Université du Littoral
Licence de Physique

Physique numérique

analyse et présentation des données, programmation

Dmitrií Sadovskií
Département de physique, Université du Littoral
e-mail: sadovski@univ-littoral.fr, web: http://pca3.univ-littoral.fr/~sadovski
Anton Sokolov
Département de physique, Université du Littoral
e-mail: Anton.Sokolov@univ-littoral.fr

Calais, automne 2015

Horaires

cours-TD-TP 13 séances de 3h

Contenu des cours

Ce module est destiné aux étudiants de la troisième année en licence sciences avec option physique à Calais. Il aura les thèmes principales suivantes.

1. introductions, notions de base sur l'architecture des ordinateurs, systèmes d'exploitation, structures de représentation des données, etc
2. solution des systèmes des équations linéaires, méthode de Gauss. L'aspect pratique : développer une programme (TP-1) en utilisant le logiciel `MapleV.3`.
3. traitement des données et notions essentielles de statistique telles, que distribution Gaussienne, écart standard, corrélations, intervalle de confiance, etc.
4. méthode des moindres carrées (mmc). Estimation des incertitudes. L'aspect pratique : on utilise les formules statistiques et la MMC pour la présentation des résultats des TP possiblement aidée par l'informatique. On applique la mmc pour modéliser les données sur les masses atomiques (TP-2), voir cours de Prof. Boris Zhilinskií.
5. problèmes de stabilité, problèmes mal définis ou insuffisants, décomposition selon les valeurs singulières (svd)
6. équations nonlinéaires, méthodes itératives (itération de Gauss, TP-3), linéarisation suivie par une itération mmc
7. solution numérique des équations différentielles ordinaires, intégrateurs, méthode Runge-Kutta, problème de Cauchy. L'aspect pratique : TP-4, équations de mouvement dans le potentiel Lennard-Jones (en collaboration avec Prof. Hadj Sarahoui et autres).
8. transformation de Fourier directe (DFT), inverse, rapide (FFT). Filtrage. L'aspect pratique : écrire une programme DFT (TP-6) ; analyser la solution des équations de mouvement obtenue précédemment.
9. calcul matriciel : diagonalisation, orthogonalisation, inversion. TP-7 : diagonalisation des matrices symétriques par la méthode de Jacobi.

Modalités de contrôles des connaissances

En TD les étudiants passent au tableau. Les TP ont lieu en salle des ordinateurs ; les étudiants doivent y posséder un compte avec l'accès au logiciel `MapleV.3`. Les comptes rendus de TP sont notés sur 5 (soit 20 points max) ; ils sont dus à la fin de la séance TP, ou, exceptionnellement, au prochain TP. Ils représentent essentiellement les texte des programmes

en `MapleV.3` avec des commentaires imprimés en TP. Il n'y a pas de partiel écrit au mi-cours. A sa place on a un projet commun avec l'enseignant du cours de vibrations. Ce projet est présenté en oral et noté sur 20. L'examen est noté sur 20, la note finale est définie avec le règle du sup contre la moyenne projet+TP.

Bilan des compétences

La théorie sera vraiment de base, accessible à tous, qui essayent de travailler sérieusement. L'idée est d'apprendre et maîtriser surtout l'aspect physique d'application des formules et notions expliqués en cours et TD.

1 Introduction

L'idée du cours. Le rôle organisateur de la thème «équations linéaires». Le projet de mi-parcours.

Ressources et bibliographie sur Maple

1. *Vérsion de démonstration de* `MapleV.3` sur le site de *MapleSoft*, la société qui édite Maple : `ftp://ftp.maplesoft.com/pub` puis suivre selon le système d'exploitation que l'on utilise.
2. *Maple V. First leaves : A Tutorial Introduction to Maple V.* ISBN : 3540976213, Waterloo Maple. 1992 (en anglais)
3. *Poly sur Maple* par Raphaël Giromini `http://usenet.alea.net/entraide/rg/maple/poly.pdf`
4. *Quelques feuilles de Maple — Maple expliqué en français* par Jean Charles Canonne (Université de Valenciennes) `http://perso.orange.fr/jeancharles.canonne/sommaire1.html`
5. *Notes de cours sur Maple 6, 7 et 8 (janvier 2004) en français* par Claude Gomez (INRIA, Rocquencourt) `http://www-rocq.inr`
6. *Computational Physics using Maple* par Marko Horbatsch (York University) `http://www.yorku.ca/marko/ComPhys`

Ressources et bibliographie sur Maxima

1. *Aide-Mémoire pour Maxima* `http://michel.gosse.free.fr/documentation/fichiers/aidememoireaxima.pdf`
2. *Introduction à Maxima* par H. Hand, traduite par M. Gosse `http://michel.gosse.free.fr/documentation/fichiers/i`
3. *The Maxima Book* par De Souza, Fateman, Moses, Yapp `http://michel.gosse.free.fr/documentation/fichiers/max`

2 Architecture des ordinateurs

3 Statistiques, analyse des données

3.1 Erreurs systématiques et accidentelles

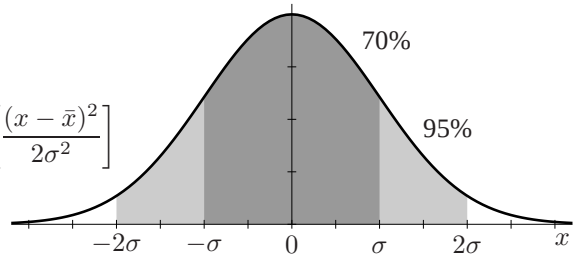
Systématiques : Ces erreurs sont indépendantes de l'opérateur et échappent en partie son contrôle. Comme leur nom l'indique, elles faussent la mesure toujours dans même sens mais on le ne connaît pas. Elles sont souvent difficiles à déceler (défauts d'appareillage, mauvaise méthode ...), chaque cas constituant un problème spécifique.

Accidentelles : Ces erreurs sont dites aussi *aléatoires* car elles faussent la mesure tantôt dans une sens tantôt dans l'autre, et n'ont pas un caractère répétitif. On peut les minimiser en refaisant un nombre raisonnable des fois la même mesure, en prenant la moyenne des résultats obtenus et en appliquant des méthodes statistiques.

3.2 Présentation statistique des résultats

Soit une grandeur x à mesurer. Dans le cas le plus fréquent, quand l'incertitude sur x est une somme de nombreuses contributions faibles et indépendantes, x a pour la densité de probabilité une distribution gaussienne

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{(x-\bar{x})^2}{2\sigma^2}\right]$$



avec *valeur moyenne* $\bar{x}$ et *écart type* (ainsi appelé dispersion ou variance) σ. On effectue un *nombre fini* N de mesures

$$x_i, \quad i = 1, 2, \ldots, N.$$

On prendra

$$x_e = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.1}$$

pour *valeur moyenne expérimentale*. Évidemment, $x_e \to \bar{x}$ quand $N \to \infty$. Notons que x_e est elle-même une quantité aléatoire avec sa propre moyenne et son écart type

$$\bar{x}_e = \bar{x}, \quad \sigma^2_{x_e} = \sigma^2/N.$$

En réalité, ni $\bar{x}$ ni σ ne sont pas connus ! Alors on prendra

$$\sigma_e^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - x_e)^2,$$

pour la *dispersion expérimentale*. Notons, que pour $N = 1$ la σ_e ne peut pas, évidemment, être déterminée, d'où le facteur $1/(N-1)$. Par contre, si $N \to \infty$, nous avons $\sigma_e \to \sigma$. Ce que nous intéresse, c'est l'écart type σ_{x_e}. Lui est donnée par

$$s^2_{x_e} = \frac{1}{N}\sigma_e^2 = \frac{1}{N(N-1)}\sum_{i=1}^{N}(x_i - x_e)^2. \tag{3.2}$$

Dans la limite de grands N nous avons

$$x_e \to \bar{x}, \quad s_{x_e} \to \sigma/\sqrt{N}.$$

Le résultat de la mesure s'écrit :

$$x = x_e \pm \delta x, \quad \text{avec} \quad \delta x = t_{PN}\, s_{x_e} \approx t_P\, \sigma/\sqrt{N}$$

On constate que l'incertitude s_{x_e} de valeur mesurée x_e décroît avec N (améliore) en $1/\sqrt{N}$. Par exemple, pour gagner un facteur de 10 sur δx_e il nous faut effectuer 100 fois plus de mesures ... . Le facteur correctif t_{PN} appelé *coefficient de Student* dépend de deux paramètres, le nombre de mesures N et la probabilité P avec laquelle la vraie valeur de $x - x_e$ se trouve dans l'intervalle $[-\delta x, +\delta x]$ dit *intervalle de confiance*. Pour $N \to \infty$ ce facteur tend vers le facteur de sécurité t_P pour la distribution gaussienne (voir tableau).

valeurs de coefficient correctif de Student t_{PN}

probabilité	nombre de mesures N					
P	3	5	6	10	∞	
0.70	1.3	1.2	1.2	1.1	1.0	
0.95	4.3	2.8	2.6	2.3	2.0	(règle de 2σ)
0.99	9.9	4.6	4.0	3.3	2.6	

Exercice 3.1. Prenons un exemple concret. On mesure plusieurs fois la concentration x.

x_1	x_2	x_3	x_4	x_5	x_6
72.361	72.357	72.352	72.346	72.344	72.340

Prenez trois valeurs quelconques ($N = 3$), et toutes les valeurs ($N = 6$). Trouvez la valeur moyenne x_e et la dispersion dans les deux cas. Chiffrez la valeur de l'incertitude δx pour l'intervalle de confiance de 70% et 95%. Comparez et interprétez vos résultats.

Écriture de résultat : Il est inutile d'énoncer une incertitude avec plus que deux chiffres significatifs. Le résultat ne peut avoir plus des chiffres significatifs, que le nombre imposé par son incertitude.

3.3 Combinaison des incertitudes

Souvent nous ne pouvons pas mesurer directement la quantité y en question. On mesure donc un nombre K des quantités $(\xi_1, \xi_2, \ldots, \xi_K)$ afin de trouver y comme une fonction de $(\xi_1, \xi_2, \ldots, \xi_K)$. Supposons, que $(\xi_1, \xi_2, \ldots, \xi_K)$ sont variables aléatoires indépendantes, chacune possédant une distribution de probabilité gaussienne $p_k(\xi_k)$ avec une moyenne $\bar{\xi}_k$ et un écart type σ_k, $k = 1, \ldots, K$.

Quelle est la moyenne $\bar{y}$ et son écart type σ_y ? Considérons d'abord quelques exemples typiques. Soit $K = 2$ et

$$y = a\xi_1 \pm b\xi_2, \tag{3.3a}$$

avec des constantes a et b. Évidemment

$$\bar{y} = a\bar{\xi}_1 \pm b\bar{\xi}_2. \tag{3.3b}$$

On trouve pour l'écart type de y, que

$$(\sigma_y)^2 = a^2\sigma_1^2 + b^2\sigma_2^2, \quad \sigma_y \le |a|\sigma_1 + |b|\sigma_2, \quad \Delta y \le |a|\Delta\xi_1 + |b|\Delta\xi_2. \tag{3.3c}$$

Soit $K = 1$ et y est une fonction lisse f quelconque de ξ_1. Dans ce cas on obtient

$$y = f(\xi_1), \quad \bar{y} = f(\bar{\xi}_1), \quad \sigma_y^2 = \left(f'(\bar{\xi}_1)\right)^2 \sigma_1^2, \quad \text{à condition que } f'(\bar{\xi}_1) = \left.\frac{df}{d\xi_1}\right|_{\bar{\xi}_1} \neq 0. \tag{3.4}$$

En particulier si $f = \log$ [et donc $f'(\xi) = 1/\xi$] nous avons

$$y = \log(\xi_1), \quad \bar{y} = \log\bar{\xi}_1, \quad \sigma_y^2 = \frac{\sigma_1^2}{\bar{\xi}_1^2}, \quad \Delta y = \frac{\Delta\xi_1}{\bar{\xi}_1}. \tag{3.5}$$

Exercice 3.2. En utilisant les règles (3.3c) et (3.5) montrerons le *règle des écarts relatifs*

$$y = \xi_1^a\xi_2^b \text{ ou } y = \frac{\xi_1^a}{\xi_2^b}, \quad \left(\frac{\sigma_y}{\bar{y}}\right)^2 = a^2\left(\frac{\sigma_1}{\bar{\xi}_1}\right)^2 + b^2\left(\frac{\sigma_2}{\bar{\xi}_2}\right)^2 \tag{3.6}$$

Exercice 3.3. On a mesuré la période $T = 2 \times 10^{-3}$ sec d'un signal avec une incertitude relative $\Delta T/T$ de 2%. Quelle est l'incertitude de la fréquence de ce signal ?

Donnons enfin le règle général.

$$y = f(\xi_1, \ldots, \xi_K), \quad \sigma_y^2 = \sum_{k=1}^{K}\left(\left.\frac{\partial f}{\partial\xi_k}\right|_{\bar{\xi}}\right)^2\sigma_k^2, \quad \sigma_y \le \sum_{k=1}^{K}\left|\left.\frac{\partial f}{\partial\xi_k}\right|_{\bar{\xi}}\right|\sigma_k \tag{3.7}$$

où les dérivées *partielles* $\partial f/\partial\xi_k$ sont calculées dans $\xi_k = \bar{\xi}_k$, $k = 1, \ldots, K$.

Exercice 3.4. Considérez le circuit électrique du diviseur de tension (voir section "Électricité"), avec deux résistances en série de valeurs R_1 et R_2, et la source de f.e.m. $\mathcal{E}$. Trouvez la tension U_2 comme fonction de R_1, R_2, et $\mathcal{E}$. En sachant, que les incertitudes relatives de R_1, R_2, et $\mathcal{E}$ sont de 5%, 10%, et 2%, trouvez l'incertitude relative $\Delta U_2/U_2$. Appliquez vos résultats dans le cas $\mathcal{E} = 10$V, $R_1 = R_2$.

Exemples des combinaisons des incertitudes.

1. En spectrophotomètrie (voir TP) on mesure la transmission ξ d'un échantillon par rapport à la transmission ξ_0 d'une cuve remplie de l'eau pure. On effectue donc *deux* mesures et prend la différence $\xi_0 - \xi$. Dans certaines cas, si l'intensité absolue peut être mesurée, on normalise $1 - \xi/\xi_0$.
2. Souvent pour déterminer la concentration c_A d'une substance A on doit passer par plusieurs étapes de purification, séparation, modification chimique, etc. À chaque étape on ne peut pas contrôler les pertes de A. Alors, on utilise la méthode du témoignage. On ajoute une substance B en concentration c_B connue d'office. Car A et B subissent les mêmes transformations en cours de l'analyse, on assume que les pertes de A et B sont les mêmes. On détermine les concentrations ξ_A et ξ_B après toutes les modifications (par la méthode chromatographique, mass–spectrométrique, spectrophotométrique, etc), et on trouve $c_A = c_B(\xi_A/\xi_B)$.

Démonstration. (*Tous démonstrations sont facultatives.*) Notons, que dans les conditions précisées au début de cette section, la fonction de densité de probabilité pour y est un produit des fonctions individuelles $p(\xi_k)$,

$$p(\xi_1, \dots, \xi_K) = p(\xi_1)p(\xi_2)\cdots p(\xi_K) = \prod_k p(\xi_k),$$

normalisée par l'intégrale

$$\int \cdots \int p(\xi_1, \dots, \xi_K)d\xi_1 \dots d\xi_K = \int p(\xi_1)d\xi_1 \int p(\xi_2)d\xi_2 \cdots \int p(\xi_K)d\xi_K = 1 \cdot 1 \cdots 1 = 1.$$

Rappelons aussi, que la valeur moyenne $\bar f$ d'une fonction $f(x)$ d'une variable aléatoire x avec la densité de probabilité $p(x)$ est donnée par l'intégrale $\bar f = \int f(x)p(x)dx$. Dans le cas (3.3) nous avons

$$\bar y = \int\int y(\xi_1, \xi_2)d\xi_2 d\xi_1 = a \int \xi_1 d\xi_1 \int d\xi_2 + b \int d\xi_1 \int \xi_2 d\xi_2 = a\bar\xi_1 \cdot 1 + 1 \cdot b\bar\xi_2.$$

De même façon, on calcule la moyenne de $(y - \bar y)^2$:

$$\overline{(y-\bar y)^2} = \int\int (y-\bar y)^2 d\xi_2 d\xi_1 = \int\int \big[a^2(\xi_1 - \bar\xi_1)^2 + b^2(\xi_2 - \bar\xi_2)^2 \pm 2ab(\xi_1 - \bar\xi_1)(\xi_2 - \bar\xi_2)\big]d\xi_2 d\xi_1$$

Les deux premiers termes donnent les deux termes de (3.3c), le troisième terme est 0 parce que

$$\int\int (\xi_1 - \bar\xi_1)(\xi_2 - \bar\xi_2)d\xi_2 d\xi_1 = \int (\xi_1 - \bar\xi_1)d\xi_1 \int (\xi_2 - \bar\xi_2)d\xi_2 = 0 \cdot 0.$$

Dans le cas de (3.4) tenez compte du fait, que la déviation $|\xi_1 - \bar\xi_1|$ est petite et faites développement limité de $f(\xi_1)$ dans le point $\bar\xi_1$,

$$f(\xi_1) \approx f(\bar\xi_1) + f'(\bar\xi_1)(\xi_1 - \bar\xi_1),$$

d'où on obtient immédiatement

$$y - \bar y = f(\xi_1) - f(\bar\xi_1) \approx f'(\bar\xi_1)(\xi_1 - \bar\xi_1).$$

Il vous ne reste qu'à prendre la moyenne

$$\overline{(y-\bar y)^2} = f'(\bar\xi_1)\overline{(\xi_1 - \bar\xi_1)^2}.$$

□

3.4 Méthode des moindres carrés, cas de régression linéaire

On mesure y_i pour chaque valeur x_i, $i = 1, \dots, N$ de la *variable du contrôle* x. Pour les valeurs y_i on estime ces incertitudes Δy_i (ou ces écart types σ_i). Par exemple, on effectue plusieurs mesures de chaque y_i et trouve Δy_i, voir sec. 3.2. Dans le cas de *régression* on suppose, que les valeurs de x sont "exactes". [En réalité, nous n'avons que $\Delta x_i \ll \Delta y_i$.] Le phénomène est décrit par la loi théorique $y = f(x;\alpha)$, où $\alpha = \alpha_1, \alpha_2, \dots$ sont des paramètres phénoménologiques. On cherche alors de déterminer α, et, bien sûr, les incertitudes $\Delta\alpha$. Bien entendu, le nombre des mesures N doit être plus important que le nombre des paramètres α.

L'idée centrale de la méthode[1] c'est de trouver telles valeurs des paramètres α, que la déviation moyenne quadratique entre la théorie et l'expérience est au minimum. On cherche donc à minimiser

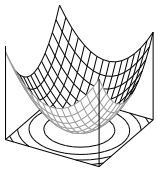
$$F(\alpha) = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i, \alpha)}{\sigma_i} \right)^2 .$$

Ici les écart types σ_i (= les incertitudes à un facteur près) de chaque mesure (x_i, y_i) jouent le rôle des "poids" : les mesures avec plus précises sont prises en compte plus. Dans le cas *linéaire* où

$$y = f(x) = a_1 x + a_0, \quad \alpha = \{a_1, a_0\}$$

on a *deux* paramètres, et on cherche

$$\min_{a_1,a_0} F(a_1, a_0) = \min_{a_1,a_0} \sum_{i=1}^{N} \left(\frac{y_i - (a_1 x_i + a_0)}{\sigma_i} \right)^2 .$$



En introduisant la notation pour les "moyennes pondérées"[2]

$$[g] := \frac{1}{S} \sum_{i=1}^{N} \frac{g_i}{\sigma_i^2}, \quad \text{où} \quad S = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}, \quad \text{et} \quad g_i = x_i, y_i, x_i y_i, x_i^2, \tag{3.8a}$$

on exprime les valeurs de a_1 et a_0,

$$a_1 = \frac{[xy] - [x][y]}{[xx] - [x][x]}, \quad a_0 = [y] - [x]\, a_1, \tag{3.8b}$$

qui minimisent $F(a_1, a_0)$. Pour les écart types de deux paramètres on trouve

$$\sigma_{a_1}^2 = \frac{1}{S} \frac{1}{[xx] - [x][x]}, \quad \sigma_{a_0}^2 = \sigma_{a_1}^2 [xx]. \tag{3.8c}$$

Notez, que dans le cas simplifié $\sigma_i \approx \sigma$ pour tous i on trouve, que

$$\frac{1}{S} \approx \frac{\sigma^2}{N}.$$

Par conséquence, les incertitudes Δa_1 et Δa_0 décroissent comme $1/\sqrt{N}$. Une fois de plus on retrouve un résultat fondamental que nous avons déjà obtenu dans la sec. 3.2.

Démonstration. Dans le minimum de $F(a_1, a_0)$ on a $dF = 0$. On obtient le système de deux équations linéaires avec variables (a_1, a_0)

$$\begin{cases} \dfrac{\partial F}{\partial a_1} = 0 \\ \dfrac{\partial F}{\partial a_1} = 0 \end{cases} \Rightarrow \begin{cases} \displaystyle\sum_{i=1}^{N} x_i (y_i - (a_1 x_i + a_0)) \sigma_i^{-2} = 0 \\ \displaystyle\sum_{i=1}^{N} (y_i - (a_1 x_i + a_0)) \sigma_i^{-2} = 0 \end{cases} \Rightarrow \begin{cases} [xy] - a_1 [xx] - a_0 [x] = 0 \\ [y] - a_1 [x] - a_0 = 0 \end{cases}$$

dont la solution est donnée par (3.8b). □

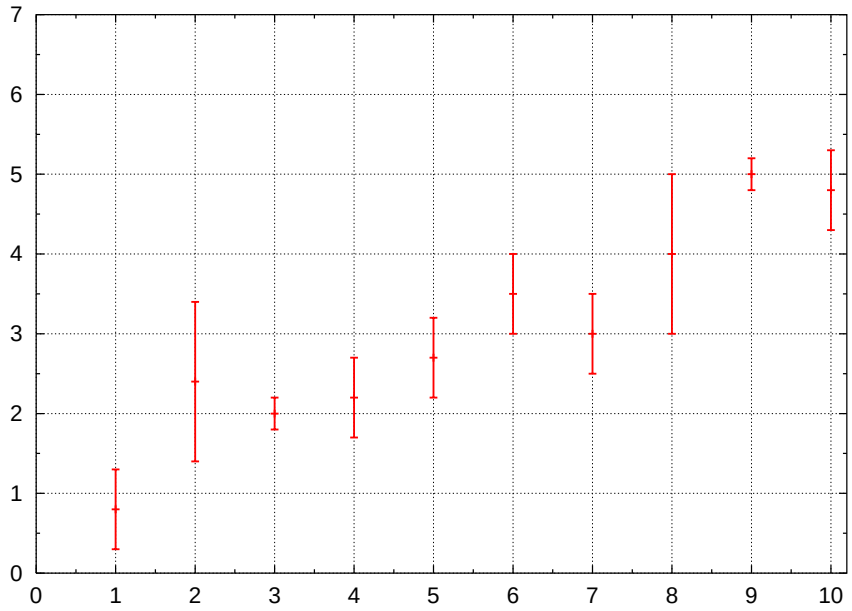
[1] en terminologie anglaise : *least squares fit*

[2] Notez, que dans le cas simplifié, où tous les σ_i sont les mêmes, $[g]$ devient la valeur moyenne $\bar{g}$.

Exemples d'application de la régression linéaire.

1. Détermination de la résistance interne ρ d'une source de tension (voir TP GBF—Schémas équivalents). On mesure la tension U aux bornes d'une résistance variable R et le courant I traversant cette résistance à l'aide d'un voltmètre et un ampèremètre. Par la théorème de Thevenin $U = U_0 - \rho I$, où U_0 est la tension "à vide" de la source (voir section "Électricité").
2. Mesure des distances à l'aide d'une résistance variable R (voir TP Capteur de déplacement). On mis R sous tension ; la distance x est une fonction linéaire de la tension V mesurée entre la borne amovible de R et la masse. On trouve la droite de calibration $x(V)$.
3. Mesure de la température en utilisant une résistance à platine (Pt) ou une thermistance (voir TP Capteurs thermiques). Dans le cas de Pt, la résistance augmente linéairement avec la température T° ; pour la thermistance la loi est exponentielle et on utilise donc l'échelle logarithmique. Dans les deux cas on obtient le paramètres de la droite de calibration $R(T^\circ)$.
4. Spectrophotomètrie (voir TP). Détermination des concentrations en utilisant la loi de Beer–Lambert (en échelle logarithmique). On construit la courbe de calibration pour l'intensité $\mathcal{I}$ de la lumière absorbée en fonction de concentration x et/ou de la longueur de la cuve l.

Exercice 3.5. Analysez un exemple concret de 10 points expérimentaux présentés sur la figure ci-dessous.



Le valeurs (y_i, x_i) et les écart types correspondants sont suivants.

$x_i = i$	1	2	3	4	5	6	7	8	9	10
y_i	0.8	2.4	2.0	2.2	2.7	3.5	3.0	4.0	5.0	4.8
σ_i	0.5	1.0	0.2	0.5	0.5	0.5	0.5	1.0	0.2	0.5

En supposant que la loi théorique $y = f(x)$ est une loi linéaire $y = a_1 x + a_0$, trouvez les valeurs des paramètres a_1 et a_0 et leurs incertitudes Δa_1 et Δa_0.

Cas général de régression, forme matricielle. Soit $f(x,\alpha)$ une fonction C^∞ de x et de paramètres $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)^T$ et $\alpha^{(0)}$ une valeur quelconque des paramètres, dite *valeur de départ*. Le n-vecteur de gradient

$$\nabla_\alpha f = \left(\frac{\partial f(x,\alpha)}{\partial \alpha_1}, \frac{\partial f(x,\alpha)}{\partial \alpha_2}, \cdots \frac{\partial f(x,\alpha)}{\partial \alpha_n} \right)$$

donne la linéarisation de cette fonction dans $\alpha^{(0)}$. Dans chaque point (x_i, y_i) on calcule

$$A_{ij}^{(0)} := \frac{\partial f(x_i, \alpha^{(0)})}{\partial \alpha_j}, \quad i = 1 \dots N \quad \text{et} \quad j = 1 \dots n.$$

On voit que afin d'approximer les données (x_i, y_i), il faudra résoudre un système de N équations linéaires pour trouver n variables $a_j = \alpha_j - \alpha_j^{(0)}$ (avec typiquement $n \ll N$). En forme matricielle on a

$$A\,a = y, \quad \text{ou graphiquement} \quad \boxed{A}\Big|_a = \Big|_y \tag{3.9a}$$

où A est une matrice $n \times N$, a est vecteur des inconnus de dimension n, et y est vecteur des données de dimension N.

Avant de resoudre (3.9a), on le modifie pour tenir compte de différences en incertitudes des données y_i. En d'autres termes, on emploie la m.m.c. pondérée. Rappelons, que dans le cas le plus général, les données sont caractérisées par la matrice de *covariance* (ou covariation en anglais) σ, une matrice symétrique de dimension $N \times N$ dont les éléments diagonaux σ_{ii} reprsentent l'écart type (la dispersion ou variance) des données y_i respectives, tandis que les éléments non-diagonaux $\sigma_{ii'}$ caractérisent corrélations entre y_i et $y_{i'}$. Notons qu'en pratique σ est rarement connue. Elle est souvent supposée d'être diagonale ($\sigma_{ii'} = 0$ pour tous $i \neq i'$), c'est à dire, on assume que pour tous $i \neq i'$ aucune correlation entre les mesures de y_i et de $y_{i'}$ existe. Dans le cas encore plus simplifié, on suppose $\sigma_{ii} = s$ pour tous i. On obtient l'équation pondérée en multipliant (3.9a) par σ^{-1} à gauche,

$$\sigma^{-1} A\,a = \sigma^{-1} y. \tag{3.9b}$$

Ainsi les données dont la variance est moins importante aurons plus de poids dans la somme F_α.

Afin de resoudre (3.9b), on le transforme en multipliant à gauche par A^T (matrice A transposée). Cela donne

$$S\,a = (A^T \sigma^{-1} A)\,a = A^T \sigma^{-1}\, y = b, \tag{3.9c}$$

ou graphiquement

$$\boxed{S}\Big|_a = \boxed{A^T}\boxed{\sigma^{-1}}\boxed{A}\Big|_a = \boxed{A^T}\boxed{\sigma^{-1}}\Big|_y = \Big|_b .$$

Au présent S est une matrice carrée de dimension $n \times n$ et a et b sont vecteurs de dimension n. On a donc à résoudre un système de n équations pour n inconnus. Si bien sur $\det S \neq 0$ (et en pratique on demande $\det S \gg 0$) la solution est

$$a = S^{-1}\, b = S^{-1} A^T \sigma^{-1}\, y, \tag{3.9d}$$

où les éléments $(S^{-1})_{jj'}$ de la matrice S^{-1} donnent la covariance des paramètres a_j et $a_{j'}$. L'écart type de a_j étant $(S^{-1})_{jj}$ on définit la matrice de correlation C dont les éléments sont

$$C_{jj'} = \frac{(S^{-1})_{jj'}}{\sqrt{(S^{-1})_{jj}\,(S^{-1})_{j'j'}}}$$

Exercice 3.6. Dans le cas $n = 2$ avec $f(x;\alpha) = \alpha_1 + \alpha_2 x$, vérifiez que la solution a trouvée ainsi dans (3.9d) correspond à celle trouvée dans (3.8b).

Exercice 3.7. Démontrez plus généralement, que la solution (3.9d) conforme au principe de la m.m.c. Pour cela exprimez $F(\alpha)$ en utilisant les matrices A et σ et donnez les équations pour trouver $\min_\alpha F(\alpha)$.

Iteration m.m.c. Dans le cas de où $f(x,\alpha)$ est une fonction linéaire de paramètres $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)^T$, la solution a dans (3.9d) est finale. (Dans ce cas, la valeur de départ $\alpha^{(0)}$ peut être choisie librement, par exemple $\alpha^{(0)} = 0$.) Autrement, on obtient

$$\alpha^{(1)} = \alpha^{(0)} + a,$$

calcule de nouveau la matrice $A = A^{(1)}$ et repete la procédure m.m.c. Évidement, nul garantit la convergence. Elle dépend largement du choix du départ $\alpha^{(0)}$ et des correlations entre les paramètres.

TP-1 Solution des systèmes des équations linéaires

But de TP. Ecrire et tester une procédure de langage `MapleV.3` permettant la solution d'un système d'équations linéaires par la méthode de Gauss sans choix d'élément pivotant.

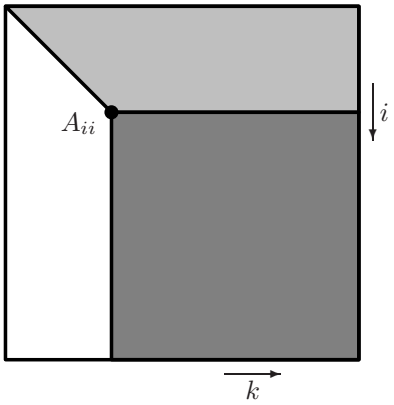
Notions de base. On cherche à obtenir une solution $x = (x_1, \ldots, x_n)$ d'un système à n équations linéaires écrit sous la forme matricielle suivante

$$A\,x = b,$$

où A est une matrice $n \times n$, x est le vecteur des inconnus, et b est un vecteur constant de dimension n. On suppose que $\det A \neq 0$ et donc la solution unique $A^{-1}\,b$ existe.

Pour chercher x on utilise une procédure standard connue comme la méthode de Gauss. Selon cette méthode, on cherche d'abord à transformer $A \to A'$ par la procédure de l'élimination (de colonnes) en forme triangulaire supérieure avec les entrées sur la diagonale principale égales à 1. En même temps b est transformé en b'. Par la suite on résout l'équation $A'\,x = b'$ en utilisant la substitution inverse (*back substitution*) en commençant par la dernière composante de x. Pour simplifier les choses, dans ce TP on ne fait aucun choix d'élément pivotant.

Rappelez vous que l'étape d'élimination Gaussienne s'achève en n pas. A chaque pas $i = 1, \ldots, n$ on descend sur ligne i de la matrice A. En ce moment, comme illustré ci-dessous, tous les éléments A_{ik} de la matrice avec $k < i$ sont déjà éliminés, c'est à dire ils sont égaux à 0.



Par ailleurs, sur toutes lignes $j > i$ les éléments A_{jk} avec $k < i$, ainsi que sur toutes lignes $j < i$ les éléments A_{jk} avec $k < j$ sont aussi tous nuls. En plus tous éléments diagonaux A_{kk} avec $k < i$ sont égaux à 1.

Car on ne fait aucun choix de l'élément pivotant, on prends $p = A_{ii}$. *Il est donc important que $p \neq 0$; si c'est n'est pas le cas, prévoir d'abandonner la procédure !* On divise tous les éléments A_{ik} avec $k = i, \ldots, n$ ainsi que b_i par p :

$$A_{ik} \to A_{ik}/p, \quad k = 1, \ldots, n, \quad b_i \to b_i/p.$$

Si $i = n$ il n'y a rien d'autre à faire, l'étape de l'élimination se termine. Autrement on procède à éliminer la colonne $k = i$ dans tous lignes $j > i$, c'est à dire au-dessous de ligne i (voir le carreau gris foncé dans la figure). Alors, de chaque ligne $j = i{+}1, \ldots, n$ on soustrait ligne i multipliée par $r = A_{ji}$ et on corrige également la partie droite b_j :

$$A_{jk} \to A_{jk} - r\,A_{ik}, \quad k = i, \ldots, n, \quad b_j \to b_j - r\,b_i.$$

(Notez que cela nécessitera une boucle double, une boucle extérieure en j et une boucle intérieure en k ; notez aussi que on est à l'intérieur de la boucle en i, laquelle est la boucle principale de l'élimination.)

Une fois la phase de l'élimination est terminée, on procède avec la substitution inverse en utilisant A' et b'. Évidemment, car tous les éléments A_{nk} avec $k < n$ sont nuls et $A_{nn} = 1$, on obtient $x_n = b'_n$. On commence donc par l'avant dernière ligne $n - 1$. Sur chaque ligne $i = n{-}1, \ldots, 1$ on utilise les composants x_k avec $k > i$ déjà trouvées précédemment pour chercher x_i et on remplace b_i par x_i :

$$b_i \to b_i - \sum_{k=i+1}^{n} b_k\,A_{ik}, \quad i = n{-}1, \ldots, 1.$$

Quelques conseils de programmation en `MapleV.3`. Afin d'accéder aux certaines opérations matricielles communes, chargez la bibliothèque standard d'algèbre linéaire

```
with(linalg):
```

Notez que si `A` est du type `matrix` (elle représente donc matrice A) les éléments A_{ik} sont accédés comme `A[i,k]`. Ainsi, pour imprimer les éléments $i = 1, \ldots, n$ de colonne k on utilise la boucle suivante

```
for i from 1 to n do
  print(A[i,k]);
od;
```

L'avantage d'utiliser les objets du type `matrix` ou `vector` (par rapport aux listes) est dans la possibilité de changer les valeurs de leurs éléments individuellement par une simple attribution, par exemple

```
for k from i to n do
  A[i,k]:=A[i,k]/p;   # rescale i-th row
od;
b[i]:=b[i]/p;         # rescale the rhs entry
```

Notez que ici le dièse # (dit *hash*) précède les commentaires. Vous trouverez ce type d'objets dans les langages les plus anciens, tels comme `FORTRAN` ou `BASIC`, où ils sont appelés `ARRAY`. Vous pouvez imposer l'usage de ces objets dans la définition de votre procédure de façon suivante :

```
# simple in-place Gauss elimination code without pivoting for Maple V.3
gausselim := proc(A:matrix, b:vector)
  local n,i,j,k,p,r;
  n:=rowdim(A);       # dimension of the matrix

  . . .


  RETURN(evalm(b));
end:
```

Notez aussi la déclaration des variables locales à votre procédure. Ces variables ne sont pas visibles ni accessibles en dehors de la procédure où les mêmes noms peuvent servir à d'autres choses. Utilisez `rowdim` pour trouver la dimension de A. Enfin, utilisez `evalm` pour retourner le vecteur de solution b par sa valeur et pas par son nom. Il vous sera aussi peut être nécessaire de utiliser `evalf` afin de convertir vos résultats finales et intermédiaires en nombres flottants. Dans ce cas utilisez `map` afin d'appliquer `evalf` à chaque élément d'une matrice ou d'un vecteur, par exemple

```
map(evalf,b);
```

Enfin, pour tester votre procédure faites

```
A := matrix([[1,2,3],[7,8,9],[4,6,5]]); A1:=copy(A):
b := vector([1,2,3]);                   b1:=copy(b):
gausselim(A1,b1);
```

Notez que notre procédure `gausselim` détruit la matrice A et le vecteur b initiaux et les remplace par la matrice en forme triangulaire supérieure et le vecteur des solutions. Pour préserver A et b, il est impératif de faire leur copies. Cependant, juste l'opération simple `A1:=A;` ne suffit pas, car contrairement aux autres objets (variables simples, listes), les matrices (`matrix`), vecteurs (`vector`), et tableaux (`table`) ne sont pas attribués en `MapleV.3` par leur valeur mais par leur nom. Afin de les attribuer par leur valeur, on utilise `copy(A)` ou `evalm(A)`. Vous pouvez donc utiliser `A` et `b` pour un teste ultime

```
linsolve(A,b);
```

et comparer avec la solution retournée par votre `gausselim`. (Voir la description de la procédure standard `linsolve` dans la bibliothèque `linalg`.)

Questions. Quel est le défaut principal de votre programme ? Votre programme, est-il stable numériquement ? Pourquoi ? Quelle est le rôle d'élément pivotant p ? Quelles sont les modifications nécessaires afin de pouvoir choisir l'élément pivotant ?

Texte du programme :

```
# simple in-place Gauss elimination code without pivoting for Maple V.3
with(linalg):
# NB: to preserve A and b in Maple V.3, use copy(A) and copy(b)
gausselim := proc(A:matrix, b:vector)
    local n,i,j,k,p,r;
    n:=rowdim(A);              # dimension of the matrix
    for i from 1 to n do
        p:=A[i,i];             # pivoting element
        if p=0 then break; fi:  # bail out to avoid division by 0
        for k from i to n do
            A[i,k]:=A[i,k]/p;   # rescale i-th row
        od;
        b[i]:=b[i]/p;          # rescale the rhs entry
        if printlevel>2 then print(i,p,row(A,i)) fi;
        if i=n then break fi;   # stop with no rows left to eliminate
        for j from i+1 to n do  # eliminate i-th column from all rows j>i
            r:=A[j,i];
            for k from i to n do
                A[j,k] := A[j,k]-A[i,k]*r;
            od;
            b[j]:=b[j]-b[i]*r;  # subtract from the j-th rhs entry
        od;
    od;
    if printlevel>1 then print(n,A,b) fi;
    for j from 1 to n-1 do      # back substitute, store the solution in b
        i:=n-j;
        for k from i+1 to n do
            b[i]:=b[i]-b[k]*A[i,k];
        od;
    od;
    RETURN(evalm(b));
end:
```

TP-2 Métode des moindres carrées

Considerez les données[3] sur les masses atomiques m des isotopes en fonction de leur charge Z et du nombre des protons p et des neutrons n (voir tableau). On définit la masse effective

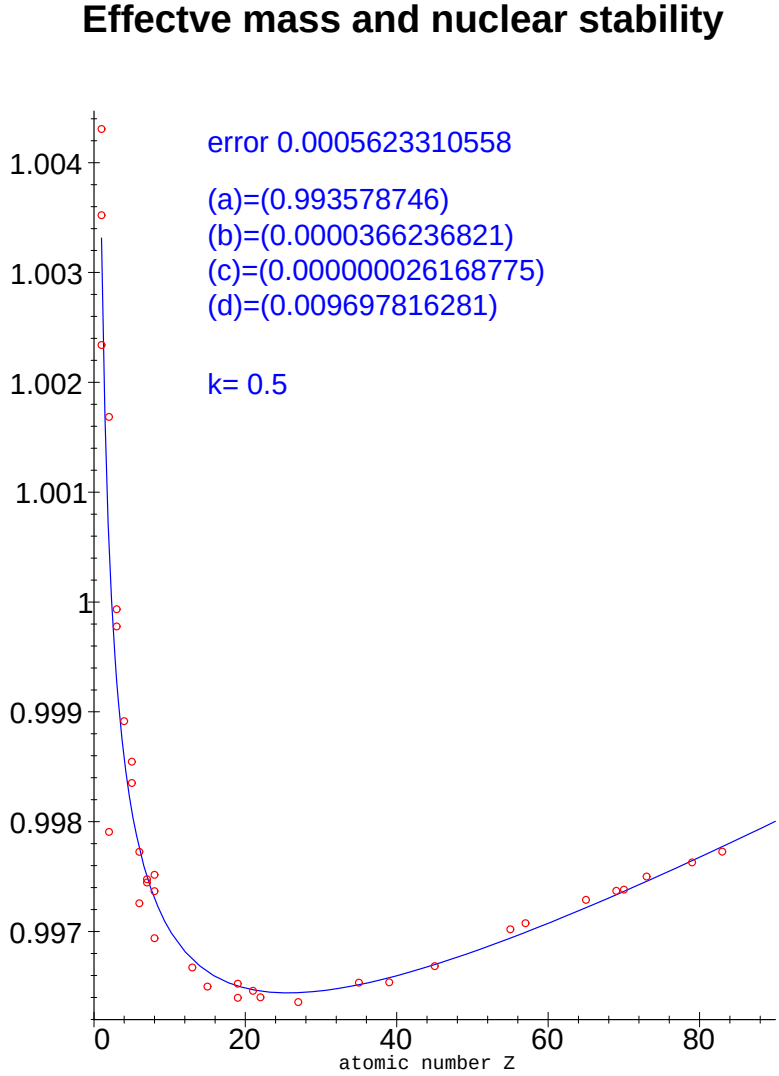
$$M_{\text{eff}} := \frac{m - m_e\, Z}{n + p}$$

du noyau (corrigée par la masse des électrons $m_e = 5.485799 \times 10^{-4}$ u.a.) et on cherche de l'approximer avec une fonction empirique de Z et des paramètres $\alpha = [a, b, c, d]$ et K,

$$M_{\text{eff}}(Z) = a + b\, Z + c\, Z^2 + \frac{d}{Z^K}.$$

Au premier temps on utilise la méthode linéaire et on fixe K à 0.5. Développez un programme de la méthode des moindres carrées et obtenez les valeurs de paramètres. Représentez vos résultats. Comparez «théorie» et «expérience» textuellement et graphiquement (voir la figure ci-dessous). Quel élément est le plus stable[4] ?

Calculez les matrices de covariation S et de correlation C. En supposant que l'écart type des données est $\sigma = 10^{-4}$ u.a., trouvez l'écart type des paramètres α. Présentez et interprétez vos résultats.



Enfin proposez et réalisez une procédure itérative pour optimiser la valeur de K.

[3] voir «Atomic weights and isotopic compositions» [http://physics.nist.gov/PhysRefData/Compositions/] par J. S. Coursey, D. J. Schwab, & R. A. Dragoset, NIST, Physics Laboratory, Office of Electronic Commerce in Scientific and Engineering Data

[4] pour l'interprétation physique, voir le matériel du cours de physique atomique sur la stabilité des noyaux et les deux sources de l'énergie nucléaire

Texte du programme :

```
# example of the nonlinear least square fit [DS, 11 Oct 2005]
# dependence of the effective atomic mass Meff on the charge (atomic number) Z
with(linalg):

# mass of the electron (all masses in a.u.)
m_e := 5.485799*10e-4;

# definition of the effective mass
Meff := (Z,n,m)->(m-m_e*Z)/n :

atomic_data := [
# [ 0, 'n' ,  1,  1.00866 ],
  [ 1, 'H' ,  1,  1.007825],
  [ 1, 'D' ,  2,  2.01410],
  [ 1, 'T' ,  3,  3.01605],
  [ 2, 'He',  3,  3.01603],
  [ 2, 'He',  4,  4.00260],
  [ 3, 'Li',  6,  6.01512],
  [ 3, 'Li',  7,  7.01600],
  [ 4, 'Be',  9,  9.01218],
  [ 5, 'B' , 10, 10.0129 ],
  [ 5, 'B' , 11, 11.0093 ],
  [ 6, 'C' , 12, 12.00000],
  [ 6, 'C' , 13, 13.00335],
  [ 7, 'N' , 14, 14.00307],
  [ 7, 'N' , 15, 15.00011],
  [ 8, 'O' , 16, 15.99491],
  [ 8, 'O' , 17, 16.99913],
  [ 8, 'O' , 18, 17.99917],
  [13, 'Al', 27, 26.9815 ],
  [15, 'P' , 31, 30.97376],
  [19, 'K' , 39, 38.9637 ],
  [19, 'K' , 41, 40.9618 ],
  [21, 'Sc', 45, 44.95591],
  [22, 'Ti', 48, 47.94795],
  [27, 'Co', 59, 58.93320],
  [35, 'Br', 79, 78.9183 ],
  [39, 'Y' , 89, 88.90586],
  [45, 'Rh',103,102.90550],
  [55, 'Cs',133,132.90543],
  [57, 'La',139,138.90636],
  [65, 'Tb',159,158.92535],
  [69, 'Tm',169,168.93423],
  [70, 'Yb',171,170.9363 ],
  [73, 'Ta',181,180.94801],
  [79, 'Au',197,196.96656],
  [83, 'Bi',209,208.98039],
  [92, 'U' ,238,238.05079],
  [94, 'Pu',238,238.04956]
]:
# estimated uncertainty of the data
sigma := 0.0001;

# approximating empirical function
Kval := 0.5;
Feff := Z -> a + b*Z + c*Z^2/2 + d/Z^Kval :

S := 0:
Svars:=[a,b,c,d]:
for t in atomic_data do
  tm:= Meff(t[1],t[3],t[4]);
  S := S + (Feff(t[1]) - tm)^2;
od:
S := evalf(expand(S));
dS:=convert(grad(S,Svars),list);
```

```
# determine parameters
ss:=solve(convert(dS,set),indets(S));

# compute the covariation matrix
M := inverse(convert([seq(map((t,v)->coeff(v,t),Svars,i),i=dS)],matrix));
dM:= [seq(sqrt(M[i,i]),i=1..nops(Svars))];
Mc:= diag(100$nops(Svars)):
for i from 1 to nops(Svars) do
  for j from 1 to i-1 do
    Mc[i,j] := round(M[i,j] / (dM[i]*dM[j]) * 100);
  od:
od:

subs(ss,Svars);
map (t->t*sigma,dM);
print(Mc);

for i from 1 to nops(Svars) do
  printf(‘%s %12.5e [%9.3e]\n‘,
         convert(Svars[i],string),subs(ss,Svars[i]),dM[i]*sigma);
od:

# present results theory vs experiment
for t in atomic_data do
  tm:= Meff(t[1],t[3],t[4]);
  printf(‘%3i %2s[%3i] %10.6f %7.5f (%5.0f)\n‘,
         t[1],t[2],t[3],t[4],tm, (subs(ss,Feff(t[1])) - tm)*1e4
        );
od:
s0 := sqrt(subs(ss,S)/nops(atomic_data)):
printf(‘  mean square deviation %7.5f for %i points\n‘,s0,nops(atomic_data));

plotsetup(ps,plotoutput=‘atomicdata.ps‘,plotoptions=‘color,portrait,noborder‘);

plots[display](
[
  plot(map( t->[t[1],Meff(t[1],t[3],t[4])], atomic_data ),
       ‘atomic number Z‘=0..90,
       color=red, style=POINT, symbol=CIRCLE),
  plot(subs(ss,Feff(x)), x=1..90, color=blue, style=LINE, thickness=2),
  plots[textplot]([15,1.0042,cat(‘error ‘,convert(s0,string))],color=blue,align=RIGHT),
  plots[textplot]([15,1.002,cat(‘k= ‘,convert(Kval,string))],color=blue,align=RIGHT),
  seq(plots[textplot]([15,1.004-i*.00032,convert(Svars[i]=subs(ss,Svars[i]),string)],
                      color=blue,align=RIGHT),
      i=1..nops(Svars))
],
       font=[HELVETICA,18],
       axesfont=[HELVETICA,18],
       labelfont=[HELVETICA,BOLD,22],
       titlefont=[HELVETICA,BOLD,24],
       title=‘Effectve mass and nuclear stability‘);
```

TP-3 Méthodes itératives. Itération de Newton, étude de covergence

Les méthodes itératives, dont celle de Newton (1643–1727) est la plus célèbre, sont employées pour résoudre les équations *nonlinéaires* de forme générale $f(x) = 0$. On note qu'en particulier, ces équations apparaissent dans la recherche locale des extremums d'une fonction $F(x)$ pour laquelle $f = dF/dx$, et que par conséquence, ces méthodes s'appliquent dans des problèmes d'optimization. *Deux règles générales s'imposent pour éviter les pièges* : bien choisir le point de départ x_0 et connaître l'intervalle $[x_{\min}, x_{\max}]$ où se trouve la solution.

TP-3.1 L'approche de Newton dans une dimension

Vers 1669 *Newton* avait proposé cette méthode afin de trouver la racine d'un polynôme cubique $f(x) = x^3 + 6x^2 + 10x - 1$. Plus généralement, considerez $f : \mathbb{R} \to \mathbb{R} : x \mapsto f(x)$ et supposez que ses dérivées f' et f'' existent au moins dans un intervalle $[x_{\min}, x_{\max}]$. En utilisant le développement limité jusqu'à premier ordre[5]

$$f(x + \Delta) = f(x) + f'(x)\,\Delta + O(\Delta^2),$$

c'est à dire, en utilisant l'approximation linéaire au voisinage du point x, nous cherchons[6] la racine dans $x + \Delta$:

$$f(x + \Delta)0 \approx f(x) + f'(x)\,\Delta \quad \Rightarrow \quad \Delta = -f(x)/f'(x).$$

Alors, on choisit une valeur de départ x_0 et on obtient une séquence x_k, $k = 1, 2, 3, \ldots$ avec

$$x_{k+1} = x_k - f(x_k)/f'(x_k). \tag{TP-3.1}$$

Cela représente *l'itération Newtonienne*.

Exercice TP-3.1. En prenant $x_0 = 2$ trouvez la racine $x^3 + 6x^2 + 10x - 1$ en utilisant la méthode de Newton. Fait le d'abord à la main, puis à l'aide de `MapleV.3`. Caractérisez la convergence de la méthode. Proposez un critère d'arrêt.

TP-3.2 Généralisation aux plusieurs dimensions

Au présent, considerons l'application

$$\vec{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} : \mathbb{R}^n \to \mathbb{R}^n : \vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto \vec{f}(\vec{x}) = \begin{pmatrix} f_1(\vec{x}) \\ \vdots \\ f_n(\vec{x}) \end{pmatrix}$$

et supposons que $\partial f_i/\partial x_i$ ainsi que tout les deuxèmes dérivées existent dans un domain $U \ni x_0$ de $\mathbb{R}^n$. Rappelez que dans ce cas le développement limité nous donne[7]

$$\vec{f}(\vec{x} + \vec{\Delta}) = \vec{f}(\vec{x}) + \mathsf{J}(\vec{x})\,\vec{\Delta} + O(\|\vec{\Delta}\|^2) \quad \text{où } \mathsf{J}(\vec{x}) = \begin{pmatrix} \nabla_x f_1(\vec{x}) \\ \vdots \\ \nabla_x f_n(\vec{x}) \end{pmatrix} \text{ et } \nabla_x f_i(\vec{x}) = \big(\partial f_i/\partial x_1, \ldots, \partial f_i/\partial x_n\big).$$

L'itération de Newton (TP-3.1) prend la forme suivante[7]

$$\vec{x}_{k+1} = \vec{x}_k - \mathsf{J}^{-1}(\vec{x}_k)\,\vec{f}(x_k). \tag{TP-3.2}$$

Notons, qu'une approche similaire basée sur la linéairization de la fonction $f(x, \alpha)$ est utilisée dans le cadré de la m.m.c. (voir sec. 3.4 et TP-2) afin d'optimiser les valeurs de paramètres α dont f dépend de façone nonlinéaire.

Exercice TP-3.2. Considerez le cas avec $n = 2$ et trouvez la racine de

$$f(x) = \begin{pmatrix} x_1^3 - 3x_1x_2^2 - 1 \\ 3x_1^2x_2 - x_2^3 \end{pmatrix}$$

au voisinage de point $x_0 = (-0.6, 0.6)^T$ en utilisant la méthode de Newton. Fait le d'abord à la main, puis à l'aide de `MapleV.3`. Caractérisez la convergence de la méthode. Proposez un ou plusieurs critères d'arrêt. Enfin considerez si ce problème corresponde à une recherche (locale) d'un extremum d'une fonction $F : \mathbb{R}^2 \to \mathbb{R} : (x_1, x_2) \mapsto F(x_1, x_2)$.

---

[5] Dans cet ordre, il s'agit tout simplement de la formule de *Leibnitz*, i.e., de la définition de f' ; pour les degrés plus élevés, c'est une série de *Taylor*.

[6] Évidemment en plus d'existence des dérivées nous devons demander que $f'(x) \neq 0$ pour tout $x \in [x_{\min}, x_{\max}]$!

[7] Ici le *Jacobian* J est une matrice carrée de dimension $n \times n$, et le *gradient* ∇f est un vecteur-ligne de dimension n, i.e., chaque ligne de J est le gradient d'une composante de f. Notez aussi que afin de pouvoir invertir J dans tout le domaine U nous devons supposer $\det \mathsf{J}(x) \neq 0$ pour tout $x \in U$, i.e., l'absence des *points critiques* de f, comparez avec le cas d'une dimension dans la note 6.

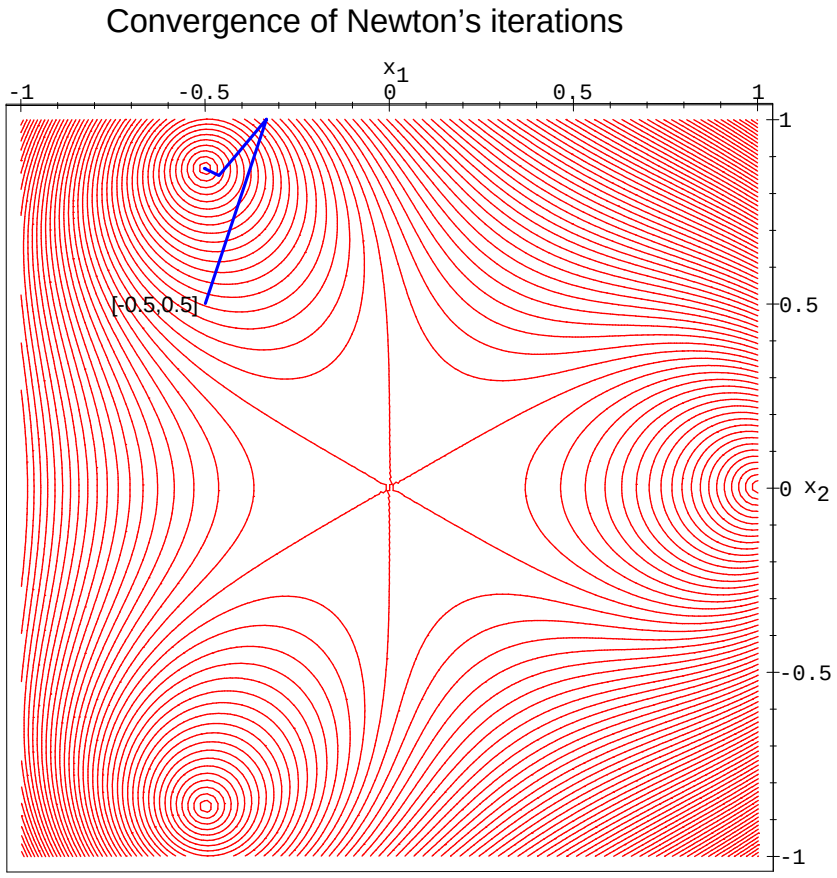
Solution de l'exercice TP-3.2 Soit $\xi = \vec{x} = (x_1, x_2)^T \in \mathbb{R}^2$. On compute :

$$\mathsf{J}(\xi) = 3\begin{pmatrix} x_1^2 - x_2^2 & -2\,x_1x_2 \\ 2\,x_1x_2 & x_1^2 - x_2^2 \end{pmatrix}, \quad \det \mathsf{J}(\xi) = \big(3(x_1^2 + x_2^2)\big)^2 = 9\|\xi\|^4, \quad \mathsf{J}^{-1}(\xi) = \frac{3}{\det \mathsf{J}(\xi)}\begin{pmatrix} x_1^2 - x_2^2 & 2\,x_1x_2 \\ -2\,x_1x_2 & x_1^2 - x_2^2 \end{pmatrix}.$$

On voit que le seul point critique de l'application $f(\xi)$ de l'exercice TP-3.2 est $\xi = (0,0)$. Par conséquent, nous devons choisir $U \subseteq \mathbb{R}^2 \setminus \{(0,0)\}$. En pratique, il vaut mieux de s'éloigner de $(0,0)$ mais de ne pas aller trop vers larges $\|\xi\| \gg 1$ car les trois racines de $f(\xi) = 0$ sont $(-1/2, \pm\sqrt{3}/2)$ et $(1,0)$. Par exemple, on pourrait travailler dans un anneau $U = \{0.1 < \|\xi\| < 2\}$. Ainsi, en choisisant $\xi^{(0)} = (-\frac{1}{2}, \frac{1}{2})$ comme le point de départ, nous obtenons pour les itérations Newtoniennes successives

k	$\xi^{(k)}$	$f(\xi^{(k)})$	$\|f(\xi^{(k)})\|$
0	$[.5, .5]$		
1	$[-.3333333333, 1.000000000]$	$[-.037037037, -.6666666667]$	.6676946807
2	$[-.4622222222, .8466666666]$	$[-.1047291957, -.064258864]$	.1228715022
3	$[-.5018742425, .8657762898]$	$[.002156538, .005250658]$	.0056762720
4	$[-.5000009238, .8660219539]$	$[-.000007578, .000007575]$	.0000107147
5	$[-.5000000000, .8660254039]$	$[0, -.2] \times 10^{-9}$	$.2 \times 10^{-9}$
6	$[-.5000000000, .8660254038]$	$[0, 0]$	0

Dans la figure ci-dessous on visualise ces itérations (trace bleue) sur le graphe de $\|f(\xi^{(k)})\|$ (contours rouges)



Pour répondre à la dernière question de l'exercice, rappelons qu'une telle $F : \mathbb{R}^2 \to \mathbb{R}$ existerait seulement si ses deuxièmes dérivées partielles mixtes sont les mêmes. Parmis les deux possibilités, on choisit

$$\partial F(\xi)/\partial x_2 = f_1(\xi) \quad \text{et} \quad \partial F(\xi)/\partial x_1 = f_2(\xi),$$

et on vérifie les deuxièmes dérivées partielles

$$\frac{\partial^2 F(\xi)}{\partial x_2\,\partial x_1} = \frac{\partial f_1(\xi)}{\partial x_1} = 3(x_1^2 - x_2^2) \quad \text{et} \quad \frac{\partial^2 F(\xi)}{\partial x_1\,\partial x_2} = \frac{\partial f_2(\xi)}{\partial x_2} = 3(x_1^2 - x_2^2).$$

Par intégration on obtient

$$F(\xi) = x_1^3x_2 - x_1x_2^3 - x_2 + \text{const} \quad \text{et donc} \quad \nabla F(\xi) = \big(f_2(\xi), f_1(\xi)\big) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} f(\xi), \quad \mathsf{G}(\xi) = \frac{\partial^2 F}{\partial \xi^2} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \mathsf{J}(\xi)$$

où la matrice symétrique $\mathsf{G}(\xi)$ des deuxèmes dérivées de $F(\xi)$ est appelée *Gaussian* de F. Alors, effectivement, il s'agit d'une recherche (locale) de l'extremum de la fonction $F(\xi)$ par une méthode d'ordre 2 (exploitant les deuxèmes dérivées).

Texte du programme :

```
# solution of the system of nonlinear eqns using Newton's method, DS 2010-11-09
# ------------------------------------------------------------------------------
with(linalg):
xi := [x1,x2]:        # variables
nvars := nops(xi):    # dimension of the system
# a handy auxiliary function to produce a list of var=value
eqzip := (a,b) -> zip( (i,j)->i=j, a,b):
# function to determine the value of vector-valued function f, return as list
# assume that the values of variables x and those of can be vectors or lists
fval := proc(x,f)
  subs( op(eqzip(xi,convert(x,list))), convert(f,list) )
end:
# prepare the quantities used in the iteration
f := [x1^3 -3*x1*x2^2 -1, 3*x1^2*x2 -x2^3]; # local diffeomorphism R2 -> R2
J := matrix(map(grad,f,xi));                # Jacobian matrix of f
factor(det(J));                             # det(J)=0 only for xi=0
# NB even though Maple thinks of grad as a vector it packs them linewise
iJ:=inverse(J);                             # inverse Jacobian matrix
df:=map(factor,convert(evalm(iJ&*f),list)); # correction vector f(x)/f'(x)

xi0 := [-0.5,0.5];    # initial point
f0  := fval(xi0,f):   # initial function value
nf0 := norm(f0,2):    # norm of f (used to measure the error) at initial point
xv := array(0..imax): # book keeping storage of iteration steps
xv[0] := [op(xi0),nf0]:

tau := 1e-2:          # relative tolerance of solution improvement
imax := 32:           # maximum number of iterations

for i from 1 to imax do                      # Newton's iterations
    xi1 := convert( evalm(xi0 - evalf(fval(xi0,df))), list);
    f1  := fval(xi1,f);                      # function vector at the new point
    nf1 := norm(f1,2):                       # its norm
    xv[i] := [op(xi1),nf1];
    lprint(i, xi1, f1, norm(f1,2));
    if i=imax or nf1=0 or abs(nf0/nf1-1) < tau then break fi;
    xi0 := xi1;
    f0 := f1;
    nf0:=nf1;
od:

plotsetup(ps,plotoutput='/tmp/Newton-iterations.ps', plotoptions='color,portrait,noborder'):
R1 := 1: R2 := R1: # sqrt(R1^2-x1^2):
plots[display]([
  plots[contourplot](sqrt(dotprod(f$2)),x1=-R1..R1,x2=-R2..R2,
                     contours=60, grid=[128$2], color=RED,axes=BOXED),
  plots[textplot3d]( [op(xv[0]),convert([op(1..2,xv[0])],string)], align={LEFT,TOP}, color=BLACK),
  plots[spacecurve]( [seq(xv[k],k=0..i)], color=BLUE,thickness=3)
], projection=1, orientation=[-90., 0], scaling=CONSTRAINED,  #view= 0..10,
   title='Convergence of Newton's iterations',  titlefont=['HELVETICA',18]);
```

TP-4 Projet : étude des vibrations d'une molécule diatomique

Introduction. Plusieurs séances de TD/TP et du cours seront consacrées à ce projet. Les étudiants se groupent en binômes. Les binômes choisissent différentes molécules. Chaque binôme prépare une présentation orale du projet défendue devant un jury de 2–3 enseignants, qui interviennent dans les cours des vibrations (Hadj Abdelhak, Robin Bocquet), de mécanique (Boris Zhilinskií, Guillaume Dhont) et de physique mathématique (Christophe Przygodsky). Le projet est noté. La note remplace celle d'un DS.

But de projet. On découvre trois thèmes principales du calcul numérique : solution des équations (méthode de Newton et autres), intégration d'un système des équations différentielles ordinaires du premier ordre (méthode RK), analyse de Fourier (transformation discrète, FFT). On apprends ainsi d'exploiter les possibilités du calcul symbolique du logiciel `MapleV.3`.

Notions de base : rappels sur la physique moléculaire On caractérise les molécules diatomiques[8] par leur distance interatomique d'équilibre r_e et leur énergie de dissociation D_e, voir quelques exemples ci-dessous[9]

molécule	état	D_e (eV)	r_e (Å)	ν_0 (cm^{-1})	molécule	état	D_e (eV)	r_e (Å)	ν_0 (cm^{-1})
LiH	Σ	2.47	1.596	1406	H_2	Σ_g	4.5	0.741	4401
CO	Σ	11.2	1.128	2170	N_2	Σ_g	9.8	1.098	2359
HCl	Σ	4.4	1.275	2991	O_2	$^3\Sigma_g$	5.2	1.208	1580
HF	Σ	5.8	0.917	4138	F_2	Σ_g	1.6	1.412	917
NO	Σ	7.0	1.151	1904	Cl_2	Σ_g	2.5	1.988	560

L'énergie totale ou l'hamiltonien est (ici μ est la masse réduite, voir problème de deux corps en cours de mécanique)

$$H = \frac{p^2}{2\mu} + V(x), \quad \text{où } x = r - r_e, \quad \text{et } \mu = \frac{m_1 m_2}{m_1 + m_2}.$$

On représente $V(x)$ par différentes fonctions empiriques, parmi lesquelles le *potentiel de Morse*[10]

$$V_{\mathrm{M}}(x) = D_e\big(1 - \mathrm{e}^{-\alpha(x/r_e)}\big)^2, \tag{TP-4.1}$$

ou le *potentiel de Lennard-Jones*[11] dit «6–12»

$$V_{\mathrm{LJ}}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] = \frac{a}{r^{12}} - \frac{b}{r^6}, \quad \text{avec } r = r_e + x, \tag{TP-4.2}$$

ou encore le *potentiel de Kratzer*

$$V_{\mathrm{K}}(r) = \frac{a_2}{r^2} - \frac{a_1}{r} + a_0, \quad \text{avec } r = r_e + x. \tag{TP-4.3}$$

sont les plus connus. Enfin, dans l'approximation *harmonique*, on utilise le potentiel quadratique

$$V_0(x) = \frac{k}{2}\,x^2 \tag{TP-4.4}$$

où k connu aussi comme k_0 ou k_{xx} est un appelé *constante de force harmonique*.

En mécanique classique, on obtiens les équations hamiltoniennes du mouvement pour les variables canoniques conjuguées x et p

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \dot{x} = \frac{\partial H}{\partial p} = \frac{p}{\mu}, \quad \frac{\mathrm{d}p}{\mathrm{d}t} = \dot{p} = -\frac{\partial H}{\partial x} = -\frac{\mathrm{d}V}{\mathrm{d}x} = -F(x).$$

Cela est un système des équations différentielles ordinaires du premier ordre, prêtes pour l'intégration numérique (voir méthodes de Euler et de RK). Pour passer en mécanique classique newtonienne, on observe que

$$\ddot{x} = \frac{\dot{p}}{\mu} = -\frac{1}{\mu}\,F(x),$$

et on considère donc $F(x)$ comme une force agissant sur la particule de masse μ.

[8] Voir les cours de la mécanique quantique (Zhilinskií, Dhont) et des vibrations (Hadj, Bocquet, Sadovskií) ainsi que l'appendice TP-4.1

[9] Huber, K. P. & Herzberg, G. (1979) *Molecular Spectra and Molecular Structure IV. Constants of Diatomic Molecules* (New York, Van Nostrand, Reinhold) ; `http://hyperphysics.phy-astr.gsu.edu/hbase/tables/diatomic.html` ; *Computational Chemistry Comparison and Benchmark Database. Part XIII – Vibrations* (Aug. 2005) NIST Standard Reference Database 101, Release 12, `http://srdata.nist.gov/cccbdb/vibrations.asp`. Notez qu'en anglais r_e est appelé *equilibrium separation* ou *bond length*

[10] Morse, P. M. (1929) Diatomic molecules according to the wave mechanics. II. Vibrational levels. *Phys. Rev.* **34**, 57–64.

[11] Lennard-Jones, J. E. (1931) Cohesion. *Proc. Phys. Soc.* **43**, 461-482.

Unités. On utilise le système des unités atomiques (u.a.) dans lequel $\hbar = 1$. L'u.a. d'énergie E_h, nommé Hartree, est égale à 2625.5 kJ/mol, 627.5 kcal/mol, 27.211 eV, ou 219474.6 cm^{-1}. Le cm^{-1} est utilisé souvent en spectroscopie pour exprimer les fréquences de vibration et/ou les énergies. Ainsi, pour obtenir la fréquence en Hz on multiplie par c en cm. L'u.a. de longeur a_0, ou rayon de Bohr, est égale 0.529 177 210(2) Å. Les u.a. de masse et de charge sont, respectivement, la masse m_e et la charge e d'électron.

Choix de la fonction potentielle, calcul des paramètres. En exploitant la mode symbolique de `MapleV.3`, trouvez les relations entre caractéristiques moléculaires r_e, D_e, et ω_0 et les paramètres des fonctions potentielles V_{M}, V_{LJ}, et/ou V_{K}. Peut on reproduire correctement les trois caractéristiques ? Faites le calcul pour le molécule de votre choix. Choisissez le potentiel le mieux adapté pour la description des petits vibrations autour r_e. Donnez approximation polynomiale ; determinez les constants de force harmonique k_0 et cubique k_{xxx}. Faites le dessin de differents fonctions potentielles et des approximations quadratique et cubique.

Calcul des points de retour r_1 et r_2. Pour une énergie donnée et en exploitant l'approximation quadratique, trouvez les intervalles de r pour chercher r_1 et r_2. Par la suite, utilisez la procédure `fsolve` pour trouvez les valeurs numériques exactes r_1 et r_2. (Alternativement utilisez la méthode de Newton étudie en TP-3.) Developpez une procedure pour une énergie quelconque $0 \leq E < D_e$. Faites le graph de $r_1(E)$ et $r_2(E)$ pour $0 \leq E < D_e$.

Intégration numérique de $x(t)$ et $p(t)$. Pour une énergie donnée choisissez les valeurs de départ $x(0)$ et $p(0)$. Par exemple, le point de retour r_1 («à gauche») ou le point $x = 0$. Trouvez symboliquement les équations hamiltoniennes du mouvement. Donnez ces équations numériquement pour la molécule choisie et en forme compatible avec `dsolve`. En exploitant l'approximation harmonique, estimez la période $T \approx T_0$ et déterminez le pas d'intégration τ, par exemple $T_0/64$. Utilisez `dsolve` en mode numérique RK-4 pour calculer $x(t)$ et $p(t)$ avec $t = 0, \tau, 2\tau, \ldots$

Intégration numérique : calcul de la période T. Dans le cas $x(0) = 0$, avancez $x(t)$ et cherchez le temps $t' > 0$ pour lequel $x(0)$ se trouve dans l'intervalle entre $x(t')$ et $x(t' + \tau)$. Dans le cas $x(0) = r_1$, avancez $p(t)$ et cherchez le temps $t'' > 0$ pour lequel $p(0) = 0$ se trouve dans l'intervalle entre $p(t'')$ et $p(t'' + \tau)$. Améliorez votre résultat de façon itérative (dichotomie, méthode des tangentes, section d'or ...) afin de trouver le temps $T/2 > 0$ pour lequel $x(T/2) = x(0)$.

Intégration numérique : exploitation et présentation des résultats. Redéfinissez τ en utilisant la valeur exacte de la période ; tabulez $x(t), p(t)$ pour une période. Calculez l'action

$$I = \frac{1}{2\pi} \int p \mathrm{d}x = \frac{1}{2\pi} \int_0^T p\dot{q}\, \mathrm{d}t.$$

(Ici l'intégrale est prise le long de la trajectoire pour un cycle.) Notez que le calcul de I peut être incorporer dans l'intégrateur `dsolve`. Dessinez vos trajectoires $x(t), p(t)$ dans l'espace des phases $\mathbb{R}^2$. Vérifiez (ou démontrez graphiquement) que l'énergie H est conservée. Calculez la période $T(E)$ et l'action $I(E)$ pour differentes énergies dans l'intervalle $0 \leq E < D_e$, faites un tableau et les graphs.

Transformation Fourier. Comparison avec la théorie des perturbations. Pour la comparison voir le texte de Hadj, section «force asymétrique». (Traduire ?)

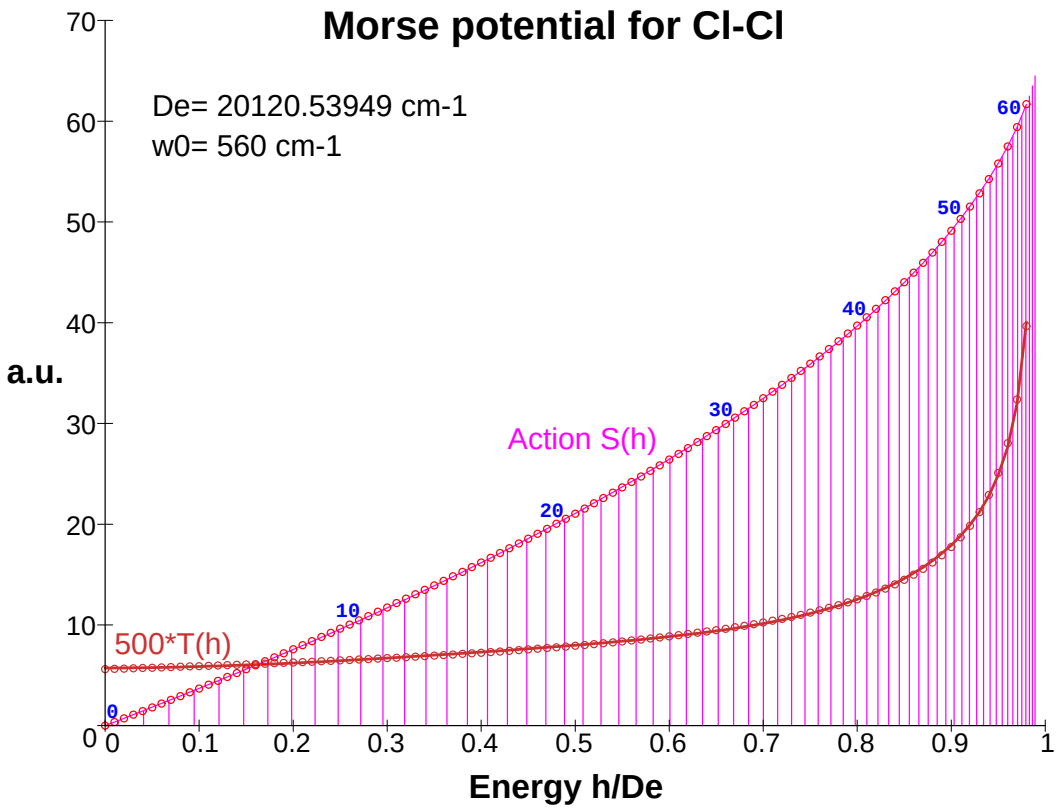
Calcul des énergies quantiques. On cherche les solutions E_n de l'équation de quantification

$$I(E_n) = n + \tfrac{1}{2} \quad \text{où } 0 \le E_n < D_e \quad \text{et } n = 0, 1, 2, \ldots$$

Faites un tableau de $T(E)$ avec un pas δ_E suffisamment petit, tel que $\delta_I \ll 1$. Approximez ces données par une fonction lisse, telle que

$$T(E) = a_0 + a_2 E^2 + a_4 E^4 + b\,(D_e - E)^{-1}.$$

(Question : quelle méthode faut il utiliser ?) Utilisez la relation $T(E) = \mathrm{d}I(E)/\mathrm{d}E$ afin d'approximer en même temps l'intégrale de l'action $I(E)$. Confirmez et démontrez vos résultats graphiquement.



Trouvez les intervalles d'énergie pour les valeurs de $I(E) = n + \frac{1}{2}$ ou $n = 0, 1, 2, \ldots$. Dans chaque intervalle, faites les itérations pour trouver les solutions numériques exactes. Il est possible ici de faire appel à la procédure `fsolve` avec votre approximation pour $I(E)$.

TP-4.1 Appendice : Potentiels moléculaires (B. I. Zhilinskiĭ)

Pour décrire les vibrations des molécules diatomiques on peut étudier le problème unidimensionnel du mouvement d'un particule de masse effective μ dans un potentiel $U(x)$, qui dépend d'un certain nombre de paramètres. Les valeurs numériques des paramètres peuvent être retrouvé à partir de données expérimentales pour les molécules concrètes.

Les données «expérimentales». On considère comme les données «expérimentales» qui caractérisent les molécules diatomiques :

1. Distance entre les noyaux dans la configuration d'équilibre r_e.
2. Fréquence harmonique ν_0, c'est à dire la fréquence d'oscillateur harmonique, qui décrit les petites vibrations prés de la position d'équilibre.
3. Énergie de la dissociation D_e, or la différence $U(r=\infty) - U(r=r_e)$. Cette énergie est nécessaire pour séparer les atomes (briser la liaison chimique).
4. Les masses m_1 et m_2 des atomes, voir le tableau périodique.

0. Sans concrétiser la forme du potentiel U, écrivez l'hamiltonien uni-dimensionel H, que vous allez utiliser pour décrire les vibrations d'une molécule diatomique. Exprimez la masse μ en fonction de m_1 et m_2.

1. Concrétisez H en utilisant le potentiel harmonique, $U(r) = k(r-r_e)^2/2$. Ce potentiel dépend de deux paramètres, k - constante de force, et r_e. Relier ces constantes avec les données expérimentales. Est ce que le modèle d'oscillateur harmonique peut reproduire la fréquence de vibration (distance d'équilibre, énergie de dissociation) avec propre choix des paramètres ? Donnez la relation entre k et la fréquence de vibration.

2. Concrétiser H en utilisant le *potentiel de Kratzer*

$$U_{\mathrm{K}}(r) = \frac{a}{x^2} - \frac{b}{r}$$

dépendant de deux paramètres a et b. Pourquoi ces paramètres doivent être positives dans le cas des vibrations moléculaires ? Pour le potentiel U_{K}, exprimez la distance d'équilibre r_e, la constante de force harmonique k, et l'énergie de la dissociation D_e, comme fonctions des paramètres (a, b). En utilisant le fait, que r_e, k, et D_e ne dépendent que de deux paramètres a et b, trouvez la relation entre r_e, k, D_e imposé par la forme de U_{K}.

3. Concrétiser le hamiltonien en utilisant le potentiel dit de Lennard-Jones :

$$U(x) = \frac{A_{12}}{x^{12}} - \frac{A_6}{x^6}$$

Ce potentiel dépend de deux paramètres, A_6, A_{12}. Pourquoi les constantes sont obligé d'être positives si vous modéliser les vibrations moléculaires ?
Exprimer pour le potentiel de Lennard-Jones la distance d'équilibre, x_e, la constante de force, k_e, l'énergie de la dissociation, D_e, comme les fonction des paramètres A_6, A_{12}.
En utilisant le fait, que x_e, D_e, k_e ne dépend que de deux paramètres A_6 et A_{12}, trouver la relation entre x_e, k_e, D_e imposé par le potentiel de Lennard-Jones.

4. Le potentiel de Morse

$$U(x) = D\left(e^{-2\alpha(x-x_0)/x_0} - 2e^{-\alpha(x-x_0)/x_0}\right)$$

dépend de trois paramètres D, α, x_0. Exprimer pour le potentiel de Morse la distance d'équilibre, x_e, la constante de force, k_e, l'énergie de la dissociation, D_e, comme les fonctions des paramètres D, α, x_0.

5. Les paramètres du potentiel de Morse pour les trois molécules H_2, HCl, I_2 sont données dans la table

Molécule	$\frac{\hbar^2}{2\mu x_e^2}$, cm^{-1}	D, cm^{-1}	α
H_2	60,8296	38292	1,440
HCl	10,5930	37244	2,380
I_2	0,0374	12550	4,954

Calculer la distance d'équilibre, la fréquence harmonique et l'énergie de dissociation pour H_2, HCl et I_2 à partir des paramètres de potentiel de Morse et de masses des atomes. Est ce que ces trois paramètres satisferont (au moins approximativement) la relation trouver pour le potentiel de Kratzer ou de Lennard-Jones ?

Est ce qu'on peut utiliser le potentiel harmonique, le potentiel de Kratzer ou le potentiel de Lennard-Jones pour les vibrations de ces trois molécules de façon approximative ? Si oui, proposer les valeurs des paramètres à prendre pour chaque modèle.

6. Pour une molécule de votre choix proposer le potentiel de Kratzer (de Lennard-Jones) qui vous permet reproduire l'énergie de dissociation et la fréquence harmonique. Répresenter ce potentiel et le potentiel de Morse initiale sur le même graphique.

TP-5 Intégrateur numérique des équations differentielles ordinaires

But de TP. Écrire et tester une procédure en langage `MapleV.3` permettant l'intégration numérique d'un système des équations differentielles ordinaires (é.d.o) d'ordre 1 par la méthode d'*Euler* et ces améliorations. Comme exemple concret, considérer le *pendule plan* de large amplitude (voir les cours de *Mécanique* et de *Vibrations*).

Généralités mathématiques Rappler la formule de *Leibnitz* définissant la derivée $\dot q = \mathrm{d}q/\mathrm{d}t$. Quelle est la différence entre les équations différentielles ordinaires et partielles ? Dans un système mécanique, qui joue le rôle de la variable indépendante (variable d'intégration) ? Comment transformer n équation(s) d'ordre $k > 1$ en un système de $n \times k$ équations d'ordre 1 ?

Généralités physiques On dérive les équations de mouvement pour notre système concret. Ainsi, en utilisant la mécanique *Lagrangienne* nous avons $\ddot\varphi + \omega_0^2 \sin\varphi = 0$ où $\omega_0 = \sqrt{g/l}$ peut être mise à 1 par un changement d'échelle de temps $t \mapsto t/\omega_0$. Par la suite, en utilisant deux variables $x(t) = \varphi(t)$ et $y(t) = \dot\varphi(t)$ on obtient le système de deux é.d.o d'ordre 1 :

$$\ddot\varphi + \omega_0^2 \sin\varphi = 0 \Rightarrow \begin{cases} \dot x = y \\ \dot y = -\sin x \end{cases} \quad \Rightarrow \quad \dot q = f(q), \text{ avec } q = \begin{pmatrix} x \\ y \end{pmatrix} \text{ et } f(q) = \begin{pmatrix} y \\ -\sin x \end{pmatrix}$$

Obtenez les mêmes équations en utilisant le formalisme *Hamiltonien* et variables dynamiques (φ, p_φ).

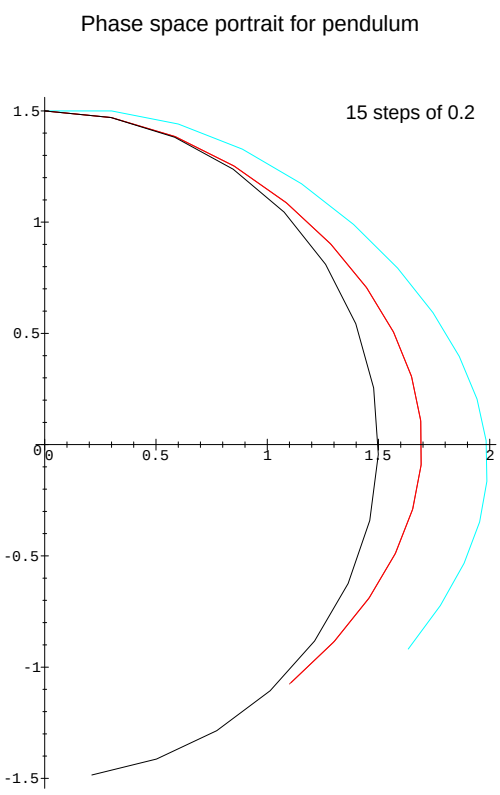
Notions de base sur la méthode d'*Euler* En fixant les conditions initiales $q_0 := q(0)$ pour l'instant $t = 0$ et le pas en temps Δ_t suffisamment petit, on essaye de retrouver les valeurs $q(t)$ pout $t > 0$. Ainsi on trouve la *trajectoire*, une courbe $\mathbb{R} \to \mathbb{R}^2 : t \mapsto q(t)$ dans l'*espace des phases* de notre systeme des é.d.o's $\mathbb{R}^2$ avec les coordonnées (x, y). Soit[12] $t_i = i\Delta_t$, $q_i = q(t_i)$ et $f_i = f(t_i, q_i)$. Afin d'avancer (intégrer) $q(t)$ en temps t selon l'équation $\dot q = f(q)$, on exploite la formule de *Leibnitz* $q(t_i + \mathrm{d}t) = q_i + \mathrm{d}\, q_i$ à l'instant $t = t_i$, $i = 1, 2, 3, \ldots$ où $\mathrm{d}q_i = f_i \mathrm{d}t \approx f_i\, \Delta_t$ pour $\mathrm{d}t \approx \Delta_t$. On effectue alors le *pas d'Euler*

$$q_{i+1} = q_i + f_i\, \Delta_t, \quad i = 0, 1, 2, \ldots \tag{TP-5.1a}$$

Cette simple méthode donne la précision est d'ordre $O(\Delta t^2)$, mais on observe aussi que les ereurs s'accumulent—ce qui est plus grave. Pour améliorer la *stabilité* par rapport à la trajectoire exacte, on modifie la procédure (TP-5.1a) de façon suivante

$$\begin{aligned} t_{i+1/2} := t_i + \tfrac{1}{2}\Delta_t, \quad q_{i+1/2} = q_i + \tfrac{1}{2} f_i\, \Delta_t, \\ f_{i+1/2} := f(t_{i+1/2}, q_{i+1/2}), \\ q_{i+1} = q_i + f_{i+1/2}\, \Delta_t, \quad i = 0, 1, 2, \ldots \end{aligned} \tag{TP-5.1b}$$

où la derivée $\mathrm{d}q/\mathrm{d}t$ pour le pas final est calculée «au milieux» de l'intervalle Δ_t. La précision devient $O(\Delta t^3)$ dans la méthode d'*Euler-Cauchy* (où effectue une ou plusieures itérations optionnelles pour affiner la valeur $\tilde f_{i+1}$ de f au point final)



$$\tilde q_{i+1} = q_i + f_i\, \Delta_t, \quad \underbrace{\tilde f_{i+1} = f_i(t_{i+1}, \tilde q_{i+1}), \quad q_{i+1} = q_i + \frac{f_i + \tilde f_{i+1}}{2}\, \Delta_t}_{\text{itération } \tilde q_{i+1} = q_{i+1}}, \quad i = 0, 1, 2, \ldots \tag{TP-5.1c}$$

Dans la figure ci-dessus, les courbes noire, cyan, orange et rouge, représentent, respectivement, l'approximation harmonique $\sin x \approx x$ (cercle), la méthode d'Euler (TP-5.1a), la méthode d'Euler modifiée (TP-5.1b), et la méthode RK-4 ; les deux dernières étant très proches de la solution exacte.

Choix de pas Δ_t On divise/multiplie Δ_t par deux et on compare les résultats. Il faut choisir Δ_t pour assure le calcul à la fois précis et économe.

[12] Dans notre exemple concret, f ne depend pas de temps, i.e., $f(t, q) = f(q)$

Texte du programme :

```
# TP on numerical integration of o.d.e. Maple 5.3  D.Sadovskii 2011-11-09
with(linalg):
with(plots):          # Maple V.3 general purpose graphics
odetype:='pendulum':          # use pendulum or Hpoly
plotsetup(ps,                 # PostScript and options
  plotoutput='/tmp/TPRK4.ps', plotoptions='color,noborder,portrait');
#   simple Euler's method (first order in tau)
Es_step := (f,u,t, tau) -> evalm( u + tau * f(u,t) ):
#   modified Euler's method (second order in tau)
Em_step := proc(f,u,t,tau) local k1,k2,k;
   k1 := tau * f(u,t);
   k2 := tau * f(evalm(u + k1/2), t+tau/2);
   k := [k1,k2];
   RETURN(evalm(u + k2));
end:
#    fourth order Runge-Kutta method
RK_step :=  proc(f,u,t,tau) local k1,k2,k3,k4,k;
   k1 := tau * f(u,t);
   k2 := tau * f(evalm(u + k1/2), t+tau/2);
   k3 := tau * f(evalm(u + k2/2), t+tau/2);
   k4 := tau * f(evalm(u + k3  ), t+tau);
   k := map(evalm,[k1,k2,k3,k4]);
   RETURN(evalm(u + k1/6 + k2/3 + k3/3 + k4/6));
end:
#    the driver routine which integrates o.d.e.'s using different methods
ode_trace := proc(step_fun, step_size, nstep:integer, ode_rhs, u0)
   local t,u,i,data;
   t := 0; u := u0;                     # initial conditions
   data := array[0..nstep];             # resulting trajectory
   for i from 0 to nstep do
      data[i] := [t,convert(u,list)];    # store current point
      u := step_fun(ode_rhs, u, t, step_size );
      t := t+step_size;                 # step forward in time
   od;
   RETURN(seq(data[i],i=0..nstep));      # return trajectory
end:
#-------------------------------------------------------------------------------
#    Hamiltonian equations of motion for a mathematical pendulum
if odetype='pendulum' then
  Ham := p^2/2 - cos(q);        # Hamiltonian for these equations of motion
  ode := (u,t) ->  vector( [u[2], -sin(u[1])] ):
  init_cond := vector([0,1.5]);          # initial conditions for t=0
  maxsteps:=15;                          # number of steps in time
  stepsize:=0.2;                         # time step size
fi;
#-------------------------------------------------------------------------------
data_Es := ode_trace(Es_step, stepsize, maxsteps, ode, init_cond ):
data_Em := ode_trace(Em_step, stepsize, maxsteps, ode, init_cond ):
data_RK := ode_trace(RK_step, stepsize, maxsteps, ode, init_cond ):
#    NB: for small q, the Hamiltonian approximates to q^2/2+p^2/2-1
if odetype='pendulum' then
  R0 := sqrt(2*(-cos(init_cond[1]) + init_cond[2]^2/2 +1));
  data_H0 := op(map(v->[v[1],[R0*sin(v[1]),R0*cos(v[1])] ], [data_Es])):
fi:
#-------------------------------------------------------------------------------
xmax := max( op(map(v->v[2][1], [data_Es])) );
ymax := max( op(map(v->v[2][2], [data_Es])) );
```

Méthode de *Runge-Kutta* (RK) est la méthode à pas fixe la plus courante (typiquement en version RK-4 d'ordre 4) pour obtenir une trajectoire stable et précise dans la plupart des cas. La méthode RK-k est basée sur les formules de l'extrapolation de *Legendre* donnant la précision $O\left(\Delta_t^{k+1}\right)$, son pas $i = 0, 1, 2, \ldots$ est effectué de façon suivante

$$\begin{aligned} K_1^{(i)} &= \Delta_t\, f(t_i, q_i), \\ K_2^{(i)} &= \Delta_t\, f\left(t_{i+1/2}, q_i + \tfrac{1}{2}K_1^{(i)}\right), \\ K_3^{(i)} &= \Delta_t\, f\left(t_{i+1/2}, q_i + \tfrac{1}{2}K_2^{(i)}\right), \\ K_4^{(i)} &= \Delta_t\, f\left(t_{i+1}, q_i + K_3^{(i)}\right), \end{aligned} \qquad q_{i+1} = q_i + \tfrac{1}{6}\left(K_1^{(i)} + 2K_2^{(i)} + 2K_3^{(i)} + K_4^{(i)}\right) \qquad \text{(TP-5.2)}$$

Présentation des résultats, graphisme Faire les graphs sous `MapleV.3` est toujours longue . . .

```
Es_color := cyan: Em_color := orange: RK_color := red: H0_color := white:
#    plot trajectory versus time x(t) and y(t)
plotsetup(plotoutput='/tmp/TPRK4xt.ps'):
plots[display]( {seq( plot( map(v->[v[1],v[2][1]], [data_.F]), color=eval(cat(F,_color)) ),
                F=[Es,Em,RK,H0])} );
plotsetup(plotoutput='/tmp/TPRK4yt.ps'):
plots[display]( {seq( plot( map(v->[v[1],v[2][2]], [data_.F]), color=eval(cat(F,_color)) ),
                F=[Es,Em,RK,H0])} );
plotsetup(plotoutput='/tmp/TPRK4xyt.ps'):
plots[display]( 'union'(seq({ plot( map(v->[v[1],v[2][1]], [data_.F]), color=eval(cat(F,_color)) ),
                              plot( map(v->[v[1],v[2][2]], [data_.F]), color=eval(cat(F,_color)) )
                                 },
                F=[Es,Em,RK,H0]) ) );
#    plot phase space trajectory (x(t),y(t))
plotsetup(plotoutput='/tmp/TPRK4xy.ps'):
plots[display]([
   textplot([xmax, ymax, cat('step=',convert(stepsize,string),' ',
                             convert(maxsteps,string),' steps')],
            color=black,align=LEFT),
   seq( plot( map(v->[v[2][1],v[2][2]], [data_.F]), scaling=CONSTRAINED,
              color=eval(cat(F,_color)) ),
        F=[Es,Em,RK,H0])
  ], title=cat('Phase space portrait for ',odetype),
     font=[HELVETICA,16],titlefont=[HELVETICA,18] );
#    print out all trajectories
for i from 1 to nops([data_Es]) do
   printf(' %6.3f   %8.3f %8.3f   %8.3f %8.3f   %8.3f %8.3f   %8.3f %8.3f \n' ,
             op(i,[data_Es])[1], seq( op(op(2,op(i,[data_.k]))), k=[Es,Em,RK,H0] ) );
od;
```

Programme optionnel Comme une option intéressante, on peut considérer les solutions de l'é.d.o d'ordre 2 définissant les polynômes d'Hermite $H_n(t)$ d'ordre n

$$\frac{\mathrm{d}^2 H_n}{\mathrm{d}t^2} - 2t\,\frac{\mathrm{d}H_n}{\mathrm{d}t} + 2n\,H_n(t) = 0, \qquad \text{avec } q(t) := \begin{pmatrix} H_n(t) \\ \mathrm{d}H_n(t)/\mathrm{d}t \end{pmatrix} \text{ et } f(t,q) := \begin{pmatrix} q_2(t) \\ 2tq_2 - 2nq_1 \end{pmatrix}$$

```
#    differential equation defining Hermite's polynomials
if odetype='Hpoly' then
  deq := (h,n) -> expand( diff(h(n,x),x,x) - 2*x*diff(h(n,x),x) + 2*n*h(n,x) ):
  n := 6;                # choose order of the polynomial
  deq(orthopoly[H],n), orthopoly[H](n,x), diff( orthopoly[H](n,x), x);
  ode := (y,x) -> vector( [ y[2],  2*x*y[2] - 2*n*y[1] ]):
  init_cond := vector( subs(x=0,[orthopoly[H](n,x), diff( orthopoly[H](n,x), x)]) );
  maxsteps:=15;  stepsize:=0.08;
fi;
```

et par la suite on initialise la trajectoire exacte

```
#    ok, compute the exact solution at the same points in time
if odetype='Hpoly' then
  data_H0 := op(map(v->[v[1], subs(x=v[1], [orthopoly[H](n,x), diff( orthopoly[H](n,x), x)])  ],
                  [data_Es])):
fi:
```

TP-6 Transformation de Fourier discrète

But de TP. Écrire et tester une procédure en langage `MapleV.3` permettant de transformer une fonction $g(x)$ définie par un tableau $g(x_1), g(x_2), \ldots$ sur une grille des points équidistants $x_1, x_2, \ldots$ vers en tableau des amplitudes du spectre Fourier, ainsi que la procédure de la transformation inverse (tableau des amplitudes vers tableau des valeurs de $g(x)$). Faire le graphe de $g(x)$ à partir du tableau donné et à partir des fréquences trouvées. Sur le même graphe, tracer l'approximation obtenue avec les hautes fréquences «coupées» ou atténuées.

Notions de base. On considère la fonction $g(x)$ échantillonnée pour des valeurs équidistantes de x telles que

$$x_k = k\Delta, \quad g_k = g(x_k), \quad \text{où } k = 0, 1, 2, \ldots, N-1.$$

Ici Δ est *l'intervalle d'échantillonnage*. Notez qu'on appelle Δ^{-1} *le taux d'échantillonnage*. On définit aussi *la fréquence critique de Nyquist*

$$f_c = \frac{1}{2\Delta} \tag{TP-6.1}$$

Si la fonction $g(x)$ est échantillonnée avec un taux de Δ^{-1}, son spectre Fourier $h(f)$ est limité par la bande passante $|f| < f_c$. Donc si les amplitudes $h(f)$ dans le vrais spectre de $g(x)$ sont nulles pour tous $|f| \geq f_c$, ce spectre est entièrement défini par un nombre fini des échantillons g_k. Ce qui est tout à fait remarquable.

Le spectre $h(f)$ est lui aussi discret. Pour N points g_k nous devons évidemment pouvoir obtenir N amplitudes dans ce spectre. En effet, les amplitudes h_n sont définies pour les fréquences f_n discrètes telles que

$$f_n = \frac{n}{N\Delta}, \quad h_n = \Delta h(f_n) \quad \text{où } n = -\frac{N}{2}, \ldots, \frac{N}{2}, \quad \text{et } h_{-\frac{N}{2}} \equiv h_{\frac{N}{2}} = 2\,\Delta\, h(f_c). \tag{TP-6.2}$$

Notez que les valeurs extrêmes des fréquences sont $\pm f_c$, et que grâce à la dernière condition (à démontrer plus tard) nous avons précisément N amplitudes h_n indépendantes. (Facteur de 2 à ne pas oublier !)

L'expression pour la transformation Fourier discrète direct est obtenue par la dicretisation de l'intégrale Fourier

$$h(f) = \int_{-\infty}^{\infty} g(x) \exp(2\pi \mathrm{i} f x)\, dt \quad \Rightarrow \quad h(f_n) = \sum_{k=0}^{N-1} g_k \exp(2\pi \mathrm{i} f_n x_k)\, \Delta = \Delta \sum_{k=0}^{N-1} g_k \exp\left(\frac{2\pi \mathrm{i}}{N} kn\right). \tag{TP-6.3}$$

Ainsi cette transformation donne N nombres

$$h_n = \sum_{k=0}^{N-1} g_k \exp\left(\frac{2\pi \mathrm{i}}{N} kn\right) \tag{TP-6.4}$$

indépendants de l'échelle Δ de l'échantillonnage. Notez que h_n est périodique en n avec la période N et que en particulier $h_{-n} = h_{N-n}$. Par conséquent, au lieu de considérer l'indice n dans l'intervalle $[-\frac{N}{2}, \ldots, \frac{N}{2}]$ comme nous avons fait auparavant, il est possible et pratique d'utiliser $n = 0, \ldots, N-1$ comme nous avons fait pour k. En adhérant à une telle convention, il faut se rappeler de la correspondance suivante

$n = 0$	$f = 0$
$1 \leq n \leq \frac{1}{2}N - 1$	$0 < f < f_c$
$n = \frac{N}{2}$	$f = \pm f_c$
$\frac{1}{2}N + 1 \leq n \leq N - 1$	$-f_c < f < 0$

Enfin la transformation inverse $\{h_n\} \to \{g_k\}$ est

$$g_k = \frac{1}{N} \sum_{n=0}^{N-1} h_n \exp\left(-\frac{2\pi \mathrm{i}}{N} kn\right) \tag{TP-6.5}$$

Notez que le calcul de h_n et g_k peut être effectué par le même programme. Il nous reste à expliquer comment reconstruire la fonction $g(x)$ à partir des amplitudes $\{h_n\}$. On utilise (TP-6.5) :

$$g_k = g(x_k) = \frac{1}{N} \sum_{n=0}^{N-1} h_n \exp\left(-\frac{2\pi \mathrm{i} n}{N\Delta} x_k\right) = \frac{1}{N} \sum_{n=0}^{N-1} h_n \exp(-2\pi \mathrm{i} f_n x_k),$$

où les fréquences f_n sont données par (TP-6.2). Par conséquent,

$$g(x) \approx \frac{1}{N} \sum_{n=0}^{N-1} h_n \exp(-2\pi \mathrm{i} f_n x).$$

Si cette fonction reconstruite fait trop des oscillations d'hautes fréquences on pourrait les atténuer ou tout simplement couper en utilisant une fréquence de coupure f_n avec $n_{\max} \ll N$.

Quelques conseils de programmation en `MapleV.3`**.** Regardez la section correspondante du TP précédent pour des informations générales. Commencez la procédure de la transformation directe de la façon suivante

```
with(linalg);
dDFT := proc(g:vector)
  local N,n,h, ... ;
  N :=  vectdim(g);
  h := vector(N);

  . . .


  RETURN(evalm(h));
end:
```

et utilisez (TP-6.4) pour remplir les entrées `h[n]` de vecteur `h`. La procédure `iDFT` de la transformation inverse s'écrit de même façon et traduisez (TP-6.5).

Pour tester vos procédures prenez un vecteur de $N = 8$ (ou 16, ou 32 etc) points, par exemple

```
  g := vector([1,2,3,1,5,7,6,4]);
  N :=  vectdim(g);
  h := dDFT(g);
  s := iDFT(h);
```

vous pouvez comparer `s` et `g`. Pour visualiser la fonction `g` et son spectre `h`, assumez que $\Delta = 1$ et utilisez la commande `plot` de façon suivante

```
  plot( zip((a,b)->[a,b], [$1..N], convert(g,list)) ) ;
```

Pour en faire le graphe de `g` ensemble avec la fonction $g(x)$ reconstruite comme une expression `gfun` selon (TP-6.5)

```
  plots[display]([
     plot( zip((a,b)->[a,b], [$1..N], convert(g,list)) ) ,
     plot( gfun(x), x=0..N )
  ], title='original and reconstructed functions');
```

Enfin, pour un teste ultime, appelez la procédure `FFT` de `MapleV.3`.

```
readlib(FFT):                 # Fast Fourier Transform library
  g1 := map(evalf,map(Re,g)): # real part of g(x)
  g2 := map(evalf,map(Im,g)): # imaginary part
  i := FFT(3,g1,g2);
```

Notez que ici 3 (le premier argument) est le $\log_2 N$ pour $N = 8$ et que `g2` représente à l'entrée la partie imaginaire de $g(x)$, tandis que en retour `g1` et `g2` contiennent les parties réelles et imaginaires des amplitudes h_n (donc les valeurs originales de `g1` seront perdues). Comparez les avec vos résultats dans `h`.

Questions. Quelle est la fréquence de Nyquist dans votre exemple ? Expliquer le choix de la fréquence de coupure. Les valeurs h_n sont ils réelles ? Pourquoi ? Démontrez que votre fonction `gfun` est réelle.

Texte du programme :

```
# direct/inverse discrete Fourier transform (DFT) (c) 2006 Sadovskii
with(linalg);
readlib(FFT):                # Fast Fourier Transform library

dDFT := proc(g:vector)       # direct transform
  local k,n,N,h,s;
  N := vectdim(g);
  h := vector(N);
  for n from 0 to N-1 do
  s := 0;
    for k from 0 to N-1 do
      s := s+g[k+1]*exp(2*Pi*I/N*k*n)
    od;
  h[n+1]:=s;
  od;
  RETURN(evalm(h));
end:

Fn := proc(n,N)
  if n=0 then RETURN(0);
  elif n>0 and n<N/2 then RETURN(n/N);
  elif n>N/2 and n<N then RETURN(n/N-1);
  else RETURN(n/N);
  fi;
end:

# testing the procedures
g := vector([1,2,3,1,5,7,6,4]);
N := vectdim(g);
h := dDFT(g);

# reconstruction of g(x)
gfun := 0:
for n from 0 to N/2-1 do
  gfun := gfun + h[n+1]*exp(-2*Pi*I*(n/N)*x);
od:
for n from N/2+1 to N-1 do
  gfun := gfun + h[n+1]*exp(-2*Pi*I*(n/N-1)*x);
od:
gfun := gfun + h[N/2+1]*( exp(-2*Pi*I*(1/2)*x) + exp(2*Pi*I*(1/2)*x) )/2:
gfun := combine(evalc(gfun / N),trig);

g1 := copy(g):         # real part of g(x)
g2 := vector([0$N]):   # imaginary part $

FFT(3,g1,g2);          # transform and compare
print(g1); evalf(map(Re,h));
print(g2); evalf(map(Im,h));

plotsetup(ps,plotoutput='/tmp/FT.ps',plotoptions='color,noborder'); # ,portrait

plots[display]([
    plot( zip((a,b)->[a,b], [$0..N-1], convert(g,list)), style=POINT, symbol=CIRCLE, color=red, thic
    plot(gfun,x=0..N,color=blue, thickness=2)
    ],
    font=[HELVETICA,18],
    axesfont=[HELVETICA,18],
    labelfont=[HELVETICA,BOLD,22],
    titlefont=[HELVETICA,BOLD,24],
    title='Fourier transform of discrete data'
);
```

TP-7 Calcul des valeurs propres

But de TP. Écrire et tester une procédure en langage `MapleV.3` permettant de trouver les valeurs propres d'une matrice symétrique réelle en utilisant la méthode de *Jacobi* (1846).

Questions générales. Donnez les définitions de la matrice symétrique, hermitienne, orthogonale, et unitaire. Combien valeurs propres possède une matrice symétrique ou hermitienne de dimension k ? Sont-elles réelles ou complexes ? Pourquoi ? En quel domaine de physique utilise-t-on des matrices symétriques et leurs valeurs propres ?

Notions de base. Afin de diagonaliser une matrice symétrique réelle A de dimension k la méthode de Jacobi utilise une séquence des rotations

$$A' = R^T(p,q)\, A\, R(p,q)$$

(appelées plus précisément transformations orthogonales de similarité ou conjugaisons avec des matrices orthogonales) définies par les matrices

$$R(p,q) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

dont les éléments diagonaux $R(p,q)_{pp} = R(p,q)_{qq} = c = \cos\phi$, le reste des éléments diagonaux sont égaux à 1, et les seuls deux élements nondiagonaux non nuls sont $R(p,q)_{qp} = -R(p,q)_{pq} = s = \sin\phi$. Notez que seuls les deux colonnes et les deux lignes avec les indices q et p sont changées par cette transformation. Plus précisément, on obtient[13]

$$A'_{rp} = cA_{rp} - sA_{rq}, \qquad A'_{rq} = cA_{rq} + sA_{rp}, \qquad r \neq q,\ r \neq qp \tag{TP-7.1a}$$

$$A'_{pp} = c^2 A_{pp} + s^2 A_{qq} - 2s\,c\,A_{pq} \qquad A'_{qq} = s^2 A_{pp} + c^2 A_{qq} + 2s\,c\,A_{pq} \tag{TP-7.1b}$$

$$A'_{qp} = (c^2 - s^2)A_{pq} + s\,c\,(A_{pp} - A_{qq}) \tag{TP-7.1c}$$

Le but de la transformation et d'éliminer l'élément A'_{qp}. Pour cela on récrit (TP-7.1c) sous forme

$$t^2 + 2t\theta - 1 = 0, \quad \text{avec} \quad \theta = \frac{A_{qq} - A_{pp}}{2\,A_{pq}},$$

où $t = s/c = \tan\phi$ et on trouve

$$t = \frac{s}{c} = \frac{\operatorname{sgn}(\theta)}{|\theta| + \sqrt{\theta^2+1}}, \quad c = \frac{1}{\sqrt{1+t^2}}, \quad s = t\,c, \quad \tau = \tan\frac{\phi}{2} = \frac{s}{1+c}. \tag{TP-7.2}$$

Par conséquent,

$$A'_{pq} = 0, \tag{TP-7.3a}$$

$$A'_{pp} = A_{pp} - t\,A_{pq}, \qquad A'_{qq} = A_{qq} + t\,A_{pq}, \tag{TP-7.3b}$$

$$A'_{rp} = A_{rp} - s\,(A_{rq} + \tau A_{rp}), \qquad A'_{rq} = A_{rq} + s\,(A_{rp} - \tau A_{rq}). \tag{TP-7.3c}$$

Quelques conseils de programmation en `MapleV.3`. Utilisez les TP précédents pour des informations générales. Notez que la transformation peut être effectuée sur place en utilisant que la moitie de A. Commencez par écrire une procédure de la transformation $R(p,q)$ (rotation de Jacobi) basée sur (TP-7.2) et (TP-7.3). Puis programmez un boucle double et appliquez $k(k-1)/2$ transformations $R(1,2),\dots,R(1,k),R(2,3),\dots,R(2,k),\dots,R(k-1,k)$. Enfin faites un boucle général avec le critère d'arrêt

$$S = \sum_{r<s} |A_{rs}| < \epsilon$$

Comptez le nombre des passages de ce dernier boucle. Présentez vos résultats. Comparez les avec les valeurs propres obtenues en utilisant la procédure standard de `MapleV.3`.

[13] n'oubliez pas que A et A' sont symétriques, par exemple $A_{ij} = Aji$