_i'$ la vitesse de S_i, $\mathbf{v}$ le vecteur des v_i (donc $v_i = \frac{du_i}{dt}$, et $\mathbf{v} = \frac{d\mathbf{u}}{dt}$)

et $\mathbf{F}$ le vecteur des forces F_i. On peut se ramener de cette façon à une équation différentielle ordinaire (dans $\mathbb{R}^{2n}$),

$$\begin{cases} \dfrac{d\mathbf{u}}{dt} = \mathbf{v} \\ \rho b \dfrac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{u}) \ . \end{cases} \tag{6}$$

1. Vitesse de propagation

On cherche à estimer la vitesse de propagation associée au modèle décrit ci-dessus, pour des ondes de faible amplitude (i.e. $u_i << a+b$). La force F_i peut s'exprimer suivant

$$\begin{aligned} F_i &= k\beta h \left(\frac{1}{a+u_i-u_{i-1}} - \frac{1}{a+u_{i+1}-u_i} \right) \\ &= k\beta h \frac{u_{i+1}-2u_i+u_{i-1}}{(a+u_i-u_{i-1})(a+u_{i+1}-u_i)} \\ &= k\beta h \frac{u_{i+1}-2u_i+u_{i-1}}{h^2} \frac{1}{\left(\alpha+\frac{u_i-u_{i-1}}{h}\right)\left(\alpha+\frac{u_{i+1}-u_i}{h}\right)} \ , \end{aligned}$$

avec $h = a+b$ et $\alpha = a/(a+b) = a/h$. On identifie maintenant les expressions faisant intervenir les u_i à des dérivées en espace de u (cf. méthode des différences finies) :

$$\frac{u_{i+1}-2u_i+u_{i-1}}{h^2} \sim \frac{\partial^2 u}{\partial x^2}(X_i) \ , \quad \frac{u_i-u_{i-1}}{h} \sim \frac{u_{i+1}-u_i}{h} \sim \frac{\partial u}{\partial x}(X_i) \ .$$

On aboutit donc, formellement, à l'équation aux dérivées partielles sur u (en remarquant que $b/h = 1-\alpha$)

$$\rho(1-\alpha)u'' - \frac{k\beta}{(\alpha+u')^2}\frac{\partial^2 u}{\partial x^2} = 0 \ .$$

Si l'on fait l'hypothèse supplémentaire que u' reste petit devant α, on obtient, toujours formellement, l'équation

$$u'' - c^2 \frac{\partial^2 u}{\partial x^2} = 0 \ , \quad \text{avec } c^2 = \frac{k\beta}{\rho(1-\alpha)\alpha^2} \, ,$$

où c est donc la vitesse de propagation de l'onde dans ce milieu.

Cette équation est l'équation des ondes. Son étude et sa simulation ont fait l'objet du TP précédent.

2. Propagation d'une onde dans un milieu homogène

On se propose de résoudre numériquement le système (6) pour modéliser la propagation d'une onde dans le milieu (sans faire, a priori d'hypothèse de type "petits déplacements").

Pour perturber le système, on va agir sur la position du point x_0, qu'on ne supposera donc plus fixe, mais dont la position va être imposée au cours du temps.

On introduit une discrétisation de l'intervalle en temps $[0,T]$:

$$t_0 = 0 < t_1 = \delta_t < t_2 = 2\delta_t < \cdots < t_M = M\delta_t = T \ ,$$

et on note u_n^m (resp. v_n^m) l'approximation de u_n (resp. v_n) au temps t_m. On discrétise alors le système (6) de la façon suivante :

$$\begin{cases} \dfrac{u_n^m - u_n^{m-1}}{\delta_t} &= v_n^m \\ \rho b \dfrac{v_n^{m+1}-v_n^m}{\delta_t} &= k\beta h \left(\dfrac{1}{a+u_n^m-u_{n-1}^m} - \dfrac{1}{a+u_{n+1}^m-u_n^m} \right) \ , \end{cases}$$

ce qu'on peut finalement écrire, en éliminant les v :

$$u_n^{m+1} = 2u_n^m - u_n^{m-1} + \delta_t^2 \mu \left(\frac{1}{a + u_n^m - u_{n-1}^m} - \frac{1}{a + u_{n+1}^m - u_n^m} \right) ,$$

avec $\mu = k\beta h/\rho b$. Les conditions aux limites en espace s'écrivent

$$u_0^m = \lambda(t) \ , \ u_N^m = 0 \ \forall \, m = 0, \cdots, M \ ,$$

où $\lambda(t)$ est donnée, et les conditions en temps

$$u_n^0 = U_n^0 \ , \ u_n^{-1} = U_n^{-1} \ \forall \, n = 0, \cdots, N \ ,$$

où les U_n^0 et U_n^{-1} sont des données qui permettent de représenter l'état du système au temps initial.

A partir de ces relations, on peut alors calculer les u_n^m pour tout n et tout M.

Calcul numérique. Calculer numériquement les u_n^m. On pourra prendre comme valeur des paramètres :

$$\rho = 1 \ , \ N = 99 \ , \ L = 1.0 \ , \ \alpha = 0.2 \ , \ \beta = 1 \ , \ k = 1.0 \ , \ \delta_t = 10^{-3} \ , \ T = 2 \ , \ h = 10^{-2} \ , \ b = 1.$$

3. Propagation d'une onde dans un milieu inhomogène

On considère maintenant que la densité, ou masse linéique, ρ n'est plus constante, soit $\rho := \rho(x)$. On discrétise comme précédemment : $\rho_n = \rho(X_n)$.

Modifier le programme de calcul précédent afin de permettre la prise en compte d'une densité non constante, i.e. un milieu inhomogène.

Reprendre les valeurs numériques précédentes, en modifiant uniquement ρ suivant :

$$\rho_n = \begin{cases} 2 \ , \text{ si } n = 50 \\ 1 \ , \text{ sinon} \end{cases} .$$

4. Exemple de contrôle non destructif

La problématique est maintenant la suivante : comment à partir des seules données et observations en $x = 0$ et/ou $x = L$ déterminer si le matériau est homogène ou non, et alors dans ce cas, de manière plus ambitieuse, déterminer la profondeur à laquelle est situé l'inhomogénéité ?

A partir des calculs et observations effectués dans les deux paragraphes précédents, donner une méthode permettant de

- détecter la présence d'une inhomogénéité dans le milieu ;
- calculer la profondeur (abscisse) à laquelle se trouve cette inhomgénéité .