

Université du Littoral – Côte d’Opale
Licence de Physique, 3^{me} année, Calais

Physique numérique

analyse et présentation des données, programmation

Dmitriï Sadovskii
Département de physique

L3 S5, automne 2026



Vos suggestions, améliorations et corrections sont bienvenues,
merci de les envoyer par mél à sadovskii@univ-littoral.fr
Pour les mises à jour, voir <http://purple.univ-littoral.fr/etudes/CalNum/>

Horaires : cours-TD-TP 13 séances de 3h

Table des matières

	page
Contenu du cours	2
<i>ce module est destiné aux étudiants de la troisième année en licence sciences avec l’option physique à Calais — il aura les thèmes principaux suivants</i>	
Listages de programmes en MAXIMA	2
Contrôles des connaissances	2
Bibliographie sur MAXIMA	2
1 Introduction	3
1.1 Architecture des ordinateurs	3
<i>notions de base sur le hardware, systèmes d’exploitation, structures de représentation des données, etc</i>	
2 Statistiques, analyse des données	3
<i>traitement des données et notions essentielles de statistique, telles, que la distribution Gaussienne, écart standard, corrélations, intervalle de confiance, etc</i>	
2.1 Erreurs systématiques et accidentelles	3
2.2 Présentation statistique des résultats	3
2.3 Combinaison des incertitudes	4
2.4 Méthode des moindres carrés, cas de régression linéaire	6
<i>estimation des incertitudes, problèmes mal définis ou insuffisants, stabilité, décomposition selon les valeurs singulières – en pratique (TP2), utiliser la statistique et la M.M.C. pour la présentation des résultats</i>	
TP1 Solution des systèmes des équations linéaires	9
<i>développer un programme pour la méthode de Gauss en utilisant le logiciel MAXIMA</i>	
TP2 Méthode des moindres carrés	12
<i>Appliquer la M.M.C. (sec. 2.4) pour modéliser les données sur les masses atomiques avec MAXIMA</i>	
TP3 Méthodes itératives. Itération de Newton, étude de convergence	14
<i>racines, équations nonlinéaires, linéarisation suivie par une itération M.M.C. (TP2) – application à la recherche numérique de la période (TP4 et TP5), et la quantification WKB de l’action (TP4)</i>	
TP3.1 L’approche de Newton dans une dimension	14
TP3.2 Généralisation aux plusieurs dimensions	14
TP4 Projet : étude des vibrations d’une molécule diatomique	17
<i>intégration numérique des équations de mouvement (TP5), calcul de la période et de l’action, quantification sémi-classique (méthode WKB), estimation de la fréquence et du nombre des niveaux vibrationnels</i>	
TP4.1 Appendice : Potentiels moléculaires (B. I. Zhilinskiï)	20
<i>les potentiels empiriques de Kratzer, Lennard–Jones, ou Morse</i>	
TP4.2 Questions théoriques sur le projet «vibrations moléculaires»	21

TP5 Intégration numérique des équations différentielles ordinaires	22
<i>solution numérique des équations différentielles ordinaires, problème de Cauchy, intégrateurs – en pratique, étude des équations de mouvement du pendule, et d’une molécule diatomique (TP4)</i>	
TP5.1 Généralités mathématiques et l’application numérique	22
TP5.2 Généralités physiques. Pendule plan	24
TP5.3 Recherche de la période et de l’action. Quantification	26
TP5.4 Etude de polynômes d’Hermite	27
TP5.5 Présentation des résultats, graphisme	27
TP6 Transformation de Fourier discrète	28
<i>transformation de Fourier directe (DFT), inverse, rapide (FFT), filtrage — en pratique, écrire un programme DFT sous MAXIMA ; analyser la solution des équations différentielles de mouvement nonlinéaires obtenue précédemment en TP5 (pendule) et TP4 (molécule)</i>	
TP7 Calcul des valeurs propres	31
<i>se familiariser au calcul matriciel sous MAXIMA : diagonalisation, orthogonalisation, inversion ; développer un programme pour trouver les valeurs propres d’une matrice symétrique par la méthode de Jacobi</i>	
TP7.1 Notions de base sur la méthode de Jacobi	31
Acronymes	35
Bibliographie	35

Listages de programmes en MAXIMA

	page
L1 <code>license.mac</code> : commande pour démontrer votre version de MAXIMA et LISP	3
L2 <code>Gelim.mac</code> : programme pour TP1 (systèmes des équations linéaires) sous MAXIMA	11
L3 <code>nonlinlsq.mac</code> : programme sous MAXIMA pour TP2	13
L4 <code>Newton-method-eqns.mac</code> : programme sous MAXIMA pour exercice TP3.2	16
L5 <code>Runge-Kutta.mac</code> : intégrateurs É.D.O. et utilitaires sous MAXIMA pour TP5	23
L6 <code>Runge-Kutta-TP.mac</code> : programme pour TP5 sous MAXIMA	25
L7 <code>DFT.mac</code> : programme pour TP6 (transformation Fourier discrète) sous MAXIMA	30
L8 <code>Jacobi.mac</code> : programme pour TP7 (valeurs propres) sous MAXIMA	32
L9 <code>qpend.mac</code> : programme sous MAXIMA pour l’exercice TP7.1 (spectre quantique du pendule)	34

License : Sauf noté autrement, tout listage pour MAXIMA est ©2026 D.A. Sadovskií (sadovski@univ-littoral.fr) ; la Licence Publique Générale GNU (GNU GPL) ver. 3 de la Free Software Foundation, Inc. (FSF), 51 Franklin St., 5th Floor, Boston, MA 02110-1301, Etats Unis, est impliquée. Ce code est un logiciel libre ; vous pouvez le redistribuer et/ou le modifier au titre des clauses de la version 3 de la GNU GPL, telle qu’est publiée par la FSF. Le code est distribué dans l’espoir qu’il sera utile, mais sans aucune garantie, ni même une garantie implicite de commerciabilité ou de conformité à une utilisation particulière. Vous devriez avoir reçu et consulté un exemplaire de la GNU GPL avec ce code ; si ce n’est pas le cas, demandez le à la FSF.

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 MAXIMA. 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 MAXIMA 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.

Ressources et bibliographie sur MAXIMA

1. *Aide-Mémoire pour Maxima*
2. *Introduction à Maxima* par H. Hand, traduite par M. Gosse
3. *The Maxima Book* par De Souza, Fateman, Moses, Yapp

Listage L1 – `license.mac` : commande pour démontrer votre version de MAXIMA et LISP

```
block([x:build_info()],
  printf(true," running on ~a with maxima/~a ~a/~a~% under GNU GPL ver.3, see ~a~%",
    time_date(), x@lisp_name, x@version,
    substring(x@lisp_version, 1, sposition(" ",x@lisp_version)), load_pathname),
  if stringp(x@maxima_frontend) then printf(true," frontend ~a ver.~a~%",
    x@maxima_frontend, x@maxima_frontend_version)
);
```

1 Introduction

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

1.1 Architecture des ordinateurs

2 Statistiques, analyse des données

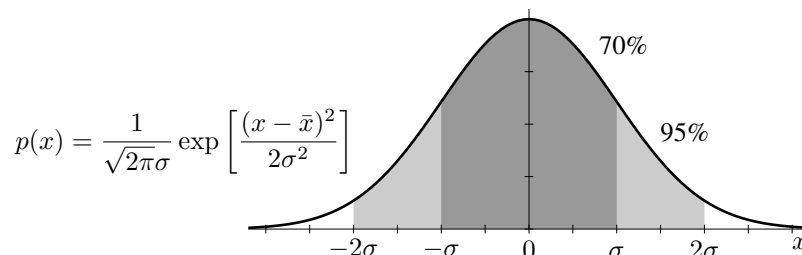
2.1 Erreurs systématiques et accidentelles

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.

2.2 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



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

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

On prendra

$$x_e = \frac{1}{N} \sum_{i=1}^N x_i \quad (2.1)$$

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

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

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 $N \rightarrow \infty$, nous avons $\sigma_e \rightarrow \sigma$. Ce que nous intéresse, c'est l'écart type σ_{x_e} . Lui est donnée par

$$s_{x_e}^2 = \frac{1}{N} \sigma_e^2 = \frac{1}{N(N-1)} \sum_{i=1}^N (x_i - x_e)^2. \quad (2.2)$$

Dans la limite de grands N nous avons

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

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 $N \rightarrow \infty$ 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 P	nombre de mesures N				
	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
0.99	9.9	4.6	4.0	3.3	2.6

Exercice 2.1. 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.

2.3 Combinaison des incertitudes

Souvent nous ne pouvons pas mesurer directement la quantité y en question. On mesure donc un nombre K des quantités $(\xi_1, \xi_2, \dots, \xi_K)$ afin de trouver y comme une fonction de $(\xi_1, \xi_2, \dots, \xi_K)$. Supposons, que $(\xi_1, \xi_2, \dots, \xi_K)$ 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 , $k = 1, \dots, K$.

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

$$y = a\xi_1 \pm b\xi_2, \quad (2.3a)$$

avec des constantes a et b . Évidemment

$$\bar{y} = a\bar{\xi}_1 \pm b\bar{\xi}_2. \quad (2.3b)$$

On trouve pour l'écart type de y , que

$$(\sigma_y)^2 = a^2 \sigma_1^2 + b^2 \sigma_2^2, \quad \sigma_y \leq |a| \sigma_1 + |b| \sigma_2, \quad \Delta y \leq |a| \Delta \xi_1 + |b| \Delta \xi_2. \quad (2.3c)$$

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

$$y = f(\xi_1), \quad \bar{y} = f(\bar{\xi}_1), \quad \sigma_y^2 = (f'(\bar{\xi}_1))^2 \sigma_1^2, \quad \text{à condition que } f'(\bar{\xi}_1) = \left. \frac{df}{d\xi_1} \right|_{\bar{\xi}_1} \neq 0. \quad (2.4)$$

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

$$y = \log(\xi_1), \quad \bar{y} = \log \bar{\xi}_1, \quad \sigma_y^2 = \frac{\sigma_1^2}{\bar{\xi}_1^2}, \quad \Delta y = \frac{\Delta \xi_1}{\bar{\xi}_1}. \quad (2.5)$$

Exercice 2.2. En utilisant les règles (2.3c) et (2.5) montrerons le *règle des écarts relatifs*

$$y = \xi_1^a \xi_2^b \text{ ou } y = \frac{\xi_1^a}{\xi_2^b}, \quad \left(\frac{\sigma_y}{\bar{y}} \right)^2 = a^2 \left(\frac{\sigma_1}{\bar{\xi}_1} \right)^2 + b^2 \left(\frac{\sigma_2}{\bar{\xi}_2} \right)^2 \quad (2.6)$$

Exercice 2.3. 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.

$$y = f(\xi_1, \dots, \xi_K), \quad \sigma_y^2 = \sum_{k=1}^K \left(\left. \frac{\partial f}{\partial \xi_k} \right|_{\bar{\xi}} \right)^2 \sigma_k^2, \quad \sigma_y \leq \sum_{k=1}^K \left| \left. \frac{\partial f}{\partial \xi_k} \right|_{\bar{\xi}} \right| \sigma_k \quad (2.7)$$

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

Exercice 2.4. 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 $\mathcal{E} = 10\text{V}$, $R_1 = R_2$.

Exemples des combinaisons des incertitudes.

1. En spectrophotométrie (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-spectrométrie, spectrophotométrie, 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

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

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 (2.3) 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 $(y - \bar{y})^2$:

$$\overline{(y - \bar{y})^2} = \int \int (y - \bar{y})^2 d\xi_2 d\xi_1 = \int \int [a^2(\xi_1 - \bar{\xi}_1)^2 + b^2(\xi_2 - \bar{\xi}_2)^2 \pm 2ab(\xi_1 - \bar{\xi}_1)(\xi_2 - \bar{\xi}_2)] d\xi_2 d\xi_1$$

Les deux premiers termes donnent les deux termes de (2.3c), 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 (2.4) 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

$$\overline{(y - \bar{y})^2} = f'(\bar{\xi}_1) \overline{(\xi_1 - \bar{\xi}_1)^2}.$$

□

2.4 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. 2.2. 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 méthode¹ 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

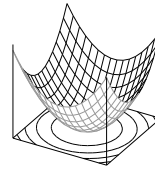
$$F(\alpha) = \sum_{i=1}^N \left(\frac{y_i - f(x_i, \alpha)}{\sigma_i} \right)^2.$$

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

$$[g] := \frac{1}{S} \sum_{i=1}^N \frac{g_i}{\sigma_i^2}, \quad \text{où} \quad S = \sum_{i=1}^N \frac{1}{\sigma_i^2}, \quad \text{et} \quad g_i = x_i, y_i, x_i y_i, x_i^2, \quad (2.8a)$$

on exprime les valeurs de a_1 et a_0 ,

$$a_1 = \frac{[xy] - [x][y]}{[xx] - [x][x]}, \quad a_0 = [y] - [x] a_1, \quad (2.8b)$$

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

$$\sigma_{a_1}^2 = \frac{1}{S} \frac{1}{[xx] - [x][x]}, \quad \sigma_{a_0}^2 = \sigma_{a_1}^2 [xx]. \quad (2.8c)$$

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. 2.2.

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)

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

dont la solution est donnée par (2.8b). □

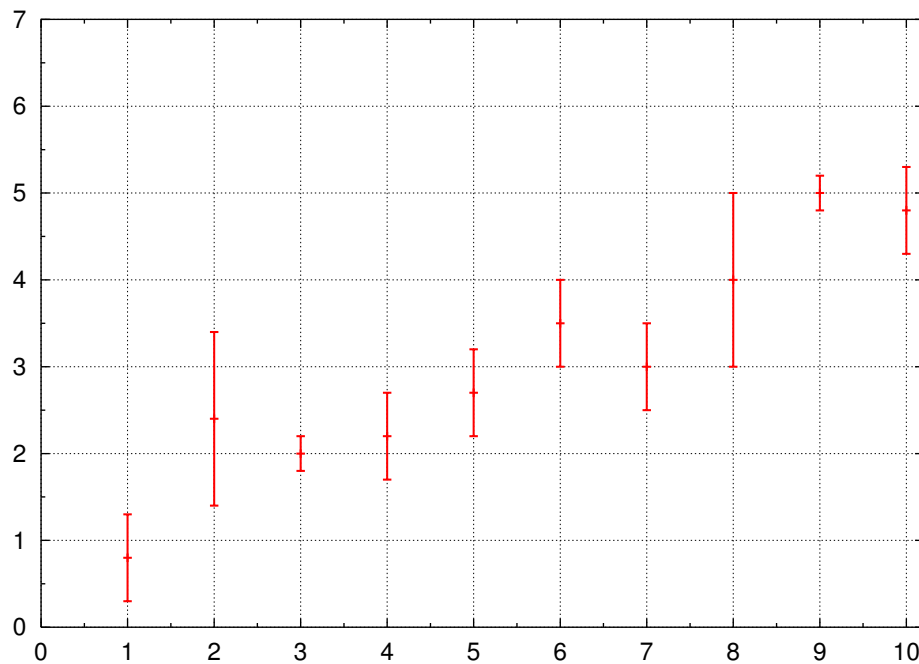
¹en terminologie anglaise : *least squares fit*

²Notez, 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. Spectrophotométrie (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 2.5. 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 $y = a_1x + a_0$, 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

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

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

$$A a = y, \quad \text{ou graphiquement} \quad \boxed{A} \boxed{a} = \boxed{y} \quad (2.9a)$$

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 résoudre (2.9a), 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} représentent 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 corrélation 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 (2.9a) par σ^{-1} à gauche,

$$\sigma^{-1} A a = \sigma^{-1} y. \quad (2.9b)$$

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

Afin de résoudre (2.9b), on le transforme en multipliant à gauche par A^T (matrice A transposée). Cela donne

$$S a = (A^T \sigma^{-1} A) a = A^T \sigma^{-1} y = b, \quad (2.9c)$$

ou graphiquement

$$\boxed{S} \boxed{a} = \boxed{A^T} \boxed{\sigma^{-1}} \boxed{A} \boxed{a} = \boxed{A^T} \boxed{\sigma^{-1}} \boxed{y} = \boxed{b}.$$

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

$$a = S^{-1} b = S^{-1} A^T \sigma^{-1} y, \quad (2.9d)$$

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 corrélation C dont les éléments sont

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

Exercice 2.6. Dans le cas $n = 2$ avec $f(x; \alpha) = \alpha_1 + \alpha_2 x$, vérifiez que la solution a trouvée ainsi dans (2.9d) correspond à celle trouvée dans (2.8b).

Exercice 2.7. Démontrez plus généralement, que la solution (2.9d) 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 (2.9d) 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. Évidemment, nul garantit la convergence. Elle dépend largement du choix du départ $\alpha^{(0)}$ et des corrélations entre les paramètres.

TP1 Solution des systèmes des équations linéaires

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

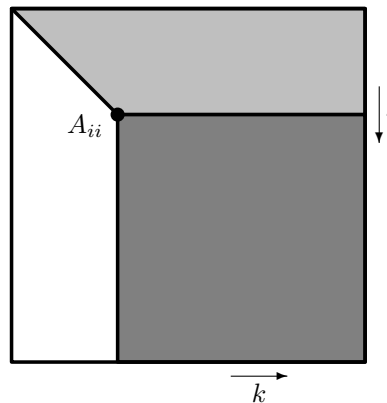
Notions de base. On cherche à obtenir une solution $x = (x_1, \dots, x_n)$ d'un système à n équations linéaires écrit sous la forme matricielle suivante

$$Ax = b,$$

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 $A \rightarrow A'$ 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 $i = 1, \dots, n$ 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 $k = i, \dots, n$ ainsi que b_i par p :

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

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 $j = i+1, \dots, n$ on soustrait ligne i multipliée par $r = A_{ji}$ et on corrige également la partie droite b_j :

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

(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 $i = n-1, \dots, 1$ on utilise les composants x_k avec $k > i$ déjà trouvées précédemment pour chercher x_i et on remplace b_i par x_i :

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

Quelques conseils de programmation en MAXIMA Notez que si la $n \times n$ matrice A est représentée par l'objet A du type **matrix**, les éléments A_{ik} de A sont accédés comme $A[i][k]$. Ainsi, dans une boucle suivante, on imprime les éléments $i = 1, \dots, n$ de la colonne k (cf. le dessin ci-dessus)

```
for i:1 thru n do print (A[i][k]); /* k-th entry of row i */
```

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

```
for k:i thru n do A[i][k] : A[i][k]/p; /* rescale i-th row above diagonal */
```

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 */
rowdim(A) := if matrixp(A) then length(first(args(A))) else 0 $
gausselim(A, b) := block(
  [n:rowdim(A), i, j, k, p, r], /* row dimension of matrix A */

  . . .

  return(b)
)$
```

Notez qu'ici les marques `/*` et `*/` entourent les commentaires. Notez aussi la déclaration et l'initialisation des variables `n, i, j, k, p, r` 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 . Remarquez que MAXIMA reconnaît les matrices comme des objets de type `matrixp` construits par un appel à `matrix`, e.g., une matrice 2×2 est définie comme `A:matrix([1,2],[3,4])`. Il n'y a pas d'objet spécial pour les vecteurs, qui sont représentés comme des matrices $1 \times n$ ou *vecteurs-colonnes*, e.g., `v:matrix([v1],[v2])`. Cependant, les listes, telles comme `[v1,v2]`, sont aussi reconnus comme des vecteurs dans le contexte des opérations d'algèbre linéaire, notamment le produit non-commutatif `A.v` dont le résultat est rendu naturellement sous forme d'un *vecteur-colonne*. Pour réduire cette forme «encombrante» à une liste simple, on peut introduire une utilitaire `vlist` suivant

```
vlist(v) := if listp(v) then v elseif matrixp(v) then flatten(args(v)) else [v];
```

et pour convertir une liste en un vecteur-colonne, on fait appel à `columnvector` (alias `covect`). Il vous sera aussi peut être nécessaire de utiliser `float` afin de convertir vos résultats finales et intermédiaires en nombres flottants. Enfin, pour tester votre procédure faites

```
A : matrix([[1,2,3],[7,8,9],[4,6,5]]); AX:copymatrix(A);
b : matrix([1],[2],[3]);          bX:copymatrix(b);
gausselim(AX,bX);
```

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 (créées par `matrix`) ne sont pas attribués en MAXIMA par leur valeur mais par leur nom (adresse). Afin de les attribuer par leur valeur, utilisez `copymatrix` (A). Vous pouvez donc utiliser vos A et b pour un teste ultime

```
v:[x1,x2,x3]; /* variables of the linear equations */
linsolve(flatten(args(A.v-b)),v);
```

et comparer avec la solution retournée par votre `gausselim` (voir la description de la procédure standard `linsolve`).

Exercice TP1.1 (pivot). Quel est le défaut principal de votre programme ? Votre programme, est-il stable numériquement ? Pourquoi ? Quelle est le rôle de l'élément pivotant p ? Quelles sont les modifications nécessaires afin de pouvoir choisir l'élément pivotant ?

Listage L2 – Gelim.mac : programme pour TPI (systèmes des équations linéaires) sous MAXIMA

```

2  /*+ simple in-place Gauss elimination code without pivoting (c) 2026 Sadvskii +*/
maxima_this_rev : "(c)DS 20261010"$ batchload("license.mac")$
debug_level : 1$ /* set to 0 to skip all extra debugging msg */
4  rowdim(A) := if matrixp(A) then length(first(args(A))) else 0 $
vlist(v) := if listp(v) then v elseif matrixp(v) then flatten(args(v)) else [v] $
6
gausselim(A, b) := block([n:rowdim(A)], /* for n x n matrix */
8   for i:1 thru n do block([p:A[i][i]], /* pivoting element */
   if p=0 then return, /* avoid division by 0 */
10  for j:i thru n do A[i][j]:A[i][j]/p, /* rescale i-th row */
   b[i]:b[i]/p, /* rescale the rhs entry */
12  if debug_level>1 then print(i,p,A[i]),
   if i=n then return, /* stop with no rows left to eliminate */
14  for j:i+1 thru n do block([r:A[j][i]], /* eliminate i-th column from all rows j>i */
   for k:i thru n do A[j][k] : A[j][k]-A[i][k]*r,
16  b[j]:b[j]-b[i]*r, /* subtract from the j-th rhs entry */
   if debug_level>1 then print(j,A,b)
18  )
   ),
20  if debug_level>0 then print(" in upper triangular form",A,"b=",vlist(b)),
   for k:1 thru n-1 do block([i:n-k],
22  if debug_level>1 then print(i,b[i]),
   for j:i+1 thru n do (
24  b[i] : b[i]-b[j]*A[i][j], /* back substitute, storing the solution in b */
   if debug_level>2 then print(i,j,b[i])
26  ),
   if debug_level>1 then print(b[i])
28  ),
   if debug_level>0 then print(" solution",vlist(b),"after backsubstitution"),
30  return(vlist(b)) /* solution vector as a simple list */
) $
32 /* preserve original A and b with copymatrix(A) and copymatrix(b), destroy AX */
A : matrix([1,2,3],[7,8,9],[4,6,5]); AX:copymatrix(A)$
34 b : matrix([1],[2],[3]); bX:copymatrix(b)$
gausselim(AX,bX); /* test the program */
36 v:[x1,x2,x3]; /* variables of the linear equations */
v:subst( linsolve(flatten(args(A.v-b)),v), v);
38 print(A,v,A.v, if zeromatrixp(v-bX) then "agree with gausselim" else "TEST NOT PASSED");

```

TP2 Méthode des moindres carrés

Considérez les données [Coursey et al., 2015] 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 ,

$$M_{\text{eff}}(Z) = a + bZ + cZ^2 + \frac{d}{Z^K}.$$

Au premier temps, on peut appliquer la méthode linéaire pour fixer la valeur de K à 0.5. Par la suite, développez un programme de la méthode des moindres carrés (M.M.C.) et obtenez les valeurs de paramètres a , b , c , et d . Calculez les matrices de covariation S et de corrélation C . En supposant que l'écart type des données est $\sigma = 10^{-4}$ u.a., trouvez l'écart types des paramètres. Représentez et interprétez vos résultats. Comparez la «théorie» et «l'expérience» textuellement et graphiquement (voir la figure 1 ci-dessous). Quel élément est le plus stable³? Enfin proposez et réalisez une procédure itérative pour optimiser la valeur de K .

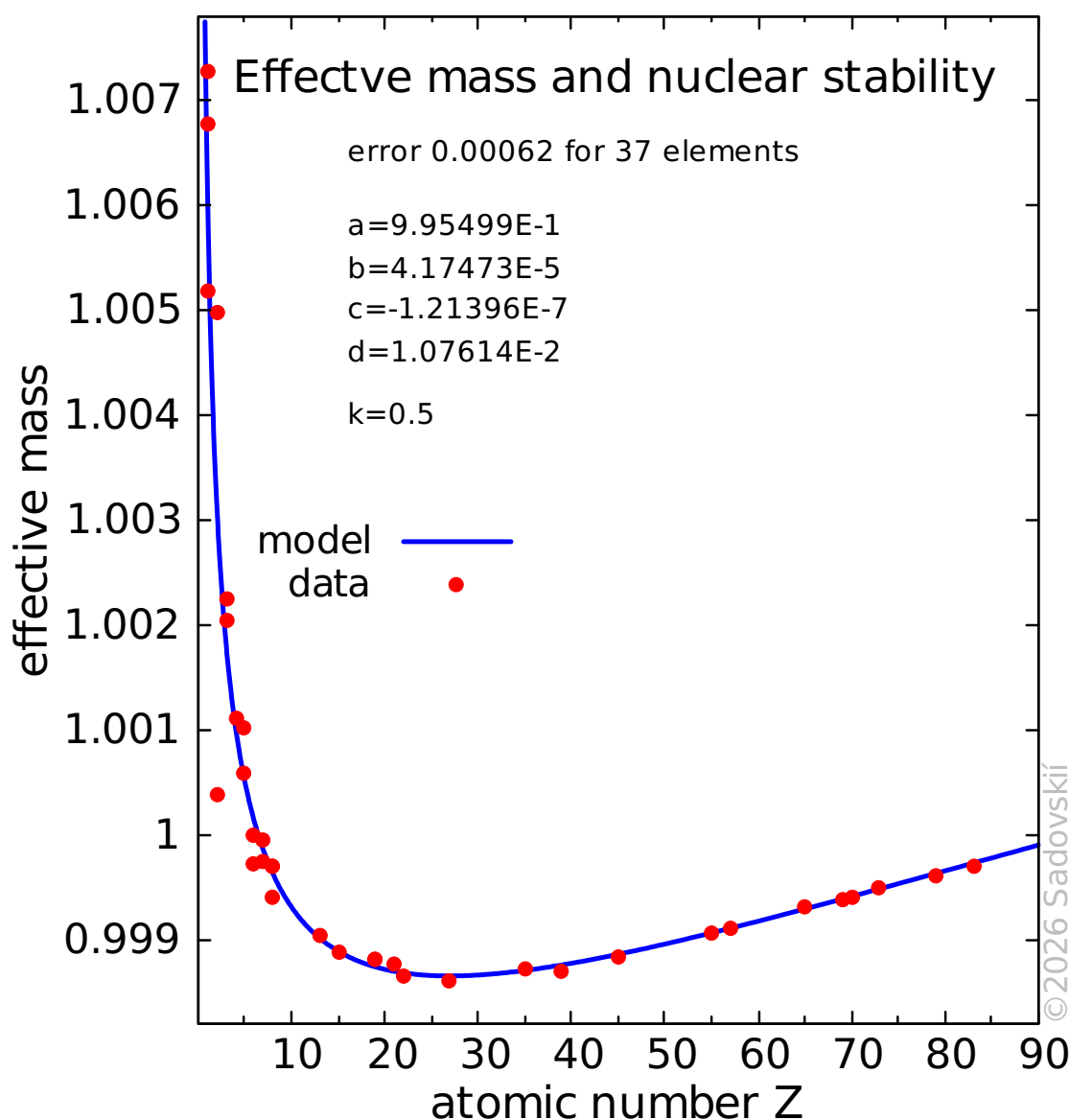


FIG. 1 – TP2 : présentation graphique des résultats de la M.M.C. (avec MAXIMA, voir la fin de listage L3)

³l'idée de ce TP a été suggérée par Prof. Boris Zhilinskiĭ ; pour l'interprétation physique, voir le cours de physique atomique

Listage L3 – nonlinlsq.mac : programme sous MAXIMA pour TP2

```

/*+ dependence of the effective atomic mass Meff on the charge (atomic number) Z +*/
4 m_e : 5.485799e-4 $ /* mass of the electron (approx m_p/1836) [a.u.] */
Meff(Z,n,m) := (m-m_e*Z)/n $ /* definition of the effective mass [a.u.] */
6 atomic_data : [ /* all masses in a.u. */
/*[ 0, "n" , 1, 1.00866],*/[ 7, "N" , 14, 14.00307],
8 [ 1, "H" , 1, 1.007825], [ 7, "N" , 15, 15.00011], [39, "Y" , 89, 88.90586],
[ 1, "D" , 2, 2.01410], [ 8, "O" , 16, 15.99491], [45, "Rh",103,102.90550],
10 [ 1, "T" , 3, 3.01605], [ 8, "O" , 17, 16.99913], [55, "Cs",133,132.90543],
[ 2, "He", 3, 3.01603], [ 8, "O" , 18, 17.99917], [57, "La",139,138.90636],
12 [ 2, "He", 4, 4.00260], [13, "Al", 27, 26.9815 ], [65, "Tb",159,158.92535],
[ 3, "Li", 6, 6.01512], [15, "P" , 31, 30.97376], [69, "Tm",169,168.93423],
14 [ 3, "Li", 7, 7.01600], [19, "K" , 39, 38.9637 ], [70, "Yb",171,170.9363 ],
[ 4, "Be", 9, 9.01218], [19, "K" , 41, 40.9618 ], [73, "Ta",181,180.94801],
16 [ 5, "B" , 10, 10.0129 ], [21, "Sc", 45, 44.95591], [79, "Au",197,196.96656],
[ 5, "B" , 11, 11.0093 ], [22, "Ti", 48, 47.94795], [83, "Bi",209,208.98039],
18 [ 6, "C" , 12, 12.00000], [27, "Co", 59, 58.93320], [92, "U" , 238,238.05079],
[ 6, "C" , 13, 13.00335], [35, "Br", 79, 78.9183 ], [94, "Pu",238,238.04956] ]$
20 sigma : 0.0001$ /* estimated data uncertainty */
atomic_data : sort(atomic_data, lambda([a,b],a[1]<b[1]))$
22 c:'c$ a:'a$ b:'b$ d:'d$ Kval:0.5; /* approximating empirical function */
Feff(Z) := a + b*Z + c*Z^2/2 + d/Z^Kval ;
24 S : 0 $
for t in atomic_data do block( [tm:Meff(t[1],t[3],t[4]), ratprint:false],
26 S : S + (Feff(t[1]) - tm)^2 /* sum of deviations squared */
);
28 block( [ratprint:false, ratmx:false, fpprintprec:4, N:length(atomic_data)],
S : float(expand(S)), Svars:listofvars(S),
30 dS: float(makelist(diff(S,v), v, Svars)), /* gradient of S */
/* find parameter values which minimize S, i.e., for which dS=0 */
32 ss: float(solve(dS,Svars)),
print( " found", Svars, " = ", subst(ss,Svars)),
34 M : float(coefmatrix(dS,Svars)),
M : invert(M), /* compute the covariation matrix */
36 dM: makelist(sqrt(M[i][i]),i,length(Svars)),
print(" parameter variance = ", sigma*dM),
38 Mc: 100*ident(length(Svars)), /* compute the correlation matrix */
for i:1 thru length(Svars) do (
40 for j:1 thru i-1 do Mc[i][j] : round(M[i][j] / (dM[i]*dM[j]) * 100)
), print(""),
42 print(" covariation matrix", M), print(""),
print(" correlation matrix", Mc ), print(""),
44 for i:1 thru length(Svars) do block( [p:Svars[i]],
printf(true,"~a ~12,5e ~9,3e~%", p, subst(ss,p), dM[i]*sigma)
46 ), /* compare lsq fit vs experiment */
for t in atomic_data do block( [tm:Meff(t[1],t[3],t[4])],
48 printf(true,"~3d ~2a[~3d] ~10,6f ~7,5f (~4d)~%",
t[1],t[2],t[3],t[4],tm, round((subst(ss,Feff(t[1]))-tm)*1e5) )
50 ),
s0 : sqrt(subst(ss,S)/N),
52 printf(true," mean square deviation of ~7,5f for ~d data points~%",s0,N)
)$
54 plot2d( [ subst(ss,Feff(x)),
[discrete, makelist([t[1],Meff(t[1],t[3],t[4])],t,atomic_data)] ],
56 [x,0.05,90], [y,0.9982,1.0078], [yx_ratio, 1.2],
[style, [lines,2,1], [points,2,2,1]], [legend, "model", "data"],
58 [xlabel, "atomic number Z"], [ylabel, "effective mass"],
[title,"Effective mass and nuclear stability"],
60 append([label,[printf(false,"error ~,5f for ~d elements",s0,length(atomic_data)),15,1.0065],
[printf(false,"k=~f",Kval),15,1.0062-0.0022]],
62 makelist([printf(false,"~a=~,5e",Svars[i], subst(ss,Svars[i])), 15, 1.0062-i*.0004],
i, length(Svars))),
64 [gnuplot_preamble,
concat( "set title offset 0,-4 font ',14'; set tics font ',13'; set key font ',12';",
66 " set xlabel font ',14'; set ylabel font ',14'; set key at graph .4,.5; set mxtics 2;",
printf(false," set label 201 '~a1'~a Sadovskii' at graph 1,0 offset 1,0 rotate by 90 font ',10' tc 'gray
';",scriptname(), substring(timedata(),1,5)))
68 )$

```

TP3 Méthodes itératives. Itération de Newton, étude de convergence

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 conséquent, ces méthodes s'appliquent aux 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.

TP3.1 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, considérez $f : \mathbb{R} \rightarrow \mathbb{R} : x \mapsto f(x)$ 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 ordre⁴

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

c'est à dire, en utilisant l'approximation linéaire au voisinage du point x , nous cherchons⁵ la racine dans $x + \Delta$:

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

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

$$x_{k+1} = x_k - f(x_k)/f'(x_k). \quad (\text{TP3.1})$$

Cela représente l'*itération newtonienne*.

Exercice TP3.1. En prenant $x_0 = 2$ trouvez la racine $x^3 + 6x^2 + 10x - 1$ en utilisant la méthode de *Newton*. Faites le d'abord à la main, puis à l'aide de MAXIMA. Caractérisez la convergence de la méthode. Proposez un critère d'arrêt.

```
/* minimize function f(x) using Newton's method and assuming that |x| << 1 */
```

TP3.2 Généralisation aux plusieurs dimensions

Au présent, considérons l'application

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

et supposons que $\partial f_i / \partial x_i$ ainsi que tout les deuxèmes dérivées existent dans un domaine $U \ni x_0$ de $\mathbb{R}^n$. Rappelez que dans ce cas le développement limité nous donne⁶

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

L'itération de Newton (TP3.1) prend la forme suivante⁶

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}^{-1}(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k). \quad (\text{TP3.2})$$

Notons, qu'une approche similaire basée sur la linéarisation de la fonction $f(x, \alpha)$ est utilisée dans le cadré de la m.m.c. (voir sec. 2.4 et TP2) afin d'optimiser les valeurs de paramètres α dont f dépend de façon nonlinéaire.

Exercice TP3.2. Considérez le cas avec $n = 2$ et trouvez la racine de

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

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 MAXIMA. Caractérisez la convergence de la méthode. Proposez un ou plusieurs critères d'arrêt. Enfin considérez si ce problème corresponde à une recherche (locale) d'un extremum d'une fonction $F : \mathbb{R}^2 \rightarrow \mathbb{R} : (x_1, x_2) \mapsto F(x_1, x_2)$.

⁴Dans 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*.

⁵Évidemment en plus d'existence des dérivées nous devons demander que $f'(x) \neq 0$ pour tout $x \in [x_{\min}, x_{\max}]$!

⁶Ici le *Jacobian* $\mathbf{J}$ 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 $\mathbf{J}$ est le gradient d'une composante de f . Notez aussi que afin de pouvoir inverser $\mathbf{J}$ dans tout le domaine U nous devons supposer $\det \mathbf{J}(x) \neq 0$ pour tout $x \in U$, i.e., l'absence des *points critiques* de f , comparez avec le cas d'une dimension dans la note 5.

Solution de l'exercice TP3.2 Soit $\xi = x = (x_1, x_2)^T \in \mathbb{R}^2$. On compute :

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

On voit que le seul point critique de l'application $f(\xi)$ de l'exercice TP3.2 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 choisissant $\xi^{(0)} = (-\frac{1}{2}, \frac{1}{2})$ comme le point de départ, nous obtenons pour les itérations Newtoniennes successives (cf la figure 2)

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

Convergence of Newton's iterations

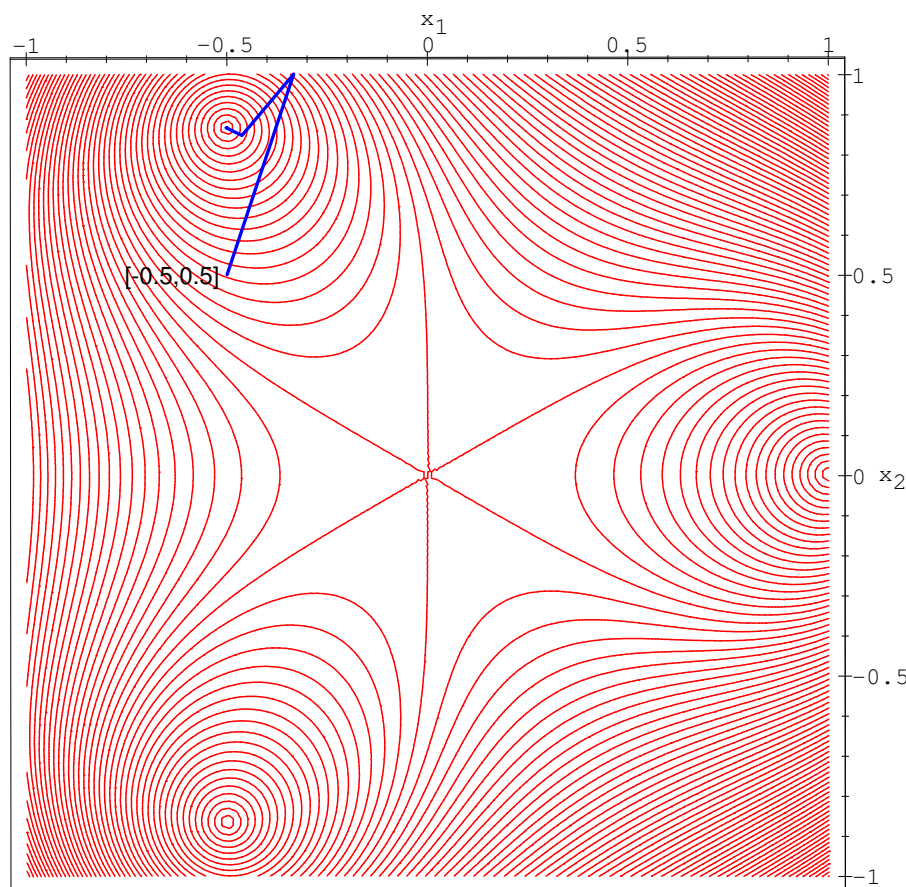


FIG. 2 – Les résultats de l'exercice TP3.2 (avec MAPLE V.3) : les itérations k (trace bleue) sur le graphe de $\|f(\xi^{(k)})\|$ (contours rouges)

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

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

où la matrice symétrique $G(\xi)$ des deuxièmes 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 deuxièmes dérivées).

Listage L4 – Newton-method-eqns.mac : programme sous MAXIMA pour exercice TP3.2

```

1 /* solve a system of nonlinear eqns using Newton's method, Maxima DS 20260924 */
vnorm(v) := sqrt(v.v)$ /* compute the norm |v| of vector v */
3 vlist(v) := if listp(v) then v elseif matrixp(v) then flatten(args(v)) else [v]$
/* names of variables and dimension of the system of (nonlinear) equations */
5 (xi : [x1,x2], nvars : length(xi))$
/* relative tolerance on solution improvement and max number of iterations */
7 ( tau : 1e-2, imax : 32 );
/* nonlinear equations defining the concrete local diffeomorphism R2 -> R2 */
9 f(x) := [ x[1]^3 - 3*x[1]*x[2]^2 - 1, 3*x[1]^2*x[2] - x[2]^3 ];
(
11 Jf(x) := jacobian(f(x),x), /* matrix of first derivatives of f(x) */
detJf(x) := factor(determinant(Jf(x))),
13 _iJf : factor(invert(Jf(xi))), /* its symbolic inverse (for xi) */
iJf(x) := factor(subst([xi[1]=x[1],xi[2]=x[2]],_iJf)) /* for concrete xi=x */
15 )$
print(" c'est bien une matrice inversée:", factor(Jf(xi)._iJf))$
17 printf(true,
" solve equations f(~{a~#~[~;~]} ) = (~{a~#~[~;~]} ) = 0% with Jacobian determinant ~a~%",
19 xi,f(xi),detJf(xi));
/* initial point, function value, and its notm (used to measure the error) */
21 ( xi0 : [-0.5,0.5], f0 : f(xi0), nf0 : vnorm(f(xi0)) );
printf(true," starting at x=(~{a~#~[~;~]} ) with f(x)=(~{~3f~#~[~;~]} ) and |f(x)|=~3f~%",
23 xi0,f0,nf0);
xv : [append(xi0,[f0,nf0])]; /* initiate iteration book keeping */
25 for i:1 thru imax do block(
[xi1,f1,nf1, ratprint:false], /* Newton's iterations i=1..imax */
27 xi1 : float(vlist(xi0 - iJf(xi0).f(xi0))), f1 : f(xi1), nf1 : vnorm(f1),
xv : endcons(append(xi1,[f1,nf1]), xv),
29 printf(true," ~2d x=(~{~9,6f~#~[~;~]} ) f(x)=(~{~9,6f~#~[~;~]} ) ~,2e~%",
i, xi1, f(xi1), nf1), /* break if changes in |f| are small */
31 if nf1=0 or abs(nf0/nf1-1) < tau then i:imax,
xi0:xi1, f0:f1, nf0:nf1 /* continue to next iteration */
33 )$

```

Remarquez que MAXIMA reconnaît les matrices comme des objets de type `matrixp` construits par un appel à `matrix`, e.g., une matrice 2×2 est définie comme `A:matrix([a11,a12],[a21,a22])`. Il n'y a pas d'objet spécial pour les vecteurs, qui sont représentés comme des matrices $1 \times n$, e.g., `v:matrix([v1],[v2])` ou⁷ `v:covect([v1,v2])`. Cependant, les listes, telles comme `[v1,v2]`, sont aussi reconnus comme des vecteurs dans le contexte des opérations d'algèbre linéaire, notamment le produit non-commutatif `A.v` dont le résultat est rendu naturellement sous forme d'une matrice $1 \times n$. Le code L4 ci-dessus introduit une utilitaire `vlist` pour réduire cette forme «encombrante» à une liste simple, voir le calcul de `xi1` dans la boucle.

Exercice TP3.3 (quantification sémi-classique). Considérez le pendule plan quantique qu'on étudiera d'avantage dans l'exercice TP7.1. Au présent, notons que selon le principe EBK⁸, dans la limite sémi-classique, ses énergies E_n correspondent aux énergies du système mécanique classique h pour lesquelles l'intégrale d'action classique $S(h)$ le long de ses trajectoires fermées $\gamma(h) \sim \mathbb{S}^1$ prend des valeurs entiers n fois $\hbar$. Cependant, le pendule possède deux régimes, rotationnel et vibrationnel, avec $h > 2$ et $h < 2$, respectivement, et pour une quantification EBK uniforme, nous devons considérer une action réduite $\tilde{S}(h)$

$$S(h) = \frac{1}{2\pi} \oint_{\gamma(h)} p_\theta d\theta, \quad \tilde{S}(h) = \begin{cases} S(h)/2 & \text{for } h \leq 2 \\ S(h) & \text{for } h \geq 2 \end{cases} = \frac{4}{\pi} \begin{cases} E(k_1) - (k'_1)^2 K(k_1) \\ k_1 E(k) \end{cases} = \hbar n_\pm, \quad (\text{TP3.3a})$$

où K et E sont les intégrales elliptiques complètes⁹ de 1er et 2me espèces en fonction du *module elliptique* (ou *excentricité*)

$$k_1 = \sqrt{\frac{h}{2}}, \quad k = \frac{1}{k_1}, \quad \text{et} \quad k'_1 = \sqrt{1 - k_1^2}. \quad (\text{TP3.3b})$$

On se propose de résoudre l'équation $\tilde{S}(h) = \hbar n_\pm$ (TP3.3a) numériquement pour les états paires avec $n_+ = 0, 1, 2, \dots$ et impaires avec $n_- = 1, 2, \dots$ et trouver ainsi les énergies $E_{n_\pm}$ du pendule quantique dans l'approximation uniforme sémi-classique.

⁷MAXIMA fonction `columnvector` et sont synonyme `covect` nécessitent la bibliothèque `eigen` qu'on peut charger avec `load("eigen")`;

⁸le principe EBK généralise l'approche de Bohr-Sommerfeld, les idées de de Broglie, ainsi que la méthode WKB, et s'applique à tous systèmes mécaniques classiques intégrables, donc, forcément, aux systèmes à un seul degré de liberté comme le pendule

⁹Sous MAXIMA, les valeurs des intégrales $K(m)$ et $E(m)$ sont rendues par `elliptic_kc(m)` et `elliptic_ec(m)` avec paramètre $m:k^2$ ou $m:1/k^2$

TP4 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 regroupent 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 Zhilinskiĭ, 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 apprend ainsi d'exploiter les possibilités du calcul symbolique du logiciel MAXIMA.

Notions de base : rappels sur la physique moléculaire On caractérise les molécules diatomiques¹⁰ par leur distance interatomique d'équilibre r_e et leur énergie de dissociation D_e , voir quelques exemples ci-dessous¹¹

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* [Morse, 1929]

$$V_M(x) = D_e (1 - e^{-\alpha(x/r_e)})^2, \quad (\text{TP4.1})$$

ou le *potentiel de Lennard-Jones* [Lennard-Jones, 1931] dit «6–12»

$$V_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] = \frac{a}{r^{12}} - \frac{b}{r^6}, \quad \text{avec } r = r_e + x, \quad (\text{TP4.2})$$

ou encore le *potentiel de Kratzer*

$$V_K(r) = \frac{a_2}{r^2} - \frac{a_1}{r} + a_0, \quad \text{avec } r = r_e + x. \quad (\text{TP4.3})$$

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

$$V_0(x) = \frac{k}{2} x^2 \quad (\text{TP4.4})$$

dont le paramètre k , connu aussi comme k_0 ou k_{xx} , est appelé *constante de force harmonique*. En mécanique classique, on obtiens les équations hamiltoniennes du mouvement pour les variables canoniques conjuguées x et p

$$\frac{dx}{dt} = \dot{x} = \frac{\partial H}{\partial p} = \frac{p}{\mu}, \quad \frac{dp}{dt} = \dot{p} = -\frac{\partial H}{\partial x} = -\frac{dV}{dx} = -F(x).$$

Ceci est un système des **É.D.O.s** du premier ordre, prêtes pour l'intégration numérique (spécifiquement, par les méthodes de *Euler* et de *RK* dans la sec. **TP5**). 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 .

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 longueur 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.

¹⁰Voir les cours de la mécanique quantique (Zhilinskiĭ, Dhont) et des vibrations (Hadj, Bocquet, Sadovskii) ainsi que l'appendice **TP4.1**

¹¹Huber and Herzberg [1979]; <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*

Choix de la fonction potentielle, calcul des paramètres. En exploitant la mode symbolique de MAXIMA, 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 la molécule de votre choix. Choisissez le potentiel le mieux adapté pour la description des petites vibrations autour r_e . Donnez l'approximation polynomiale ; déterminez les constantes de force harmonique k_0 et cubique k_{xxx} . Faites le dessin de différentes 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 trouver les valeurs numériques exactes r_1 et r_2 . (Alternativement utilisez la méthode de Newton étudiée en TP3.) Développez une procédure 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 $t = 0, \tau, 2\tau, \dots$

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

$$I = \frac{1}{2\pi} \int p dx = \frac{1}{2\pi} \int_0^T p \dot{q} dt.$$

(Ici l'intégrale est prise le long de la trajectoire pour un cycle.) Notez que le calcul de I peut être incorporé 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 différentes énergies dans l'intervalle $0 \leq E < D_e$, faites un tableau et les graphs.

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

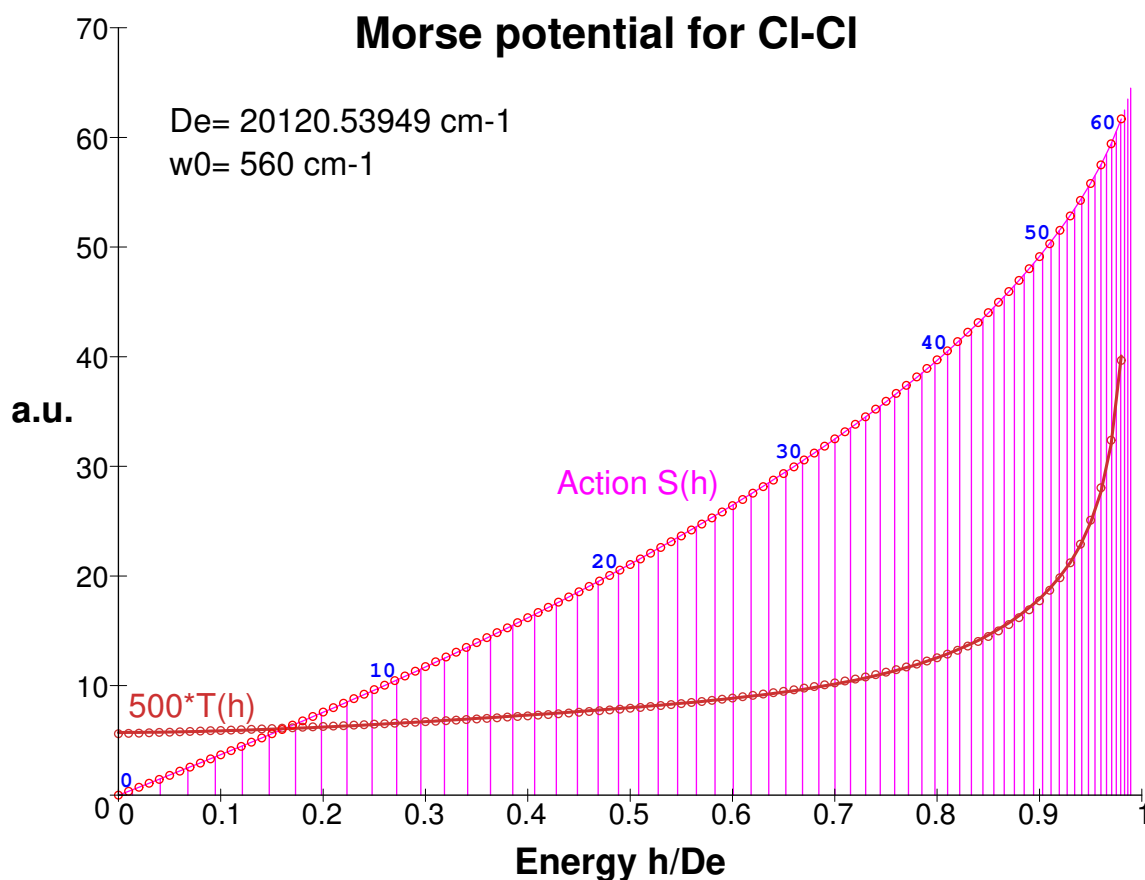
Calcul des énergies quantiques. On cherche les solutions E_n de l'équation de quantification

$$I(E_n) = n + \frac{1}{2} \quad \text{où } 0 \leq E_n < D_e \quad \text{et } n = 0, 1, 2, \dots$$

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 $T(E) = dI(E)/dE$ 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 $n = 0, 1, 2, \dots$. 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 $I(E)$.

TP4.1 Appendice : Potentiels moléculaires (B. I. Zhilinskiĭ)

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

Les données expérimentales Considérez les caractéristiques suivantes des 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 près de la position d'équilibre.
3. Énergie de la dissociation D_e , ou la différence $U(\infty) - U(r_e)$. Cette énergie est nécessaire pour séparer les atomes (briser la liaison chimique).
4. La masse réduite $\mu = m_1 m_2 / (m_1 + m_2)$ où m_1 et m_2 sont les masses des atomes dans le tableau périodique.

Les potentiels modèles

0. Sans concrétiser la forme du potentiel U , écrivez l'hamiltonien uni-dimensionnel H (l'énergie du système).
1. Concrétisez H en utilisant le potentiel harmonique, $U_0(r) = k(r - r_e)^2/2$, qui dépend de deux paramètres, la *constante de force* k , et r_e . Reliez 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 un choix des paramètres approprié ? Donnez la relation entre k et la fréquence de vibration.
2. Concrétisez H en utilisant le potentiel de *Kratzer*

$$U_K(r) = \frac{a}{x^2} - \frac{b}{r}$$

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. Concrétisez l'hamiltonien en utilisant le potentiel de *Lennard-Jones* dit «6-12» :

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

Ce potentiel dépend de deux paramètres A_6 et A_{12} . Pourquoi les constantes sont obligées d'être positives si vous modélisez 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 dépendent que de deux paramètres A_6 et A_{12} , trouvez la relation entre x_e, k_e, D_e imposée 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 , et x_0 . Exprimez pour ce potentiel 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	$\frac{\hbar^2}{2\mu x_e^2}, \text{ cm}^{-1}$	$D, \text{ cm}^{-1}$	α
H_2	60,8296	38292	1,440
HCl	10,5930	37244	2,380
I_2	0,0374	12550	4,954

Calculez 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. Représenter ce potentiel et le potentiel de Morse initiale sur le même graphique.

TP4.2 Questions théoriques sur le projet «vibrations moléculaires»

Vous avez 2h30. Aucun document (notes, topos, livres) ne sont autorisés. Une calculatrice simple suffit.

- Unités atomiques (u.a.) : expliquez le choix des unités microscopiques pour la distance, énergie, masse, action, charge. A noter : l'u.a. d'énergie et de longueur égalent 27.211 eV ou 219474.6 cm⁻¹ et 0.529 Å, respectivement.
- Quelles sont les différentes contributions dans l'énergie moléculaire. Quel(s) terme(s) est(sont) négligé(s) dans l'approximation Born-Oppenheimer. Pourquoi peut-on appliquer cette approximation ?
- Molécule. Classifiez les degrés de liberté d'une molécule quasi-rigide. Donnez le nombre de degrés de chaque type. Prenez les exemples de H₂, H₂O, CO, et CO₂. Pourquoi associe-t-on la molécule diatomique à un problème de Kepler ? Quels sont les degrés de liberté dans ce cas ?
- Molécule diatomique. Donnez l'allure de l'énergie potentielle $U(r)$ en fonction de la distance entre les atomes $r = |r_1 - r_2|$. Précisez la définition de l'énergie de dissociation D_e , de la distance d'équilibre r_e , et de la constante de force harmonique k_{xx} et cubique k_{xxx} . Pour quelles énergies h le mouvement classique et fini/infini ? Quel est l'aspect qualitatif du spectre des énergies quantiques correspondant ?

- Potentiel de *Kratzer*. Soit

$$U(r) = ar^{-2} - br^{-1}, \quad \text{avec } a, b > 0 \text{ et } r \geq 0.$$

Exprimez r_e , D_e , et k_{xx} en fonction des paramètres a et b .

- Molécule diatomique. Exprimez l'énergie vibrationnelle d'une molécule diatomique sans préciser la forme du potentiel $U(r)$. Précisez la relation entre la masse réduite et les masses de deux atomes. Donnez l'approximation harmonique et obtenez l'expression pour la fréquence harmonique ω_0 .
- Potentiel de *Morse*. Exprimez le potentiel de *Morse* $U(r)$ avec les paramètres r_e , α , D_e . Du point de vue qualitatif, quel est son défaut ? Trouvez la constante de force harmonique k_{xx} . Interpretez le paramètre α .
- Molécule diatomique. Utilisez le potentiel du *Morse* pour approximer l'énergie vibrationnelle de la molécule Cl-Cl. En sachant que pour cette molécule $D_e = 2.5$ eV, $r_e = 1.988$ Å, et $\omega_0 = 560$ cm⁻¹, trouvez la valeur numérique du paramètre α .
- Approximation harmonique. Quelle est la forme du potentiel $U(r)$? Pourquoi appelle-t-on cette approximation «harmonique» et ce système «linéaire». Toujours pour la molécule Cl-Cl, prenez l'équilibre pour zéro d'énergie et donnez les énergies quantiques dans l'approximation harmonique.
- Vibration anharmonique. Dans l'espace des phases $\mathbb{R}^2$, tracez la trajectoire pour une énergie fixe $h > 0$. Donnez l'allure correspondant $r(t)$. Comparez qualitativement avec la trajectoire du système linéaire. Expliquez le principe de la transformation Fourier discrète (TFD) et donnez l'aspect qualitatif du spectre Fourier de $r(t)$.
- Expliquez le principe de quantification de l'action (autrement connu comme Einstein-Brillouin-Keller). Appliquez ce principe à l'oscillateur harmonique et trouvez son spectre. Quelle serait la déformation de ce spectre pour le cas d'un oscillateur anharmonique ? Dans ce cas, pour le potentiel de *Morse*, le nombre des niveaux discrets, est-il fini ou infini ?
- Energie de rotation. Donnez l'expression de l'énergie de rotation. Quelles sont les énergies quantiques (le spectre) ?
- Énergie de rotation-vibration. Pour l'énergie potentielle des vibrations prenez votre potentiel de *Morse* $U(r)$. Quelle est la modification de la description dans le cas où la rotation est présent. Quels sont les (nouveaux) aspects qualitatifs du spectre quantique ? Expliquez l'effet de distortion centrifuge. Pour l'illustrer, trouvez r_e .

TP5 Intégration numérique des équations différentielles ordinaires

But de TP Écrire et tester une procédure en langage MAXIMA pour l'intégration numérique d'un système des équations différentielles ordinaires (É.D.O.) d'ordre 1 par la méthode d'Euler et ces améliorations. Comme un exemple concret, considérer le *pendule plan* d'amplitude large (voir les cours de la *Mécanique classique* et *quantique*, et des *Vibrations*).

TP5.1 Généralités mathématiques et l'application numérique

Rappelez la formule de *Leibnitz* définissant la dérivée $\dot{q} = dq/dt$. 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 ?

Notions de base sur la méthode d'Euler En fixant les conditions initiales $q_0 := q(0)$ à l'instant $t = 0$ et le pas en temps Δ_t suffisamment petit, on essaye de retrouver les valeurs $q(t)$ pour $t > 0$. Ainsi on trouve la *trajectoire*, une courbe $\mathbb{R} \rightarrow \mathbb{R}^2 : t \mapsto q(t)$ dans l'espace des phases de notre système des É.D.O.'s $\mathbb{R}^2$ avec les coordonnées (x, y) . Soit¹² $t_i = i \Delta_t$, $q_i = q(t_i)$ 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* $q(t_i + dt) = q_i + dq_i$ à l'instant $t = t_i$, $i = 1, 2, 3, \dots$ où $dq_i = f_i dt \approx f_i \Delta_t$ pour $dt \approx \Delta_t$. On effectue alors le *pas d'Euler*

$$q_{i+1} = q_i + f_i \Delta_t, \quad i = 0, 1, 2, \dots \quad (\text{TP5.1a})$$

Cette simple méthode donne la précision est d'ordre $O(\Delta_t^2)$, mais on observe aussi que les erreurs s'accumulent—ce qui est plus grave. Pour améliorer la *stabilité* par rapport à la trajectoire exacte, on modifie la procédure (TP5.1a) de façon suivante

$$\begin{aligned} t_{i+1/2} &:= t_i + \frac{1}{2} \Delta_t, & q_{i+1/2} &:= q_i + \frac{1}{2} f_i \Delta_t, \\ f_{i+1/2} &:= f(t_{i+1/2}, q_{i+1/2}), \\ q_{i+1} &:= q_i + f_{i+1/2} \Delta_t, & i &= 0, 1, 2, \dots \end{aligned} \quad (\text{TP5.1b})$$

où la dérivée dq/dt pour le pas final est calculée «au milieu» de l'intervalle Δ_t . La précision devient $O(\Delta_t^3)$ dans la méthode d'Euler-Cauchy (où effectue une ou plusieurs itérations optionnelles pour affiner la valeur $\tilde{f}_{i+1}$ de f au point final)

$$\tilde{q}_{i+1} = q_i + f_i \Delta_t, \quad \underbrace{\tilde{f}_{i+1} = f_i(t_{i+1}, \tilde{q}_{i+1}), \quad q_{i+1} = q_i + \frac{f_i + \tilde{f}_{i+1}}{2} \Delta_t}_{\text{itération } \tilde{q}_{i+1} = q_{i+1}}, \quad i = 0, 1, 2, \dots \quad (\text{TP5.1c})$$

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 $O(\Delta_t^{k+1})$, son pas $i = 0, 1, 2, \dots$ est effectué de façon suivante

$$\begin{aligned} K_1^{(i)} &= \Delta_t f(t_i, q_i), \\ K_2^{(i)} &= \Delta_t f(t_{i+1/2}, q_i + \frac{1}{2} K_1^{(i)}), \\ K_3^{(i)} &= \Delta_t f(t_{i+1/2}, q_i + \frac{1}{2} K_2^{(i)}), \\ K_4^{(i)} &= \Delta_t f(t_{i+1}, q_i + K_3^{(i)}), \end{aligned} \quad q_{i+1} = q_i + \frac{1}{6} (K_1^{(i)} + 2K_2^{(i)} + 2K_3^{(i)} + K_4^{(i)}) \quad (\text{TP5.2})$$

Programmation sous MAXIMA Dans le code L5 ci-dessous, chaque point (ou état) $\mathbf{p}$ dans l'espace des phases $\mathbb{R}^N$ du système de N É.D.O.'s est représenté par une liste de N éléments $[p_1, p_2, \dots, p_N]$. La procédure `ode_trace` permet de lancer des différents intégrateurs, notamment, `Es_step`, `Em_step`, ou `RK_step`, au «temps» $t_0 = 0$ avec le pas $\Delta_t = \text{step_size}$ depuis l'état initial $\mathbf{p}(t_0) = \mathbf{u}_0$. La fonction `ode_rhs` encode l'équation différentielle concrète à intégrer. Elle reçoit $\mathbf{p}$ et est expectée de retourner une liste $\dot{\mathbf{p}} = [\dot{p}_1, \dot{p}_2, \dots, \dot{p}_N]$. La procédure `ode_trace`, quant à elle, retourne la «trace» (i.e., la trajectoire) comme une «liste des listes» sous format $[[t_0, \mathbf{p}(t_0)], \dots, [t_{\text{nstep}}, \mathbf{p}(t_{\text{nstep}})]]$.

Choix de pas : Δ_t assure le calcul à la fois précis et économe ; divisez/multipliez Δ_t par deux et comparez vos résultats.

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

Listage L5 – Runge-Kutta.mac : intégrateurs É.D.O. et utilitaires sous MAXIMA pour TP5

```

/** Maxima 5.44 utilities for numerical o.d.e. integration **/
if not integerp(debug_level) then debug_level:1$ /* 0 skips all messages */
/* simple Euler's method (1st order in tau) */
Es_step(f,u,t, tau) := u + tau * f(u,t);
/* modified Euler's method (2nd order in tau) */
Em_step(f,u,t,tau) := block([k],
  k : tau * f(u,t),
  k : tau * f(u+k/2, t+tau/2),
  u + k
);
/* 4th order Runge-Kutta method */
RK_step(f,u,t,tau) := block([k1,k2,k3,k4],
  k1 : tau * f(u,t),
  k2 : tau * f(u + k1/2, t+tau/2),
  k3 : tau * f(u + k2/2, t+tau/2),
  k4 : tau * f(u + k3, t+tau),
  u + k1/6 + k2/3 + k3/3 + k4/6
);
/* the driver routine which integrates o.d.e.'s using different methods
   returns the list of nstep+1 trajectory points in the format [t,[q,p]] */
ode_trace(step_fun, step_size, nstep, ode_rhs, u0) := block(
  [t:0,u:u0, data:[]], /* initial conditions and storage */
  for i:0 thru nstep do (
    data : endcons([t,u], data), /* store current point */
    u : step_fun(ode_rhs, u, t, step_size),
    t : t+step_size /* step forward in time */
  ),
  return(data) /* return trajectory 0..nstep */
);
/* auxiliary utilities to extract data from the list returned by ode_trace */
phs_trace(data) := [discrete, makelist(last(d),d,data)]$
tmp_trace(data,i) := [discrete, makelist([d[1],d[2][i]],d,data)]$

/* for the list "data" in the format returned by ode_trace, find the crossings
   of the q=0 axis in any direction, i.e., negative from q>0 to q<0 or positive
   q<0 to q>0, and return the N segments of the trace bracketing the roots of
   q(t) as a straight list of 2N points with two points for each interval */
brack_roots(data) := block([roots : []],
  for i:2 thru length(data)-1 do (
    if data[i][2][1]>0 and data[i+1][2][1]<=0 or
       data[i][2][1]<0 and data[i+1][2][1]>=0 then
      roots : append([data[i],data[i+1]], roots)
  ),
  if debug_level > 0 and length(roots)/2 >= 1 then
    printf(true, " ~d step~:P, bracketed ~d root~:P in [~{~,3f~#[~;~,~]}]~%",
      length(data), length(roots)/2, makelist(t[1],t,roots)),
  return(roots) /* NB: empty list when no root found */
)$

root_tabs : 1e-6$ /* tolerance on t for roots of x(t) */
/* find bracketed roots using method 0 dichotomy, 1 trapeze, 2 Newton */
solve_roots(step_fun, ode_rhs, roots, maxiter, method) := block(
  [roots:roots, s:[], r, t, x],
  for k:1 thru length(roots)/2 while length(roots)>1 do block([t0:roots[1][1]],
    r : firstn(roots,2), roots : rest(roots,2),
    for i:1 thru maxiter do (
      block([a:r[1][2],ta:r[1][1],b:r[2][2],tb:r[2][1]],
        if method=2 then t : ta - a[1] / a[2]
        elseif method=1 then t : (ta*abs(b[1])+tb*a[1]) / (a[1]+abs(b[1]))
        else t : (ta+tb)/2,
        x : last(last(ode_trace(step_fun, t-ta, 1, ode_rhs, a))),
        if debug_level > 0 then
          printf(true, "~2d [~,3f : ~,5f : ~,3f] x(t)=~11,4,2,1e~%",i,ta,t,tb,x[1]),
          if x[1]<0 then r : [ r[1], [t,x] ] else r : [ [t,x], r[2] ]
        ),
        if abs(t-t0) < root_tabs then return(t),
        t0:t ),
      /* continue if t changes significantly */
      s : endcons([t,x],s)
    ),
    return(s) /* the list of roots found */
  )$

```

Notez, que la commande `batchload("Euler-RK.mac")` ; permet d'inclure le code TP5 ci-dessus dans des applications à condition que la liste `file_search_maxima` contient le chemin pour le répertoire où ce fichier se trouve.

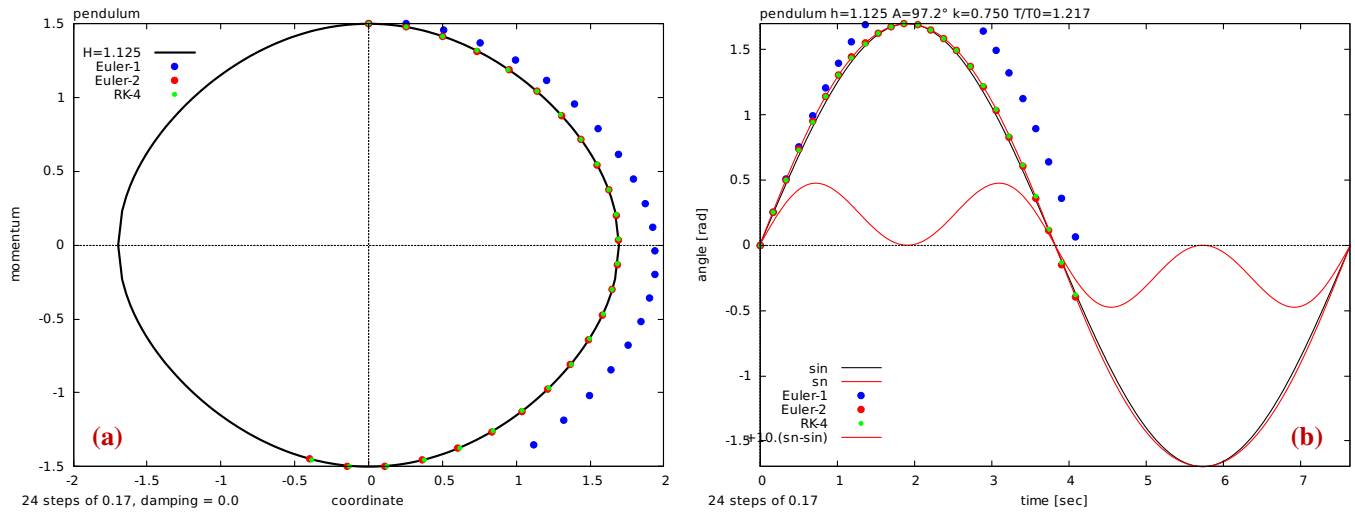


FIG. 3 – Trajectoire dans l'espace des phases (a) et la solution en fonction de temps (b) avec le programme L6 pour MAXIMA

TP5.2 Généralités physiques. Pendule plan

On dérive les équations du mouvement pour le pendule plan d'amplitude A arbitraire. Selon la mécanique *lagrangienne*, $\ddot{\varphi} + \omega_0^2 \sin \varphi = 0$ où $\omega_0 = \sqrt{g/l}$ peut être mise à 1 par un changement de l'échelle de temps $t \mapsto t/\omega_0$. En utilisant deux variables $x(t) = \varphi(t)$ et $y(t) = \dot{\varphi}(t)$, on obtient le système de deux É.D.O.'s d'ordre 1 :

$$\ddot{\varphi} + \sin \varphi = 0 \Rightarrow \begin{cases} \dot{x} = y \\ \dot{y} = -\sin x \end{cases} \Rightarrow \dot{q} = f(q), \text{ avec } q = \begin{pmatrix} x \\ y \end{pmatrix} \text{ et } f(q) = \begin{pmatrix} y \\ -\sin x \end{pmatrix}. \quad (\text{TP5.3})$$

Notez que l'énergie du système est donnée par son *hamiltonien*

$$H = E_{\text{cin}} + E_{\text{pot}} = \frac{1}{2} p_\varphi^2 + (1 - \cos \varphi) = \frac{1}{2} \dot{\varphi}^2 + 2 \sin^2 \frac{\varphi}{2}. \quad (\text{TP5.4})$$

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

L'amplitude A du pendule en régime oscillatoire, i.e., pour les valeurs $0 < h < 2$ de l'énergie (TP5.4), est la valeur maximale de $|\varphi(t)|$ qu'on observe aux *points de retour* avec $p_\varphi = \dot{\varphi} = 0$, où

$$1 - \cos A = 2 \sin^2 \frac{A}{2} = 2k^2 = h. \quad (\text{TP5.5})$$

Pour l'amplitude A suffisamment petite, nous pouvons utiliser le développement limité de l'énergie potentielle

$$E_{\text{pot}}(\varphi) = 1 - \cos \varphi = \frac{1}{2} \varphi^2 + O(\varphi^4) \quad (\text{TP5.6a})$$

et obtenir ainsi une *approximation harmonique* avec la pulsation propre $\omega_0 = 1$, la période $T_0 = 2\pi$, et la solution générale

$$\varphi(t) = A \sin(\omega_0(t - t_0)). \quad (\text{TP5.6b})$$

Ici t_0 est le temps de passage par l'axe vertical $\varphi = 0$ (i.e., par l'équilibre) vers les $\varphi > 0$ (i.e., au sens anti-horaire). Cependant, quand l'amplitude A augmente, on doit utiliser la solution générale exacte plus complexe

$$\varphi(t) = 2 \sin^{-1}[k \operatorname{sn}(\omega_0(t - t_0); k)] \quad \text{avec le module elliptique (ou excentricité) } k = \sin \frac{A}{2} \leq 1. \quad (\text{TP5.7a})$$

Dans cette solution, $\operatorname{sn}(x; k)$ est une *fonction elliptique de Jacobi*¹³ dont la période exacte (pour $k < 1$)

$$T = \frac{4}{\omega_0} K(k) = \frac{4}{\omega_0} \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-k^2 t^2)}} = T_0 \left[1 + \sum_{n=1}^{\infty} \left(\frac{(2n)!}{(2^n n!)^2} k^{2n} \right)^2 \right]. \quad (\text{TP5.7b})$$

est donnée par l'intégrale elliptique complète de 1er espèce¹⁴ $K(k)$. Par ailleurs, en *régime rotationnel*, i.e., pour $A = \pi$, $h > 2$, et $k > 1$, le pendule «tournoie» avec la période

$$T = \frac{2k^{-1}}{\omega_0} K(k^{-1}) \quad \text{où, comme auparavant, } k = \sqrt{\frac{h}{2}}. \quad (\text{TP5.7c})$$

¹³Sous MAXIMA, les valeurs de cette fonction sont accédées par `jacobi_sn(x, m)` avec paramètre $m: k^2$

¹⁴voir la note 9 en bas de la page 16

Listage L6 – Runge-Kutta-TP.mac : programme pour TP5 sous MAXIMA

```

3  /* TP on numerical integration of o.d.e. Maxima 5.44 D.Sadovskii 2026-09-24 */
batchload("Runge-Kutta.mac")$          /* load integration methods and drivers */
(odetype:"pendulum", damping_coeff:0)$    /* aka gamma or zeta*omega/2 */
5
/* set initial conditions for (q,p) at t=0, the number and size of time steps */
7 (init_cond : [0, 1.5], maxsteps : 24, stepsize : 0.17)$
if numberp(damping_coeff) and damping_coeff#0 then maxsteps : maxsteps*10 $
9
(
11  Ham : p^2/2 - cos(q) + 1,
    /* NB: for small q, this Hamiltonian approximates to q^2/2+p^2/2-1 */
13  Ham0: factor(taylor(Ham,q,0,2)),
    printf(true," Hamiltonian equations of motion for the mathematical pendulum~%"),
15  printf(true," H = ~a -> ~a for q=0~%", Ham, Ham0 ),
    odefun(u,t) := [ u[2], -sin(u[1]) - 2*damping_coeff*u[2] ]
17 )$

19 data_Es : ode_trace(Es_step, stepsize, maxsteps, odefun, init_cond )$
data_Em : ode_trace(Em_step, stepsize, maxsteps, odefun, init_cond )$
21
data_RK : ode_trace(RK_step, stepsize, maxsteps, odefun, init_cond )$
23 roots_RK : solve_roots(RK_step, odefun, brack_roots(data_RK), 10, 1);
for r in roots_RK do
25  printf(true," t=~ , 4f (q,p)=~{~,4f~# [~,~,~]~}% ", r[1], r[2])$
  ( /* phase space plot */
27  phs_plots : makelist( phs_trace(d), d, [data_Es, data_Em, data_RK] ),
    phs_style : [[points,2,1,1], [points,2,2,1], [points,1.25,3,1]],
29  phs_keys : ["Euler-1","Euler-2", "RK-4"],
    preamble : "set key at graph 0.22,0.95",
31  preamble : concat(preamble,"; set linetype 2 lc rgb 0x0057b7"),
    preamble : concat(preamble,"; set linetype 3 lc rgb 0xffdd00"),
33  if damping_coeff # 0 then
    preamble : concat(preamble, printf(false,"; set label 103 'damping = ~,3f' at graph 1,1 right offset
      0,0.75 tc rgb 0x080808",damping_coeff)),
35  preamble : concat(preamble, printf(false,"; set label 101 '~d steps of ~f' at graph 0,0 offset -6,-2.5",
    maxsteps,stepsize))
  )$
37 /* compute the exact trajectory (possibly, at the same points in time) */
block( [qstep,mstep : 63, ratprint:false, solvetrigwarn:false],
39  h : subst([q:=init_cond[1],p:=init_cond[2]],Ham), /* value of H = energy */
    m_val : h/2, k_val : sqrt(m_val), /* elliptic modulus k */
41  T_val : 4*elliptic_kc(m_val), /* exact period 4*K(k) */
    w : 2*%pi/T_val, /* oscillation frequency */
43  x_sol(t) := 2*asin(k_val*jacobi_sn(t,m_val)), /* exact solution for x(t) */
    x_sol0(t) := A_val*sin(w*t), /* sin of exact period */
45  /* positive branch of constant h-level set of H as function p(q) > 0 */
    p_sol : first(sublist( solve(Ham = h,p), lambda([p],subst(q=0,rhs(p))>0))),
47  s : first(solve(subst([p=0,q=q0],Ham=h),q0)), /* turning point gives A */
    A_val : float(rhs(s)),
49  printf(true," ~a energy h=~f, turning at ~a = ~,3f rad = ~,2f deg~%",
    odetype, h, s, A_val, A_val/%pi*180 ),
51  printf(true," elliptic modulus k=~,4f and half-period T=~,4f~%", k_val, T_val/2),
    preamble : concat(preamble, printf(false,"; set label 100 '~a h=~f A=~,1f k=~,3f T/T0=~,3f' at graph 0,1
      offset 0,0.75",
53  odetype, h, A_val/%pi*180, k_val,T_val/(2*%pi))),
    qstep : subst(s,q0 / mstep),
55  data_H0 : makelist( [qstep*i,subst(q=qstep*i,rhs(p_sol))], i, 0, mstep),
    data_H0 : append( data_H0, makelist( [p[1],-p[2]], p, rest(reverse(data_H0)))) ),
57  data_H0 : append( data_H0, makelist( [-p[1],p[2]], p, rest(reverse(rest(data_H0)))) ),
    phs_plots : cons( [discrete, data_H0], phs_plots),
59  phs_style : cons( [lines,2,-1], phs_style),
    phs_keys : cons(printf(false,"H=~,3f",h), phs_keys)
61 )$
plot2d( phs_plots, cons(style, phs_style), cons(legend, phs_keys),
63  [gnuplot_preamble, preamble],
    [x,-2.0,2.0], [xlabel, "coordinate [rad]"],
65  [y,-1.51,1.51], [ylabel, "momentum"]
  );
67 block( [k:k_val,m:k_val^2,mstep : 255,stp,zoom:10], /* q(t) plot */
    read("Hit any key followed by ; or $ to see the q(t) plot"),
69  tmp_plots : makelist( tmp_trace(d,1), d, [data_Es, data_Em, data_RK] ),
    tmp_style : [[points,2,1,1], [points,2,2,1], [points,1.25,3,1]],
71  tmp_keys : ["Euler-1","Euler-2", "RK-4"],
    stp : T_val/mstep, data_H0 : makelist( [stp*i,x_sol(stp*i)], i, 0, mstep),

```

suite au listage L6...

```

84 tmp_plots : cons([discrete, data_H0], tmp_plots),
tmp_style : cons([lines,1,2],tmp_style),
tmp_keys : cons("sn",tmp_keys),
86 tmp_plots : cons(x_sol0(t), tmp_plots),
tmp_style : cons([lines,1,-1],tmp_style),
88 tmp_keys : cons("sin",tmp_keys),
tmp_plots : endcons((x_sol(t)-x_sol0(t))*zoom, tmp_plots),
90 tmp_style : endcons([lines,1,2],tmp_style),
tmp_keys : endcons(sprintf(false,"%~.0f(sn-sin)",zoom),tmp_keys),
92 preamble : concat(preamble, "; set key at graph 0.22,0.05 bottom")
)$
94 plot2d( tmp_plots, [t,0,T_val], [y,-A_val,A_val],
cons(style, tmp_style), cons(legend, tmp_keys),
96 [gnuplot_preamble, preamble],
[xlabel, "time [sec]"],
98 [ylabel, "angle [rad]"]
);

```

Comparaison de différentes méthodes dans l'espace des phases Dans la figure 3 ci-dessus, la courbe grise, et les cercles bleus, rouges, et verts représentent, respectivement, la trajectoire exacte (le niveau constant de l'énergie correspondant aux conditions initiales $x = 0$, $\dot{x} = 1.5$, la méthode d'Euler (TP5.1a), la méthode d'Euler modifiée (TP5.1b), et la méthode RK-4 ; les deux dernières étant très proches de la solution exacte.

Exercice TP5.1 (amortissement). Incorporez dans les O.D.E. du pendule (TP5.3) le terme représentant le frottement (ou, plus généralement, une perte d'énergie). Que observez vous dans l'espace de phases ?

TP5.3 Recherche de la période et de l'action. Quantification

Tant que l'amortissement (voir `damping_coeff` dans le programme L6) reste nul et, par conséquent, l'énergie totale est conservée, la trajectoire $\gamma(t) : t \mapsto (\varphi(t), \dot{\varphi}(t))$ revient au point de départ (voir `init_cond`) après un certain temps T appelé la période. Notons, que grâce aux symétries $\varphi \mapsto -\varphi$ et $\dot{\varphi} \mapsto -\dot{\varphi}$ de $\gamma(t)$, il suffit de la suivre seulement pour un quart de sa période. La détermination numérique de la valeur de T se passe en deux étapes : (1) «encadrement» de la racine de x ou $\dot{x}$, voir la procédure `brack_roots` du listage L5, et (2) rectification itérative selon la/les méthode(s) de TP3 réalisé dans `solve_roots`.

Exercice TP5.2 (action). Revenez à l'équation (TP3.3a) de l'exercice TP3.3. Calculez numériquement l'intégrale

$$I(t) = \frac{1}{2\pi} \int_{\gamma(t)} p_{\varphi} d\varphi = \frac{1}{2\pi} \int_{\gamma(t)} \dot{\varphi}^2 dt.$$

Consigne pour l'exercice TP5.2 : à chaque pas d'intégration, les valeurs de $\dot{\varphi}$ sont désormais disponibles. Il suffit donc ajouter une 3me variable d'intégration et suivre la trajectoire $\gamma(t) : t \mapsto (\varphi(t), \dot{\varphi}(t), I(t))$ pour avoir les valeurs instantanées de I

```
odefun(u,t) := [ u[2], -sin(u[1]), u[2]^2 ]; init_cond : [0, 1.5, 0];
```

Exercice TP5.3. Etudiez la période T et l'action $S := I(T)$ en fonction de l'énergie totale $h > 0$ dans les domaines d'oscillation $h < 2$ et de rotation $h > 2$. Vérifiez que $T(h) = dS(h)/dh$.

Consigne pour l'exercice TP5.3 : on se propose d'avancer $h > 0$ en pas suffisamment petits (par rapport à l'énergie de l'équilibre instable à $\varphi = \pi$), spécifiquement, $\Delta h = 0.1$ ou 0.2 . À chaque pas, il faut s'assurer que le nombre de points de trajectoire `maxsteps` est suffisant pour atteindre le premier changement de signe de φ (demi-période $T/2$) ou de $\dot{\varphi}$ ($T/4$). Si `maxsteps` est grand, envisagez d'ajouter une option d'arrêt au changement de signe de φ et/ou de $\dot{\varphi}$ à la procédure `ode_trace` dans le code L5 pour éviter l'intégration longue et inutile. Envisagez également d'ajouter une option de la recherche des racines de $\dot{\varphi}$ à la procédure `solve_roots`. Comparez vos résultats numériques aux formules théoriques (TP5.7b), et (TP5.7c) de la sec. TP5.2, et (TP3.3a).

Exercice TP5.4 (quantification sémi-classique WKB/EBK). Si le pendule dans la sec. TP5.2 est un système quantique, l'énergie E_n de ses états (dit «niveaux») et discrétisée, voir l'exercice TP3.3. Trouvez les E_n numériquement. Trouvez le nombre des niveaux oscillatoires ? Observez et expliquez la dégénérescence des niveaux rotationnels pour $E \gg 2$.

Consigne pour l'exercice TP5.4 Dans l'exercice TP5.3, la fonction empirique $S(h)$ est obtenue comme un tableau de points pour certaines valeurs de h . Modélisez (interpolez) $S(h)$ par la méthode M.M.C. de la sec. TP2, et utilisez la méthode itérative de TP3 pour chercher la solution de $S(h)/\hbar = (n + \frac{1}{2})$. Notez, que $S(h)$ est continue et monotone, mais dS/dh est discontinue dans $h = 2$.

TP5.4 Etude de polynômes d'Hermite

Comme une application intéressante, considérez les solutions de l'É.D.O. d'ordre 2

$$\frac{d^2 H_n}{dx^2} - 2x \frac{dH_n}{dx} + 2n H_n(x) = 0 \Rightarrow \mathbf{q}(x) := \begin{pmatrix} H_n(x) \\ dH_n(x)/dx \end{pmatrix} \text{ et } f(x, \mathbf{q}) := \begin{pmatrix} q_2(x) \\ 2x q_2 - 2n q_1 \end{pmatrix} \quad (\text{TP5.8})$$

définissant les polynômes d'Hermite $H_n(x)$ d'ordre n qui satisfont les relations de récurrence (vérifier avec MAXIMA)¹⁵

$$H_{n+1}(x) = 2x H_n(x) - H'_n(x) = 2x H_n(x) - 2n H_{n-1}(x). \quad (\text{TP5.9})$$

Ces polynômes jouent un rôle très important en *physique quantique*. Spécifiquement, la fonction d'onde $\psi_n(x)$ de l'oscillateur harmonique, l'un des systèmes quantique le plus fondamentaux, s'exprime comme produit de $H_n(x)$ et la fonction «cloche» *Gaussienne* (fois une constante de normalisation¹⁶)

$$\psi_n(x) = H_n(x) \exp(-x^2/2), \quad x \in \mathbb{R} \text{ et } n = 0, 1, 2, \dots \quad (\text{TP5.10a})$$

Cette fonction représente l'état propre $|n\rangle$ de l'oscillateur avec la valeur propre $\lambda_n = n + \frac{1}{2}$. Pour le démontrer (avec MAXIMA), rappelons que l'opérateur hamiltonien (l'énergie en régime stationnaire) de l'oscillateur harmonique est

$$\hat{\mathcal{H}} = \frac{1}{2} \hat{p}^2 + \frac{1}{2} \hat{x}^2 = -\frac{1}{2} \Delta_x + \frac{1}{2} x^2 = -\frac{1}{2} \frac{d^2}{dx^2} + \frac{1}{2} x^2, \quad (\text{TP5.10b})$$

et agissons avec cet opérateur sur ψ_k et en utilisant la relation de récurrence (TP5.9) pour simplifier le résultat.

```
rec : hermite(k,x) = 2*x*hermite(k-1,x) - 2*(k-1)*hermite(k-2,x);
factor(subst(solve(rec,hermite(k-2,x)),expand(-diff(psi(k,x),x,2)/2 + x^2/2*psi(k,x))));
```

Notons, qu'il s'agit du calcul formel, avec k représentant tout ordre possible. Ainsi nous vérifions que

$$\hat{\mathcal{H}} \psi_k(x) = \left(k + \frac{1}{2}\right) \psi_k(x) \quad (\text{TP5.10c})$$

```
# 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 calcule la trajectoire exacte (pour les mêmes valeurs de x)

```
# 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;
```

TP5.5 Présentation des résultats, graphisme

Faire les graphs sous MAXIMA est toujours longue quand on cherche la perfection ...

¹⁵Notez, qu'ici, x correspond à la variable de contrôle t ou «temps» dans la sec. TP5.1, et que dH_n/dx est noté $H'_n(x)$ au lieu de $\dot{H}_n(t)$

¹⁶Calculez la norme $\int_{-\infty}^{\infty} \psi_n(x)^2 = 2^n n! \pi^{1/2}$ pour différentes valeurs de n avec MAXIMA

TP6 Transformation de Fourier discrète

But de TP Écrire et tester une procédure en langage MAXIMA permettant de transformer une fonction $g(x)$ définie par un tableau $g(x_1), g(x_2), \dots$ sur une grille des points équidistants $x_1, x_2, \dots$ vers un 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 Considérez la fonction $g(x)$ échantillonnée pour des valeurs équidistantes de x , telles que

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

avec l'intervalle d'échantillonnage Δ et le taux d'échantillonnage Δ^{-1} . On définit aussi la fréquence critique de Nyquist

$$f_c = \frac{1}{2\Delta} \quad (\text{TP6.1})$$

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 vrai 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

$$f_n = \frac{n}{N\Delta}, \quad h_n = \Delta h(f_n) \quad \text{où } n = -\frac{N}{2}, \dots, \frac{N}{2}, \quad \text{et } h_{-\frac{N}{2}} \equiv h_{\frac{N}{2}} = 2\Delta h(f_c). \quad (\text{TP6.2})$$

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 discretisation de l'intégrale *Fourier*

$$h(f) = \int_{-\infty}^{\infty} g(x) \exp(2\pi i f x) dt \Rightarrow h(f_n) = \sum_{k=0}^{N-1} g_k \exp(2\pi i f_n x_k) \Delta = \Delta \sum_{k=0}^{N-1} g_k \exp\left(\frac{2\pi i}{N} kn\right). \quad (\text{TP6.3})$$

Ainsi cette transformation donne N nombres

$$h_n = \sum_{k=0}^{N-1} g_k \exp\left(\frac{2\pi i}{N} kn\right) \quad (\text{TP6.4})$$

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 $[-\frac{N}{2}, \dots, \frac{N}{2}]$ comme nous avons fait auparavant, il est possible et pratique d'utiliser $n = 0, \dots, N-1$ comme nous avons fait pour k . En adhérant à une telle convention, il faut se rappeler de la correspondance suivante

$$\begin{array}{ll} n = 0 & f = 0 \\ 1 \leq n \leq \frac{1}{2}N - 1 & 0 < f < f_c \\ n = \frac{N}{2} & f = \pm f_c \\ \frac{1}{2}N + 1 \leq n \leq N - 1 & -f_c < f < 0 \end{array}$$

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

$$g_k = \frac{1}{N} \sum_{n=0}^{N-1} h_n \exp\left(-\frac{2\pi i}{N} kn\right) \quad (\text{TP6.5})$$

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 (TP6.5) :

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

où les fréquences f_n sont données par (TP6.2). Par conséquent,

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

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 MAXIMA. Regardez la section correspondant du TP précédent pour des informations générales. Commencez la procédure de la transformation directe de façon suivante

```
dDFT(g) := block( [ N : length(g), n, h : [], ... ],
  . . .
  return(h) );
```

et utilisez (TP6.4) pour remplir les entrées $h[n]$ de vecteur h . La procédure `iDFT` de la transformation inverse s'écrit de même façon et traduisez (TP6.5). Pour tester vos procédures prenez un vecteur de $N = 8$ (ou 16, 32 etc) points, e.g.,

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

et comparez s et g .

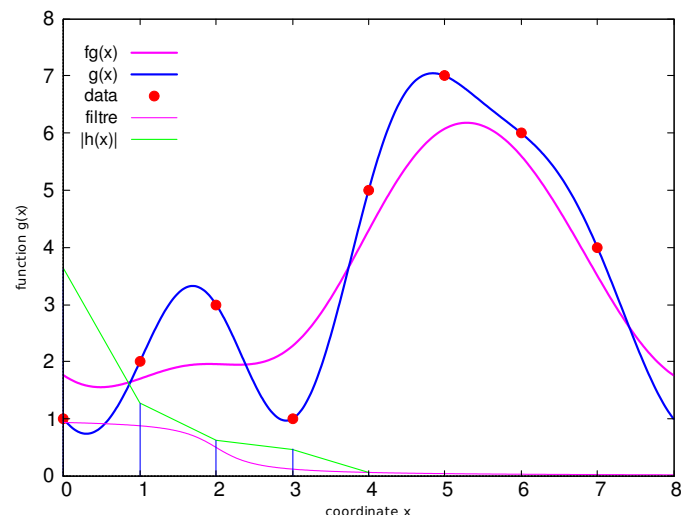


FIG. 4 – Données $\{g_k\}$, leur image $\{h_n\}$, et la fonction $g(x)$.

Pour visualiser la liste g des valeurs $g_k = g(x_k)$ comme des points $[x_k, g_k]$, employez la commande suivante

```
N:length(g); plot2d( makelist( [k, g[k]], k, 1, N) );
```

et complétez ce graphe avec celui de la fonction $g(x)$ reconstruite selon (TP6.5) comme une expression `gfun`

```
plot2d([gfun, makelist( [k, g[k]], k, 1, N) ], [x, 1, N]);
```

Enfin, pour un teste ultime, faites appel à la procédure `fft` ou `real_fft` (pour une fonction réelle g) de MAXIMA,

```
load("fft")$ _h : real_fft(g);
```

et comparez ses résultats numériques `_h` avec la liste de vos valeurs dans `float(h)`.

Exercice TP6.1 (questions). Quelle est la fréquence de Nyquist dans votre exemple ? Les valeurs $h[n+1] = h_n$, sont elles réelles ? Vérifiez que h_n et h_{N-n} avec $n = 1, \dots, \frac{1}{2}N - 1$ sont conjuguées. Pourquoi ? Quelle est la taille de la liste `_h` retournée par `real_fft` ? Pourquoi ? Démontrez que votre fonction reconstruite `gfun` est réelle.

Exercice TP6.2 (filtrage). Réalisez un filtre «passe bas». Expliquez le choix de la fréquence de coupure. Faites le graph de la fonction transformée par le filtre ensemble avec la fonction reconstruite complète `gfun` (fig. 4) et les données g .

Exercice TP6.3 (mouvement anharmonique). Retournez au TP5.1 (intégration numérique) et la fig. 3 où on étudie la trajectoire $\varphi(t)$ et la période T des oscillations (et rotations) d'un pendule à l'amplitude arbitraire $A \in [0, \pi]$. Pour une valeur de l'énergie suffisamment large, suivez $\varphi(t)$ pendant une période et tablez la en 2^k points avec $k = 5, 6$, ou même 7. Par la suite, trouvez le spectre Fourier de $\varphi(t)$ avec `real_fft` et interprétez le. Spécifiquement, identifiez la fréquence «principale» (cf la courbe «sin» dans la fig. 3b) et ses «harmoniques» dues à la caractère nonlinéaire de cette É.D.O..

Listage L7 – DFT.mac : programme pour TP6 (transformation Fourier discrète) sous MAXIMA

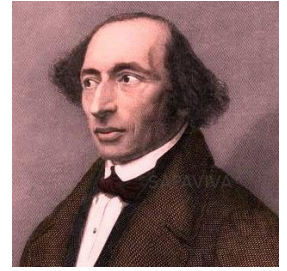
```

/*+ direct/inverse discrete Fourier transform (DFT) (c) 2026 Sadovskii +*/
2 batchload("license.mac");
load("fft"); /* Fast Fourier Transform library */
4
dDFT(g) := block( [ N : length(g), h:[ ] ],
6   for n:0 thru N-1 do block( [r:0],
8     for k:0 thru N-1 do r : r + g[k+1]*factor(exp( 2*pi*i*k*n/N)),
10    h : endcons(expand(r),h) /* spectrum element h_n */
12  ),
14  return(h)
15 )$
iDFT(h) := block( [ N : length(h), g:[ ] ],
16   for k:0 thru N-1 do block( [r:0],
18     for n:0 thru N-1 do r : r + h[n+1]*factor(exp(-2*pi*i*k*n/N)),
20    g : endcons(expand(r/N),g) /* function sample g_k */
22  ),
24  return(g)
25 )$
Fn(n,N) := ( /* index n in [0..N] -> frequency f_n */
26   if n=0 then 0 /* constant (dc) part */
27   elseif n>0 and n<N/2 then n/N
28   elseif n>N/2 and n<N then n/N-1
29   else n/N
30 )$
/* equidistantly sampled function (Delta=1) to test the dDFT/iDFT procedures */
31 g : [1,2,3,1,5,7,6,4];
/* number of samples, frequency spectrum, and its real and imaginary parts */
32 (N : length(g), h : dDFT(g), hRe : realpart(h), hIm : imagpart(h));
33 block([d:unique(imagpart(g))], /* determine whether function g(t) is real */
34   _h : if length(d)=1 and first(d)=0 then real_fft(g) else fft(g) );
35 block([d:unique(iDFT(h)-g)],
36   if length(d)=1 and first(d)=0 then
37     printf(true," TEST: inverse FT of FT(g) restored g=~a of ~d samples~%",g,N)
38   );
39
40 block([n:0], /* zero (dc) frequency h has only Re part */
41   printf(true," ~d h~d = ~16a = ~a~%", Fn(n,N), n, h[n+1]/N, _h[n+1])
42 )$
43 for n:1 thru N/2-1 do /* compare complex components of h */
44   printf(true," ~d/~d h~d ~c h~d = ~16a + ~21a = ~,4f~7,4@f*i ~a~%",
45     Fn(n,N)*N, N, N-n, if h[n+1] = conjugate(h[N-n+1]) then "=" else "#", n,
46     hRe[n+1]/N, %i*(hIm[n+1]/N), realpart(_h[n+1]), imagpart(_h[n+1]),
47     if abs(float(h[n+1]/N - _h[n+1]))<1e-8 then "OK" else "??" );
48 block([n:N/2], /* Nyquist frequency h has only Re part */
49   printf(true," ~d/~d h~d = ~16a = ~a~%", Fn(n,N)*N, N, n, h[n+1]/N, _h[n+1])
50 )$
51 /* smooth reconstruction of real function g(x), cf frequencies Fn(n,N) */
52 (f0:2, fslope:2.5, f(w) := 1/2-atan(fslope*(w-f0))/pi)$ /* attenuator */
53 block([w:-2*pi*i, ratprint:false],
54   gfun : h[N/2+1]*( exp(w*(1/2)*x) + exp(-w*(1/2)*x) )/2
55     + sum( h[n+1]*exp(w*(n/N)*x), n, 0, N/2-1)
56     + sum( h[n+1]*exp(w*(n/N-1)*x), n, N/2+1, N-1),
57   gfun : trigrat(expand(demoivre(gfun / N))),
58   ffun : h[N/2+1]*f(N/2)*( exp(w*(1/2)*x) + exp(-w*(1/2)*x) )/2
59     + sum( h[n+1]*f(n)*exp(w*(n/N)*x), n, 0, N/2-1)
60     + sum( h[n+1]*f(N-n)*exp(w*(n/N-1)*x), n, N/2+1, N-1),
61   ffun : float(expand(trigrat(demoivre(ffun / N))))
62 )$ print(" plotting ...");
63 s : makelist([n,abs(_h[n+1])/(if n=N/2 then 2 else 1)],n,0,N/2)$ /* spectre */
64 plot2d( [ffun, float(gfun), [discrete, makelist([k-1,g[k]],k,N)],
65   f(x), [discrete, s],[discrete, s]
66 ], [x,0,N],
67   [style,[lines,2,4],[lines,2,1],[points,3,2,1],[lines,1,4],[lines,1,3],[impulses,1]],
68   [gnuplot_preamble,
69     "set tics font 'Helvetica,14'; set key top left at graph 0,.95 font 'Helvetica,12';",
70   [legend, "fg(x)", "g(x)", "data", "filtre", "|h(x)|", ""],
71   [xlabel, "coordinate x"],
72   [ylabel, "function g(x)"]
73 );

```

TP7 Calcul des valeurs propres

But de TP Écrire et tester une procédure en langage MAXIMA permettant de trouver les valeurs propres d'une matrice symétrique réelle en utilisant la méthode de *Jacobi*¹⁷ (1846).



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

TP7.1 Notions de base sur la méthode de Jacobi

Afin de diagonaliser une matrice réelle symétrique A de dimension k la méthode de *Jacobi* déploie une séquence des rotations [Press et al., 2002, Chap. 11. Eigensystems]

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

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

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

dont les éléments diagonaux $R(p, q)_{pp} = R(p, q)_{qq} = c = \cos \phi$, le reste des éléments diagonaux égalent 1, et les seuls deux éléments 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 concrètement, on obtient¹⁸

$$A'_{rp} = cA_{rp} - sA_{rq}, \quad A'_{rq} = cA_{rq} + sA_{rp}, \quad r \neq q, r \neq qp \quad (\text{TP7.1a})$$

$$A'_{pp} = c^2 A_{pp} + s^2 A_{qq} - 2s c A_{pq} \quad A'_{qq} = s^2 A_{pp} + c^2 A_{qq} + 2s c A_{pq} \quad (\text{TP7.1b})$$

$$A'_{qp} = (c^2 - s^2)A_{pq} + s c (A_{pp} - A_{qq}) \quad (\text{TP7.1c})$$

Puisque le but de la transformation est d'éliminer l'élément A'_{qp} , on récrit (TP7.1c) sous forme

$$t^2 + 2t\theta - 1 = 0, \quad \text{avec} \quad \theta = \frac{A_{qq} - A_{pp}}{2A_{pq}} \quad \text{et} \quad t = s/c = \tan \phi,$$

et on trouve

$$t = \frac{s}{c} = \frac{\text{sgn}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}}, \quad c = \frac{1}{\sqrt{1 + t^2}}, \quad s = t c, \quad \tau = \tan \frac{\phi}{2} = \frac{s}{1 + c}. \quad (\text{TP7.2})$$

Par conséquent,

$$A'_{pq} = 0, \quad (\text{TP7.3a})$$

$$A'_{pp} = A_{pp} - t A_{pq}, \quad A'_{qq} = A_{qq} + t A_{pq}, \quad (\text{TP7.3b})$$

$$A'_{rp} = A_{rp} - s (A_{rq} + \tau A_{rp}), \quad A'_{rq} = A_{rq} + s (A_{rp} - \tau A_{rq}). \quad (\text{TP7.3c})$$

Quelques conseils de programmation en MAXIMA Utilisez les TP précédents pour des informations générales. Notez que la transformation peut être effectuée sur place en utilisant que la moitié de A . Commencez par écrire une procédure de la transformation $R(p, q)$ (rotation de *Jacobi*) basée sur (TP7.2) et (TP7.3). Par la suite programmez une 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 une boucle générale avec le critère d'arrêt

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

Comptez le nombre des passages de cette dernière boucle. Présentez vos résultats. Comparez les avec les valeurs propres obtenues en utilisant la procédure standard `eigenvalues` ou `eigens_by_jacobi` de MAXIMA.

¹⁷Carl Gustav Jacob Jacobi 1804–1851, mathématicien allemand

¹⁸n'oubliez pas que A et A' sont symétriques, par exemple $A_{ij} = A_{ji}$

Listage L8 – `Jacobi.mac` : programme pour TP7 (valeurs propres) sous MAXIMA

```

/*+ diagonalize real symmetric matrices by the Jacobi method (c) 2026 Sadovskii +*/
2 maxima_this_rev : "(c)DS 20261018"$ batchload("license.mac")$
debug_level : 1$ /* set to 0 to skip all extra debugging msg */
4 rowdim(A) := if matrixp(A) then length(first(args(A))) else 0 $

6 Jrot(A,q,p,eps) := block( /* perform a single Jacobi rotation R(q,p) */
    [theta, t, s, c, tau, r, a], /* unless the element q,p is already small */
8     if abs(A[q][p]) < eps then return(0),
    theta : (A[q][q]-A[p][p]) / (2*A[q][p]),
10    if debug_level > 1 then print(A[q][p],theta),
    t : float(1/(abs(theta)+sqrt(1+theta^2))), if theta < 0 then t:-t,
12    A[p][p]:A[p][p]-t*A[p][q],
    A[q][q]:A[q][q]+t*A[p][q],
14    A[p][q]:0,
    A[q][p]:0,
16    c : 1/sqrt(1+t^2),
    s : t*c,
18    tau:s/(1+c),
    if debug_level > 1 then print(t,s,c, tau),
20    for r:1 thru rowdim(A) do (
        if r#p and r#q then block( [a:A[r][p]],
22            A[r][p]: a - s*(A[r][q]+a*tau), A[p][r]:A[r][p],
            A[r][q]:A[r][q] + s*(a-tau*A[r][q]), A[q][r]:A[r][q]
24        )
    ),
26    return(1)
) $

28 Jdiag(A,releps) := block( /* diagonalize matrix A in-place */
30    [n:rowdim(A), Nmax:10, eps : float(mat_norm(A,frobenius))/rowdim(A)*releps, fpprintprec:5],
    if debug_level > 0 then print(A, "abs precision",eps),
32    for i from 1 thru Nmax do block( [r:0], /* count iterations i */
        for p:2 thru n do
34            for q:1 thru p-1 do (
                r : r + Jrot(A,p,q,eps), /* count rotations r */
36                if debug_level > 1 then print("no of rotations", r,p,q)
            ),
38            if r = 0 then return(r), /* stop if no rotations were needed */
            if debug_level > 0 then print(A,i,r)
40        ), /* A becomes diagonalized */
        return(makelist(A[i][i],i,1,n)) /* return diagonal elements of new A */
42    ) $

44 A:matrix([1,5,7],[5,2,8],[7,8,3])$ A0:copymatrix(A)$
/* compute eigenvalues of test matrix A using Maxima library function and Jdiag */
46 print("A =", A, "lambda =", first(eigens_by_jacobi(A))), fpprintprec=5$
s : Jdiag(A0,1e-6), debug_level=0$
48 print(A0, " our s =", s), fpprintprec=5$

```

Exercice TP7.1 (spectre quantique). Comme une application physique, considérez le calcul des valeurs propres de l'hamiltonien quantique du pendule plan dont l'analogue classique a été étudié dans la sec. TP5.2, voir les exercices TP5.2, TP5.3, et TP5.4. Traitez $\hbar$ comme un paramètre permettant d'obtenir un spectre vibrationnel suffisamment dense. Dessinez quelques fonctions propres $\psi(\theta)$.

Consignes pour l'exercice TP7.1 : L'analogue quantique de (TP5.4) avec $\theta := \varphi$ est un opérateur différentiel

$$\hat{H} = -\frac{\hbar^2}{2} \frac{d^2}{d\theta^2} - \cos \theta + 1 \quad (\text{TP7.4})$$

dont le seul «paramètre» $\hbar$ contrôle le nombre des valeurs propres vibrationnelles (aux énergies au dessous de 2). Ses fonctions propres peuvent être exprimées en utilisant les fonctions de Mathieu [Condon, 1928, Aldrovandi and Ferreira, 1980]. Pour nous, surtout que nous ne sommes intéressés principalement que dans les valeurs propres de (TP7.4), une solution numérique serait parfaitement suffisante, avec l'avantage de pouvoir traiter facilement la symétrie des solutions.

L'équation de Schrödinger (stationnaire) $\hat{H}\psi = E\psi$ pour les fonctions d'onde propres ψ et les énergies quantiques (valeurs propres) correspondantes $E \in \mathbb{R}$ de (TP7.4) peut être résolu numériquement par la méthode spectrale. Spécifiquement, il s'agit d'une application de la théorie de Sturm–Liouville de XIXe siècle à une É.D.O. d'ordre 2 sur l'espace $\mathbb{S}^1$. Cette théorie [Courant and Hilbert, 1953], exploite l'existence sur tout espace compact P d'une base discrète (aka comptable) des fonctions $\psi_k : P \rightarrow \mathbb{R}$ permettant de représenter tout autre ψ comme une combinaison (série) de ψ_k . Et, cerise sur le gâteau, pour $P = \mathbb{S}^1$, il s'agit, bien évidemment, de la série de Fourier déjà étudiée dans le TP6.

Nous tenons compte de la symétrie de (TP7.4) par rapport à $\theta \mapsto -\theta$ en utilisant une base des fonctions propres du rotateur libre unidimensionnel $\psi : \mathbb{S}^1 \rightarrow \mathbb{R}$, paires $\psi_m^+(\theta)$ avec $m \in \mathbb{Z}_{\geq 0}$ et impaires $\psi_m^-(\theta)$ avec $m \in \mathbb{Z}_{>0}$, spécifiquement

$$\psi_0^+(\theta) = 1, \quad \psi_m^+(\theta) = \sqrt{2} \cos(m\theta), \quad \psi_m^-(\theta) = \sqrt{2} \sin(m\theta), \quad \text{où } m \in \mathbb{Z}_{>0} \text{ et } \|\psi_m^\pm\| = 1. \quad (\text{TP7.5a})$$

Dans cette base, $\hat{H}$ est représenté par une simple matrice réelle symétrique *tridiagonale* ayant deux blocs, pair et impair, dont les fonctions de base (TP7.5a) sont indexées par $m=0, 1, 2, \dots$ et $m=1, 2, 3, \dots$, respectivement. Notez que la norme des fonctions (TP7.5a) et les éléments $H_{m', m''}$ de la matrice de $\hat{H}$ sont donnés, respectivement, par les intégrales

$$\|\psi_m^\pm\|^2 = \oint_{\mathbb{S}^1} (\psi_m^\pm)^2 = \int_{-\pi}^{+\pi} \psi_m^\pm(\theta)^2 d\theta = 1 \quad \text{et} \quad H_{m', m''} = \int_{-\pi}^{+\pi} \psi_{m'}^\pm(\theta) \hat{H} \psi_{m''}(\theta) d\theta. \quad (\text{TP7.5b})$$

L'énergie cinétique contribue $\hbar^2 m^2/2$ sur la diagonale principale, alors que le potentiel est représenté par les éléments non-nuls de la 2me diagonale. Dans le bloc pair, celle-là commence par $1/\sqrt{2}$, tandis que tout autres éléments dans les deux blocs égalent $1/2$. En tronquant à une valeur maximale suffisamment grande $m_{\max}$, telle que $\hbar^2 m_{\max}^2 \gg 1$, i.e., bien haut dans le domaine rotationnel, où les éléments hors diagonaux deviennent peu importants, les valeurs propres inférieures peuvent être obtenues par une diagonalisation limpide. Pour construire les fonctions (TP7.5), vérifier leur norme, et obtenir la matrice de (TP7.4) dans la base (TP7.5) sous MAXIMA, inspirez vous du listage L9 sur la page 34. Enfin, comparez aux résultats des exercices TP3.3 et TP5.4.

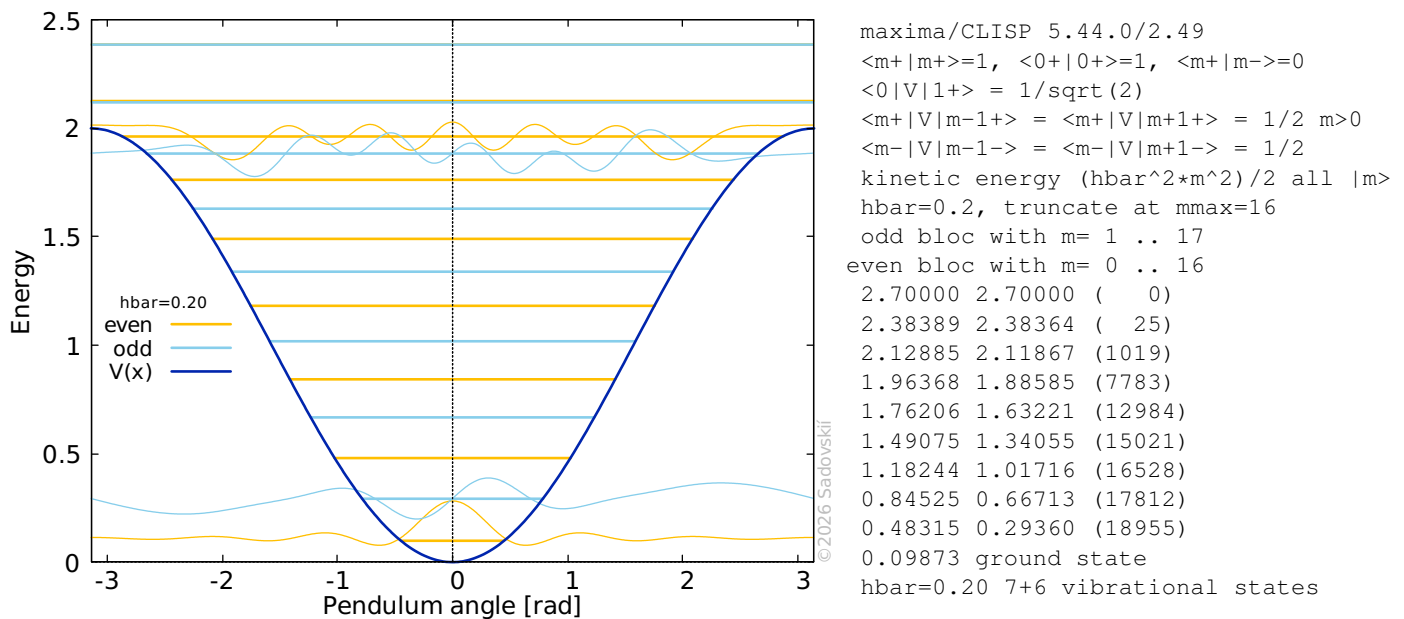


FIG. 5 – Exercice TP7.1 : présentation graphique des niveaux d'énergie paires (+) et impaires (−) du hamiltonien quantique (TP7.4) du pendule plan dans son puits potentiel (avec MAXIMA, voir la fin de listage L9). On observe un léger écartement de 10^{-2} dans premier doublet rotationnel à l'énergie 2.1 dû à la rotation contrarié, alors que les niveaux plus hauts correspondent aux états $|\pm m\rangle$ d'un rotateur quasi-libre.

Notez qu'au lieu de `Jdiag` ou de `eigens_by_jacobi` utilisés auparavant par le programme L8, dans le code L9, on fait appel à l'utilitaire `zheev` du paquet numérique "lapack" qui s'applique à toutes matrices hermitiennes. Toutefois, le mieux adapté à la tâche serait une version de l'algorithme QL pour les matrices réelles symétriques tridiagonales, voir, par exemple, la `tqli` de [Press et al., 2002, chap. 11.3, p. 480] aux décalages implicites. Quand à l'utilitaire `eigenvalues` de MAXIMA, elle est symbolique et peu utile pour des matrices larges. Pour chaque bloc, pair (+) et impair (−), la matrice H de dimension $n \times n$ est construite, et les procédures `eigens_by_jacobi(H)` ou `zheev(H, true)` rendent une liste de deux éléments : la liste $E^\pm$ de n valeurs propres $E_i^\pm$ et la matrice $n \times n$ orthogonale $U^\pm$ dont les lignes $U[i, :]$ sont des vecteurs propres correspondant $u_i^\pm$. Les composantes j de $u_i^\pm$ donnent les contributions de fonctions de base $\psi_m^\pm$ dans la fonction propre $\varphi_i^\pm$. On résume :

$$U^T H U = E^\pm, \quad H u_i^\pm = E_i^\pm u_i^\pm, \quad \hat{H} \varphi_i^\pm = E_i^\pm \varphi_i^\pm, \quad \text{et } \varphi_i^\pm = \sum_{j=1}^n U_{ij} \psi_m^\pm \quad \text{où } m = \begin{cases} j-1 & \text{pour type +} \\ j & \text{pour type -} \end{cases} \quad \text{et } i, j = 1, \dots, n.$$

La reconstruction de $\varphi^\pm(x)$ pour un vecteur u (liste) et parité $p=0$ (−) et 1 (+) est déléguée dans le listage L9 à l'utilitaire `ufun(u, p, x)` (lignes 9-10) qui est engagée pour quelques vecteurs propres (lignes 62-65) afin de tracer $\varphi^\pm(x)$ dans la fig. 5.

Listage L9 – `qpend.mac` : programme sous MAXIMA pour l'exercice TP7.1 (spectre quantique du pendule)

```

/* quantum spectrum of planar pendulum (c)2026 Sadovskii */
2 batchload("license.mac"); debug_level:1;
/* free one-dimensional rotor basis functions S1 -> R with m=0,1,2,... (even) and m=1,2,... (odd) */
4 hbar:hbar$ assume(hbar>0)$
Tmax:10$ /* x2 maximal kinetic (rotational) energy for the basis cutoff (mmax) */
6 hbar_val : 0.2$ /* scale oscillator frequency through a fictitious value of hbar */
psi(m,p,t) := if p>0 then (if m=0 then 1 else sqrt(2)) * cos(m*t) /* even parity */
8 else (if m=0 then 0 else sqrt(2)) * sin(m*t); /* odd */
ufun(u,p,t) := block([p:if p>0 then 1 else 0], /* vector u -> (eigen)function */
10 apply("+", makelist(float(u[k]*psi(k-p,p,t)), k, 1, length(u))) );
/* integrals over S1 of psi(m,t,p) times f (aka "matrix elements") */
12 mel(m,p,f,t) := factor(subst(sin(2*%pi*m)=0, integrate(psi(m,p,t)*f,t,-%pi,%pi)/(2*%pi)))$
(h01 : mel(0,1,cos(t)*psi( 1,1,t),t), /* 1/sqrt(2) for m=0, parity + */
14 hml1 : mel(m,1,cos(t)*psi(m-1,1,t),t), /* 1/2 for m>0, and any parity */
hmm : mel(m,1,-hbar^2*diff(psi(m,1,t),t,2)/2,t) /* diagonal term m^2*hbar^2/2 */
16 )$
printf(true," verify normalization: <m|m+>=~a, <0|0+>=~a, <m|m->=~a .~%",
18 mel(m,1, psi(m,1,t), t), mel(m, 0, psi(m,0,t), t),
mel(m,1, psi(m,0,t), t) )$
20 printf(true," non-0 matrix elements of pendulum potential V=cos(t) occupy the 2nd diagonal~% <0|V|1+> = ~a,~%
<m|V|m-1+> = ~a and <m|V|m+1+> = ~a for m>0 and even parity,~% <m|V|m-1-> = ~a and <m|V|m+1-> = ~a
for odd parity.~%",
h01, hml1, mel(m,1,cos(t)*psi(m+1,1,t),t), /* 1/2 for m>0, parity + */
22 mel(m,0,cos(t)*psi(m-1,0,t),t), mel(m,0,cos(t)*psi(m+1,0,t),t))$ /* 1/2 for m>0, parity - */
printf(true," kinetic energy gives diagonal elements ~a for all |m>~%", hmm-1 )$ /* m^2*hbar^2/2 */
24 block([w0:hbar_val, Vmax:2, ratprint:false],
mmax : ceiling(rhs(first(sublist(solve((hbar_val*m)^2=Tmax,m),lambda([s],rhs(s)>0))))),
26 printf(true," estimated num. of vib. levels ~d for hbar=~f, truncate at mmax=~d~%",
ceiling(Vmax/hbar_val), hbar_val, mmax)
28 )$
load ("lapack")$ /* numerical diagonalization */
30 h:[ ]$ hvib:[ ]$ U:[ ]$
for p:0 thru 1 do block( /* p=0/1 for odd/even parity */
32 [ h01:float(h01), hml1:float(hml1),
hmm:float(subst(hbar=hbar_val,hmm))+1, /* add 1 to potential energy */
34 fpprintprec:4, H, m0:1-p, mmax:mmax], mmax:mmax+m0,
H : zeromatrix(mmax+p,mmax+p), /* matrix of the Hamiltonian */
36 if debug_level>0 then print(if p=0 then " odd" else "even", "bloc with m=", m0, "..", mmax),
for m:m0 thru mmax do block([i:m+p,j], j:i+1, H[i][i] : subst('m=m,hmm),
38 if m<mmax then H[i][j] : subst('m=m,if m=0 then h01 else hml1),
if debug_level>1 then (if m=mmax then print(m,i, H[i][i]) else print(m,i,j, H[i][i], H[i][j]))
40 ), /* combined eigenvalue and eigenvector lists */
block([s:zheev(H,true)], h : cons(first(s),h), U : cons(last(s),U)),
42 hvib : cons(sublist(first(h),lambda([x],x<2)), hvib),
if debug_level>0 and mmax+p <= 5 then print(H,first(h))
44 );
kmax:length(h[1])-1$ /* converged level counter */
46 block([d0:inf, eps:1e-9], /* list quantum energies */
for i:kmax step -1 thru 1 do block( [a:h[1][i+1],b:h[2][i],d], d:abs(a-b),
48 if d<d0 and d>eps then kmax:i else printf(true," ~,5f ~,5f (~4d)~%",a,b, round(d/1e-5)),
d0:d ),
50 printf(true,
" ~,5f ground state~% hbar=~,2f 2x~d levels (of 2x~d) converged to <~,0e with ~d~d vibr. states~%",
52 first(h[1]), hbar_val, kmax, length(h[1]), eps, length(hvib[1]), length(hvib[2]) )
), fpprintprec=5$
54 levels : [ ]$
for p:0 thru 1 do /* level lines between turning points */
56 levels : cons( map(lambda([p],[discrete,[[p[1],p[2]],p]]),
map(lambda([x],[if x<2 then acos(1-x) else %pi,x]), firstn(h[2-p],kmax))), levels) $
58 styles : cons(style, apply(append,makelist(map(lambda([x],[lines,2,p]],levels[p]),p,1,2))) $
titles : cons(legend, apply(append, makelist(cons(["even","odd"][p],map(lambda([x,""],rest(levels[1]))), p
,1,2) )) $
60 levels : apply(append, levels);

62 for l in [[1,1],[0,1],[1,7],[0,6]] do block([p:l[1], k:l[2], i: if l[1]>0 then 1 else 2],
print(ufun(U[i][k],p,x)), /* reconstruct eigenfunction k */
64 levels : endcons( ufun(U[i][k],p,x)/20.0 + h[i][k], levels),
styles : endcons([lines,1,i],styles), titles : endcons("",titles) ), fpprintprec=3;
66
plot2d( endcons(1-cos(x),levels), [x,-%pi,%pi], [y,0,2.5],
68 endcons([lines,2,3],styles), endcons("V(x)",titles), [gnuplot_preamble,
concat(last(get_plot_option(gnuplot_preamble)), "set key bottom left at graph 0.02,.33;"]),
70 [label, [printf(false,"hbar=~2f",hbar_val), -%pi+.2, 1.2]],
[xlabel, "Pendulum angle [rad]"], [ylabel, "Energy"] )$

```

Acronymes

Numérique

É.D.O. équation différentielle ordinaire, voir **O.D.E.**

M.M.C. méthode des moindres carrés

O.D.E. ordinary differential equation, *anglais*, voir **É.D.O.**

Physique mathématique

WKB principe sémi-classique de *Wentzel, Kramers*, et *Brillouin*, 1926

EBK méthode de quantification sémi-classique selon *Einstein–Brillouin–Keller*, une généralisation de **WKB**

Références

R. Aldrovandi and P. L. Ferreira. Quantum pendulum. *Am. J. Phys.*, 48(8) :660–664, 1980.

E. U. Condon. The physical pendulum in quantum mechanics. *Phys. Rev.*, 31 :891–894, May 1928.

R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1, chapter V : Vibration and Eigenvalue Problems. Wiley-VCH, 1953.

J. S. Coursey, D. J. Schwab, and R. A. Dragoset. Atomic weights and isotopic compositions. Technical report, NIST, Physics Laboratory, Gaithersburg, MD, 2015. URL <http://physics.nist.gov/PhysRefData/Compositions>. Ver. 4.1.

K. P. Huber and G. Herzberg. *Molecular Spectra and Molecular Structure IV. Constants of Diatomic Molecules*. Van Nostrand - Reinhold, New York, 1979.

J. E. Lennard-Jones. Cohesion. *Proc. Phys. Soc.*, 43 :461–482, 1931.

P. M. Morse. Diatomic molecules according to the wave mechanics. II. Vibrational levels. *Phys. Rev.*, 34 :57–64, 1929.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2 edition, 2002. ISBN 0-521-43108-5. URL <https://numerical.recipes>.