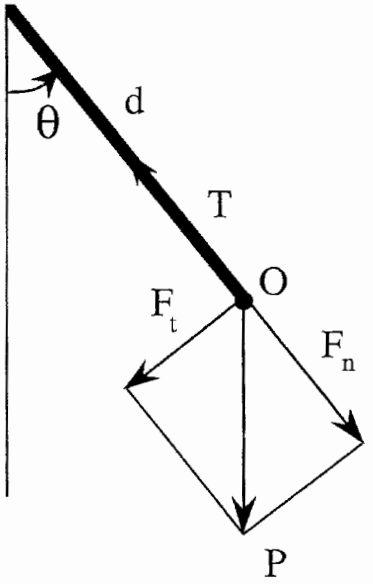
polycopié : • 4 semaines (i.e 2 séances)

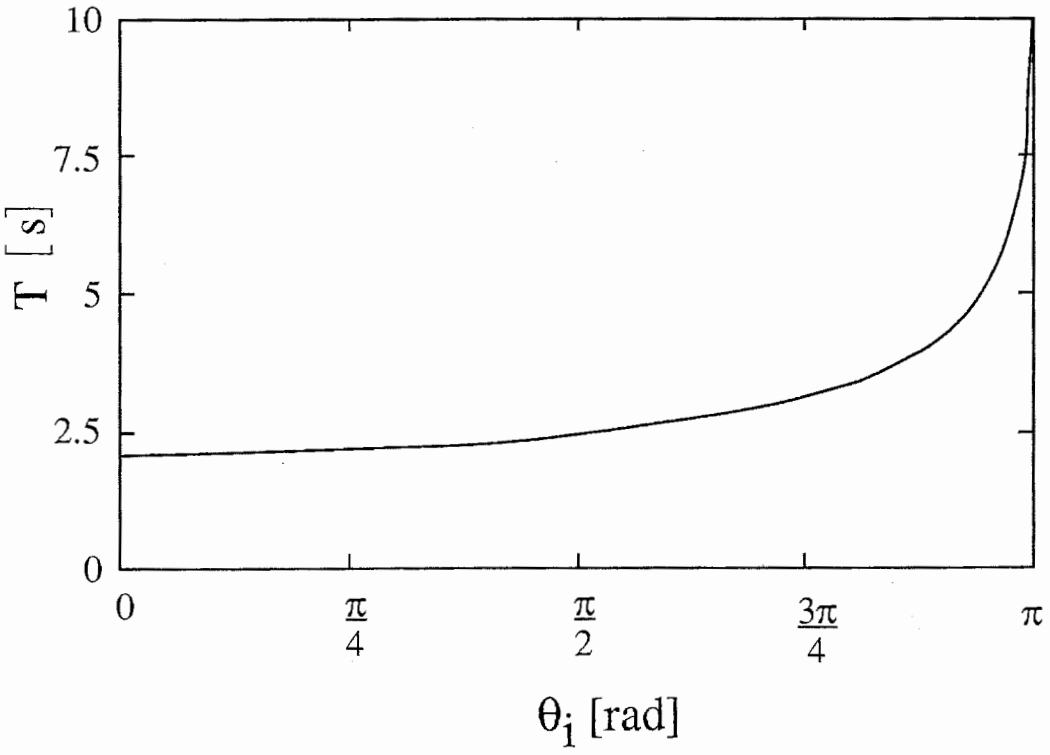
OSCILLATIONS D'UN PENDULE PLAN

Un point matériel O de masse M est attaché à l'extrémité d'un fil rigide de longueur d = 1 m et de masse négligeable, dont l'autre extrémité est fixée à un point immobile.

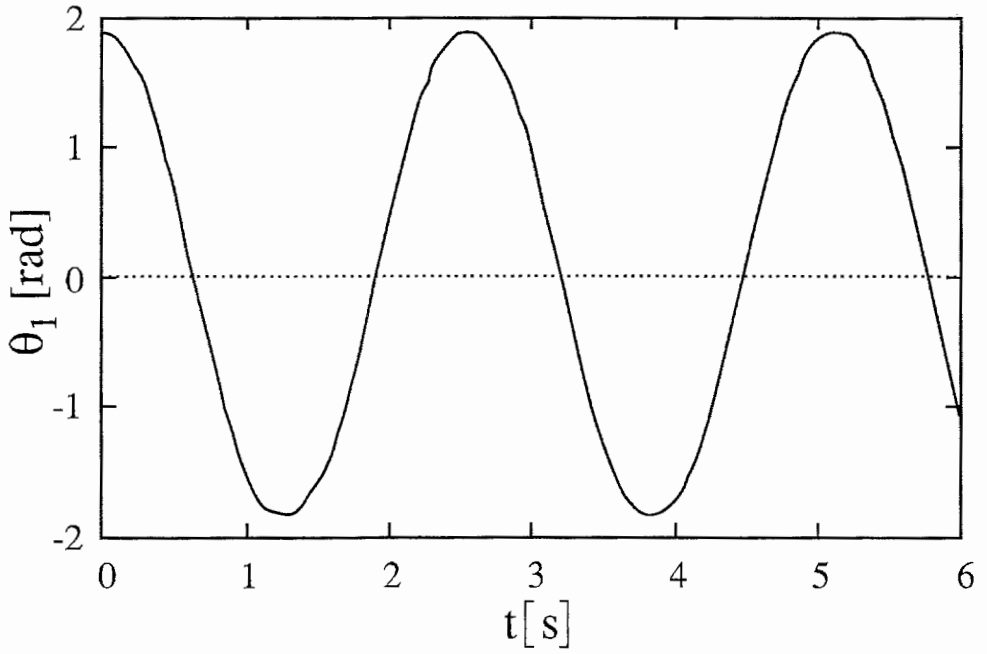


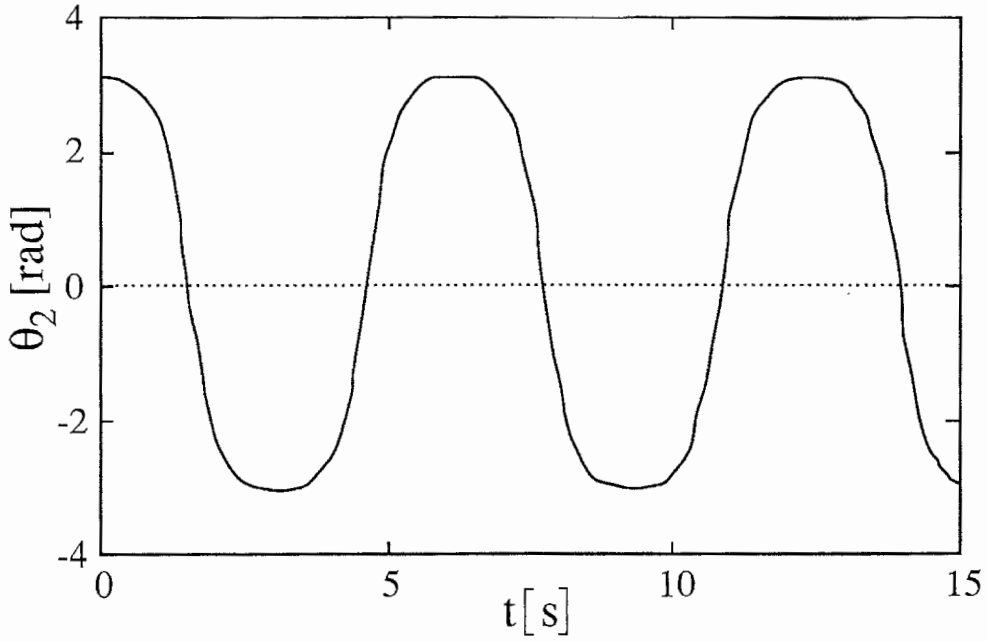
En t = 0, le pendule est lâché sans vitesse de la position initiale θ_i $[0 \leq \theta_i < \pi]$. Le mouvement de O est plan et périodique de période $T(\theta_i)$. Après un quart de période, le point passe pour la première fois par la verticale $[\theta = 0]$.

1. Etudier la fonction $T(\theta_i)$ dans l'intervalle $0 \leq \theta_i < \pi$.



2. *Déterminer les valeurs α_1 et α_2 de l'angle initial θ_i telles que la période du mouvement soit respectivement $T_1 = 2.56$ s et $T_2 = 6.20$ s .*
Etudier dans l'intervalle $0 \leq t \leq 6$ s le mouvement $\theta_1(t)$ qui correspond à la position initiale α_1 et dans l'intervalle $0 \leq t \leq 15$ s le mouvement $\theta_2(t)$ qui correspond à la position initiale α_2 .





3. *Analyser en série de Fourier les fonctions périodiques $\theta_1(t)$ et $\theta_2(t)$.*

UNITES ET EQUATION DE MOUVEMENT

Forces

$$\vec{P} = M\vec{g} = -M g \sin\theta\, \vec{e}_t + M g \cos\theta\, \vec{e}_n$$

$$\vec{T} = -M g \cos\theta\, \vec{e}_n$$

où $\vec{e}_n$ et $\vec{e}_t$ sont les vecteurs unitaires dans la direction normale et tangentielle

et $g = 9.8\ ms^{-2}$.

Equation de mouvement (composantes tangentielles)

$$\frac{d^2\theta}{dt^2} = -\frac{g}{d}\sin\theta$$

avec les conditions initiales $\theta(0) = \theta_i$ et $\theta'(0) = 0$;

ou les conditions aux bords $\theta'(0) = 0$ et $\theta(T/4) = 0$.

Unités caractéristiques du système

⊗ unité de temps : $t^* = \sqrt{d/g} = 0.3194383$ s

unité d'angle : $\theta^* = 1$ rad

Nouvelles variables

$$y = \theta / \theta^* \quad ; \quad x = t / t^*$$

p. ex. pour $t = 3.01$, on a : $x = \frac{3.0}{t^*}$

Equation de mouvement dans les nouvelles variables

$$\frac{d^2y}{dx^2} + \sin y = 0$$

avec les conditions initiales $y(0) = \theta_i / \theta^*$ et $y'(0) = 0$;

ou les conditions au bord $y'(0) = 0$ et $y(T/4t^*) = 0$.

Petites oscillations ($y \sim 0$)

Dans ce cas on a $\sin y \cong y$ et l'équation de mouvement devient $y'' + y = 0$.
La solution générale est

$$y = y_0 \cos(x + x_0)$$

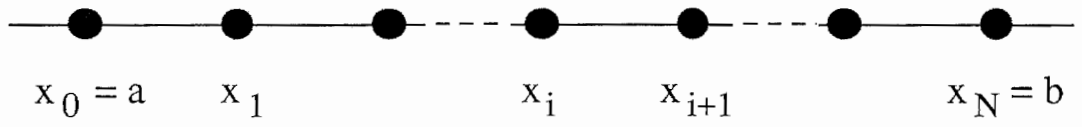
correspondant à un mouvement périodique harmonique de période $T^* = 2\pi t^* = 2.007090$ s.
En imposant les conditions initiales du problème, on a $x_0 = 0$ et $y_0 = \theta_i / \theta^*$.

EQUATIONS DIFFERENTIELLES ORDINAIRES DE PREMIER ORDRE

Soit une fonction $y = y(x)$ définie par l'équation

$$\frac{dy}{dx} = y'(x) = f(x,y)$$

avec la condition initiale $y = \alpha$ pour $x = a$.
Déterminer la valeur β de y en $x = b$.



Définitions :

$$h = (b - a) / N$$
$$x_i = a + ih$$
$$y_i = y(a + ih)$$
$$y_i' = y'(a + ih) = f(x_i, y_i)$$

METHODE D'EULER

$$y_{i+1} = y_i + h\, y_i' + O(h^2)$$

$$\boxed{\overline{y}_{i+1} = \overline{y}_i + h\, f(x_i, \overline{y}_i)}$$

Dans chaque intervalle $[x_i, x_{i+1}]$ l'erreur locale est $O(h^2)$. L'erreur globale sur l'intervalle [a,b] est $O(N^{-1})$. En règle générale, la précision est trop faible.

METHODE DE TAYLOR (2ème ordre)

$$y'(x) = f(x,y)$$

$$y''(x) = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y} f(x,y) = g(x,y)$$

$$y_{i+1} = y_i + h\, y_i' + \frac{1}{2} h^2\, y_i'' + O(h^3)$$

$$\boxed{\bar{y}_{i+1} = \bar{y}_i + h\, f(x_i, \bar{y}_i) + \frac{1}{2} h^2\, g(x_i, \bar{y}_i)}$$

Dans chaque intervalle $[x_i, x_{i+1}]$ l'erreur locale est $O(h^3)$. L'erreur globale sur l'intervalle [a,b] est $O(N^{-2})$.

METHODE DE TAYLOR (4ème ordre)

$$y'''(x) = \frac{\partial^2 f(x,y)}{\partial x^2} + 2\frac{\partial^2 f(x,y)}{\partial x \partial y} f(x,y) + \frac{\partial^2 f(x,y)}{\partial y^2} [f(x,y)]^2 + \frac{\partial f(x,y)}{\partial y} g(x,y)$$

$$= \frac{\partial g(x,y)}{\partial x} + \frac{\partial g(x,y)}{\partial y} f(x,y) = p(x,y)$$

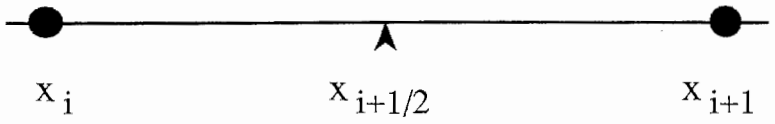
$$y''''(x) = \frac{\partial p(x,y)}{\partial x} + \frac{\partial p(x,y)}{\partial y} f(x,y) = q(x,y)$$

$$y_{i+1} = y_i + h\, y_i' + \frac{1}{2} h^2\, y_i'' + \frac{1}{6} h^3\, y_i''' + \frac{1}{24} h^4\, y_i'''' + O(h^5)$$

$$\boxed{\begin{aligned} \bar{y}_{i+1} = \bar{y}_i + h\, f(x_i, \bar{y}_i) + \frac{1}{2} h^2\, g(x_i, \bar{y}_i) + \\ + \frac{1}{6} h^3\, p(x_i, \bar{y}_i) + \frac{1}{24} h^4\, q(x_i, \bar{y}_i) \end{aligned}}$$

Dans chaque intervalle $[x_i, x_{i+1}]$ l'erreur locale est $O(h^5)$. L'erreur globale sur l'intervalle [a,b] est $O(N^{-4})$.

METHODE DE RUNGE-KUTTA (2ème ordre)



$$y_{i+1} - y_i = h\, y'_{i+1/2} + \frac{1}{24} h^3\, y'''_{i+1/2} + \ldots.$$

$$= h\, f\left(x_{i+1/2}, y_{i+1/2}\right) + O(h^3)$$

Il faut calculer $y'_{i+1/2} = f\left(x_{i+1/2}, y_{i+1/2}\right)$ avec une erreur $O(h^2)$. Puisque

$$f(x, y+\varepsilon) = f(x,y) + \varepsilon \frac{\partial f(x,y)}{\partial y} + \ldots.$$

il faut calculer $y_{i+1/2}$ avec une erreur $O(h^2)$:

$$y_{i+1/2} = y_i + \frac{1}{2} h\, f\left(x_i, y_i\right) + O(h^2)$$

et on obtient

$$y'_{i+1/2} = f[x_{i+1/2},\ y_i + \frac{1}{2} h\, f(x_i, y_i)] + O(h^2)$$

Résultat :

$$\boxed{\bar{y}_{i+1} = \bar{y}_i + t_2}$$

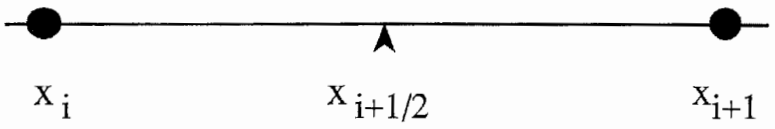
où

$$t_1 = h\, f\left(x_i, \bar{y}_i\right)$$

$$t_2 = h\, f\left(x_{i+1/2}, \bar{y}_i + t_1/2\right)$$

Dans chaque intervalle $[x_i, x_{i+1}]$ l'erreur locale est $O(h^3)$. L'erreur globale sur l'intervalle [a,b] est $O(N^{-2})$.

METHODE DE RUNGE-KUTTA (4ème ordre)



$$y_{i+1} - y_i = h\, y'_{i+1/2} + \frac{1}{24} h^3\, y'''_{i+1/2} + O(h^5)$$

Il faut calculer $y'_{i+1/2}$ avec une erreur $O(h^4)$ et $y'''_{i+1/2}$ avec une erreur $O(h^2)$.

Résultat :

$$\boxed{\bar{y}_{i+1} = \bar{y}_i + (t_1 + 2t_2 + 2t_3 + t_4)/6}$$

$\bar{y}_1 = \bar{y}_0 + \frac{h}{6}\left(f(x_0, \bar{y}_0) + f\left(x_{0+1/2}, \bar{y}_0 + \frac{t_1}{2} \right) \ldots \right.$

où

$$t_1 = h\, f(x_i, \bar{y}_i)$$

$$t_2 = h\, f(x_{i+1/2}, \bar{y}_i + t_1/2)$$

$$t_3 = h\, f(x_{i+1/2}, \bar{y}_i + t_2/2)$$

$$t_4 = h\, f(x_{i+1}, \bar{y}_i + t_3)$$

Dans chaque intervalle $[x_i, x_{i+1}]$ l'erreur locale est $O(h^5)$. L'erreur globale sur l'intervalle [a,b] est $O(N^{-4})$.

ERREUR GLOBALE DANS L'INTERVALLE [a, b]

La détermination de l'erreur globale peut se faire à l'aide de la méthode de Richardson en intégrant deux fois l'équation différentielle sur tout l'intervalle [a, b] avec le même algorithme d'intégration mais en utilisant deux pas d'intégration différents.

1ère fois : $N_1 = N \quad , \quad h_1 = (b - a) / N$

on obtient pour y(b) la valeur approchée $\overline{y}(b,N)$

2ème fois : $N_2 = 2N \quad , \quad h_2 = h_1 / 2$

on obtient pour y(b) la valeur approchée $\overline{y}(b,2N)$.

L'erreur globale dépend de l'ordre de l'algorithme utilisé :

Méthode de 2ème ordre

$$\begin{cases} y(b) = \overline{y}(b, 2N) + \varepsilon \\ y(b) \cong \overline{y}(b, N) + 4\,\varepsilon \end{cases}$$

et donc

$$\begin{cases} \varepsilon \cong [\overline{y}(b, 2N) - \overline{y}(b, N)] / 3 \\ \overline{y}(b, extr) \cong [4\overline{y}(b, 2N) - \overline{y}(b, N)] / 3 \end{cases}$$

⊛ *Méthode de 4ème ordre*

$$\begin{cases} y(b) = \overline{y}(b, 2N) + \varepsilon \\ y(b) \cong \overline{y}(b, N) + 16\,\varepsilon \end{cases}$$

et donc

estimation de l'erreur

$$\begin{cases} \varepsilon \cong [\overline{y}(b, 2N) - \overline{y}(b, N)] / 15 \\ \overline{y}(b, extr) \cong [16\,\overline{y}(b, 2N) - \overline{y}(b, N)] / 15 \end{cases}$$

ERREUR DE TRONCATION DANS L'INTERVALLE $[x_i, x_{i+1}]$

L'intégration de l'équation différentielle de x_i à x_{i+1} permet d'obtenir une valeur approchée $\bar{y}_{i+1}$ de la solution "exacte" y_{i+1} à partir de la valeur (elle aussi approchée !) $\bar{y}_i$.

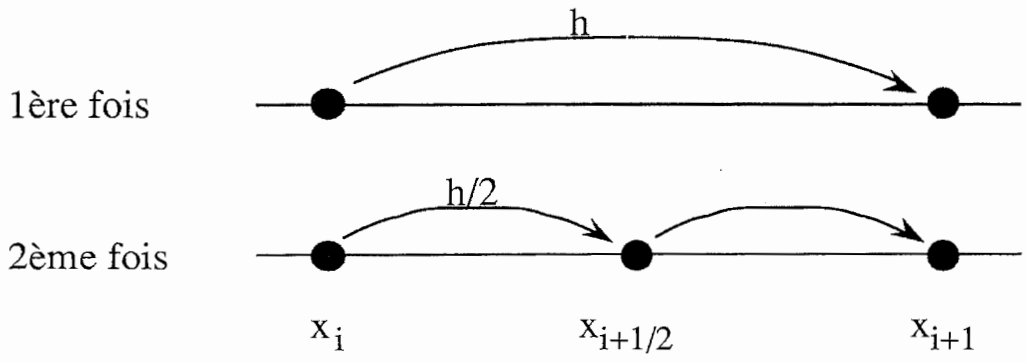
L'erreur de troncation est

$$\Delta_i = | y_{i+1} - \bar{y}_{i+1} |$$

Pour la méthode de Taylor (4ème ordre) on a

$$\Delta_i \cong \frac{h^5}{120} | y_i^{\prime\prime\prime\prime\prime} |$$

Le calcul de Δ_i pour la méthode de Runge-Kutta (4ème ordre) peut se faire en intégrant l'équation différentielle de x_i à x_{i+1} une première fois avec un seul pas de longueur $h = x_{i+1} - x_i$ et une deuxième fois avec deux pas de longueur h/2.



Si on indique par $\bar{y}_{i+1}(h)$ et $\bar{y}_{i+1}(h/2)$ les résultats des deux intégrations, on a

$$\begin{cases} y_{i+1} = \bar{y}_{i+1}(h/2) + 2\,\delta\,(h/2)^5 \\ y_{i+1} \cong \bar{y}_{i+1}(h) + \delta\,h^5 \end{cases}$$

d'où on obtient

$$\bar{y}_{i+1}(h/2) - \bar{y}_{i+1}(h) \cong \frac{15}{16} \delta h^5$$

qui implique une erreur de troncation pour la valeur $\bar{y}_{i+1}(h/2)$ obtenue avec deux pas de longueur h/2

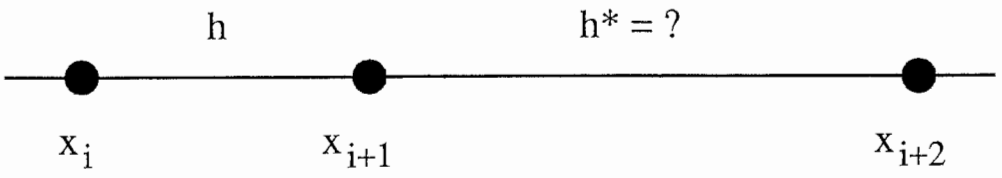
$$\Delta_i \cong \frac{1}{15} | \bar{y}_{i+1}(h/2) - \bar{y}_{i+1}(h) |$$

Remarque :

après le calcul de l'erreur de troncation, il conviendra d'approcher y_{i+1} avec la valeur $\bar{y}_{i+1}(h/2)$ qui est plus précise que $\bar{y}_{i+1}$ (h).

MODIFICATION DU PAS h AU COURANT DE L'INTEGRATION

Le calcul de l'erreur de troncation Δ_i dans l'intervalle $[x_i, x_{i+1}]$ permet de calculer la fonction y(x) avec une précision ε donnée et/ou modifier le pas d'intégration h en fonction des variations plus ou moins importantes de y avec x.



Pour obtenir $\bar{y}_{i+1}$ avec une erreur ε (méthode de 4ème ordre)

$$h \rightarrow h^* = S\, h \left| \frac{\Delta^*}{\Delta_i} \right|^{1/5}$$

où $\Delta^* = \varepsilon \, | \bar{y}_{i+1} |$ et $S < 1$ est un facteur de sécurité (valeur recommandée $S \cong 0.9$).

Pour obtenir la variation ($\bar{y}_{i+1}$ - $\bar{y}_i$) avec une erreur ε (méthode de 4ème ordre)

$$h \rightarrow h^* = S\,h \left| \frac{\Delta^*}{\Delta_i} \right|^{1/4}$$

où $\Delta^* = \varepsilon \, | \bar{y}_{i+1} - \bar{y}_i |$.

En effet, contrairement à $\bar{y}_{i+1}$ du cas précédent, la variation $(\bar{y}_{i+1} - \bar{y}_i)$ dépend linéairement de h.

Remarques :

1. Pour éviter la construction de sous-programmes différents pour traiter les différents types d'erreur, on peut se tenir à la stratégie globale suivante

$$h \rightarrow h^* = \begin{cases} S\,h \left| \Delta^* / \Delta_i \right|^{1/5} & \Delta^* \geq \Delta_i \\ S\,h \left| \Delta^* / \Delta_i \right|^{1/4} & \Delta^* < \Delta_i \end{cases}$$

   qui est valable dans les deux cas considérés en haut (il s'agit d'accepter dans chaque cas la plus petite des deux valeurs de h^*).

2. Si dans l'intervalle $[x_i, x_{i+1}]$ on obtient $\Delta_i < \Delta^*$, on retiendra la valeur $\bar{y}_{i+1}$ déjà calculée et on utilisera h^* pour effectuer le pas suivant, soit de x_{i+1} à $x_{i+2} = x_{i+1} + h^*$.

3. Par contre, si dans l'intervalle $[x_i, x_{i+1}]$ on obtient $\Delta_i > \Delta^*$, la valeur $\bar{y}_{i+1}$ doit être rejetée parce que elle est de précision insuffisante. Dans ce cas, il faudra revenir en x_i [où la fonction vaut $\bar{y}_i$] et intégrer de x_i à $x'_{i+1} = x_i + h^*$.

⊗ SYSTEMES D'EQUATIONS DIFFERENTIELLES ORDINAIRES DU PREMIER ORDRE

Soit à intégrer dans l'intervalle [a, b] le système

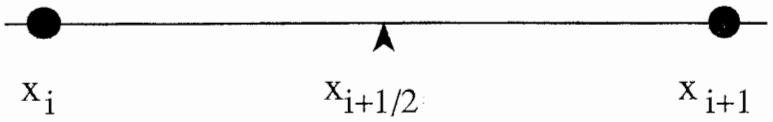
$$\begin{cases} \dfrac{dy}{dx} = f(x, y, z) \\ \dfrac{dz}{dx} = g(x, y, z) \end{cases}$$

avec les conditions initiales

$$y(a) = \alpha_1 \qquad \text{et} \qquad z(a) = \alpha_2 .$$

Le système peut être intégré à l'aide des méthodes de Taylor ou de Runge-Kutta en généralisant les formules données pour le cas d'une seule équation du premier ordre.

METHODE DE RUNGE-KUTTA (4ème ordre)



$h = (b - a)/N$

$$t_1 = h\, f(x_i , \bar{y}_i , \bar{z}_i), \qquad u_1 = h\, g(x_i , \bar{y}_i , \bar{z}_i)$$

$$t_2 = h\, f(x_{i+1/2} , \bar{y}_i + t_1/2 , \bar{z}_i + u_1/2), \qquad u_2 = h\, g(x_{i+1/2} , \bar{y}_i + t_1/2 , \bar{z}_i + u_1/2)$$

$$t_3 = h\, f(x_{i+1/2} , \bar{y}_i + t_2/2 , \bar{z}_i + u_2/2), \qquad u_3 = h\, g(x_{i+1/2} , \bar{y}_i + t_2/2 , \bar{z}_i + u_2/2)$$

$$t_4 = h\, f(x_{i+1} , \bar{y}_i + t_3 , \bar{z}_i + u_3), \qquad u_4 = h\, g(x_{i+1} , \bar{y}_i + t_3 , \bar{z}_i + u_3)$$

$$\begin{cases} \bar{y}_{i+1} = \bar{y}_i + (t_1 + 2t_2 + 2t_3 + t_4)/6 \\ \bar{z}_{i+1} = \bar{z}_i + (u_1 + 2u_2 + 2u_3 + u_4)/6 \end{cases}$$

Dans chaque intervalle $[x_i, x_{i+1}]$ l'erreur locale est $O(h^5)$. L'erreur globale de troncature sur l'intervalle [a, b] est $O(N^{-4})$.

⊗ On peut facilement généraliser les formules de Runge-Kutta pour un système de *n* équations du première ordre.

(/ . . /)

⊗ EQUATIONS DIFFERENTIELLES ORDINAIRES D'ORDRE SUPERIEUR A UN

Toute équation différentielle ordinaire d'ordre n peut se remplacer par un système de n équations du premier ordre.

Considérons, par exemple, l'équation du deuxième ordre

$$\frac{d^2y}{dx^2} = f(x, y, y')$$

où $y' = \frac{dy}{dx}$, avec les conditions initiales $y(a) = \alpha_1$ et $y'(a) = \alpha_2$.

Nous pouvons remplacer cette équation par le système

$$\begin{cases} \frac{dy}{dx} = z(x) \\ \frac{dz}{dx} = f(x, y, z) = \frac{d^2y}{dx^2} \end{cases}$$

⇒ syst. de premier ordre

avec les conditions initiales $y(a) = \alpha_1$ et $z(a) = \alpha_2$.

De même, l'équation du troisième ordre

$$\frac{d^3y}{dx^3} = g(x, y, y', y'')$$

où $y'' = \frac{d^2y}{dx^2}$, peut se remplacer par le système

$$\begin{cases} \frac{dy}{dx} = z(x) \\ \frac{dz}{dx} = w(x) \\ \frac{dw}{dx} = g(x, y, z, w). \end{cases}$$

EQUATIONS DIFFERENTIELLES AVEC CONDITIONS AUX BORDS

La solution générale d'une équation différentielle d'ordre n

$$y^{(n)} = F(x, y, y', y'', ..., y^{(n-1)})$$

dépend de n constantes arbitraires. Une solution particulière $\bar{y}(x)$ est caractérisée par les valeurs de ces n constantes qui sont déterminés, par exemple, par n conditions imposées à la solution générale.

Jusqu'à présent, nous n'avons considéré que des problèmes avec *conditions initiales* où on fixe la valeur de la fonction et celles de ses n–1 premières dérivées au point initial x_0 :

$$\bar{y}(x_0) = y_0 \quad ; \quad \bar{y}'(x_0) = y_0' \quad ; \quad \ldots \quad \bar{y}^{(n-1)}(x_0) = y_0^{(n-1)} .$$

Ce type de problème est le seul qui puisse se présenter pour des équations différentielles du premier ordre.

Dans le cas d'une équation d'ordre n > 1, une solution particulière $\bar{y}(x)$ définie sur l'intervalle $a \leq x \leq b$ peut être caractérisée par n conditions, les unes au point x = a, les autres au point x = b. On parle dans ce cas de *conditions aux bords* et le problème à résoudre est généralement plus difficile que celui avec conditions initiales. Une méthode de résolution simple de problèmes avec conditions aux bords est la *méthode de tir*.

METHODE DE TIR

La méthode sera présentée dans le cas le plus simple : une équation différentielle du deuxième ordre dont la solution particulière est caractérisée par deux conditions, une à chaque extrémité de l'intervalle $a \leq x \leq b$.

Soit l'équation

$$\frac{d^2y}{dx^2} = F(x, y, y')$$

avec les conditions aux bords $y(a) = \alpha$, $y(b) = \beta$ qui caractérisent la solution particulière $\bar{y}(x)$.

La méthode de tir consiste à déterminer les solutions particulières $\tilde{y}(x, t)$ qui sont caractérisées par les conditions initiales

$$\begin{cases} y(a) = \alpha \\ y'(a) = t \end{cases}$$

dont la première est une donnée et la deuxième est l'*inconnue* de notre problème.
On sait résoudre ce problème avec conditions initiales (par exemple avec la méthode de Runge-Kutta) pour chaque valeur de l'inconnue t et on détermine ainsi la solution particulière $\tilde{y}(x, t)$ qui dépend paramétriquement de t.
La méthode de tir consiste à déterminer la valeur t^* du paramètre t, solution de l'équation

$$\tilde{y}(b, t) = \beta.$$

Il s'agit d'un problème de recherche de zéros que l'on peut résoudre, par exemple, par la méthode de bissection ou la méthode de Newton.
Une fois déterminé t^*, la solution particulière $\tilde{y}(x, t^*)$ obtenue en imposant les conditions initiales $y(a) = \alpha$, $y'(a) = t^*$ est la solution du problème avec les conditions au bord $y(a) = \alpha$, $y(b) = \beta$.

La généralisation à une équation d'ordre n ou à des systèmes est triviale. Considérons par exemple l'équation

$$\frac{d^3y}{dx^3} = F(x, y, y', y'')$$

dont on cherche dans l'intervalle $a \leq x \leq b$ la solution particulière telle que

$$y(a) = \alpha \quad ; \quad y(b) = \beta \quad ; \quad y'(b) = \beta'.$$

On introduit les conditions initiales $y(b) = \beta$, $y'(b) = \beta'$ (données) et $y''(b) = t$ (inconnue). Ces conditons initiales caractérisent la solution particulière $\tilde{y}(x, t)$. Il s'agira après de résoudre l'équation $\tilde{y}(a, t) = \alpha$.

Remarque :
La méthode de tir s'adapte bien aux équations linéaires. On peut rencontrer des difficultés dans l'application aux équations non-linéaires.

SERIE DE FOURIER D'UNE FONCTION PERIODIQUE

Soit une fonction f(x) périodique de période T

$$f(x+T) = f(x).$$

On peut développer f(x) en série de Fourier

$$f(x) = \frac{1}{2} a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(n\omega x) + b_n \sin(n\omega x)\right]$$

où $\omega = 2\pi/T$ est la pulsation associée à T et les coefficients a_n et b_n sont donnés par

$$a_n = \frac{2}{T} \int_{x_0}^{x_0+T} f(x) \cos(n\omega x)\, dx$$

$$b_n = \frac{2}{T} \int_{x_0}^{x_0+T} f(x) \sin(n\omega x)\, dx ,$$

x_0 étant arbitraire.

La constante $a_0/2$ est la valeur moyenne de f(x). Il est évident que si f(x) est une fonction paire on a $b_n = 0$ ($\forall n$) et que si f(x) est une fonction impaire on a $a_n = 0$ ($\forall n$).
La représentation exacte de f(x) demande un nombre infini de termes de Fourier mais on obtient en général une bonne approximation en ne considérant qu'un nombre fini de termes

$$f_N(x) = \frac{1}{2} a_0 + \sum_{n=1}^{N} \left[a_n \cos(n\omega x) + b_n \sin(n\omega x)\right] .$$

La quantité

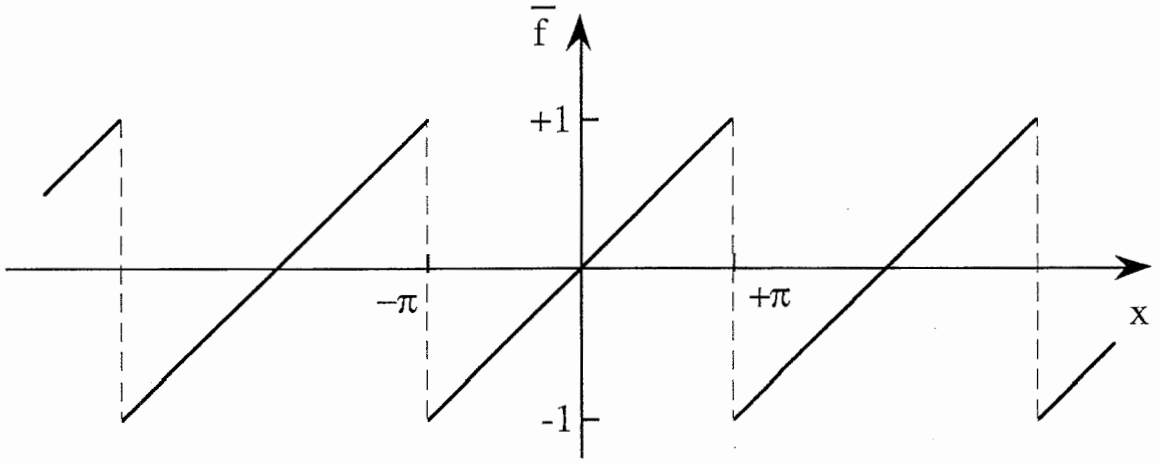
$$\Delta_N = \max \left| f(x) - f_N(x) \right| \text{ pour } x_0 \le x \le x_0 + T$$

est une mesure de l'erreur qu'on fait en remplaçant f(x) avec $f_N(x)$.

Remarque

On peut représenter par série de Fourier dans un intervalle [a, b] toute fonction f(x) définie sur l'intervalle. Il suffit de construire la fonction périodique $\bar{f}(x)$ de période T = (b - a) qui coïncide avec f(x) dans l'intervalle $a \leq x \leq b$. Remarquer que si f(x) est continue et $f(a) \neq f(b)$, $\bar{f}(x)$ est une fonction discontinue.

Exemple : Pour représenter en série de Fourier dans l'intervalle $-\pi < x < \pi$ la fonction $f(x) = x/\pi$, on construit la fonction périodique $\bar{f}(x)$ donnée à la figure.

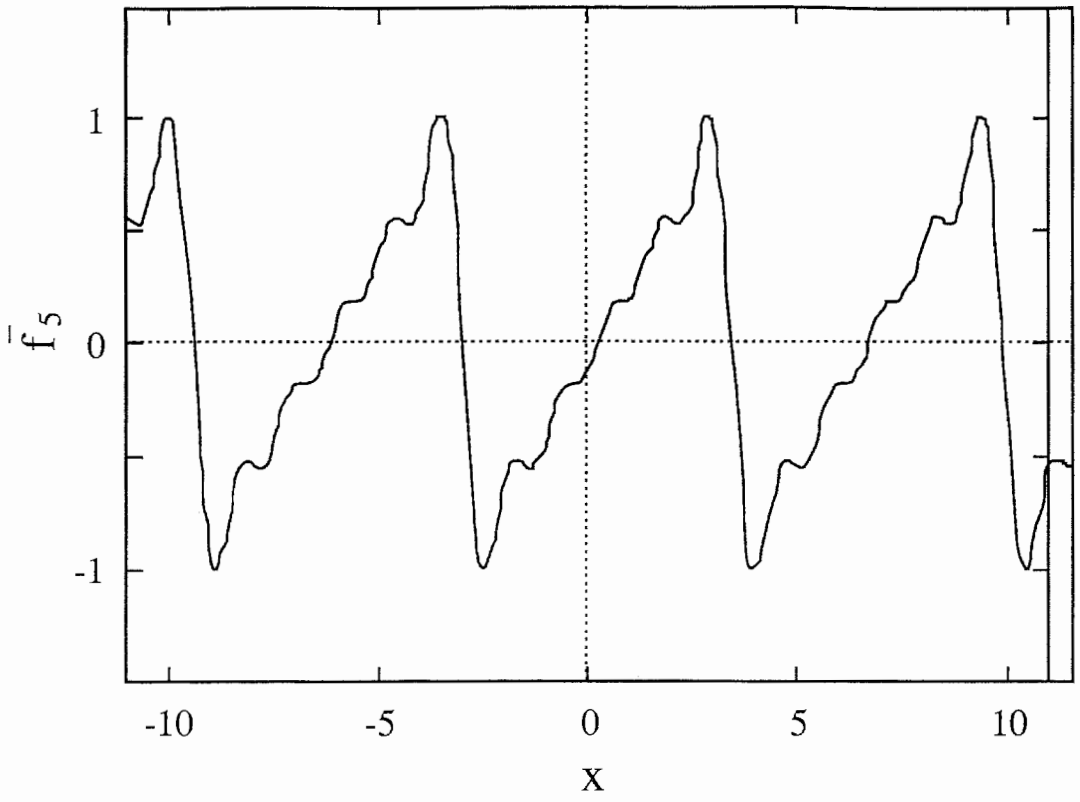


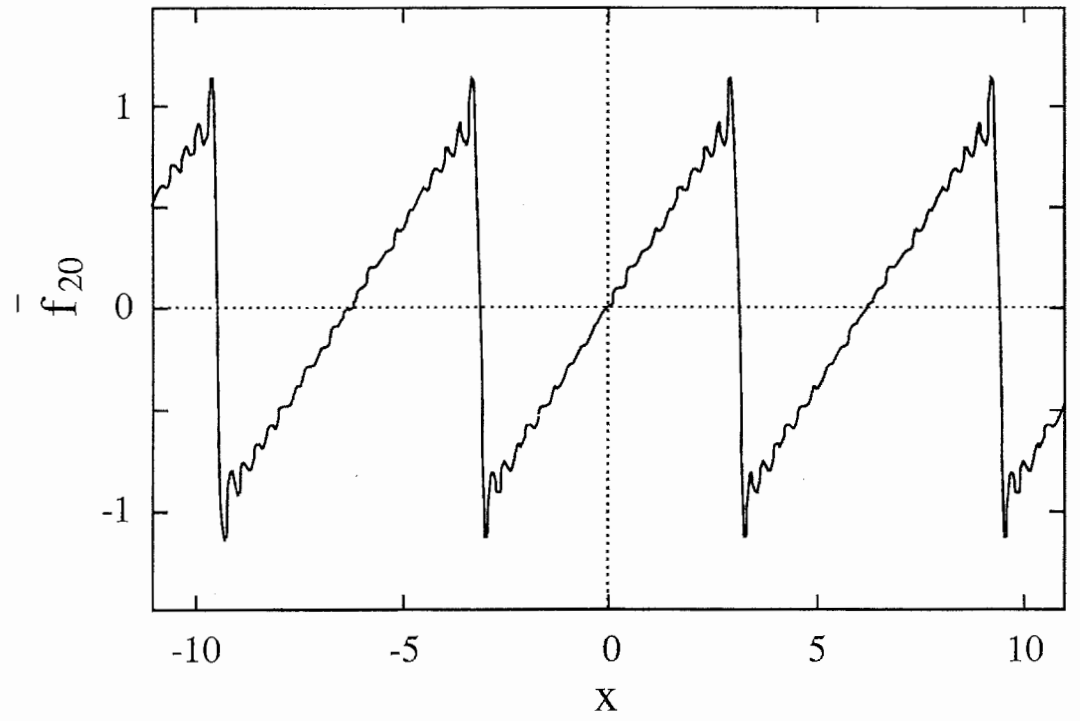
dont le développement en série de Fourier est

$$\bar{f}(x) = -\frac{2}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^n}{n} \sin(nx) \ .$$

On pourra remplacer f(x) avec ce développement pour tout x dans l'intervalle $-\pi < x < \pi$.

En ne considérant que les N premiers termes du développement de $\bar{f}(x)$, on obtient la représentation approchée $\bar{f}_N(x)$. Les cas N = 5 et N = 20 sont donnés aux figures suivantes.

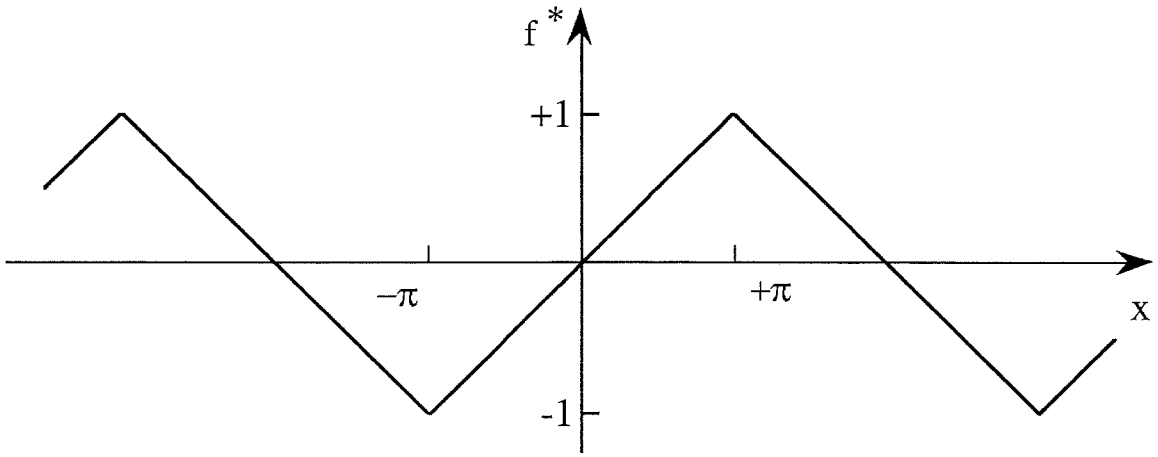




Les fortes oscillations qu'on observe dans $\overline{f}_5(x)$ et $\overline{f}_{20}(x)$ sont dues aux discontinuités de $\overline{f}(x)$ en $x = (2\,i + 1)\pi$ pour $i = \pm 1, \pm 2, ...$

Une meilleure représentation de $f(x) = x/\pi$ dans l'intervalle $[-\pi, \pi]$ s'obtient à l'aide de la fonction périodique $f^*(x)$ de période 4π et définie dans l'intervalle $[-\pi, 3\pi]$ comme suit

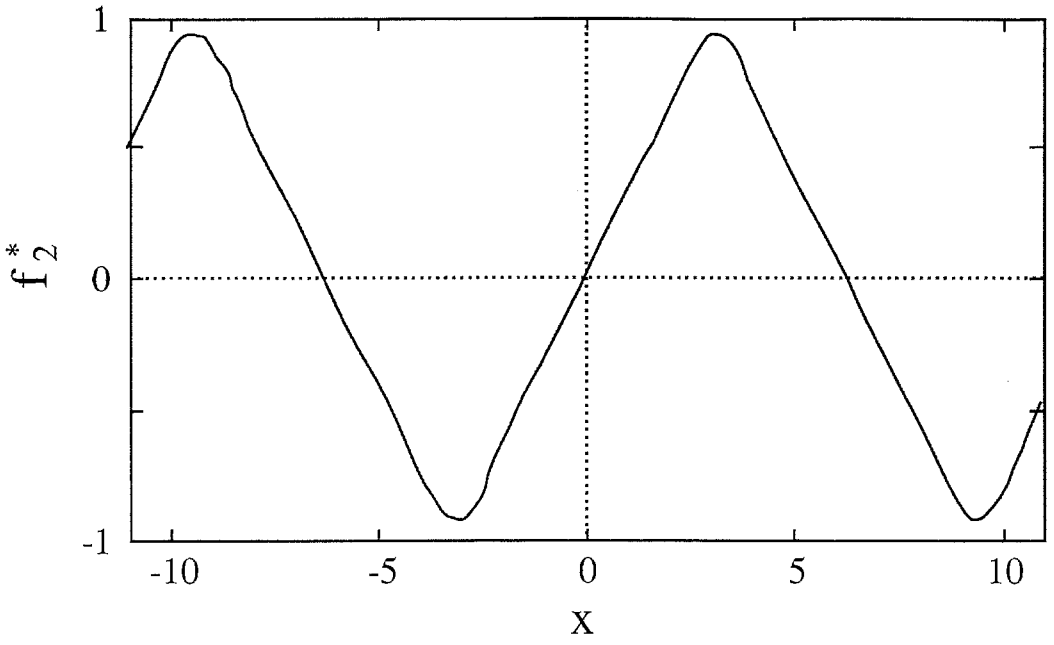
$$f^*(x) = \begin{cases} f(x) & -\pi < x < \pi \\ f(2\pi - x) & \pi < x < 3\pi \end{cases} .$$

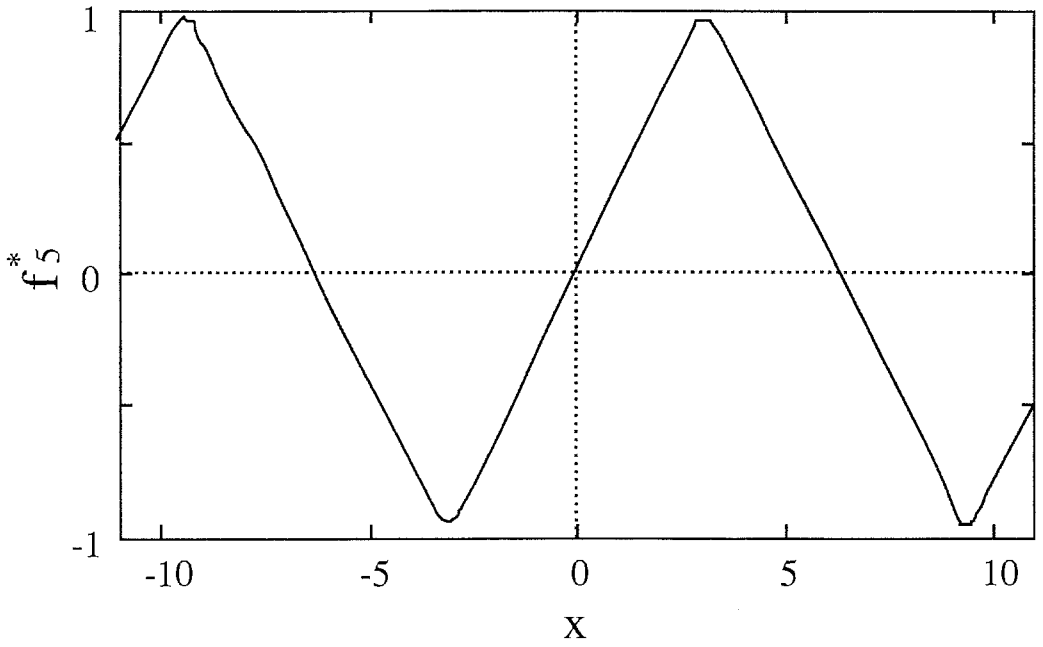


Le développement en série de Fourier de $f^*(x)$ est

$$f^*(x) = \frac{8}{\pi^2} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} \sin\left(\frac{2n+1}{2} x\right).$$

Les représentations approchées $f_N^*(x)$ qu'on obtient en ne considérant que les $N = 2, 5$ premiers termes du développement sont données aux figures suivantes.





On peut constater que les $f_N^*(x)$ représentent mieux la fonction $f(x) = x/\pi$ dans l'intervalle $[-\pi, \pi]$ que les $\overline{f}_N(x)$.

EXERCICES

Exercice 1

1a) Construire une procédure de type sous-programme

subroutine rk4 (fct, xi, yi, xf, n, yf, x, y)

pour intégrer l'équation différentielle ordinaire

$$\frac{dy}{dx} = f(x,y) \quad \text{avec la condition initiale } y(x_i) = y_i$$

de $x = x_i$ jusqu'à $x = x_f$, en utilisant la méthode de Runge-Kutta de 4ème ordre et n pas d'intégration de même longueur $h = (x_f - x_i) / n$.

Le sous-programme reçoit les données suivantes :

xi, yi = conditions initiales x_i et $y_i = y(x_i)$

xf = valeur x_f de x où on veut calculer la fonction y

n = nombre n des pas d'intégration.

Le sous-programme fera appel à

function fct (x, y)

dont l'interface devra être déclarée et qui pour chaque paire de valeurs des variables x, et y fournit la valeur de $\frac{dy(x)}{dx} = f(x,y)$.

possible !

Le sous-programme fournira :

yf = valeur cherchée de $y(x_f)$

(x, y) = tableaux de dimension (n+1) contenant les valeurs intermédiaires de la variable indépendante x ainsi que celles correspondantes de la fonction y(x).

1b) Tester le sous-programme *rk4* à l'aide de la fonction $s(x) = [e^x - (x^2 + 2x + 2)]$ qui est la solution de l'équation différentielle.

$$\frac{ds}{dx} = s + x^2 \text{ avec la condition initiale } s(0) = -1.$$

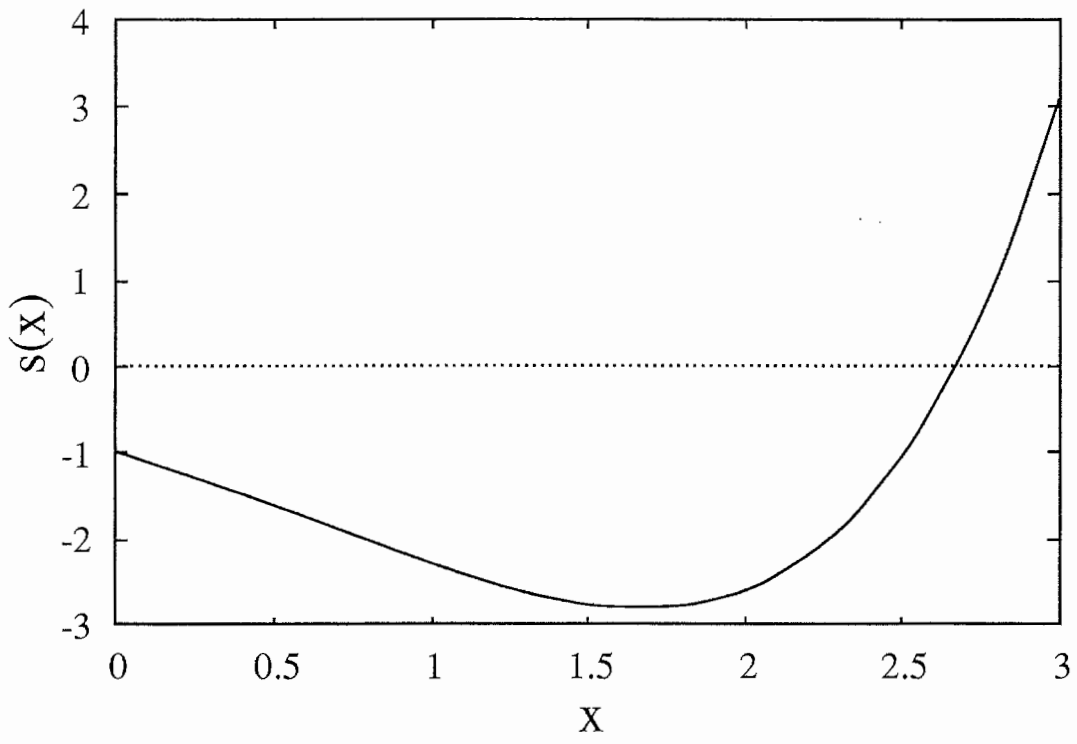
$f(x,y) = x^2 + y$ i.e. $\frac{dy}{dx} = f(x,y)$

Intégrer cette équation de $x_i = 0$ jusqu'à $x_f = 3$ avec n = 5, 10 et 20 pas d'intégration.

En sachant que s(3) = 3.08553692318..... , déterminer l'erreur s(3)–s(3,n) des valeurs s(3,n) calculées en x = 3.

[Réponse :

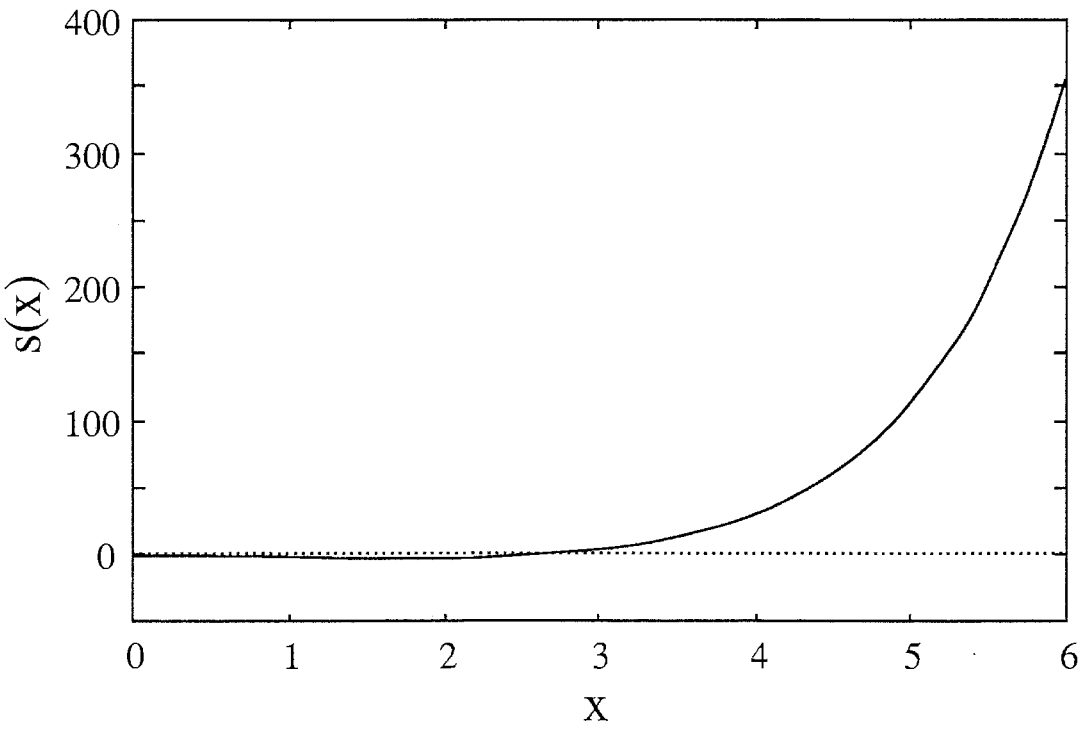
n	*s(3,n)*	*s(3) – s(3,n)*
5	*3.08352*	*0.00202*
10	*3.08513*	*0.00041*
20	*3.08550*	*0.00004*] .



1c) Tester aussi le sous-programme *rk4* en intégrant l'équation différentielle de l'exercice 1b de $x_i = 0$ jusqu'à $x_f = 6$ avec n = 5, 10, 20 et 40 pas d'intégration et déterminer l'erreur des valeurs s(6,n) ainsi obtenues en sachant que s(6) = 353.428793492..... .

[Réponse :

n	*s(6,n)*	*s(6) − s(6,n)*
5	*346.79*	*6.64*
10	*352.63*	*0.80*
20	*353.36*	*0.07*
40	*353.42*	*0.00*] .



Exercice 2

2a) En utilisant la méthode de Richardson, améliorer le sous-programme *rk4* afin de calculer la valeur $y_f = y(x_f)$ avec une erreur relative ε donnée. Le nouveau sous-programme sera du type suivant

subroutine rk4 (fct, xi, yi, xf, itmax, eps, yf, n, x, y)

où :

itmax	=	nombre maximum d'itérations permis (à la fin de chaque itération, le nombre d'intervalles est doublé)
eps	=	erreur relative donnée
n	=	nombre d'intervalles qui a permis d'atteindre la précision désirée (n+1 est donc la dimension des tableaux x et y).

Le sous-programme intégrera l'équation différentielle de $x = x_i$ jusqu'à $x = x_f$ avec un nombre de pas n = 1, 2, 4, 8, ... jusqu'à avoir obtenu la précision désirée pour $y(x_f)$.

2b) A l'aide du sous-programme *rk4*, calculer avec cinq chiffres significatifs la valeur à x = 3 de la fonction s(x) de l'exercice 1b.

[Réponse : $s(3) \cong 3.0855$] . ⇒ erreur absolue

2c) A l'aide du sous-programme *rk4*, calculer avec cinq chiffres significatifs la valeur à x = 6 de la fonction s(x) de l'exercice 1b.

[Réponse : $s(6) \cong 353.43$] .

Exercice 3

A l'aide du logiciel graphique *gnuplot*, représenter sur l'écran la fonction s(x) de l'exercice 1b dan l'intervalle $0 \le x \le 3$ et dans l'intervalle $0 \le x \le 6$.

```
plot [0:6]  exp(x) -(x**2+2*x+2) , "6eqdif3a.txt" , "6eqdif3b.txt
```

Exercice 4

4a) Construire un sous-programme

subroutine rk4sys (fct, xi, yi, xf, n, yf, x, y)

pour intégrer de $x = x_i$ jusqu'à $x = x_f$ le système de N équations différentielles

$$\begin{cases} \dfrac{dy_1}{dx} = f_1\left(x, y_1, y_2, \ldots, y_N\right) \\ \dfrac{dy_2}{dx} = f_2\left(x, y_1, y_2, \ldots, y_N\right) \\ \ldots\ldots \\ \dfrac{dy_N}{dx} = f_N\left(x, y_1, y_2, \ldots, y_N\right) \end{cases}$$

avec les conditions initiales $y_1(x_i) = y_{i,1}$, $y_2(x_i) = y_{i,2}$, ... , $y_N(x_i) = y_{i,N}$. Le sous-programme utilise la méthode de Runge-Kutta du 4ème ordre et n pas d'intégration de même longueur $h = (x_f - x_i) / n$.

Le sous-programme reçoit les données suivantes :

fct	=	fonction qui fournit en résultat le tableau contenant les N termes de droite des équations différentielles
xi, xf	=	extrémités de l'intervalle d'intégration $[x_i, x_f]$
yi	=	tableau contenant les N conditions initiales $y_{i,1}, y_{i,2}, \ldots, y_{i,N}$
n	=	nombre n des pas d'intégration

Le sous-programme fait appel à

function fct (x, y)

qui pour chaque ensemble des valeurs des variables x et $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$ fournit le tableau *fct* contenant les valeurs des fonctions $f_1(x, \mathbf{y}), f_2(x, \mathbf{y}), \ldots, f_N(x, \mathbf{y})$.

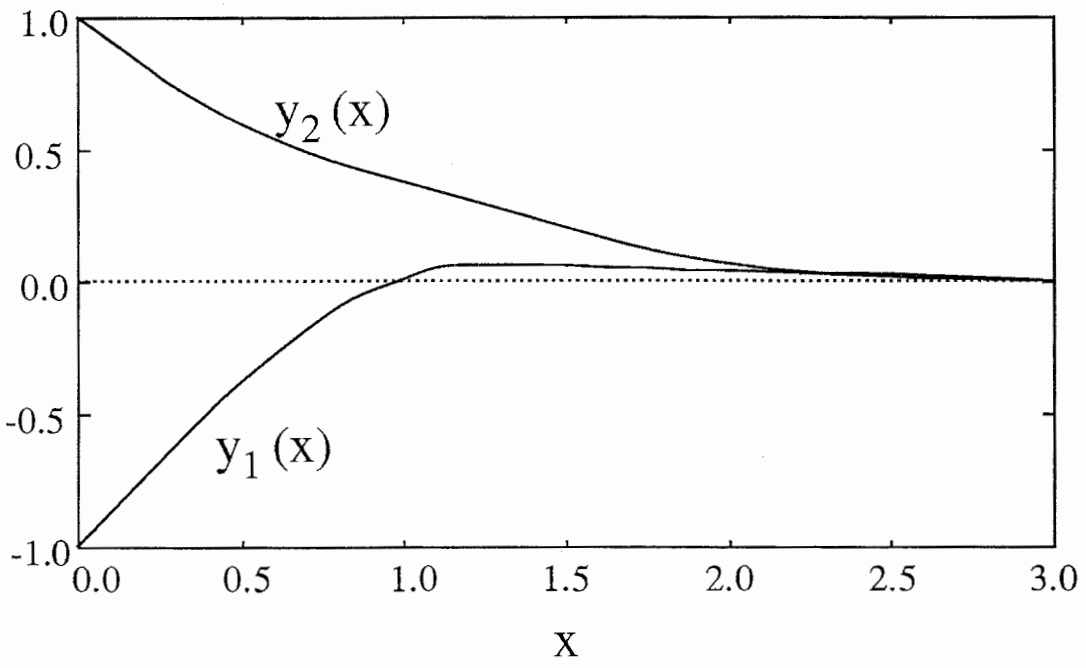
Le sous-programme fournit :

yf	=	tableau contenant les valeurs des N variables dépendantes calculées pour $x = x_f$.
x	=	tableau contenant les (n+1) valeurs intermédiaires de la variable x
y	=	tableaux à deux indices de dimension (N, n+1) contenant les valeurs intermédiaires des N variables dépendantes.

4b) Tester le sous-programme *rk4sys* dans le cas des fonctions $y_1(x) = (x - 1)\, e^{-x^2}$ et $y_2(x) = (x^2 - x + 1)\, e^{-x^2}$ qui sont la solution du système

$$\begin{cases} \dfrac{dy_1}{dx} = y_2 - 3xy_1 \\ \dfrac{dy_2}{dx} = (x - 2x^2)\, y_1 - y_2 \end{cases}$$

avec les conditions initiales $y_1(0) = -1$ et $y_2(0) = 1$.



Intégrer le système de $x_i = 0$ jusqu'à $x_f = 1.5$ avec n = 5, 10, 20 et 40 pas d'intégration et calculer avec cinq chiffres significatifs les valeurs $y_1(1.5, n)$ et $y_2(1.5, n)$.
En sachant que $y_1(1.5) = 0.05269961228.....$ et $y_2(1.5) = 0.18444864298.....$ calculer les erreurs $\delta y_1(n) = [y_1(1.5) - y_1(1.5, n)]$ et $\delta y_2(n) = [y_2(1.5) - y_2(1.5, n)]$.

[Réponse :

n	$y_1(1.5, n) \cdot 10^2$	$\delta y_1(n) \cdot 10^2$	$y_2(1.5, n) \cdot 10^2$	$\delta y_2(n) \cdot 10^2$
5	*5.1052*	*0.1647*	*18.302*	*0.143*
10	*5.2632*	*0.0067*	*18.440*	*0.005*
20	*5.2696*	*0.0003*	*18.445*	*0.000*
40	*5.2699*	*0.0000*	*18.445*	*0.000*]

Exercice 5

5a) En utilisant la méthode de Richardson, modifier le sous-programme *rk4sys* afin de calculer les valeurs $y_{f,1} = y_1(x_f)$, $y_{f,2} = y_2(x_f)$, ..., $y_{f,N} = y_N(x_f)$ avec des erreurs relatives ε_1, ε_2, ..., ε_N données. Le nouveau sous-programme sera :

subroutine rk4sys (fct, xi, yi, xf, itmax, eps, yf, n, x, y)

où

itmax	=	nombre maximum d'itérations permis (à la fin de chaque itération, le nombre d'intervalles est doublé)
eps	=	tableau de dimension N contenant les erreurs relatives données
n	=	nombre d'intervalles qui permet d'atteindre la précision désirée [(n+1) est donc la dimension du tableau *x* et (N, n+1) est la dimension du tableau *y*].

5b) A l'aide de la nouvelle version du sous-programme *rk4sys* intégrer le système d'équations différentielles considéré en 4b) de $x_i = 0$ jusqu'à $x_f = 1.5$, 2.0 et 3.0 et calculer avec cinq chiffres significatifs les valeurs $y_1(1.5)$, $y_2(1.5)$, $y_1(2.0)$, $y_2(2.0)$, $y_1(3.0)$ et $y_2(3.0)$.

[Réponse : $y_1(1.5) = 0.052700$, $y_1(2.0) = 0.018316$, $y_1(3.0) = 0.00024682$
$y_2(1.5) = 0.18445$, $y_2(2.0) = 0.054947$, $y_2(3.0) = 0.00086387$ *]*

Exercice 6

6a) Considérer l'équation de mouvement du pendule

$$\frac{d^2y}{dx^2} + \sin y = 0,$$

$$\left.\begin{array}{l} z = \frac{dy}{dx} \\ \frac{dz}{dx} = -\sin(y) \end{array}\right\} \quad y(x) = \ldots$$

la transformer en un système de deux équations du premier ordre et résoudre ce dernier par la méthode de Runge-Kutta.

6b) A l'aide du sous-programme *rk4sys* , calculer avec erreur inférieure à 1×10^{-5} rad la position θ (t) du pendule pour $t_1 = 3.0$ s, $t_2 = 3.5$ s et $t_3 = 4.0$ s en sachant qu'il est laché sans vitesse de la position intiale $\theta(0) = \theta_0 = 0.5$ rad.
Comparer les résultats obtenus avec ceux de la solution analytique $\tilde{\theta}(t) = \theta_0 \cos\left(\sqrt{g/d}\, t\right)$ qui est valable pour $\theta_0 \to 0$.

$x = \frac{t}{\sqrt{\frac{g}{d}}}$

$\Rightarrow t = \sqrt{\frac{g}{d}} x$

$\Rightarrow \theta(t) = \theta_0 \cdot \cos\left(\sqrt{\frac{g}{d}} x\right)$

[Réponse :

$\theta(t_1) = -0.49202$ rad ; $\tilde{\theta}(t_1) = -0.49972$ rad ;

$\theta(t_2) = -0.10467$ rad ; $\tilde{\theta}(t_2) = -0.01942$ rad ;

$\theta(t_3) = +0.48584$ rad ; $\tilde{\theta}(t_3) = +0.49951$ rad]

6c) Déterminer, avec erreur inférieure à 1×10^{-5} rad, une borne supérieure à la valeur absolue de $\Delta\theta(t) = \theta(t) - \tilde{\theta}(t)$ pour $0 \leq t \leq 4$ s.

[Réponse :

$|\Delta\theta(t)| < 0.08675$ rad]

6d) Mêmes questions que 6b) et 6c) mais avec les conditions initiales $\theta(0) = \theta_0 = 0.1$ rad, $\theta'(0) = 0$ rad s^{-1} et avec erreur inférieure à 1×10^{-6} rad.

[Réponse :

$\theta(t_1) = -0.099923$ rad ; $\tilde{\theta}(t_1) = -0.099945$ rad ;

$\theta(t_2) = -0.004568$ rad ; $\tilde{\theta}(t_2) = -0.003883$ rad ;

$\theta(t_3) = +0.099864$ rad ; $\tilde{\theta}(t_3) = +0.099901$ rad ;

$|\Delta\theta(t)| < 0.000689$ rad]

Exercice 7

7a) Calculer avec cinq chiffres significatifs la période T du pendule s'il est lâché sans vitesse des positions initiales $\theta_1 = \pi/32$ rad , $\theta_2 = \pi/16$ rad , $\theta_3 = \pi/8$ rad et $\theta_4 = \pi/4$ rad

[Réponse :

$T(\theta_1) = 2.0083$ s ; $T(\theta_2) = 2.0119$ s ;

$T(\theta_3) = 2.0266$ s ; $T(\theta_4) = 2.0873$ s]

7b) Sur la base des résultats obtenus en 7a), déterminer le comportement de la fonction $T(\theta_i)$ pour $\theta_i \sim 0$.

[Réponse : $T(\theta_i) = T^* (1 + c\, \theta_i^2)$

où $T^* = 2\pi t^* = 2.007090$ s et $c = 0.0625$ rad $^{-2}$]

Exercice 8

8a) Calculer avec quatre chiffres significatifs la période T du pendule s'il est lâché sans vitesse des positions initiales $\theta_1 = 63\ \pi/64$ rad, $\theta_2 = 127\ \pi/128$ rad, $\theta_3 = 255\ \pi/256$ rad et $\theta_4 = 511\ \pi/512$ rad.

[Réponse :

$$T(\theta_1) = 6.509\ s \quad ; \quad T(\theta_2) = 7.394\ s\ ;$$

$$T(\theta_3) = 8.280\ s \quad ; \quad T(\theta_4) = 9.165\ s\ ]$$

8b) Sur la base des résultats obtenus en 8a), déterminer le comportement de la fonction $T(\theta_i)$ pour $\theta_i \sim \pi$.

[Réponse : $T(\theta_i) \sim c\ ln[(\pi - \theta_i) / \theta_0\ ]$
où $c = -1.278\ s$ *et* $\theta_0 = 8.00\ rad\ ]$

Exercice 9

Calculer avec trois chiffres significatifs les valeurs de $T(\theta_i)$ pour $\pi/100 \leq \theta_i \leq 99\ \pi/100$ avec incrément $\pi/100$ et faire un graphique de la fonction $T(\theta_i)$ dans l'intervalle $0 \leq \theta_i < \pi$.

```
plot [0:pi] "6eqdif9a.txt"
      ↑
    3.1415
```

Exercice 10

Déterminer avec cinq chiffres significatifs l'angle initial α_2 tel qu'en lâchant le pendule sans vitesse, la période du mouvement correspondant est $T_2 = 6.20$ s. Représenter graphiquement la loi du mouvement correspondant $\theta_2(t)$ pour $0 \leq t \leq 15$ s.

[Réponse : $\alpha_2 = 3.0790\ rad\ ]$ ⟹ trouver $Y_1(0)$ tq $Y_1\left(\frac{1}{4}\frac{T_2}{t}\right) = 0$ et $Y_1'\left(\frac{1}{4}\frac{T_2}{t}\right) = 0$

Mais comme ∄ frottement, α_2 est le même que après $\frac{T_2}{2}$; T_2 , $\frac{3}{2}T_2$ ...

$\theta(T/4) = 0$ et $\theta'(T/2) = 0$ ⟹ $Y(1, \frac{T}{4}) = 0$ et $Y(2, \frac{T}{2}) = 0$

Exercice 11

On considère $\theta_2(t)$ obtenu sous 10) qui est une fonction périodique de période $T_2 = 6.20$ s et peut être écrite en série de Fourier sous la forme

$$\theta_2(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos n \frac{2\pi t}{T_2} + b_n \sin n \frac{2\pi t}{T_2} \right] .$$

11a) Montrer analytiquement que

(i) $b_n = 0 \quad \forall\, n$

(ii) $a_n = 0 \quad \forall\, n$ pair

et que par conséquent on a

$$\theta_2(t) = \sum_{\nu=0}^{\infty} a_{2\nu+1} \cos (2\nu+1) \frac{2\pi t}{T_2} .$$

11b) Déterminer, avec erreur inférieure à 1×10^{-4} rad, les coefficients a_1, a_3 et a_5 des trois premiers termes du développement (pour l'intégration, utiliser la méthode de Simpson).

[Réponse : $a_1 = +3.5331$ rad ; $a_3 = -0.5536$ rad ; $a_5 = +0.1250$ rad]

11c) Déterminer une borne supérieure à la valeur absolue de l'erreur $\Delta(t) = \theta_2(t) - \tilde{\theta}_2(t)$ faite en remplaçant $\theta_2(t)$ par la somme $\tilde{\theta}_2(t)$ des trois premiers termes de son développement.

[Réponse : $|\Delta(t)| < 0.0436$ rad]

Exercice 12

Déterminer avec cinq chiffres significatifs l'angle initial α_1 tel qu'en lâchant le pendule sans vitesse, la période du mouvement correspondant est $T_1 = 2.56$ s. Représenter graphiquement la loi du mouvement correspondant $\theta_1(t)$ pour $0 \le t \le 6$ s.

[Réponse : $\alpha_1 = 1.8686$ rad]

Exercice 13

Mêmes questions que 11b) et 11c), mais pour la fonction $\theta_1(t)$ de période $T_1 = 2.56$ s obtenue sous 12).

[Réponse : $a_1 = +1.9108\ rad$; $a_3 = -0.0438\ rad$; $a_5 = +0.0017\ rad$;

$|\ \Delta(t)\ | < 0.0001\ rad$ *]*

Ex. 14
Ex. 15 ⟶ dans 4 sem. à répondre

Salles: - libres tous les vendredis (réservé pour nous)
- toutes les 2 semaines, séance ordinaire

Modules: - on va recevoir des fichiers

Méthode pour trouver l'angle initial correspondant à une période

p. exemple : $T = 4$, $\alpha_0 = ?$ (évtl. modifier la fonction)

6eqdif100.exe ⟶ modifier la période cherchée dans le programme:

p_c = 4

→ donner des valeurs pour cerner une première fois la valeur de la période le mieux possible: augmente les précisions jusqu'à la non convergence, angle init. et final: ~ 0.01 ; 3.1

→ passe à un autre prg, p.ex a trouvé ici: $\alpha_0 = 2,78415$

6eqdif10.exe ⟶ modifier la période cherchée dans le prg:

p_c = 4

→ donner la valeure trouvée avant : 2,78415

{ précision exigée (Δ période, période calculée) : 1e-x
{ incréments de l'angle : 1e-(x+1)

→ augmente tjs la précision:

→ $\alpha_0 = 2,78414$

Trouver le développement de Fourier

6eqdif11.exe ⟶ avec la période trouvée; modifier ce prg.
évtl. modifier la boucle sur les coeff. de fourier par calculer

$a_0, a_1, \ldots$
$b_0, b_1, \ldots$

Cas général: 6eqdi140.exe : permet de voir quels coefficients sont nuls.

1) Construction de matrices d_i+d_j et d_i-d_j
A B

EXERCICES

Exercice 1

Construire la base de $(2n+1)$ fonctions $|\alpha_i, l\rangle$ où $i = 0, \pm 1, \ldots, \pm n$, et évaluer la matrice de recouvrement $S_{\nu,\mu}$ $(\nu, \mu = 1, \ldots, 2n+1)$ correspondante en fonction des paramètres α_0, q, l et n donnés.

A l'aide de la procédure ssyev de la bibliothèque LAPACK, diagonaliser la matrice de recouvrement pour $\alpha_0 = 0.27$, $q = 1.4$, $n = 2$, $l = 0, 1, 2$. Déterminer les cinq valeurs propres $\sigma_\nu(l)$.

$|\alpha_i, l\rangle = A_{ir^l} r^l e^{-\alpha_i r}$; $2n+1$ fct. base
$\alpha_i = \alpha_0(l) \cdot q(l)^i$

[Réponse:

	ν=1	2	3	4	5
$\sigma_\nu(l=0)$	0.6140e−04	0.2314e−02	0.4736e−01	0.6205e+00	0.4330e+01
$\sigma_\nu(l=1)$	0.2728e−03	0.7337e−02	0.1035e+00	0.8873e+00	0.4002e+01
$\sigma_\nu(l=2)$	0.7772e−03	0.1588e−01	0.1689e+00	0.1078e+01	0.3736e+01]

si on a des v.p. < 0 , alors ∃ un problème
⇒ permet de vérifier si notre matrice a une allure correcte

Exercice 2

En sachant que les valeurs propres d'une matrice de recouvrement entre fonctions linéairement indépendantes sont positives, diagonaliser la matrice de recouvrement $S_{\nu,\mu}$ pour $\alpha_0 = 0.27$, $q = 1.4$, $l = 0$, et $n = 3$, 4, 5, 6, 7, et déterminer la valeur propre minimum $\sigma_1(l=0)$ et maximum $\sigma_{2n+1}(l=0)$ dans chaque cas. Constater qu'en augmentant le nombre de fonctions de base, la matrice n'est plus définie positive pour l'ordinateur.

[Réponse:

	$\sigma_1(l=0)$	$\sigma_{2n+1}(l=0)$
n=3	0.4202e−05	0.5444e+01
4	0.5237e−06	0.6260e+01
5	0.2634e−07	0.6862e+01
6	−0.5825e−07	0.7311e+01
7	−0.5044e−07	0.7653e+01]

↓ la matrice n'est plus définie positive ⇒ problème.

Exercice 3

En utilisant la même base qu'à l'Exercice 1, construire la matrice d'énergie $E^{(l)}_{\nu,\mu}$ de l'opérateur radial

$$K_l = -\frac{d^2}{dr^2} - \frac{2}{r}\frac{d}{dr} + \frac{l(l+1)+2B_l}{r^2} - \frac{2Z}{r},$$

pour l donné, et en fonction des paramètres Z et B_l.

Considérer le cas de l'hydrogène ($Z = 1$, $B_l = 0\ \forall\, l$), et à l'aide de la procédure ssygv de la bibliothèque LAPACK, calculer les quatre valeurs propres $\varepsilon_\nu\ (l)$ les plus basses ($\nu = 1,\ 2,\ 3,\ 4$) pour $l = 0,\ 1,$ et 2. Utiliser la base $\alpha_0 = 0.27$, $q = 1.4$, $n = 3$, et comparer les résultats avec les valeurs analytiques.

[Réponse:

	$\nu=1$	2	3	4
$\varepsilon_\nu\ (l=0)$	-0.9988	-0.2500	-0.1111	-0.0624
$\varepsilon_\nu\ (l=1)$	-0.2500	-0.1111	-0.0625	-0.0397
$\varepsilon_\nu\ (l=2)$	-0.1111	-0.0625	-0.0400	-0.0274]

Exercice 4

Utiliser les mêmes données qu'à l'Exercice 3, et étudier pour $l = 0$ comment varient les valeurs propres $\varepsilon_\nu\ (l = 0)$ lorsqu'on augmente le nombre de fonctions de base avec $n = 4,\ 5,$ et 6.

[Réponse:

	$\varepsilon_1\ (l=0)$	$\varepsilon_2\ (l=0)$	$\varepsilon_3\ (l=0)$	$\varepsilon_4\ (l=0)$
n=4	-1.0000	-0.2500	-0.1111	-0.0625
n=5	-1.0000	-0.2500	-0.1111	-0.0624
n=6	...	...	...	...]

Exercice 5

Mêmes questions qu'à l'Exercice 3, mais pour l'atome de sodium Na, pour lequel $Z = 1$, $B_0 = 0.51047$, $B_1 = 0.18288$, $B_2 = 0.02450$. Utiliser $\alpha_0 = 0.27$, $q = 1.4$, et $n = 5$.

[Réponse:

	ν=1	2	3	4
$\varepsilon_\nu\,(l=0)$	−0.3774	−0.1448	−0.0760	−0.0466
$\varepsilon_\nu\,(l=1)$	−0.2231	−0.1029	−0.0590	−0.0381
$\varepsilon_\nu\,(l=2)$	−0.1118	−0.0628	−0.0402	−0.0278

]

SYSTEMES ATOMIQUES A UN ELECTRON

EXERCICES (2ème série)

Exercice 6

Toujours pour l'atome de Na, étudier la convergence en fonction de n des quatre valeurs propres $\varepsilon_\nu(l)$ les plus basses ($\nu = 1, 2, 3, 4$) pour $l = 0$, 1 et 2 en utilisant la base définie par $\alpha_0 = 0.03$ et $q = 1.55$ ($l = 0$), $q = 1.50$ ($l = 1$), et $q = 1.45$ ($l = 2$). Donner les valeurs convergées de $\varepsilon_\nu(l)$ arrondies à 10^{-4} Ry près.

[Réponse :

	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$
$\varepsilon_\nu (l = 0)$	*–0.3776*	*–0.1449*	*–0.0760*	*–0.0467*
$\varepsilon_\nu (l = 1)$	*–0.2231*	*–0.1029*	*–0.0590*	*–0.0382*
$\varepsilon_\nu (l = 2)$	*–0.1118*	*–0.0628*	*–0.04015*	*–0.0278*]

Exercice 7

En utilisant la même base qu'aux exercices 1 et 3, construire la matrice représentative de l'opérateur r (distance électron-noyau) $R^{(l)}_{\nu,\mu} = \langle \alpha_\nu, l | r | \alpha_\mu, l \rangle = \frac{2l+3}{\alpha_\nu + \alpha_\mu} S^{(l)}_{\nu,\mu}$.

Considérer le cas de l'hydrogène et à l'aide de la procédure *ssygv*, calculer les coefficients des vecteurs propres corresondants aux trois niveaux d'énergie $\varepsilon_\nu(l)$ les plus bas ($\nu = 1, 2, 3$)]

Utiliser la même base que celle de l'exercice 6 avec n = 10.

A l'aide des coefficients des vecteurs propres et des éléments de matrice $R^{(l)}_{\nu,\mu}$, calculer pour chaque état propre la valeur moyenne de l'opérateur r arrondie à 10^{-2} a_0 près.

[Réponse :

	$\nu = 1$	$\nu = 2$	$\nu = 3$
$\langle r \rangle_{l=0}$	*1.50*	*6.00*	*13.57*
$\langle r \rangle_{l=1}$	*5.00*	*12.51*	*23.09*
$\langle r \rangle_{l=2}$	*10.50*	*21.01*	*34.57*]

Exercice 8

En utilisant les coefficients des vecteurs propres corresondants aux états 1s et 2s (c.à.d $\nu = 1$ et 2 pour $l = 0$) de l'hydrogène obtenus à l'exercice 7, tabuler les fonctions radiales $F_{10}(r)$ et $F_{20}(r)$ correspondantes dans l'intervalle $0 \leq r \leq 10\, a_0$ avec incrément de $0.1\, a_0$.

Afficher les deux fonctions radiales à l'écran à l'aide du logiciel *gnuplot*.

Exercice 9

Même question et mêmes données qu'à l'exercice 7, mais pour l'atome de sodium.

[Réponse :

	$\nu = 1$	$\nu = 2$	$\nu = 3$
$\langle r \rangle_{l=0}$	*3.46*	*9.85*	*19.32*
$\langle r \rangle_{l=1}$	*5.54*	*13.40*	*24.29*
$\langle r \rangle_{l=2}$	*10.44*	*20.92*	*34.45*

]

SYSTÈMES ATOMIQUES À UN ÉLECTRON
POTENTIEL MODÈLE DE FUES-SIMONS

opérateur de correction de Simons

projecteur sur le sous espace ℓ (élément angulaire)

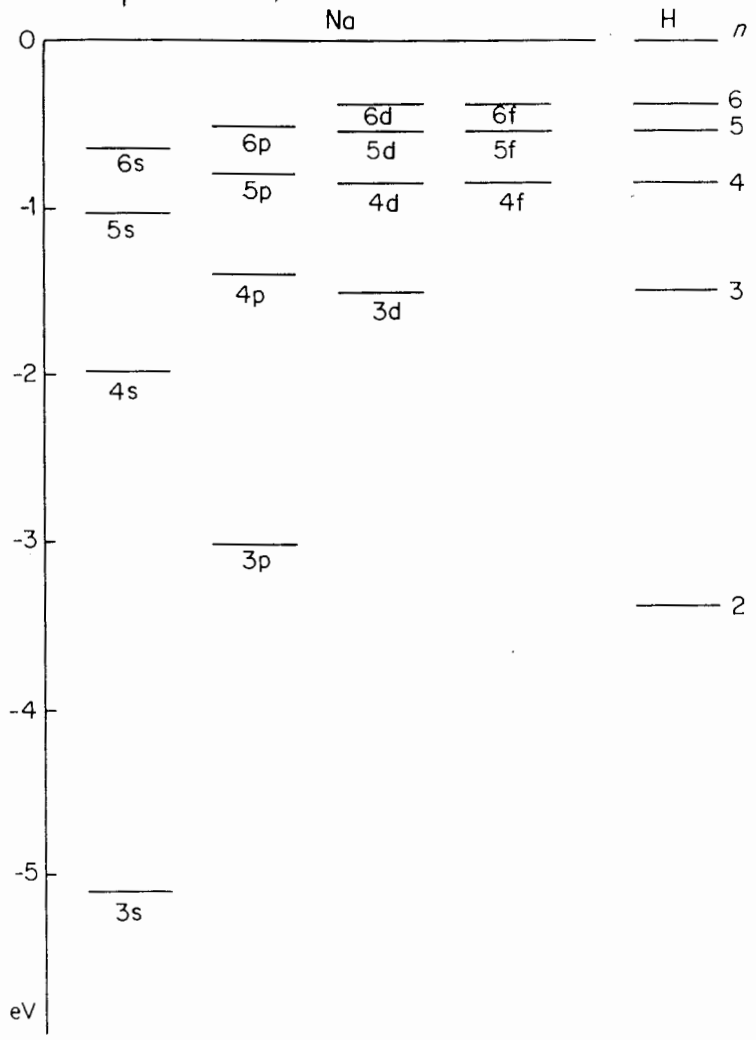
$$H = \frac{p^2}{2m} - \frac{Ze^2}{r} + \underbrace{\sum_{\ell=0}^{\infty} \frac{B_\ell}{r^2} P_\ell}_{\text{correction}}$$

ex: $Z = 1$ pour le Na, ou H

Exemple : atome de Na $\left[1s^2\ 2s^2\ 2p^6\ \underbrace{3s^1}_{\text{électron extérieur}}\right]$; $Z = 1$,

$B_0 = 0.51047$, $B_1 = 0.18288$, $B_2 = 0.02450$, $B_\ell = 0$ pour $\ell > 2$.

(immenses corrections comparé à e^2)



test: connaît le spectre analytiquement, compare ce que on obtient avec le spectre connu.

The energy levels of the sodium atom in electron volts for $l = 0, 1, 2$ and 3 with, on the right, the levels of the hydrogen atom for $n = 2$ to 6.

Constantes du mouvement

$$[H, \ell^2] = [H, \ell_z] = [\ell^2, \ell_z] = 0$$

Unités atomiques

$$a_0 = \frac{\hbar^2}{m e^2} \quad ; \quad R_y = \frac{m e^4}{2\hbar^2} = \frac{e^2}{2 a_0} \doteq 13{,}6\,[\text{eV}]$$

Hamiltonien en unités atomiques

$$H = -\Delta - \frac{2Z}{r} + \sum_{\ell=0}^{\infty} \frac{2 B_\ell}{r^2} P_\ell$$

Etats propres (potentiel central)

$$|n, \ell, m\rangle \equiv F_{n\ell}(r) \cdot Y_{\ell m}(\vartheta, \varphi)$$

Orthonormalité

$$\int Y^*_{\ell m}(\vartheta, \varphi)\, Y_{\ell' m'}(\vartheta, \varphi)\, d\Omega = \delta_{\ell\ell'}\, \delta_{mm'}$$

($d\Omega$: angle solide ; $Y_{\ell' m'}$: connu)

$$\int_0^\infty r^2 F^*_{n\ell}(r)\, F_{n'\ell}(r)\, dr = \delta_{nn'}$$

($F_{n'\ell}$: inconnu)

Equation radiale pour ℓ et m donnés

$$\left[-\frac{d^2}{dr^2} - \frac{2}{r}\frac{d}{dr} + \frac{\ell(\ell+1)}{r^2} - \frac{2Z}{r} + \frac{2B_\ell}{r^2}\right] F_{n\ell}(r) = \varepsilon_{n\ell} F_{n\ell}(r)$$

éqn radiale pour chaque ℓ $\Rightarrow$ le projecteur a disparu

Comportement de $F_{n\ell}(r)$ pour $r \sim 0$

$$F_{n\ell}(r) \sim c_{n\ell} r^{\gamma} + d_{n\ell} r^{\gamma+1} + \cdots$$

avec γ tel que

$$\begin{cases} \gamma > 0 \\ \gamma(\gamma+1) = \ell(\ell+1) + 2B_\ell \end{cases}$$

En particulier :

$\gamma = \ell$ si $B_\ell = 0$

FONCTIONS RADIALES DE BASE POUR ℓ DONNÉ

FONCTIONS RADIALES pour 1 ℓ donné

$$|\alpha_i, \ell\rangle \equiv \underbrace{A_{i\ell}}_{\text{cte normalis.}} \underbrace{r^{\ell}}_{\text{pour le comportement à l'origine ; pas de correction de Simons}} e^{-\alpha_i r} = \varphi_{i,\ell}(r)$$ } fonctions de base

$$\alpha_i = \alpha_0(\ell) \cdot q(\ell)^{i} \qquad (i = 0, \pm 1, \pm 2, \cdots, \pm n)$$

$$\alpha_{i+1} = q \cdot \alpha_i$$ } série géométrique

$\alpha_0(\ell)$
$\alpha_1 = \alpha_0 \cdot q^1$
$\alpha_2 = q \cdot \alpha_1 = \alpha_0 \cdot q^2$
$\alpha_3 = \alpha_0 \cdot q^3$

Normalisation $\{\alpha_0, q\}$ $\Rightarrow$ détermine les fct. cherchées

$\{\alpha_0, q, n\}$ détermine les fct. propres

$$\int_0^{\infty} r^{\nu} e^{-ar}\, dr = \frac{\nu!}{a^{\nu+1}}$$

$$\langle \alpha_i, \ell \,|\, \alpha_i, \ell \rangle = A_{i,\ell}^2 \int_0^{\infty} r^{2\ell+2} e^{-2\alpha_i r}\, dr =$$

$$= A_{i,\ell}^2 \frac{(2\ell+2)!}{(2\alpha_i)^{2\ell+3}} = 1$$

$$A_{i,\ell} = \frac{(2\alpha_i)^{\ell+\frac{3}{2}}}{\sqrt{(2\ell+2)!}}$$

de de normalisation

$|\alpha_i, \ell\rangle = A_{i\ell} \cdot r^{\ell} \cdot e^{-\alpha_i r}$

Recouvrement $\Leftrightarrow$ $\not\exists$ orthonormalité

$$S_{i,j}^{(\ell)} = \langle \alpha_i, \ell \,|\, \alpha_j, \ell \rangle =$$

$$= A_{i,\ell}\, A_{j,\ell} \int_0^{\infty} r^{2\ell+2} e^{-(\alpha_i+\alpha_j) r}\, dr =$$

$$= \frac{(4\alpha_i\alpha_j)^{\ell+3/2}}{(2\ell+2)!} \cdot \frac{(2\ell+2)!}{(\alpha_i+\alpha_j)^{2\ell+3}}$$

$$= \left[\frac{4\alpha_i\alpha_j}{(\alpha_i+\alpha_j)^2}\right]^{\ell+3/2}$$

orthonormalité : $a_{ij} = \delta_{ij}$

ici : $a_{ij} = \ldots$

ELEMENTS DE MATRICE DE L'HAMILTONIEN

$$\begin{cases} E^{(\ell)}_{i,j} = \langle \alpha_i, \ell \,|\, K_\ell \,|\, \alpha_j, \ell \rangle \\ K_\ell = -\frac{d^2}{dr^2} - \frac{2}{r}\frac{d}{dr} + \frac{\ell(\ell+1)+2B_\ell}{r^2} - \frac{2Z}{r} \end{cases}$$

→ à diagonaliser avec recouvrement pour obtenir les énergies

(c.l.p 3 : éqn. radiale)

Action de K_ℓ sur $\varphi_{j,\ell}(r)$

$$\frac{d\varphi_{j,\ell}(r)}{dr} = \left(\frac{\ell}{r} - \alpha_j\right)\varphi_{j,\ell}(r)$$

$$\frac{d^2\varphi_{j,\ell}(r)}{dr^2} = \left[\frac{\ell(\ell+1)}{r^2} - \frac{2\alpha_j \ell}{r} + \alpha_j^2\right]\varphi_{j,\ell}(r)$$

$$K_\ell\,\varphi_{j,\ell}(r) = \Big[-\frac{\ell(\ell-1)}{r^2} + \frac{2\alpha_j\ell}{r} - \alpha_j^2 - \frac{2\ell}{r^2} + \frac{2\alpha_j}{r} + \frac{\ell(\ell+1)+2B_\ell}{r^2} - \frac{2Z}{r}\Big]\varphi_{j,\ell}(r) =$$

$$= \left[\frac{2B_\ell}{r^2} + 2\frac{(\ell+1)\alpha_j - Z}{r} - \alpha_j^2\right]\varphi_{j,\ell}(r).$$

Eléments de matrice fondamentaux

$$\langle \alpha_{i,\ell} \mid \alpha_{j,\ell} \rangle = \left[\frac{4\alpha_i \alpha_j}{(\alpha_i + \alpha_j)^2} \right]^{\ell + 3/2} = S_{i,j}^{(\ell)}$$

recouvrement — matrice de recouvrement

$$\langle \alpha_{i,\ell} \mid \frac{1}{r^2} \mid \alpha_{j,\ell} \rangle = A_{i,\ell}\, A_{j,\ell} \int_0^\infty r^{2\ell} e^{-(\alpha_i + \alpha_j) r}\, dr =$$

⇓ pour représenter la partie radiale de l'Hamiltonien

$$= \frac{(4\alpha_i \alpha_j)^{\ell+3/2}}{(2\ell+2)!} \frac{\ell!}{(\alpha_i + \alpha_j)^{2\ell+1}}$$

$$= \frac{(\alpha_i + \alpha_j)^2}{(2\ell+1)(2\ell+2)} S_{i,j}^{(\ell)}$$

$$\langle \alpha_{i,\ell} \mid \frac{1}{r} \mid \alpha_{j,\ell} \rangle = \frac{\alpha_i + \alpha_j}{2\ell + 2} S_{i,j}^{(\ell)}$$

$$\langle \alpha_{i,\ell} \mid r \mid \alpha_{j,\ell} \rangle = \frac{2\ell + 3}{\alpha_i + \alpha_j} S_{i,j}^{(\ell)}$$

$$\langle \alpha_{i,\ell} \mid K_\ell \mid \alpha_{j,\ell} \rangle = \left[2B_\ell \frac{(\alpha_i + \alpha_j)^2}{(2\ell+1)(2\ell+2)} + 2 \frac{[(\ell+1)\alpha_j - Z](\alpha_i + \alpha_j)}{2\ell+2} - \alpha_j^2 \right] S_{i,j}^{(\ell)} =$$

hamiltonien — nb. atomique

$$= \left[2B_\ell \frac{(\alpha_i + \alpha_j)^2}{(2\ell+1)(2\ell+2)} + \alpha_i \alpha_j - \frac{Z(\alpha_i + \alpha_j)}{\ell+1} \right] S_{i,j}^{(\ell)}$$

$O\psi = \varepsilon\psi$

$\langle e_i | e_j \rangle = S_{ij}$ (s et par S)

$\Rightarrow \boxed{\det(O_{ij} - \lambda S_{ij}) = 0}$

$\psi = \sum_i c_i \cdot e_i$

$\Rightarrow O\left(\sum_i c_i |e_i\rangle\right) = \varepsilon\left(\sum_i c_i |e_i\rangle\right)$

$\cdot \langle e_j | \Rightarrow$

$\sum_i c_i \langle e_j | O | e_i \rangle = \varepsilon \sum_i c_i \langle e_j | e_i \rangle$

$\Rightarrow \sum_i c_i (O_{ji} - \varepsilon S_{ji}) = 0 \quad . j = 1, \dots, n$

$\Rightarrow \det(O_{ij} - \lambda \cdot S_{ij})$ *

$a_{ij} = [\ ]\, S_{ij}^{(\ell)}$ t.q. $A^+ = A$

A DIAGONALISER POUR TROUVER LES ENERGIES ⇒ spectre

$\alpha_i, \alpha_j = ?$

on doit diagonaliser:

$\langle \alpha_{i\ell} | k_\ell | \alpha_{j\ell} \rangle - \lambda \cdot \langle \alpha_{i\ell} | \alpha_{j\ell} \rangle$

$\Rightarrow \det\left[[\dots] S_{ij} - \lambda \cdot S_{ij} \right] = 0$

$A \cdot x = \lambda \cdot x$

NAME

SSYEV - compute all eigenvalues and, optionally, eigenvectors of a real symmetric matrix A

SYNOPSIS

```
SUBROUTINE SSYEV( JOBZ, UPLO, N, A, LDA, W, WORK, LWORK, INFO )

    CHARACTER      JOBZ, UPLO

    INTEGER        INFO, LDA, LWORK, N

    REAL           A( LDA, * ), W( * ), WORK( * )
```

RPOSE

SSYEV computes all eigenvalues and, optionally, eigenvectors of a real symmetric matrix A.

ARGUMENTS

JOBZ (input) CHARACTER*1
= 'N': Compute eigenvalues only;
= 'V': Compute eigenvalues and eigenvectors.

UPLO (input) CHARACTER*1
= 'U': Upper triangle of A is stored;
= 'L': Lower triangle of A is stored.

N (input) INTEGER
The order of the matrix A. N >= 0.

A (input/output) REAL array, dimension (LDA, N)
On entry, the symmetric matrix A. If UPLO = 'U', the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A. On exit, if JOBZ = 'V', then if INFO = 0, A contains the orthonormal eigenvectors of the matrix A. If JOBZ = 'N', then on exit the lower triangle (if UPLO='L') or the upper triangle (if UPLO='U') of A, including the diagonal, is destroyed.

LDA (input) INTEGER
The leading dimension of the array A. LDA >= max(1,N).

N = dim

W (output) REAL array, dimension (N)
If INFO = 0, the eigenvalues in ascending order.

WORK (workspace/output) REAL array, dimension (LWORK)
On exit, if INFO = 0, WORK(1) returns the optimal LWORK.

↳ tableau temporaire de travail

LWORK (input) INTEGER
The length of the array WORK. LWORK >= max(1,3*N-1). For optimal efficiency, LWORK >= (NB+2)*N, where NB is the blocksize for SSYTRD returned by ILAENV.

INFO (output) INTEGER
= 0: successful exit
< 0: if INFO = -i, the i-th argument had an illegal value

> 0: if INFO = i, the algorithm failed to converge; i off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

⚠ instabilité si recouvrement proche de 1

SSYGV(1) LAPACK driver routine (version 2.0) SSYGV(1)

NAME

SSYGV - compute all the eigenvalues, and optionally, the eigenvectors of a real generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x

SYNOPSIS

```
SUBROUTINE SSYGV( ITYPE, JOBZ, UPLO, N, A, LDA, B, LDB, W, WORK, LWORK,
                  INFO )

    CHARACTER      JOBZ, UPLO

    INTEGER        INFO, ITYPE, LDA, LDB, LWORK, N

    REAL           A( LDA, * ), B( LDB, * ), W( * ), WORK( * )
```

PURPOSE

SSYGV computes all the eigenvalues, and optionally, the eigenvectors of a real generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. Here A and B are assumed to be symmetric and B is also positive definite.

ARGUMENTS

ITYPE (input) INTEGER
↓ type de pb.
Specifies the problem type to be solved:
= 1: A*x = (lambda)*B*x
= 2: A*B*x = (lambda)*x
= 3: B*A*x = (lambda)*x

JOBZ (input) CHARACTER*1
↓ N = val. prop.
V = val. prop. + vect. p.
= 'N': Compute eigenvalues only;
= 'V': Compute eigenvalues and eigenvectors.

UPLO (input) CHARACTER*1
= 'U': Upper triangles of A and B are stored;
= 'L': Lower triangles of A and B are stored.

N (input) INTEGER
dim de la matrice
The order of the matrices A and B. N >= 0.

A (input/output) REAL array, dimension (LDA, N)
On entry, the symmetric matrix A. If UPLO = 'U', the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A.

On exit, if JOBZ = 'V', then if INFO = 0, A contains the matrix Z of eigenvectors. The eigenvectors are normalized as follows: if ITYPE = 1 or 2, Z**T*B*Z = I; if ITYPE = 3, Z**T*inv(B)*Z = I. If JOBZ = 'N', then on exit the upper triangle (if UPLO='U') or the lower triangle (if UPLO='L') of A, including the diagonal, is destroyed.

LDA (input) INTEGER
↓
The leading dimension of the array A. LDA >= max(1,N).

dimension de la matrice

dans notre cas. N = LDA

B (input/output) REAL array, dimension (LDB, N)
On entry, the symmetric matrix B. If UPLO = 'U', the leading N-by-N upper triangular part of B contains the upper triangular part of the matrix B. If UPLO = 'L', the leading N-by-N lower triangular part of B contains the lower triangular part of the matrix B.

On exit, if INFO <= N, the part of B containing the matrix is overwritten by the triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

LDB (input) INTEGER
The leading dimension of the array B. LDB >= max(1,N).
↳ LDA = N

W (output) REAL array, dimension (N)
If INFO = 0, the eigenvalues in ascending order.
↳ contenant les N v.p. de notre pb. (val. propres)

WORK (workspace/output) REAL array, dimension (LWORK)
On exit, if INFO = 0, WORK(1) returns the optimal LWORK.

LWORK (input) INTEGER
The length of the array WORK. LWORK >= max(1,3*N-1). For optimal efficiency, LWORK >= (NB+2)*N, where NB is the blocksize for SSYTRD returned by ILAENV.

INFO (output) INTEGER
↳ la valeur (param. de sortie) dépend de l'exécution de la phase de calcul
= 0: successful exit
< 0: if INFO = -i, the i-th argument had an illegal value
> 0: SPOTRF or SSYEV returned an error code:
<= N: if INFO = i, SSYEV failed to converge; i off-diagonal elements of an intermediate tridiagonal form did not converge to zero;
> N: if INFO = N + i, for 1 <= i <= N, then the leading minor of order i of B is not positive definite. The factorization of B could not be completed and no eigenvalues or eigenvectors were computed.

LAPACK : — systèmes linéaires
— syst. surdéterminés
— pb. aux valeurs propres

f90 file1.f90 file2.f90 ... -xlic_lib=sunperf
écrit notre prg. avec les modules
ex: f90 prog.f90 biblio.f90 -xlic_lib=sunperf

(dec : f90 file1.f90 file2.f90 ... -ldxml)

```
module procedures
  implicit none
  real :: z,q                      !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl       !facteurs de correction de fues-simons
  integer :: l                     !nombre quantiques l
  integer :: n                     !définition de la taille de la base
  integer :: dim                   !la dimension des matrices: dim=2*n+1
  real :: alph_mid                 !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

!    subroutine hamilt   !hamiltonien de notre système
!      implicit none
!      real :: lval
!
!      k = 0.0
!      where (mask) k =
!    end subroutine hamilt

end module procedures


!-------------------------------------------------------------------------

program atome
  use procedures
```

le vecteur

construit une matrice triangulaire inférieure

```
  implicit none
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.27              !c'est alpha zéro
  q = 1.4                      !facteur pour calculer alpha
  n=2
  dim = 2*n+1

  lwork = 3*dim-1
  allocate(work(lwork))
  allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
  allocate(val_propres(dim))

  call basis(q,alph_mid,n)

  do l=0,2
    call overlap
    call ssyev('N','L',dim,s,dim,val_propres,work,lwork,info)
    if (info /= 0) then
      print *,'ssyev problème'
      stop
    end if
    write (*,'(5(e11.4,1x))') val_propres
  end do

  deallocate(alf,alf_s,alf_p,mask,s,k)
  deallocate(val_propres,work)



end program atome
```

```
module procedures
  implicit none
  real :: z,q                          !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl           !facteurs de correction de fues-simons
  integer :: l                         !nombre quantiques l
  integer :: n                         !définition de la taille de la base
  integer :: dim                       !la dimension des matrices: dim=2*n+1
  real :: alph_mid                     !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

!    subroutine hamilt   !hamiltonien de notre système
!      implicit none
!      real :: lval
!
!      k = 0.0
!      where (mask) k =
!    end subroutine hamilt

end module procedures


!-------------------------------------------------------------------------

program atome2
  use procedures
```

```
  implicit none
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.27              !c'est alpha zéro
  q = 1.4                      !facteur pour calculer alpha

  l=0

  do n=3,7
    dim = 2*n+1
    lwork = 3*dim-1
    allocate(work(lwork))
    allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
    allocate(val_propres(dim))

    call basis(q,alph_mid,n)


    call overlap
    call ssyev('N','L',dim,s,dim,val_propres,work,lwork,info)
    if (info /= 0) then
      print *,'ssyev problème'
      stop
    end if

    write (*,'(i3,"  ",e10.4,"  ",e10.4)') n,minval(val_propres),maxval(val_propres)

    deallocate(alf,alf_s,alf_p,mask,s,k)
    deallocate(val_propres,work)
  end do



end program atome2
```

```
module procedures
  implicit none
  real :: z,q                          !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl           !facteurs de correction de fues-simons
  integer :: l                         !nombre quantiques l
  integer :: n                         !définition de la taille de la base
  integer :: dim                       !la dimension des matrices: dim=2*n+1
  real :: alph_mid                     !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

    subroutine hamilt   !hamiltonien de notre système
      implicit none
      real :: lval
      k = 0.0
      where (mask) k = (2*bl(l)*alf_s**2/((2*l+1)*(2*l+2))+alf_p-z*alf_s/(l+1))*s
    end subroutine hamilt

end module procedures


!-------------------------------------------------------------------------

program atome3
  use procedures
  implicit none
```

```
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.27              !c'est alpha zéro
  q = 1.4                      !facteur pour calculer alpha
  n=3

  dim = 2*n+1
  lwork = 3*dim-1
  allocate(work(lwork))
  allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
  allocate(val_propres(dim))

  call basis(q,alph_mid,n)

  do l=0,2
    call overlap
    call hamilt
    call ssygv(1,'N','L',dim,k,dim,s,dim,val_propres,work,lwork,info)
    if (info /= 0) then
      print *,'ssygv problème'
      stop
    end if

    write (*,'(5(e11.4,1x))') val_propres
  end do

  deallocate(alf,alf_s,alf_p,mask,s,k)
  deallocate(val_propres,work)

end program atome3
```

```
module procedures
  implicit none
  real :: z,q                          !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl           !facteurs de correction de fues-simons
  integer :: l                         !nombre quantiques l
  integer :: n                         !définition de la taille de la base
  integer :: dim                       !la dimension des matrices: dim=2*n+1
  real :: alph_mid                     !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

    subroutine hamilt   !hamiltonien de notre système
      implicit none
      real :: lval
      k = 0.0
      where (mask) k = (2*bl(l)*alf_s**2/((2*l+1)*(2*l+2))+alf_p-z*alf_s/(l+1))*s
    end subroutine hamilt

end module procedures


!-------------------------------------------------------------------------

program atome4
  use procedures
  implicit none
```

```
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.27              !c'est alpha zéro
  q = 1.4                      !facteur pour calculer alpha
  l=0


do n=4,8

  dim = 2*n+1
  lwork = 3*dim-1
  allocate(work(lwork))
  allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
  allocate(val_propres(dim))

  call basis(q,alph_mid,n)

  call overlap
  call hamilt
  call ssygv(1,'N','L',dim,k,dim,s,dim,val_propres,work,lwork,info)
  if (info /= 0) then
    print *,'ssygv problème'
    stop
  end if

  write (*,'(i3,"  ",e10.4,"  ",e10.4,"  ",e10.4,"  ",e10.4)') n,val_propres(1),val_propre
  deallocate(alf,alf_s,alf_p,mask,s,k)
  deallocate(val_propres,work)

end do


end program atome4
```

```
module procedures
  implicit none
  real :: z,q                          !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl           !facteurs de correction de fues-simons
  integer :: l                         !nombre quantiques l
  integer :: n                         !définition de la taille de la base
  integer :: dim                       !la dimension des matrices: dim=2*n+1
  real :: alph_mid                     !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

    subroutine hamilt   !hamiltonien de notre système
      implicit none
      real :: lval
      k = 0.0
      where (mask) k = (2*bl(l)*alf_s**2/((2*l+1)*(2*l+2))+alf_p-z*alf_s/(l+1))*s
    end subroutine hamilt

end module procedures

!-------------------------------------------------------------------------

program atome5
  use procedures
  implicit none
```

```
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.27              !c'est alpha zéro
  q = 1.4                      !facteur pour calculer alpha
  n=5

  dim = 2*n+1
  lwork = 3*dim-1
  allocate(work(lwork))
  allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
  allocate(val_propres(dim))

  call basis(q,alph_mid,n)

  do l=0,2
    call overlap
    call hamilt
    call ssygv(1,'N','L',dim,k,dim,s,dim,val_propres,work,lwork,info)
    if (info /= 0) then
      print *,'ssygv problème'
      stop
    end if
    write (*,'(i3,"  ",e10.4,"  ",e10.4,"  ",e10.4,"  ",e10.4)') l,val_propres(1),val_prop
  end do

  deallocate(alf,alf_s,alf_p,mask,s,k)
  deallocate(val_propres,work)

end program atome5
```

```
module procedures
  implicit none
  real :: z,q                          !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl           !facteurs de correction de fues-simons
  integer :: l                         !nombre quantiques l
  integer :: n                         !définition de la taille de la base
  integer :: dim                       !la dimension des matrices: dim=2*n+1
  real :: alph_mid                     !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

    subroutine hamilt   !hamiltonien de notre système
      implicit none
      real :: lval
      k = 0.0
      where (mask) k = (2*bl(l)*alf_s**2/((2*l+1)*(2*l+2))+alf_p-z*alf_s/(l+1))*s
    end subroutine hamilt

end module procedures


!-------------------------------------------------------------------------

program atome6
  use procedures
  implicit none
```

```
    character (len=10) :: formule
    integer :: info,lwork
    real, allocatable, dimension(:) :: val_propres
    real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev

    write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
    read *,formule
    select case(formule)
      case ('H')
        bl = (/0.,0.,0.,0./)
        z = 1
      case ('Li2+')
        bl = (/0.,0.,0.,0./)
        z = 3
      case ('Na')
        bl = (/0.51047,0.18288,0.02450,0./)
        z = 1
      case default
        print *,'cas non prévu'
    end select

    write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
    read *,n
    !write (*,'("Nombre quantique l: ")', advance = 'no')
    !read *,l
    alph_mid = 0.03              !c'est alpha zéro


    dim = 2*n+1
    lwork = 3*dim-1
    allocate(work(lwork))
    allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
    allocate(val_propres(dim))

    do l=0,2
      if (l==0) then
        q=1.55
      end if
      if (l==1) then
        q = 1.5
      end if
      if (l==2) then
        q = 1.45
      end if
      call basis(q,alph_mid,n)
      call overlap
      call hamilt
      call ssygv(1,'N','L',dim,k,dim,s,dim,val_propres,work,lwork,info)
      if (info /= 0) then
        print *,'ssygv problème'
        stop
      end if
      write (*,'(i3,"  ",e10.4,"  ",e10.4,"  ",e10.4,"  ",e10.4)') l,val_propres(1),val_prop
      deallocate(mask)
    end do

    deallocate(alf,alf_s,alf_p,s,k)
    deallocate(val_propres,work)

end program atome6
```

```
module procedures
  implicit none
  real :: z,q                      !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl       !facteurs de correction de fues-simons
  integer :: l                     !nombre quantiques l
  integer :: n                     !définition de la taille de la base
  integer :: dim                   !la dimension des matrices: dim=2*n+1
  real :: alph_mid                 !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

    subroutine hamilt   !hamiltonien de notre système
      implicit none
      real :: lval
      k = 0.0
      where (mask) k = (2*bl(l)*alf_s**2/((2*l+1)*(2*l+2))+alf_p-z*alf_s/(l+1))*s
    end subroutine hamilt

    function rayon(no)
    !Calcule le rayon moyen dans un état propre no
    !Besoin: K qui contient les vecteurs propres
    !no donne l'état propre correspondant
      implicit none
      real :: rayon
      integer, intent(in) :: no
      integer :: i,j
```

```
      real :: lval
      real, allocatable, dimension(:,:) :: R
      lval = real(l)
      allocate(R(dim,dim))
      R = ((2*lval + 3)/alf_s)*S
      rayon = 0.
      do i=1,dim
        do j=1,dim
          rayon = rayon + K(i,no)*K(j,no)*R(i,j)
        end do
      end do
      deallocate(R)
    end function rayon

end module procedures


!-------------------------------------------------------------------------

program atome7
  use procedures
  implicit none
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev
  real, dimension(3) :: r     !dans ce tableau on stocke les 3 rayons moyens

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.03               !c'est alpha zéro
  n=10

  dim = 2*n+1
  lwork = 3*dim-1
  allocate(work(lwork))
  allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
  allocate(val_propres(dim))

  do l=0,2
    if (l==0) then
      q=1.55
    end if
    if (l==1) then
      q = 1.5
    end if
    if (l==2) then
      q = 1.45
```

```
    end if
    call basis(q,alph_mid,n)
    call overlap
    call hamilt
    call ssygv(1,'V','L',dim,k,dim,s,dim,val_propres,work,lwork,info)
    !Les vecteurs propres seront contenus dans l'hamiltonien K, par colonne
    if (info /= 0) then
      print *,'ssygv problème'
      print *,info
      stop
    end if
    !On doit recalculer la matrice de recouvrement complètement avec ces 2 lignes
    !suivantes pour obtenir un rayon correct
    mask = .true.
    call overlap
    r(1) = rayon(1)
    r(2) = rayon(2)
    r(3) = rayon(3)
    write (*,'(i3,"   ",f10.4,"   ",f10.4,"   ",f10.4)') l,r(1),r(2),r(3)
    deallocate(mask)
  end do

  deallocate(alf,alf_s,alf_p,s,k)
  deallocate(val_propres,work)

end program atome7
```

```
module procedures
  implicit none
  real :: z,q                    !numéro de l'élément,facteur pour calculer alpha
  real, dimension(0:3) :: bl     !facteurs de correction de fues-simons
  integer :: l                   !nombre quantiques l
  integer :: n                   !définition de la taille de la base
  integer :: dim                 !la dimension des matrices: dim=2*n+1
  real :: alph_mid               !c'est alpha zéro

  real, allocatable, dimension(:) :: alf
  !temporaire pour basis,c'est la matrice alpha tout court
  real, allocatable, dimension(:,:) :: alf_s, alf_p
  !_s et _p c'est somme et produit
  real, allocatable, dimension(:,:) :: s,k
  !c'est la matrice s=recouvrement, h=hamiltonien
  logical, allocatable, dimension(:,:) :: mask
  !variable logique de test

  contains

    subroutine basis(q,alf_mid,n_alf)
    !calcule les matrices ai*aj et ai + aj
    !alf_mid = c'est alpha zéro
    !n_alf = le nombre cherché de fonctions
    !routine indépendante de l
    !alloue MASK(DIM,DIM) ici; a désallouer après
      implicit none
      integer, intent(in) :: n_alf
      real, intent(in) :: q,alf_mid
      integer :: i
      alf = (/(alf_mid/q**i,i=n_alf,1,-1), alf_mid, (alf_mid*q**i,i=1,n_alf)/)
      !on construit le tableau des alphai = alpha0*q**i pour i=-n,...,+n
      alf_s = 0.0        !c'est la matrice ai + aj
      alf_p = 0.0        !c'est la matrice ai*aj
      allocate(mask(dim,dim))   !dimensions corresp au nb de fct propres
      mask =.false.
      do i=1,dim
        alf_s(i,:) = alf(i)+alf  !matrice somme
        alf_p(i,:) = alf(i)*alf    !matrice produit
        mask(i,1:i) = .true.
      end do
    end subroutine basis

    subroutine overlap   !matrice de recouvrement
      implicit none
      real :: lval
      lval = real(l)
      s = 0.0
      where (mask) s = (4.*alf_p/(alf_s)**2)**(lval+3./2)
    end subroutine overlap

    subroutine hamilt   !hamiltonien de notre système
      implicit none
      real :: lval
      k = 0.0
      where (mask) k = (2*bl(l)*alf_s**2/((2*l+1)*(2*l+2))+alf_p-z*alf_s/(l+1))*s
    end subroutine hamilt

    function fac(p)
    !calcule la factorielle du nombre p
      implicit none
      integer :: fac,i
      integer, intent(in) :: p
      fac = 1
      do i=1,p
        fac = fac*i
```

```
      end do
    end function fac

    function F(r,no)
    !Fonction qui donne la fonction radiale évaluée au point r en fct des
    !paramètres établis précédemment
    !no = fonction propre choisie
      implicit none
      real, intent(in) :: r
      integer, intent(in) :: no
      real :: F
      integer :: i
      real :: lval
      lval = real(l)
      F = 0.
      do i=1,dim
        F = F + K(i,no)*r**(lval)*exp(-r*alf(i))*((2*alf(i))**(lval+3/2.))/(fac(2*l+1))**(
      end do
    end function F

end module procedures


!-------------------------------------------------------------------------

program atome8
  use procedures
  implicit none
  character (len=10) :: formule
  integer :: info,lwork
  real, allocatable, dimension(:) :: val_propres
  real, allocatable, dimension(:) :: work        !tableau temporaire pour ssyev
  integer :: i

  write (*,'("Donner le symbole de l''atome : ")', advance = 'no')
  read *,formule
  select case(formule)
    case ('H')
      bl = (/0.,0.,0.,0./)
      z = 1
    case ('Li2+')
      bl = (/0.,0.,0.,0./)
      z = 3
    case ('Na')
      bl = (/0.51047,0.18288,0.02450,0./)
      z = 1
    case default
      print *,'cas non prévu'
  end select

  !write (*,'("Définition de la base n t.q. dimB = 2*n +1 : ")', advance = 'no')
  !read *,n
  !write (*,'("Nombre quantique l: ")', advance = 'no')
  !read *,l
  alph_mid = 0.03              !c'est alpha zéro
  n=10
  q=1.55

  dim = 2*n+1
  lwork = 3*dim-1
  allocate(work(lwork))
  allocate(alf(dim),alf_s(dim,dim),alf_p(dim,dim),s(dim,dim),k(dim,dim))
  allocate(val_propres(dim))

  l=0
  call basis(q,alph_mid,n)
```

```
  call overlap
  call hamilt
  call ssygv(1,'V','L',dim,k,dim,s,dim,val_propres,work,lwork,info)
  !Les vecteurs propres seront contenus dans l'hamiltonien K, par colonne
  if (info /= 0) then
    print *,'ssygv problème'
    print *,info
    stop
  end if
  open (unit=1, file='7ato8a.txt')
  do i=1,100
    write (1,*) 0.1*i,F(0.1*i,1)
  end do
  open (unit=1, file='7ato8b.txt')
  do i=1,100
    write (1,*) 0.1*i,F(0.1*i,2)
  end do
  deallocate(alf,alf_s,alf_p,s,k,mask)
  deallocate(val_propres,work)

end program atome8
```