Calcul Scientifique

Ecole Hassania des Travaux Publics

Salem Nafiri

Modélisation et simulation Numérique

Les équations sont partout

Chaque problème de la vie courante a son équation !

Principes mathématiques de la philosophie naturelle est l'œuvre maîtresse d'Isaac Newton (1643-1727). Cet ouvrage en latin, divisé en trois parties (ou livres, du latin liber), est publié à Londres en 1687.

La traduction française fut publiée à Paris en 1756, sous le titre Principes mathématiques de la philosophie naturelle ; elle est l'œuvre d'Émilie du Châtelet (1706-1749).

Modélisation mathématique et simulation numérique

La modélisation mathématique est l'art (ou la science) de représenter (ou de transformer) une réalité physique en des modèles abstraits accessibles à l'analyse et au calcul.

La simulation numérique est le processus qui permet de calculer sur ordinateur les solutions de ces modèles et donc de simuler la réalité physique.

La modélisation mathématique et la simulation numérique ont pris une importance considérable ces dernières décennies dans tous les domaines de la science et des applications industrielles (ou sciences de l'ingénieur).

Modélisation mathématique et simulation numérique

Attention :

- il faut pas trop simplifier pour ne pas perdre des propriétés qualitatives importantes du problème que l'on cherche à modéliser ;
- il faut pas garder un système trop compliqué sur lequel toute étude théorique serait hors de portée ;

A quoi sert la simulation numérique ?

La simulation numérique est avant tout un objet mathématique. Concrètement, il s'agit de la représentation mathématique d'un objet ou d'un système.

A quoi servent ces représentations mathématiques ?
L'utilité de la simulation réside souvent dans le fait de réaliser ce que l'on appelle des expériences virtuelles. Imaginez que j'ai une télécommande dans la main : je peux essayer de la compresser, de la casser, de la chauffer, etc. Si pour diverses raisons je n'ai pas accès à cette télécommande, mais que je possède un modèle mathématique de cette dernière [la physique est le lieu où l'on créé des modèles mathématiques des objets réels] au lieu d'agir sur elle dans le monde réel, je vais pouvoir essayer de mener un calcul numérique qui me donnera une idée de la réponse de cet objet à différentes sollicitations que l'on pourrait avoir sur lui.

Quelques applications

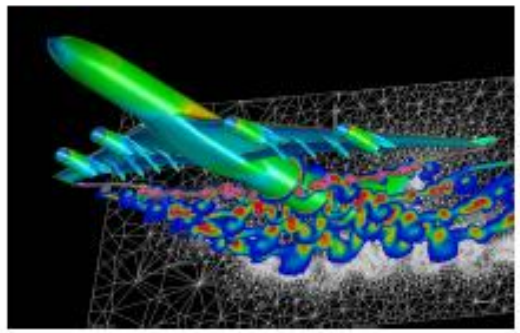
Les types d'application sont nombreux et variés.

- On peut faire une simulation numérique lorsque l'objet est inaccessible. Par exemple, en astrophysique : les étoiles, les galaxies ou l'univers dans son entier sont des objets de simulation parce qu'ils sont inaccessibles et ne peuvent faire l'objet d'une expérience en laboratoire.
  Ce problème se pose aussi pour l'infiniment petit. Certains systèmes quantiques sont en effet à ce point petits qu'ils sont inaccessibles à l'expérience. Pour essayer de les comprendre, on tente donc de les simuler sur un ordinateur.
- On peut faire de la simulation numérique lorsque l'on veut étudier le comportement d'un objet qui n'existe pas encore physiquement : la conception du dernier avion d'Airbus, l'A380, a été entièrement réalisée sur ordinateur. Cette simulation a permis de diminuer les coûts de développement de manière significative.

Conception de voitures et d'avions

Aérodynamique, écoulement dans les moteurs, crash tests, commande optimale, structure, pneus, carrosserie,....

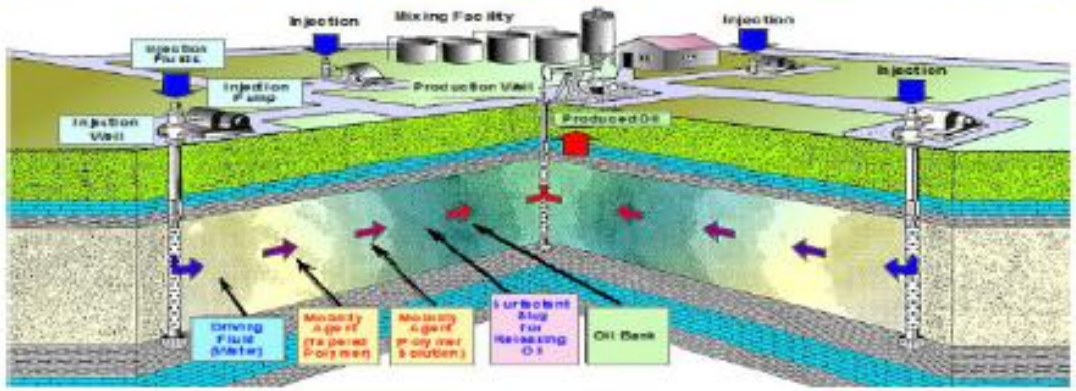
Crash test réel (à gauche), crash test numérique (à droite)



Airbus A380 (L72m,H24m)(à gauche), Simulation numérique (à droite)

Ingénierie pétrolière

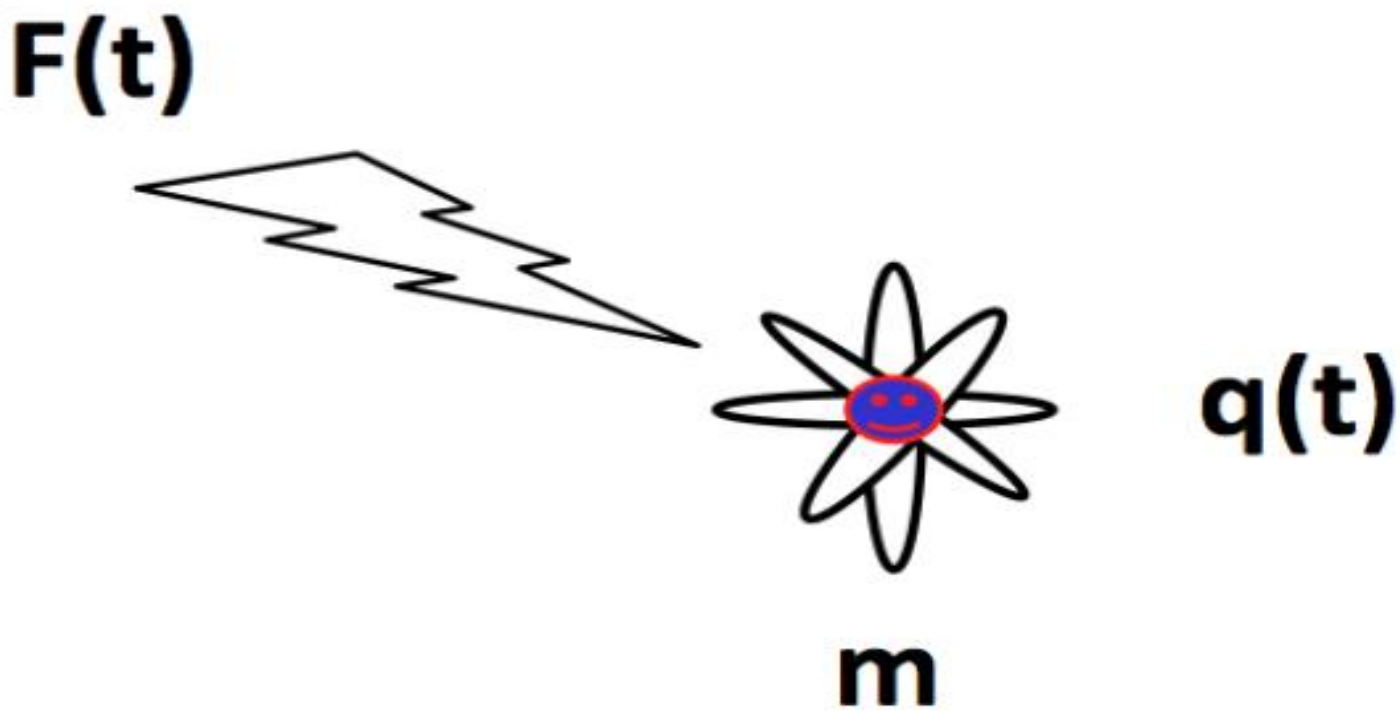
Comprendre la migration des hydrocarbures
Améliorer la production des gisements pétroliers, ...
Récupération secondaire du pétrole. Injecter de l'eau afin de déplacer le pétrole vers les puits de production.



- On peut faire de la simulation numérique lors de la conception de bâtiments et d'infrastructures.
- On peut faire de la simulation numérique pour imaginer des objets abstraits complètement imaginaires, comme des objets fractals ou des systèmes dynamiques.
- On peut aussi faire de la simulation du cerveau. Malheureusement, nous sommes encore loin de comprendre complètement le fonctionnement du cerveau.
- On peut faire de la simulation numérique lorsque la complexité d'un calcul est grande. C'est notamment le cas lorsque l'on souhaite casser un code. Les problèmes de cryptographie pour casser un code supposent en effet de très grandes puissances de calcul.

Partie 1 : Modélisation par l'exemple

La seconde loi de Newton



Exemple 1 : La seconde loi de Newton

La seconde loi de Newton indique que la force exercée sur un corps/un point matériel produit une accélération proportionnelle,

$$mq''(t) = F(t). \tag{1}$$

Ici $q(t)$ représente la position à l'instant t du corps de masse m, $F(t)$ la force qui lui est appliquée.

Exemple 1 : La seconde loi de Newton

La seconde loi de Newton indique que la force exercée sur un corps/un point matériel produit une accélération proportionnelle,

$$mq''(t) = F(t). \tag{1}$$

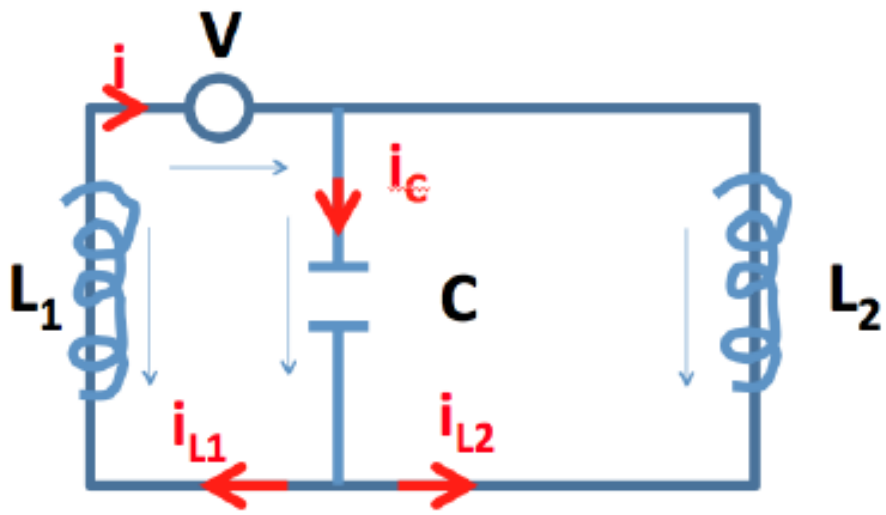
Ici $q(t)$ représente la position à l'instant t du corps de masse m, $F(t)$ la force qui lui est appliquée.

La solution $q(t)$?

$$q(t) = \frac{1}{m}\int_0^t \int_0^s F(r)drds + tC_1 + C_2 \tag{2}$$

C_1 et C_2 sont des constantes arbitraires dans $\mathbb{R}$.

Exemple 2 : Réseau éléctrique



- V désigne la source de tension.
- L_1, L_2 représentent l'inductance des bobines.
- C désigne la capacité du condensateur.

Exemple 2 : Réseau éléctrique

Appliquons les lois de bases en éléctricité :

- le courant I_L et la tension V_L à travers la bobine d'inductance L sont liés par la relation suivante :

$$V_L(t) = L\frac{dI_L}{dt}(t) \tag{3}$$

- le courant I_C et la tension V_C du condensateur dont la capacité est C vérifient

$$I_C(t) = C\frac{dV_C}{dt}(t) \tag{4}$$

Exemple 2 : Réseau éléctrique

La conservation de la charge et de l'énergie sont décrites par les lois de Kirchhoff. La première loi de Kirchhoff énonce que la somme des courants qui entrent par un noeud est égale à la somme des courants qui en sortent. La seconde loi de Kirchhoff dit que dans une maille fermée d'un réseau électrique, la somme des tensions le long de cette maille est toujours nulle. En appliquant ces lois à notre exemple, on obtient le système d'équations différentielles suivant

$$L_1 \frac{dI_{L_1}}{dt}(t) = V_{L_1}(t) = V_C(t) + V(t). \tag{5}$$

$$L_2 \frac{dI_{L_2}}{dt}(t) = V_{L_2}(t) = V_C(t). \tag{6}$$

$$C \frac{dV_C}{dt}(t) = I_C(t) = -I_{L_1}(t) - I_{L_2}(t). \tag{7}$$

Exemple 2 : Réseau éléctrique

On suppose qu'on ne peut mesurer que le courant I_{L_1} , et on pose,

$$y(t) = I_{L_1}(t)$$

La formule (5) est équivalente à :

$$L_1 y^{(1)}(t) = V_c(t) + V(t)$$

En multiplions par C et en différenciant une fois nous obtenons

$$L_1 C y^{(2)}(t) = C V_C^{(1)}(t) + C V^{(1)}(t)$$

D'après (7), il s'en suit

$$L_1 C y^{(2)}(t) = -y(t) - I_{L_2}(t) + C V^{(1)}(t) \tag{8}$$

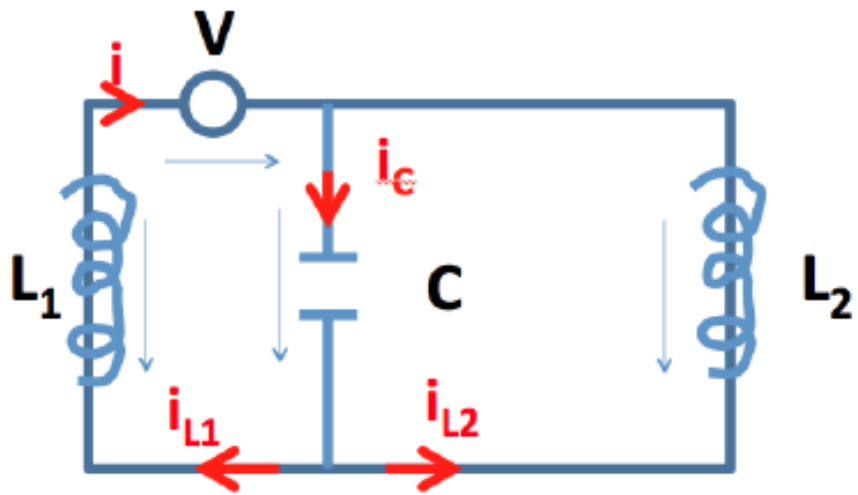
Différencions cette équation une deuxième fois et utilisons (6)

$$L_{(1)} C y^{(3)}(t) = -y^{(1)}(t) - \frac{dI_{L_{(2)}}}{dt}(t) + C V^{(2)}(t) \tag{9}$$

$$= -y^{(1)}(t) - \frac{1}{L_2} V_c(t) + C V^{(2)}(t) \tag{10}$$

$$= -y^{(1)}(t) - \frac{1}{L_2}\left(L_1 y^{(1)}(t) - V(t)\right) + C V^{(2)}(t). \tag{11}$$

Exemple 2 : Réseau éléctrique



L'équation différentielle ordinaire suivante

$$L_1 C y^{(3)}(t) + \left(1 + \frac{L_1}{L_2}\right) y^{(1)}(t) = \frac{1}{L_2} V(t) + C V^{(2)}(t). \qquad (12)$$

décrit le comportement de I_{L_1} à l'intérieur du réseau.
Exercice : Trouver l'expression de $y(t)$.

Exemple 3 : Equation Logistique

La croissance d'une population est habituellement modélisé par une équation de la forme

$$\frac{dP}{dt} = kP$$

où P représente le nombre d'individus à un temps donné t. Ce modèle suppose que le taux de croissance de la population est proportionnelle à la taille de la population.
Une solution de cette équation est la fonction exponentielle :

$$P(t) = Ce^{kt}$$

Remarque : $P(t) \longrightarrow \infty$ **qd** $t \to \infty$ **!!!**

Exemple 3 : Equation Logistique, 1844–1845

Un modèle plus réaliste tient compte du fait que **tout environnement a une capacité de charge limitée** K, donc si P atteint K la population arrête de croître. Le modèle dans ce cas est le suivant :

$$\frac{dP}{dt} = kP\left(1 - \frac{P}{K}\right) \qquad (13)$$

On l'appelle **l'équation différentielle logistique**.

Remarque : C'est une équation différentielle ordinaire non linéaire ! ! !

Exercice : Trouver l'expression de $P(t)$.

Exemple 4 : Modèle proie-prédateur, 1925-1926

Quand des observations montrèrent que la proportion de poissons prédateurs augmentait quand la pêche diminuait, un mathématicien italien, Vito Volterra, conçut un modèle pour décrire le phénomène (peu après la première guerre mondiale). Ce modèle, désormais connu sous le nom d'équations de Lotka-Volterra.

ALFRED JAMES LOTKA

VITO VOLTERRA

Exemple 4 : Modèle proie-prédateur

$$\begin{cases} x'(t) = ax(t) - bx(t)y(t) \\ y'(t) = cx(t)y(t) - dy(t) \end{cases} \qquad (14)$$

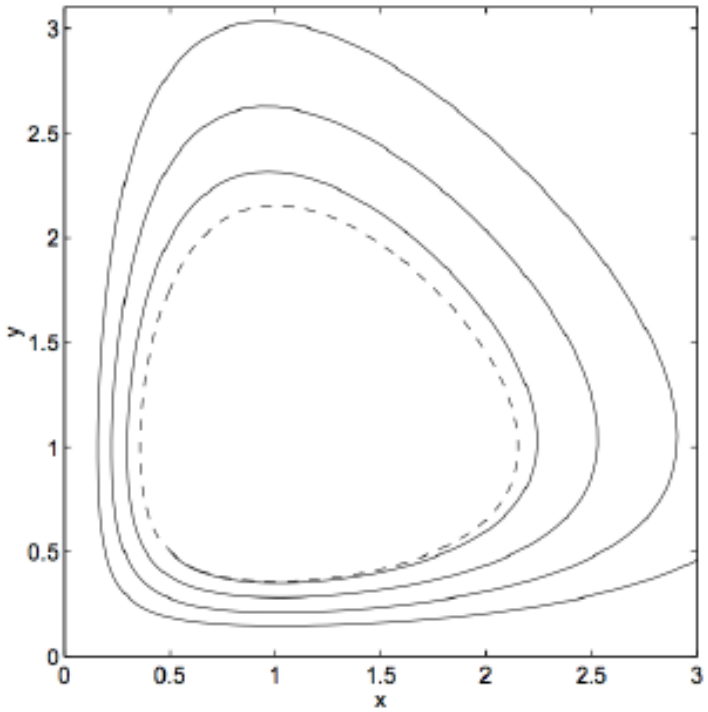
où x est une fonction représentant **l'effectif de la population des proies** à un instant donné, idem avec y pour les **prédateurs**, et a, b, c, d des réels strictement positifs.
Le couplage dans (14) modélise les rencontres entre les deux espèces et leur évolution.

Exemple 4 : Modèle proie-prédateur

Remarque : Un calcul explicite de la solution, ne peut pas s'effectuer.

$$\begin{cases} x'(t) = ax(t) - bx(t)y(t) \\ y'(t) = cx(t)y(t) - dy(t) \end{cases} \tag{15}$$

On doit recourir au **calcul numérique.**
La méthode d'Euler explicite donne le graphe spiral suivant des solutions y en fonction de x, avec $x_0 = 0.5$ et $y_0 = 0.5$.



Equation de Laplace

Pour certains choix du terme source f, la solution de l'équation de la chaleur atteint un état stationnaire, c'est-à-dire que $u(t,x)$ admet une limite $u_\infty(x)$ quand le temps t tend vers l'infini. Souvent, il est intéressant de calculer directement cet état stationnaire. Dans ce cas, pour un terme source $f(x)$ indépendant du temps, on résout une équation du deuxième ordre en espace

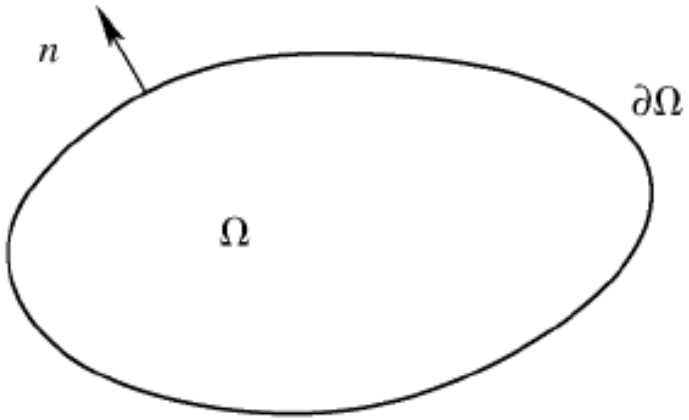
$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ u = 0 & \text{sur } \partial\Omega \end{cases} \tag{19}$$

Remarque :

- C'est une équation elliptique.
- Cette equation est aussi la version stationnaire de l'équation des ondes.
- Cette equation peut modéliser le déplacement vertical d'une membrane élastique soumise à une force normale f et fixée sur son contour.

Equation de la chaleur

L'équation aux dérivées partielles qui modélise le transfert de la chaleur le long de Ω s'appelle **l'équation de la chaleur**



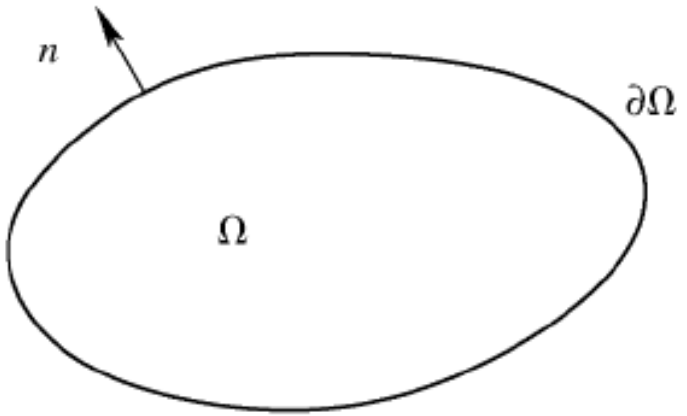
$$\begin{cases} c\dfrac{\partial\theta}{\partial t} - k\Delta\theta = f & \text{pour } (x,t) \in \Omega \times \mathbb{R}_*^+ \\ \theta(t,x) = 0 & \text{pour } (x,t) \in \partial\Omega \times \mathbb{R}_*^+ \\ \theta(t=0,x) = \theta_0(x) & \text{pour } x \in \Omega \end{cases} \tag{16}$$

Remarque :

- C'est une équation parabolique.
- $\theta(t,x)_{|\partial\Omega} = 0$: est la condition aux limites de Dirichlet, i.e. ,le domaine est supposé baigner dans un thermostat à température constante. D'autres conditions aux limites peuvent aussi être envisagées.

Equation de la chaleur

$$\begin{cases} c\dfrac{\partial\theta}{\partial t} - k\Delta\theta = f & \text{pour } (x,t) \in \Omega \times \mathbb{R}_*^+ \\ \theta(t,x) = 0 & \text{pour } (x,t) \in \partial\Omega \times \mathbb{R}_*^+ \\ \theta(t=0,x) = \theta_0(x) & \text{pour } x \in \Omega \end{cases}$$



Remarque :

- $\dfrac{\partial\theta}{\theta n}(t,x)_{|\partial\Omega} = 0$: est la condition aux limites de Neumann, i.e. , le domaine est supposé adiabatique ou thermiquement isolé de l'extérieur $\Longrightarrow$ le flux de chaleur sortant au bord est nul.
- $\dfrac{\partial\theta}{\theta n}(t,x)_{|\partial\Omega} + \alpha\theta(t,x)_{|\partial\Omega} = 0$: est la condition aux limites de Fourier/Robin, i.e. , le flux de chaleur sortant au bord est proportionnel au saut de température entre l'extérieur et l'intérieur.
- On peut imaginer des situations ou les conditions aux limites sont mélangées. Dirichlet sur $\partial\Omega_D$, Neumann sur $\partial\Omega_N$ et Fourier sur $\partial\Omega_F$, avec $\partial\Omega_D, \partial\Omega_N, \partial\Omega_F$ formant une partition de la frontière $\partial\Omega$.

Equation de la chaleur

$$\begin{cases} c\dfrac{\partial\theta}{\partial t} - k\Delta\theta = f & \text{pour } (x,t) \in \Omega \times \mathbb{R}_*^+ \\ \theta(t,x) = 0 & \text{pour } (x,t) \in \partial\Omega \times \mathbb{R}_*^+ \\ \theta(t=0,x) = \theta_0(x) & \text{pour } x \in \Omega \end{cases}$$

Remarque : Ce système n'est pas seulement un modèle de propagation de la chaleur. Il a en fait un caractère universel, et on le retrouve comme modèle de nombreux phénomènes sans aucun rapport entre eux.

- Ce système est aussi connu sous le enom d'**équation d diffusion**, et modélise la diffusion ou migration d'une concentration ou densité à travers le domaine Ω : imaginer un polluant diffusant dans l'atmosphère, ou bien une espèce chimique migrant dans un substrat.
- le modèle de la chaleur intervient aussi en finance et il porte le nom de **modèle de Black and Scholes**.

Modèle de Black and Scholes

C'est une variante du modèle de la chaleur qui permet de trouver le prix de l'option d'achat (ou call) d'une action qui vaut initialement x et qu'on pourra acheter au prix k dans un temps ultérieur T. Ce prix est la solution u de

$$\begin{cases} \dfrac{\partial u}{\partial t} - ru + \dfrac{1}{2r}x\dfrac{\partial u}{\partial x} + \dfrac{1}{2\sigma^2}x^2\dfrac{\partial^2 u}{\partial x^2} = 0 & \text{pour } (x,t) \in \mathbb{R} \times (0,T) \\ u(t=T,x) = \max(x-k,0) = 0 & \text{pour } x \in \mathbb{R} \end{cases} \tag{17}$$

- σ étant la volatilité de l'action.
- r le taux d'intérêt

Equation de la chaleur

$$\frac{\partial \theta}{\partial t} - \alpha\left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2} + \frac{\partial^2 \theta}{\partial z^2}\right) = 0.$$

Une équation similaire a été étudié par Fornasier et Mars en 2007 pour recolorer les anciennes fresques italiennes fragmentées. Voir

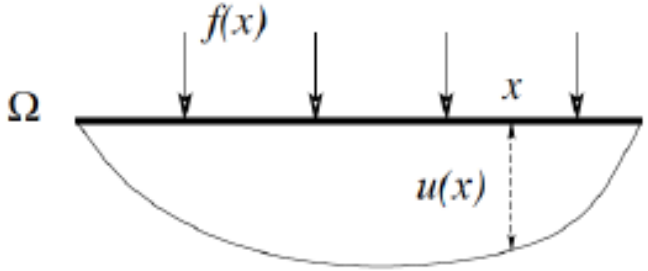
M. Fornasier and R. March. Restoration of color images by vector valued BV functions and variational calculus, SIAM J. Appl. Math., Vol. 68 No. 2, 2007, pp. 437-460.

Equation des ondes ou de D'Alembert

L'équation des ondes modélise des phénomènes de propagation d'ondes ou de vibration. Par exemple, en deux dimensions d'espace elle est un modèle pour étudier les vibrations d'une membrane élastique tendue (comme la peau d'un tambour).

Equation des ondes

En une dimension d'espace, elle est aussi appelée équation des cordes vibrantes. Au repos, la membrane occupe un domaine plan Ω. Sous l'action d'une force normale à ce plan d'intensité f, elle se déforme et son déplacement normal est noté u. On suppose qu'elle est fixée sur son bord, ce qui donne une condition aux limites de Dirichlet.
L'équation des ondes dont u est solution est donnée par



$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} - \Delta u = f & \text{dans } \Omega \times \mathbb{R}_*^+ \\ u = 0 & \text{sur } \partial\Omega \times \mathbb{R}_*^+ \\ u(t=0) = u_0 & \text{dans } \Omega \\ \dfrac{\partial u}{\partial t}(t=0) = u_1 & \text{dans } \Omega \end{cases} \tag{18}$$

Remarque : C'est une équation hyperbolique.

Equation de Schrödinger

L'équation de Schrödinger décrit l'évolution de la fonction d'onde u d'une particule soumise à un potentiel V . Rappelons que $u(t,x)$ est une fonction de $\mathbb{R}^+ \times \mathbb{R}^N$ à valeurs dans $\mathbb{C}$ et que son module au carré $|u|^2$ s'interprète comme la densité de probabilité pour détecter que la particule se trouve au point (t,x). Le potentiel $V(x)$ est une fonction à valeurs réelles. La fonction d'onde u est solution de

$$\begin{cases} i\dfrac{\partial u}{\partial t} + \Delta u - Vu = 0 & \text{dans } \mathbb{R}_*^+ \times \mathbb{R}^N \\ u(t=0) = u_0 & \text{dans } \mathbb{R}^N \end{cases} \tag{20}$$

Remarque : Il n'y a pas de condition aux limites apparentes, puisque l'équation a lieu dans tout l'espace (qui n'a pas de bord).

Equation de Maxwell

$$\begin{cases} rot(E) = -\dfrac{\partial B}{\partial t}, & rot(B) = \mu_0 \epsilon_0 \dfrac{\partial E}{\partial t} \\ div(E) = 0, & div(B) = 0 \end{cases} \tag{21}$$

- E et B étant respectivement le champ électrique et le champ magnétique.
- Ces équations signifient que la variation spatiale du champ électrique donne naissance à un champ magnétique variant dans le temps, et vice versa.

Exercice : Montrer que l'eqn. de Maxwell $\Longrightarrow$ l'éqn. des ondes.

Ce n'est pas tout...il y en a encore ! ! !

La liste n'est pas exhaustive :

- L'équation de transport
- L'équation des plaques
- L'équation de Navier-Stokes
- L'équation de Boltzman
- L'équation de Burger
- **Les systèmes thermoélastiques**
- etc...

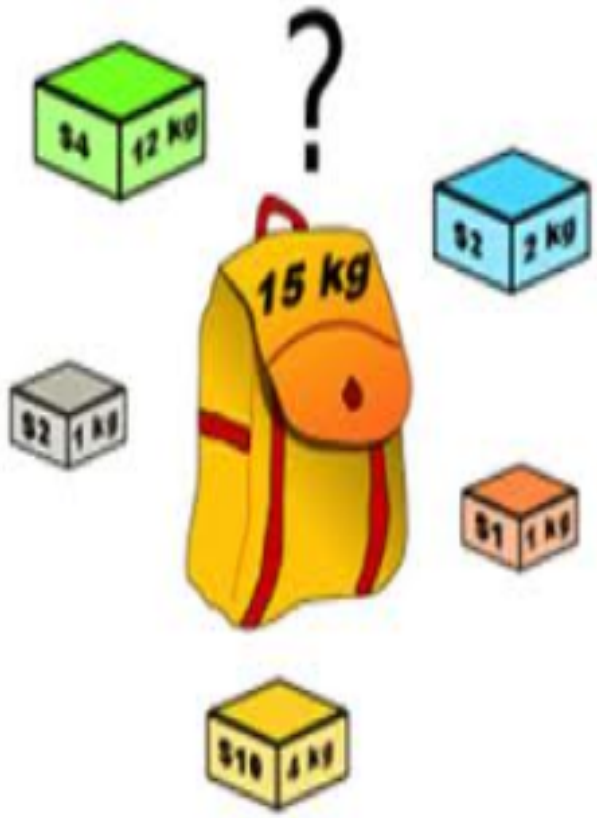
Recherche Opérationnelle

Problème du sac-à-dos.

Soit n objets : bouteille d'eau, lampe, briquet, ...

- $p1$, p_2, ... p_n : poids respectifs des objets
- u_1, u_2, u_n : utilités respectives des objets
- $x_i = 1$ si on met l'objet dans le sac-à-dos
- $x_i = 0$ sinon

Le porteur peut porter un poids maximal qu'on note P.



Formuler le problème :

Maximiser l'utilité du sac-à-dos sous la contrainte de poids.

Modèle mathématique

Chercher le maximum de la fonction :

$$x_1 u_1 + x_2 u_2 + \cdots + x_n u_n$$

telle que

$$x_1 p_1 + x_2 p_2 + \cdots + x_n p_n \leq P.$$

Simulation

Trouver un algorithme pour calculer une solution dans le cas général.

Application directe en gestion des stocks.

Bibliographie

1. Christian Magnan**, Le théorème du jardin**, morceaux choisis.
2. Jean-Louis LE MOIGNE, **La théorie du système général, théorie de la modélisation**, 1994.
3. John M. Henshaw, **Le théorème de la fourmi géante**, Collection: Science à plumes, Berlin, 2016.
4. Gregoire Allaire, **Analyse Numérique et Opitimisation**, Éditions de l'École Polytechnique 2005, 2ème édition (revue et corrigée) en 2012.
5. Gregoire Allaire, **Numerical Analysis and Optimisation**, 2007
6. R.D. Sharma & Umesh kumar, **Basic Applied Mathematics For Physical Science** (Based on the university of Delhi Syllabus), 2007.
7. Sébastien CHARNOZ, **Calcul haute performance : simuler le Monde dans un Ordinateur,** 2014

Des outils pour le calcul scientifique

- **LATEX**: qui permet de concevoir des documents de qualité professionnelle sans connaissances en typographie et mise en page.

https://www-iuem.univ-brest.fr/feiri/soutiens-scientifique/calcul-scientifique/wiki/installation-et-configuration-de-latex-sous-windows

N.B: Il faut suivre à la lettre les instructions du site pour avoir une installation réussite sans bugs.

- **Scilab:** qui est un logiciel de calcul numérique.

Scilab pour TP

C’est quoi Scilab ?

Logiciel libre, développé par l’INRIA (Institut National de Recherche en Informatique et en Automatique)

Usages :

– calcul numérique

– modélisation des problèmes des Sciences de I ’ingénieur (SI)

Proche de Matlab (logiciel privateur).

Fonctionne sur de nombreux systèmes d’exploitation.

Site web: http://www.scialb.org (scilab-6.0.0 (64-bit))

Scilab

Logiciel

Scilab est un logiciel libre de calcul numérique multi-plate-forme fournissant un environnement de calcul pour des applications scientifiques. Il possède un langage de programmation orienté calcul numérique de haut niveau. Wikipédia

Type : Calcul numérique

Licence : CeCILL v2, GPL v2

Dernière version : 6.0.0 (15 février 2017)

Environnements : Linux, Windows, Mac OS X, BSD

Programmé en : C++, Java, Fortran

Mathematica

Logiciel

Mathematica est un logiciel de calcul formel édité par Wolfram Research depuis 1988 et utilisé dans les milieux scientifiques pour effectuer des calculs algébriques et créer des programmes. Wikipédia

Date de sortie initiale : 23 juin 1988

Licence : propriétaire

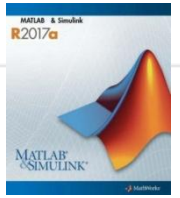
Dernière version : 11.2.0 (septembre 2017)

Type : Logiciel de calcul formel

Programmé en : Wolfram Language, Java

MATLAB

Langage de programmation



MATLAB est un langage de programmation de quatrième génération émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Wikipédia

Conçu Par : Cleve Moler

Première version : 1984

Dernière version : R2017a (9 mars 2017)

Type : Calcul numérique

Environnements : Linux, Unix, Mac OS, Windows

Licence : Propriétaire

GNU Octave

GNU Octave est un logiciel libre de calcul numérique comparable à MATLAB et à Scilab. Ce n'est pas un logiciel de calcul formel. Le logiciel est développé puis maintenu pour le projet GNU par John W. Eaton. Wikipédia

Première version : 1988

Dernière version : 4.2.1 (24 février 2017)

Licence : GNU GPL

Environnement : GNU/Linux, FreeBSD, NetBSD, OpenBSD, Mac OS X, MS Windows

Type : Calcul numérique

Commandes Scilab

1. MANIPULATION DE VECTEURS ET MATRICES

1.1. **Création de matrices.** D'abord quelques briques élémentaires utiles pour construire des choses plus compliquées.

`1:4.5`	nombres compris entre 1 et 4.5 par incrément de 1
`1:1.5:3`	nombres compris entre 1 et 3 par incrément de 1,5
`linspace(a,b,n)`	vecteur constitué de n nombres régulièrement espacés entre a et b
`[]`	matrice vide
`[1,4,3]`	vecteur ligne $(1, 4, 3)$
`[1,2;3,4]`	matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$
`ones(3,4)`	matrice de taille 3×4 remplie de 1
`zeros(3,4)`	matrice de taille 3×4 remplie de 0
`eye(5,4)`	matrice de taille 5×4 avec des 1 sur la diagonale et des 0 ailleurs
`diag(x)`	matrice carrée diagonale de diagonale le vecteur x
`diag(x,3)`	matrice carrée de i-ème diagonale diagonale le vecteur x
`toeplitz(x)`	matrice de toeplitz bâtie sur le vecteur x
`rand(m,n)`	matrice aléatoire (loin uniforme sur $[0, 1]$) de taille $m \times n$

`A*B A^2`	multiplication matricielle et puissance d'une matrice
`A+B, A-B, A.*B, A./B, A.^B`	opérations terme à terme
`A', u'`	conjugué de la transposée d'une matrice, d'un vecteur
`sin(A), exp(A)`	calcule le sinus et l'exponentielle de chaque coef
`size(A)`	nombre de lignes et de colonnes de A
`det(A), rank(A)`	le déterminant de A et son rang
`inv(A)`	l'inverse de A
`sum(u), prod(u)`	la somme et le produit des coefs d'un vecteur
`max(u), min(u)`	le plus grand et le plus petit coef d'un vecteur ou d'une matrice
`[m,xmin]=min(u)`	renvoie le minimum d'un vecteur ainsi que la coordonnée où celui-ci est atteint
`norm(u), norm(u,1)`	les normes ℓ^2 et ℓ^1 d'un vecteur

`spec(A)`	la liste des valeurs propres de A
`[V,l]=spec(A)`	la liste des valeurs propres et des vecteurs propres de A
`bdiag(A)`	la décomposition de Jordan de A

Algèbre linéaire	SCILAB
déterminant de H	`det(H)`
conditionnement de H	`cond(H)`
valeurs propres de H	`spec(H)`
polynôme caractéristique de H	`poly(H,'s')`
inverse de H	`inv(H)`
trace de H	`trace(H)`
rang de H	`rank(H)`
norme de H $\|H\|_{frobenius}$ $\|H\|_{\infty}$ $\|H\|_2$	`norm(H,flag)` si flag=`'fro'` si flag=`'inf'` par défaut
taille de H	`size(H)`

1.3. **Concaténation de tableaux.** Un exemple suffira :
`A=(1;2;3) , B=eye(3,3) , C=ones(1,4), X=[A,B;C]`
donne le résultat

$$A = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} , \; B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} , \; C = (1,1,1,1) , \; X = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

1.4. **Extraction d'une partie d'un tableau.** On rentre d'abord une matrice
`M=[1,2,3;4,5,6;7,8,9]`
dont extrait ensuite différentes parties :

`M(2,3)`	élément $M_{2,3}$
`M(:,1)`	première colonne
`M(1,:)`	première ligne
`M([2,3],:)`	matrice constituée des 2ième et 3ième lignes
`M([1,3],[1,2])`	matrice $\begin{pmatrix} 1 & 2 \\ 7 & 8 \end{pmatrix}$

Et en combinant extraction et affectations :

`M(3,3)=0`	l'élément $M_{3,3}$ de M devient 0, le reste est inchangé
`M(1,:)=2*M(1,:)`	la première ligne est multipliée par 2
`M(:,[1,2])=M(:,[2,1])`	les colonnes 1 et 2 sont échangées
`M($,:)=[]`	remplace la dernière ligne par une ligne vide. La taille de M devient 2×3

2. Quelques fonctions numériques ou vectorielles

Fonctions trigonométriques : `sin`, `cos`, `tan`
Fonction trigonométriques inverses : `asin`, `acos`, `atan`
Exponentielle, logarithme néperien ou décimal : `exp`, `log`, `log10`
Racine carrée : `sqrt`
valeur absolue, partie entière, signe : `abs`, `floor`, `sign`vspace5mm
Somme des coefficients d'un vecteur ou d'une matrice : `sum`
Produit des coefficients d'un vecteur ou d'une matrice : `prod`
Réordonner les coefficients d'un vecteura par ordre décroissant : `sort(a)`. La commande `gsort(a)` fait la même chose, mais offre des arguments supplémentaires.
Norme euclidienne d'un vecteur : `norm`
Déterminant d'une matrice : `det`
Solution du système matriciel $Ax = b$: `A\b`

Syntaxe pour la programmation

`function y=f(x)`
`  y=x.^2;`
`endfunction`
`f(3)`
`f([1;2;3])`
	Fonction qui à une matrice `A` associe la matrice dont les éléments sont les carrés de ceux de `A`

`if (1==2)`
`  a=1;`
`elseif (1==1)`
`  a=2;`
`else`
`  a=0;`
`end;`
	Boucle “if”

`a=0;`
`for i=1:10`
`  a=a+1;`
`end;`
	Boucle “for”

`i=0;`
`while i<>10`
`  i=i+1;`
`end;`
	Boucle “while”

Opérateurs logiques

%T	Le Booléen “vrai”
%F	Le Booléen “faux”
1==2	Teste si $1 = 2$
1<=2	Teste si $1 \leq 2$
1<2	Teste si $1 < 2$
1>=2	Teste si $1 \geq 2$
1>2	Teste si $1 > 2$
1<>1	Teste si $1 \neq 2$
%T&%T	Opérateur Booléen “et”
%F|%T	Opérateur Booléen “ou”

Divers

`%i`	Le nombre complexe i
`%e`	La base e de l'exponentielle
`%pi`	La constante π
`%eps`	Le "zéro machine", valant $2^{-52} \simeq 2.22 \times 10^{-16}$. Il s'agit de la précision relative maximale d'un calcul dans Scilab. Par exemple : `(1+%eps)-1` vaut `%eps`, mais `(1+0.5*%eps)-1` vaut `0`.
`%inf`	L"'infini machine". Désigne tout nombre trop grand ($\geq 2^{1024} \simeq 2 \times 10^{308}$) pour être stocké en mémoire.
`%nan`	La constante `Nan`, pour *not a number*, désigne une quantité non définie, comme par exemple `0*%inf`
`scf(2)`	Définit la fenêtre 2 comme fenêtre graphique courante
`clf()`	Efface la fenêtre graphique courante
`plot2d([0 2 5], [0 1 -1])`	Trace une courbe (fonction de $\mathbb{R}$ dans $\mathbb{R}$)
`plot3d([0 1],[-1 0], [0 1; 0 -1])`	Trace une surface (fonction de $\mathbb{R}^2$ dans $\mathbb{R}$)
`help unecommande`	Aide au sujet de la commande `unecommande`
`apropos toto`	Cherche le mot clé "toto" dans l'aide
`exec('toto.sci')`	Exécute les commandes contenues dans le fichier `toto.sci`
`abort`	Annuler un calcul préalablement interrompu par *Ctrl-C*
`resume`	Relance un calcul préalablement interrompu par *Ctrl-C*

Exercices:

Qu’est ce que ça donne le script suivant ?

```
// un script passionnant
a=input(" Rentrer la valeur de a : ");
b=input(" Rentrer la valeur de b : ");
c=input(" Rentrer la valeur de c : ");
x=[0:.01:1] ;
y=a*x.*x+b*x+c*ones(x);
plot2d(x,y)
```

Exercices:

Interpréter les deux codes suivants.

1.
```
s=0
for i=1:1000
  s=s+1/i^2
end
```

2.
```
timer()
s=0; for i=1:1000000, s=s+i, end ;
temps=timer() // la variable temps contient le temps d'exécution du programme.
```

Exercices:

Find the eigenvalues and eigenvectors of the following matrix by hand:

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

Find the eigenvalues and eigenvectors of the following matrix by hand:

$$B = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}.$$

Can you guess the eigenvalues of the matrix

$$C = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}?$$

Remarque: Les résultats numériques des vecteurs propres diffèrent de ceux calculés analytiquement !!!
Perturbation causée par les erreurs d’arrondis !
Est-ce qu’on peut faire un autre algorithme numérique qui résout ce problème numérique ?

Exercices:

Construire les matrices suivantes en utilisant le moins d'opérations possible.

$$A = \begin{pmatrix} 1 & 2 \\ 9 & 7 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} .$$

Exercice 2 (Manipulation de vecteurs). On définit le vecteur X par `X=[1:0.01:10]`.

a) Mettre dans une variable n la taille du vecteur X.

b) Afficher à l'écran la valeur du troisième élément de X.

c) Créer un vecteur Y qui contient tous les éléments de X en sens inverse.

d) Échanger le cinquième et le septième élément de X.

Exercice 3 (Résolution d'un exercice avec Scilab). On donne les matrices

$$A = \begin{pmatrix} -1 & 1 & 1 & -2 \\ 1 & -1 & -2 & 1 \\ 1 & -2 & -1 & 1 \\ -2 & 1 & 1 & -1 \end{pmatrix}, \quad J = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Les questions qui suivent seront à traiter au maximum à l'aide de Scilab.

a) Calculer J^2 et vérifier que l'on a $AJ = JA$. Que vaut J^{-1} ?

b) Montrer que $(A + 2I_4 + 3J)^2 = 4(A + 2I_4 + 3J)$ et que $A(A + 6J) = -5I_4$ (où I_4 est la matrice identité 4×4). En déduire que A est inversible et donner explicitement A^{-1} .

c) La matrice A est-elle diagonalisable ?

d) Pouvez-vous trouver une expression de A^n ?

Exercice 6 : Construire la matrice de taille 100*100 de la forme suivante.

$$\begin{pmatrix} 2 & 1 & 0 & \cdots & 0 \\ 0 & 2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & 2 \end{pmatrix}$$

Exercice 7 : Calculer la somme des entiers de 1 jusqu'à 100, calculer 100!. Donner une façon de calculer toutes les valeurs du cosinus sur les entiers de 1 à 100 le plus rapidement possible.

Exercice 8 : Calculer les normes ℓ^2 et ℓ^1 d'un vecteur sans passer par la commande `norm`.

Exercice 9 : Résoudre le système linéaire suivant

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

Exercice: Gram Schmidt

1) Dans $\mathbb{R}^3$ muni du produit scalaire canonique, orthonormaliser en suivant le procédé de Schmidt, la famille (u, v, w) avec

$$u = (1, 0, 1), v = (1, 1, 1), w = (-1, 1, 0)$$

2)

a) Énoncer le procédé d'orthonormalisation de Gram-Schmidt.

b) Orthonormaliser la base canonique de $\mathbb{R}_2[X]$ pour le produit scalaire

$$(P, Q) \mapsto \int_{-1}^{1} P(t)Q(t)\,\mathrm{d}t$$

1.1. Programmer la décomposition QR d'une matrice $A \in GL_n(\mathbb{R})$ à l'aide du procédé de Gram-Schmidt.

1.2. Que pensez-vous de l'algorithme de Gram-Schmidt modifié ci-dessous ? Donne-t-il le même résultat ? Quel est son intêret ?

```
Q=A;R=zeros(A);
for k=1:(n-1)
  R(k,k)=norm(Q(:,k));
  if  R(k,k)==0 then  abort; end;
  Q(:,k)=Q(:,k)/R(k,k);
  R(k,k+1:n)=Q(:,k)'*Q(:,k+1:n);
  Q(:,k+1:n)=Q(:,k+1:n)-Q(:,k)*R(k,k+1:n);
end
R(n,n)=norm(Q(:,n));
Q(:,n)=Q(:,n)/R(n,n);
```

1.3. Programmer la décomposition QR d'une matrice $A \in \mathcal{M}_{m,n}(\mathbb{R})$ à l'aide des matrices de Householder.

1.4. Comparer les trois méthodes précédentes (vérifier en particulier l'orthogonalité de la matrice Q obtenue) pour la matrice de Hilbert définie par $H_{i,j} = \dfrac{1}{i+j-1}$, $1 \leq i, j \leq n$ pour $n = 10$.

2. Problème des moindres carrés

Soient n points (x_i, y_i), $1 \leq i \leq n$. Faire un programme qui trouve la droite $y = ax + b$ minimisant $\sum_{i=1}^{n} |y_i - (ax_i + b)|^2$. Prendre un exemple et tracer sur un graphique les points et la droite.

Exercice:

Deux récipients A et B sont séparés par une membrane perméable dans les deux sens. On place dans les récipients A et B deux solutions contenant respectivement a_0 molécules (dans A) et b_0 molécules (dans B). On suppose que, toutes les heures, 20% des molécules passent de A dans B et 10% des molécules passent de B dans A. On note a_n et b_n les nombres respectifs de molécules présentes dans A et B au bout de n heures.

1) Montrer que $\begin{cases} a_{n+1} = 0{,}8a_n + 0{,}1b_n \\ b_{n+1} = 0{,}2a_n + 0{,}9b_n \end{cases}$ et donner l'interprétation matricielle de ce système en considérant la matrice colonne $p_n = \begin{pmatrix} a_n \\ b_n \end{pmatrix}$.

Les deux récipients n'ayant d'échanges qu'entre eux.

2) Sachant que si $a_0 = 150$ et $b_0 = 20$ (unités), quelles instructions écrire pour connaître les quantités de molécules après 10 heures ?

Quelle méthode appliqueriez vous pour connaître la répartition limite, si elle existe, entre les deux milieux ?

3) Quels sont les dosages initiaux nécessaires pour obtenir après 1 heure, une répartition égale à $a_1 = 130$ et $b_1 = 40$ (unités).

Ecrire l'instruction Scilab permettant d'expliciter le résultat.

Exercices

On considère la suite récurrente suivante

$$\begin{cases} u_0 &= 2 \\ u_1 &= -4 \\ u_n &= 111 - \dfrac{1130}{u_{n-1}} + \dfrac{3000}{u_{n-1}u_{n-2}}. \end{cases}$$

1. Montrer que u_n converge vers 6.
2. Montrer que

$$u_n = \frac{\alpha \cdot 100^{n+1} + \beta \cdot 6^{n+1} + \gamma \cdot 5^{n+1}}{\alpha \cdot 100^{n} + \beta \cdot 6^{n} + \gamma \cdot 5^{n}},$$

3. Ecrire un programme Scilab qui affiche u_n et la valeur exacte de u_n .

n	Computed value	Exact value
3	18.5	18.5
4	9.378378378378379	9.3783783783783784
5	7.8011527377521679	7.8011527377521613833
6	7.1544144809753334	7.1544144809752493535
11	6.2744386627644761	6.2744385982163279138
12	6.2186967691620172	6.2186957398023977883
16	6.1661267427176769	6.0947394393336811283
17	7.2356654170119432	6.0777223048472427363
18	22.069559154531031	6.0639403224998087553
19	78.58489258126825	6.0527217610161521934
20	98.350416551346285	6.0435521101892688678
21	99.898626342184102	6.0360318810818567800
22	99.993874441253126	6.0298473250239018567
23	99.999630595494608	6.0247496523668478987
30	99.999999999998948	6.0067860930312057585
31	99.999999999999943	6.0056486887714202679

Exercice: Calcul approché de π

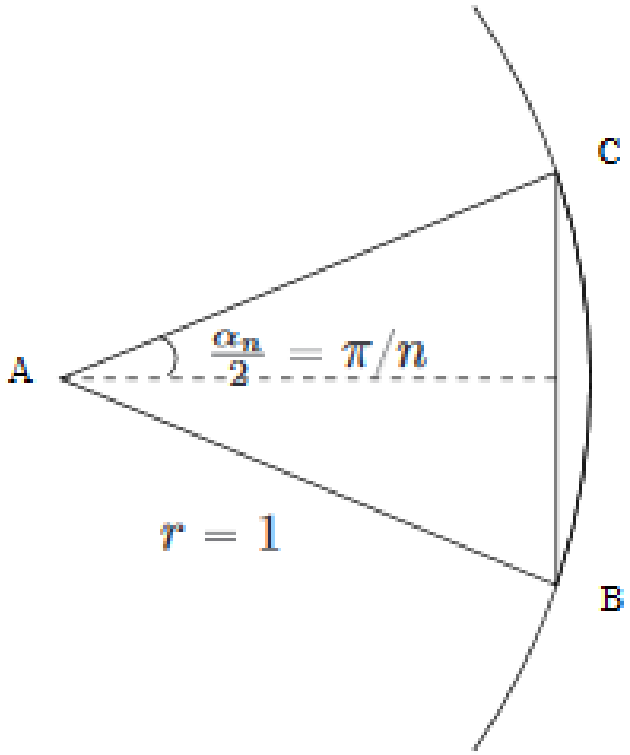
Regardons l'algorithme de calcul par les polygônes inscrits. On considère un cercle de rayon $r = 1$ et on note A_n l'aire associée au polygône inscrit à n côtés. En notant $\alpha_n = \frac{2\pi}{n}$, A_n est égale à n fois l'aire du triangle ABC représenté sur la figure 1.1, c'est-à-dire

$$A_n = n \cos\frac{\alpha_n}{2} \sin\frac{\alpha_n}{2},$$

que l'on peut réécrire

$$A_n = \frac{n}{2}(2\cos\frac{\alpha_n}{2}\sin\frac{\alpha_n}{2}) = \frac{n}{2}\sin\alpha_n = \frac{n}{2}\sin(\frac{2\pi}{n}).$$

Exercice: Calcul approché de π



Exercice: Calcul approché de π

Comme on cherche à calculer π à l'aide de A_n, on ne peut pas utiliser l'expression ci-dessus pour calculer A_n, mais on peut exprimer A_{2n} en fonction de A_n en utilisant la relation

$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\cos\alpha_n}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}.$$

Ainsi, en prenant $n = 2^k$, on définit l'approximation de π par récurrence

$$x_k = A_{2^k} = \frac{2^k}{2}s_k, \quad \text{avec } s_k = \sin(\frac{2\pi}{2^k}) = \sqrt{\frac{1-\sqrt{1-s_{k-1}}}{2}}$$

En partant de $k = 2$ (i.e. $n = 4$ et $s = 1$) on obtient l'algorithme suivant :

Algorithm 1.1 Algorithme de calcul de π, version naïve

```
1: s ← 1, n ← 4                                   ▷ Initialisations
2: Tantque s > 1e − 10 faire                      ▷ Arrêt si s = sin(α) est petit
3:     s ← sqrt((1 − sqrt(1 − s ∗ s))/2           ▷ nouvelle valeur de sin(α/2)
4:     n ← 2 ∗ n                                  ▷ nouvelle valeur de n
5:     A ← (n/2) ∗ s                              ▷ nouvelle valeur de l'aire du polygône
6: fin Tantque
```

Exercice: Calcul approché de π

1. Montrer que x_n converge vers π.
2. Ecrire un code Scilab qui approche la valeur de π.
3. Conclure.
4. Modifier votre code, en considérant la formule ci-après. Conclure.

$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}} = \sqrt{\frac{1-(1-\sin^2\alpha_n)}{2(1+\sqrt{1-\sin^2\alpha_n})}} = \frac{\sin\alpha_n}{\sqrt{2(1+\sqrt{1+\sin^2\alpha_n})}}.$$

Références

1. Sébastien CHARNOZ, **Calcul haute performance : simuler le Monde dans un ordinateur, 2013-2014.**
2. Sébastien CHARNOZ : https://www.dailymotion.com/video/x1dsu5z