

UTILISATION DE MATLAB POUR LA RESOLUTION DE PROBLEMES PHYSIQUES

François AUGER, Bruno VELAY
IUT, 58, rue Michel Ange, B.P. 420
44606 Saint Nazaire Cedex
E-mail : f.auger@ieee.org.

1 Introduction

L'utilisation de l'outil informatique dans l'enseignement des sciences physiques se justifie également par l'emploi de logiciels interactifs de calcul scientifique, qui permettent la résolution numérique ou analytique de problèmes physiques concrets, la visualisation des résultats sous une forme efficace et pédagogique, et cela avec une durée d'exécution suffisamment rapide pour pouvoir ensuite étudier la variation de la solution en fonction de ses paramètres.

Outre les logiciels *Maple* et *Mathematica* présentés dans un numéro récent du "Bulletin de l'Union des Physiciens", *Matlab* répond également à ce besoin. Développé initialement par des universitaires américains pour faciliter l'enseignement de l'analyse numérique, ce logiciel est en train de devenir une sorte d'*esperanto* moderne des scientifiques et des ingénieurs désireux de partager des modèles physiques et des méthodes de résolution. Ses quelques 600.000 utilisateurs de part le monde sont en effet séduits par

- sa simplicité d'utilisation: il suffit de faire `help fft` pour faire afficher à l'écran la syntaxe d'appel de la fonction `fft` et la description de son action.
- ses nombreuses bibliothèques de fonctions préexistantes, simplifiant la résolution des problèmes. Ce n'est donc plus nécessaire de programmer les techniques classiques de régression, d'interpolation, d'intégration, de calcul matriciel, polynômial, ...
- son langage de développement simple et performant. Assez proche du pascal, *Matlab* permet l'utilisation des structures conditionnelles et répétitives classiques (`if then else`, `for`, `while`, ...), et facilite la mise au point des programmes par un "débugueur" efficace.
- ses fonctions graphiques puissantes et simples d'utilisation permettent différents modes de visualisation des courbes et surfaces.
- la disponibilité de *Matlab* à un prix raisonnable sur de nombreux environnements (PC, Mac, stations Unix, machines DEC, IBM, ...) et la portabilité totale des programmes *Matlab*: un programme écrit avec *Matlab* sur PC peut être exécuté sur un mac, une station unix, etc ...). Précisons également que la plupart des exemples présentés aux paragraphes suivants ont été réalisés avec la version étudiante de *Matlab* [7], vendue approximativement 320 francs en librairie (des versions PC et macintosh existent).
- ses capacités à la fois numériques et formelles.

*Une version préliminaire de ce document a été préparée à l'occasion de l'université d'été sur "les outils informatiques de l'investigation scientifique en sciences physiques", organisée par l'INRP et l'UDP à Nantes du 28 août au 1^{er} septembre 1995.

- son groupe de news très actif sur internet (`comp.soft-sys.matlab`). Chaque jour, plusieurs dizaines d'utilisateurs débutants ou confirmés partagent leurs difficultés et leurs connaissances sur ce forum électronique libre. De plus, des programmes matlab du domaine public peuvent également être récupérés (et déposés!) sur le serveur `ftp.mathworks.com`

L'objet de ce texte est de présenter quelques exemples d'utilisation de Matlab pour des problèmes présentés dans [5, 6]. Disposant ainsi d'un panorama plus complet des logiciels existants, le lecteur sera alors plus à même de juger, et de ce faire une opinion sur le logiciel le plus approprié pour ses besoins, ou le plus conforme à son mode de raisonnement.

2 Quelques exemples

L'objectif des exemples d'utilisation présentés ci-dessous n'est pas de permettre un apprentissage exhaustif des possibilités de Matlab. En effet, les différents problèmes physiques ci-dessous n'illustrent que certains des aspects de ce logiciel, en particulier ses capacités de résolution de systèmes différentiels et ses fonctions graphiques. Pour une présentation plus complète, on pourra se référer à différents ouvrages [1, 3, 4], ou au "tutoriel" fourni avec le logiciel. Par contre, la mise en parallèle des solutions proposées ci-dessous avec celles présentées dans [5, 6] offre des éléments de comparaison entre les différents environnements de calcul scientifique existants.

2.1 Potentiel Electrostatique de deux charges ponctuelles

L'objectif de cet exercice est de tracer la valeur du potentiel statique créée par deux charges $+q$ et $-q$ dans un plan passant par ces deux charges. En prenant comme centre du repère le point milieu des deux charges, et comme axe x la droite reliant les deux charges, l'expression de ce potentiel s'écrit:

$$\begin{aligned} U(x, y) &= \frac{q}{4\pi\epsilon_0\sqrt{(x-a)^2 + y^2}} - \frac{q}{4\pi\epsilon_0\sqrt{(x+a)^2 + y^2}} \\ &= \frac{q}{4\pi\epsilon_0} \left(\frac{1}{\sqrt{(x-a)^2 + y^2}} - \frac{1}{\sqrt{(x+a)^2 + y^2}} \right) \end{aligned}$$

Le programme ci-dessous trace des coupes du potentiel parallèles à l'axe x , puis la surface $U(x, y)$ et ses courbes de niveaux. Les résultats sont présentés aux figures 1, 2, et 3. Rappelons qu'une aide sur les différentes fonctions utilisées dans ce programme peut être obtenue en tapant son nom précédé de `help` dans la fenêtre de commande (par exemple `help plot`, `help contour`, etc...).

```
% exemple 1: potentiel electrostatique

% genere Nx valeurs entre xmin et xmax
%      Ny valeurs entre ymin et ymax
Nx=100; x=linspace(-10,10,Nx);
Ny= 32; y=linspace(-5,5,Ny);

% calcul du potentiel U
[x2,y2]=meshdom(x,y);
U=1.0 ./ sqrt((x2-1).^2+y2.^2) - 1.0 ./ sqrt((x2+1).^2+y2.^2);

% trace
hold off; clf;
plot(x,U(13,:),x,U(15,:),x,U(16,:));
```

```
axis([-10 10 -5 5]); grid;
xlabel('axes paralleles a x'); ylabel('potentiel electrostatique');
title('coupes transversales du potentiel electrostatique');

pause; clf;
contour(x,y,U,[-8 -4 -2 -0.5 -0.2 -0.1 0.0 0.1 0.2 0.5 2 4 8]);
grid; xlabel('axe x'); ylabel('axe y');
title('courbes de niveau du potentiel electrostatique'); pause;

mesh(x,y,U);
xlabel('axe x'); ylabel('axe y'); zlabel('potentiel electrostatique');
```

Le point de vue sous lequel la surface est examinée peut être modifié avec la fonction `view`. De plus, ce point de vue peut également être modifié avec une très grande souplesse grâce à la “boîte à outils” `matdraw` disponible par internet sous <ftp://ftp.mathworks.com/pub/contrib/graphics/matdraw.zip>.

2.2 oscillateur amorti

L'objectif de ce second exercice est d'étudier le régime libre d'une équation différentielle linéaire du second ordre, notée

$$\frac{d^2 x}{dt^2}(t) + 2\lambda_0 \frac{dx}{dt}(t) + \omega_0^2 x(t) = 0$$

avec comme conditions initiales $x(0) = 1$ et $\dot{x}(0) = 1$. Pour permettre la résolution numérique de ce problème, il faut tout d'abord transformer cette équation différentielle du second ordre en un système différentiel de deux équations du premier ordre:

$$\begin{cases} x_1(t) = x(t) \\ x_2(t) = \dot{x}(t) = \frac{dx}{dt}(t) \end{cases} \quad \text{soit} \quad \begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -2\lambda_0 \frac{dx}{dt}(t) - \omega_0^2 x(t) \\ \quad = -2\lambda_0 x_2(t) - \omega_0^2 x_1(t) \end{cases}$$

Cette transformation permet ensuite d'utiliser une des fonctions qui réalisent l'intégration numérique des équations du type $\dot{X} = f(X)$. On a arbitrairement choisi la fonction `ode45`, qui intègre (par une méthode de Runge-Kutta à pas variable d'ordre 4 et 5) une fonction dont le nom est précisé en paramètre. Pour l'utiliser, il suffit donc de définir à l'aide de la commande `new` du menu `file` une fonction qui calcule les dérivées de $x_1(t)$ et $x_2(t)$, et de la sauvegarder ensuite dans un fichier `ecolder1.m`:

```
function derX=ecolder1(t,X);
% fonction de derivation pour l'exemple 2

global lambda0 omega0;

derX(1)=X(2);
derX(2)=-2.0*lambda0*X(2) - omega0^2 * X(1);
```

Ce fichier peut ensuite être appelé par le programme ci-dessous, qui réalise la résolution numérique de l'équation différentielle avec ses conditions initiales, et le tracé de la trajectoire dans le plan de phase (ou espace des états) (x_1, x_2) . Les résultats sont présentés aux figures 4 et 5.

% exemple 2: oscillateur amorti

```
omega0=2*pi*1.0; lambda0=1.0;
global lambda0 omega0;
```

```

ti=0.0; tf= 8.0; X0=[1.0 1.0];
[t,X] = ode45('ecolder1', ti, tf, X0);

% trace
plot(t,X(:,1)); grid;
xlabel('temps (s)'); ylabel('x'); pause;

plot(X(:,1),X(:,2)); grid;
xlabel('position'); ylabel('vitesse');

```

Il faut noter que la “boite à outils” de calcul formel, dont il existe une version simplifiée dans la version étudiante de matlab, permet de résoudre analytiquement cette équation, grâce à la fonction `dsolve`. En exécutant par exemple les instructions

```
x = simplify(dsolve('D2x + 2 * Dx + (2 * pi)^2 * x = 0','x(0) = 1','Dx(0) = 1')); pretty(x)
```

l'expression analytique de la solution apparaît alors à l'écran:

$$\frac{(\cos((-1 + 4\pi^2)^{1/2} t) (-1 + 4\pi^2)^{1/2} + 2 \sin((-1 + 4\pi^2)^{1/2} t)) \exp(-t)}{(-1 + 4\pi^2)^{1/2}}$$

La commande `latex(x)` permet également de traduire cette expression dans le langage de description d'expressions mathématiques de $\text{\LaTeX}$, avec lequel ce document a été réalisé. Elle permet donc d'écrire directement la solution sous la forme:

$$x(t) = \frac{(\cos(\sqrt{-1 + 4\pi^2}t)\sqrt{-1 + 4\pi^2} + 2 \sin(\sqrt{-1 + 4\pi^2}t)) e^{-t}}{\sqrt{-1 + 4\pi^2}}$$

2.3 Oscillateur de Van der Pol

Le problème précédent aurait pu être résolu formellement. Ceci n'est généralement plus possible lorsque l'équation à résoudre est non-linéaire, ce qui est le cas par exemple pour une équation de Van der Pol:

$$\frac{d^2x}{dt^2}(t) - 2\lambda_0 \left(1 - \frac{x(t)^2}{x_0^2}\right) \frac{dx}{dt}(t) + \omega_0^2 x(t) = 0$$

Pour obtenir une solution numérique de cette équation, quelques modifications de la fonction de calcul des dérivées et du programme précédent suffisent. Les résultats obtenus (voir figure 6) incitent à conclure que cette équation possède une solution périodique.

Programme de calcul des dérivées (`ecolder2.m`):

```

function derX=ecolder2(t,X);
% fonction de derivation pour l'exemple 3

global lambda0 omega0 x0;

```

```

derX(1)=X(2);
derX(2)=2.0*lambda0*(1.0-X(1).^2/x0^2)*X(2) - omega0^2 * X(1);

```

Résolution numérique de l'équation différentielle:

```
% exemple 3: oscillateur de Van der Pol
omega0=2*pi*1.0; lambda0=1.0; x0=1.0;
global lambda0 omega0 x0;

trace=0.0;

ti=0.0; tf= 8.0; X0=[1.5 0.0];
[t1,X1] = ode45('ecolder2', ti, tf, X0, 1.0e-5, trace);

plot(t1,X1(:,1)); grid; pause;
plot(X1(:,1),X1(:,2)); grid; pause;

X0=[3.0 0.0];
[t2,X2] = ode45('ecolder2', ti, tf, X0, 1.0e-5, trace);

plot(t2,X2(:,1)); grid; pause;
axis([-3 3.5 -16 15]);
plot(X1(:,1),X1(:,2),X2(:,1),X2(:,2)); grid;
xlabel('position'); ylabel('vitesse');title('plan de phase');
```

2.4 Trajectoire d'un projectile

Les mêmes principes peuvent être utilisés pour résoudre un problème classique de balistique [2]. On considère donc une masse ponctuelle de masse m lancée dans le champ de pesanteur terrestre avec une vitesse initiale $\vec{v}_0$ qui fait un angle α avec le plan horizontal. Si on tient compte d'un frottement fluide proportionnel à la vitesse du projectile, les équations du mouvement de cet projectile s'écrivent:

$$\begin{cases} m \ddot{x}(t) = -f \dot{x}(t) \\ m \ddot{z}(t) = -mg - f \dot{z}(t) \end{cases} \quad \text{soit} \quad \begin{cases} \ddot{x}(t) = -\frac{f}{m} \dot{x}(t) \\ \ddot{z}(t) = -g - \frac{f}{m} \dot{z}(t) \end{cases}$$

En l'absence de frottement, la portée du projectile avec $v_0 = 810 \text{ m s}^{-1}$ et $\alpha = 45^\circ$ serait de plus de 60 kilomètres. Quelques essais répétés ont permis de montrer que pour obtenir en simulation une portée de 23.2 km identique à celle obtenue dans un essai réel, il faut prendre un rapport f/m égal à 0.02092 s^{-1} . Les trajectoires obtenues avec $v_0 = 810 \text{ m s}^{-1}$ et $\alpha = 45$ sont présentées figure 7.

Programme de calcul des dérivées (ecolder3.m):

```
function derX=ecolder3(t,X);
% fonction de derivation pour l'exemple 4

global f_sur_m g;

derX(1)=X(2);
derX(2)=-f_sur_m * X(2);
derX(3)=X(4);
derX(4)=-g-f_sur_m * X(4);
```

Résolution numérique des équations différentielles:

```
% exemple 4: trajectoire d'un obus de 155.

f_sur_m=0.0; g=9.81;
global f_sur_m g;
```

```

v0=input('vitesse initiale (m/s): ');
alpha=input('angle de tir (degres): '); alpha=alpha*pi/180.0;
ti=0.0; tf= 130.0;
X0=[0.0 v0*cos(alpha) 0.0 v0*sin(alpha)];
[t,X] = ode45('ecolder3', ti, tf, X0, 1e-8);

f_sur_m=0.02092;
[t2,X2] = ode45('ecolder3', ti, tf, X0, 1e-8);

% trace
x=linspace(0,7e4,200);
plot(x/1000, polyval(polyfit(X(:,1),X(:,3),3),x)/1000,...
      X2(:,1)/1000,X2(:,3)/1000,...
      [0 20],[0 20*tan(alpha)], '--');
axis([0 70 0 20]); grid;
title('trajectoire d'un obus de 155');
xlabel('x (km)'); ylabel('z (km)'); grid;

```

2.5 Illustration de principes d'optique physique

L'objectif de cet exercice est d'utiliser les fonctions graphiques de matlab pour faire apparaître les figures de diffraction obtenues d'abord par deux fentes d'Young, puis par une fente rectangulaire, de cotés a et b . Celles-ci sont données par des fonctions d'intensité égales à [5]

$$I(x, y) = I_0 \frac{\sin^2\left(\frac{\pi x}{k}\right)}{\left(\frac{\pi x}{k}\right)^2} \cos^2(\pi x)$$

$$I(\alpha, \beta) = I(0, 0) \frac{\sin^2\left(\frac{\pi \alpha a}{\lambda}\right)}{\left(\frac{\pi \alpha a}{\lambda}\right)^2} \frac{\sin^2\left(\frac{\pi \beta b}{\lambda}\right)}{\left(\frac{\pi \beta b}{\lambda}\right)^2}$$

Les figures 8 et 9 correspondent aux résultats obtenus avec le programme ci-dessous. Son utilisation avec la version étudiante de matlab est possible en réduisant la valeur des variables N_x et N_y à 256 et 32, mais fournit alors des images dont la résolution est plus faible.

% exemple 5 : exemples en optique physique

```

k=6;
Nx=400; x=linspace(-20,20,Nx);

% Une fente d'Young
I1=sinc(x/k).^2;
clf; plot(x,I1); grid; pause;

% Deux fentes d'Young
I2=I1.*cos(pi*x).^2;
subplot(211); title('intensite dans le plan focal');
plot(x,I2,x,I1,'.'); axis([-20 20 0 1]); grid;
Ny=5; y=linspace(-1,1,Ny);
[x2,y2]=meshdom(x,y);
I2=sinc(x2/k).^2 .* cos(pi*x2).^2;
subplot(212);
colormap(gray(10));
pcolor(x,y,I2.^(0.5)); shading interp; pause;

% Diffraction par une fente rectangulaire

```

```

subplot(111);
Nx=400; x=linspace(-2.5,2.5,Nx);
Ny=200; y=linspace(-2.5,2.5,Ny);
[x2,y2]=meshdom(x,y);
I=sinc(x2).^2 .* sinc(y2).^2 ;
contour(x,y,I,[0.01 0.05 0.2 0.8 0.9]); grid; pause;
colormap(gray(10)); axis('square');
pcolor(x,y,I.^(0.5)); shading interp; pause;

I=sinc(x2).^2 .* sinc(2.0*y2).^2 ;
contour(I,[0.01 0.05 0.2 0.8 0.9],x,y); grid; pause;

I=sinc(x2).^2 .* sinc(5.0*y2).^2 ;
contour(x,y,I,[0.01 0.05 0.2 0.8 0.9]); grid; pause;

```

On pourra remarquer que comme dans tous les exemples précédents, aucune structure de répétition n'a été utilisée, bien que les fonctions d'intensité aient été calculées sur plusieurs centaines de points. Ceci est dû aux capacités de vectorisation de Matlab, qui permettent d'appliquer une fonction à tous les éléments d'une matrice.

3 conclusion

Bien qu'ils aient été utilisés auparavant pour illustrer les possibilités de deux environnements de calcul *formel*, les problèmes évoqués ici sont essentiellement *numériques*. Il ne faudrait cependant pas en déduire l'inexistence de problèmes de calcul formel en sciences physiques. Mais il paraît cependant raisonnable de penser que les problèmes numériques prédominent.

Enfin, ces différents exemples montrent qu'il est relativement aisé avec ce type de logiciel d'illustrer des principes physiques par des expérimentations numériques et des représentations graphiques. Il semble raisonnable cependant de considérer que celles-ci peuvent utilement compléter, mais pas se substituer à d'autres formes pédagogiques qui développent les capacités d'abstraction essentielles des étudiants.

Bibliographie

- [1] A. Biran, M. Breiner, "Matlab for engineers", Addison-Wesley, 1995.
- [2] P. Brammand, P. Faye, G. Thomassier, "Physique, terminales C et E", Hachette, ISBN 2-01-009055-1, 1983.
- [3] D.M. Etter, "Engineering problem solving with Matlab", Prentice Hall, 1993.
- [4] G. Lindfield, J. Penny, "Numerical methods using Matlab", Ellis Horwood, 1995.
- [5] B. Lepretre, R. Adad, "Étude de quelques thèmes d'optique physique", Bulletin de l'Union des Physiciens, Vol 89, pp 25-44, Juin 1995.
- [6] A. Leroux, "Les logiciels de calcul formel et leur usage en sciences physiques", Bulletin de l'Union des Physiciens, Vol 89, pp 5-24, Juin 1995.
- [7] The Math Works inc, "The Student edition of Matlab", Prentice Hall, ISBN 0-13-184979-4, 1995.

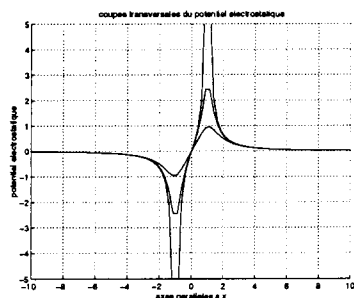


Figure 1: potentiel électrostatique dans un plan comprenant deux charges opposées: coupes transversales.

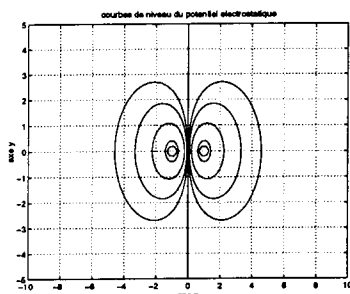


Figure 2: potentiel électrostatique dans un plan comprenant deux charges opposées: courbes de niveaux.

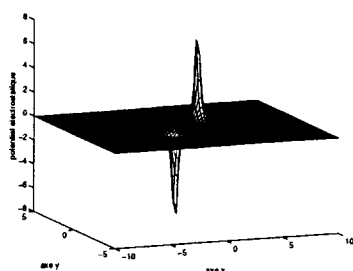


Figure 3: potentiel électrostatique dans un plan comprenant deux charges opposées: perspective en 3 dimensions.

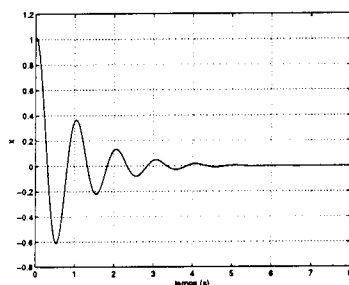


Figure 4: oscillateur amorti: représentation de la solution en fonction du temps.

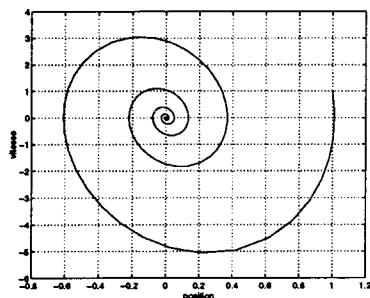


Figure 5: oscillateur amorti: représentation de la solution dans l'espace des phases.

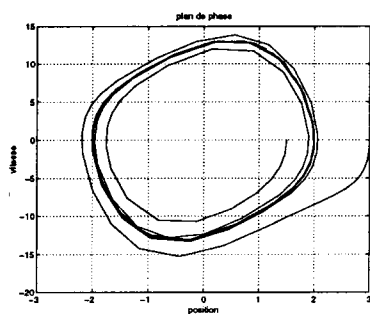


Figure 6: oscillateur de Van der Pol: représentation de deux solutions dans l'espace des phases.

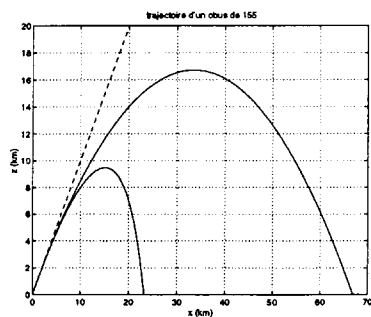


Figure 7: trajectoire d'un projectile dans le champ de pesanteur avec ou sans frottement fluide.

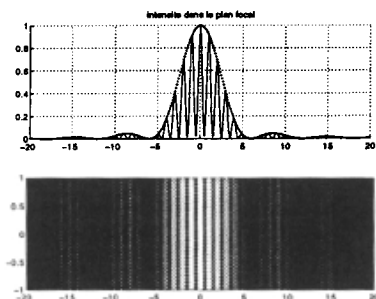


Figure 8: figure de diffraction obtenue avec deux fentes d'Young.

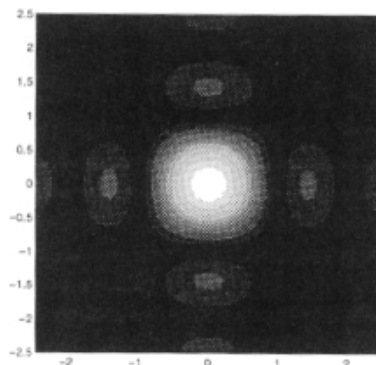


Figure 9: figure de diffraction obtenue avec une fente rectangulaire.