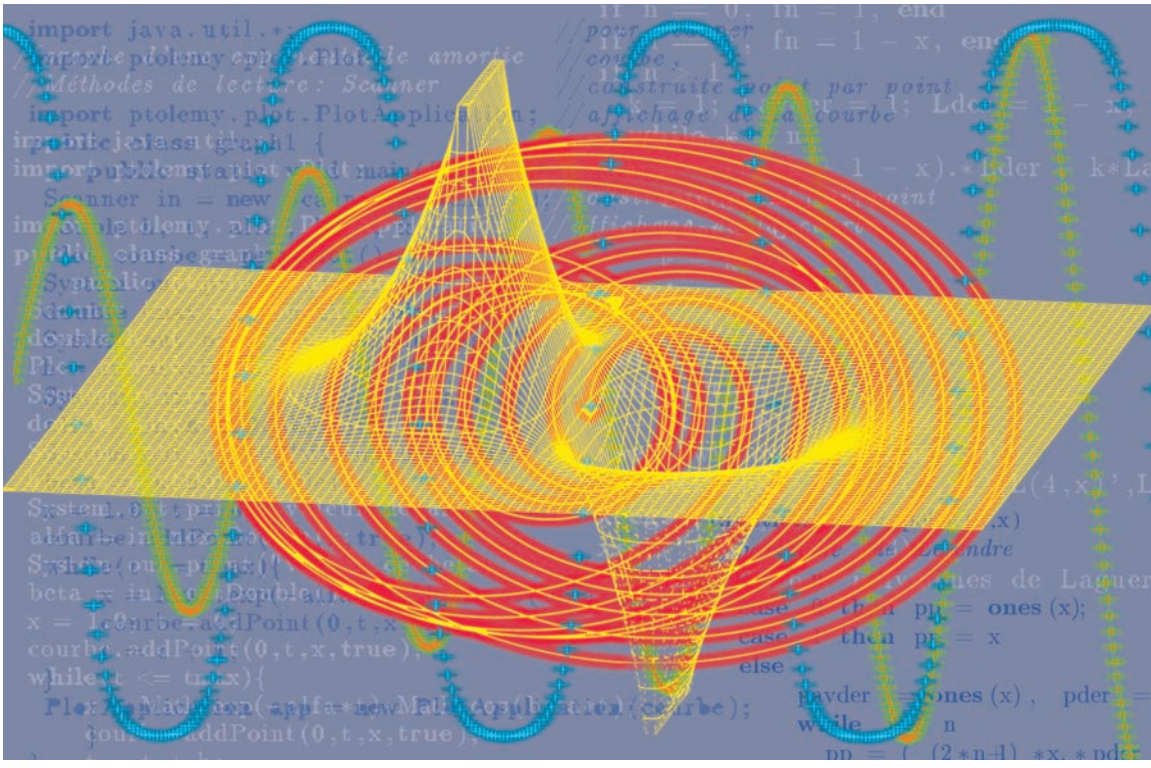
COLLECTION GRENOBLE SCIENCES
DIRIGÉE PAR JEAN BORNAREL

MÉTHODES NUMÉRIQUES APPLIQUÉES

POUR LE SCIENTIFIQUE ET L'INGÉNIEUR

Jean-Philippe GRIVET



EDP SCIENCES

This page intentionally left blank

MÉTHODES NUMÉRIQUES APPLIQUÉES
POUR LE SCIENTIFIQUE ET L'INGÉNIEUR

Grenoble Sciences

Grenoble Sciences poursuit un triple objectif :

- réaliser des ouvrages correspondant à un projet clairement défini, sans contrainte de mode ou de programme,
- garantir les qualités scientifique et pédagogique des ouvrages retenus,
- proposer des ouvrages à un prix accessible au public le plus large possible.

Chaque projet est sélectionné au niveau de Grenoble Sciences avec le concours de referees anonymes. Puis les auteurs travaillent pendant une année (en moyenne) avec les membres d'un comité de lecture interactif, dont les noms apparaissent au début de l'ouvrage. Celui-ci est ensuite publié chez l'éditeur le plus adapté.

Contact : Tél. : (33)4 76 51 46 95 - e-mail : Grenoble.Sciences@ujf-grenoble.fr
Information : ***http://grenoble-sciences.ujf-grenoble.fr***

Deux collections existent chez EDP Sciences :

- la ***Collection Grenoble Sciences***, connue pour son originalité de projets et sa qualité
- ***Grenoble Sciences - Rencontres Scientifiques***, collection présentant des thèmes de recherche d'actualité, traités par des scientifiques de premier plan issus de disciplines différentes.

Grenoble Sciences reçoit le soutien du **Ministère de l'Enseignement supérieur et de la Recherche** et de la **Région Rhône-Alpes**.
Grenoble Sciences est rattaché à l'**Université Joseph Fourier de Grenoble**.

Réalisation et mise en pages : **Centre technique Grenoble Sciences**
Illustration de couverture : **Alice GIRAUD**, d'après les éléments fournis par l'auteur

ISBN 978-2-7598-0386-6

MÉTHODES NUMÉRIQUES APPLIQUÉES
POUR LE SCIENTIFIQUE ET L'INGÉNIEUR

Jean-Philippe GRIVET

17, avenue du Hoggar
Parc d'Activité de Courtabœuf - BP 112
91944 Les Ulis Cedex A - France

Ouvrages Grenoble Sciences édités par EDP Sciences

Collection Grenoble Sciences

Chimie. Le minimum à savoir *(J. Le Coarer)* • Electrochimie des solides *(C. Déportes et al.)* • Thermodynamique chimique *(M. Oturan & M. Robert)* • CD de Thermodynamique chimique *(J.P. Damon & M. Vincens)* • Chimie organométallique *(D. Astruc)* • De l'atome à la réaction chimique *(sous la direction de R. Barlet)* • Spectroscopies infrarouge et Raman *(R. Poilblanc & F. Crasnier)* • Chemogénomique. Des petites molécules pour explorer le vivant *(sous la direction de E. Maréchal, S. Roy & L. Lafanechère)*

Introduction à la mécanique statistique *(E. Belorizky & W. Gorecki)* • Mécanique statistique. Exercices et problèmes corrigés *(E. Belorizky & W. Gorecki)* • La cavitation. Mécanismes physiques et aspects industriels *(J.P. Franc et al.)* • La turbulence *(M. Lesieur)* • Magnétisme : I - Fondements, II - Matériaux et applications *(sous la direction d'E. du Trémolet de Lacheisserie)* • Du Soleil à la Terre. Aéronomie et météorologie de l'espace *(J. Lilensten & P.L. Blelly)* • Sous les feux du Soleil. Vers une météorologie de l'espace *(J. Lilensten & J. Bornarel)* • Mécanique. De la formulation lagrangienne au chaos hamiltonien *(C. Gignoux & B. Silvestre-Brac)* • Problèmes corrigés de mécanique et résumés de cours. De Lagrange à Hamilton *(C. Gignoux & B. Silvestre-Brac)* • La mécanique quantique. Problèmes résolus, T. 1 et 2 *(V.M. Galitsky, B.M. Karnakov & V.I. Kogan)* • Description de la symétrie. Des groupes de symétrie aux structures fractales *(J. Sivardière)* • Symétrie et propriétés physiques. Du principe de Curie aux brisures de symétrie *(J. Sivardière)* • Physique des plasmas collisionnels. Application aux décharges haute fréquence *(M. Moisan & J. Pelletier)* • Energie et environnement. Les risques et les enjeux d'une crise annoncée *(B. Durand)* • Hydrothermalisme. Spéciation métallique hydrique et systèmes hydrothermaux *(M. Chenevoy & M. Piboule)* • Les roches, mémoire du temps *(G. Mascle)* • Physique des diélectriques *(J.C. Peuzin & D. Gignoux)*

Exercices corrigés d'analyse, T. 1 et 2 *(D. Alibert)* • Introduction aux variétés différentielles *(J. Lafontaine)* • Mathématiques pour les sciences de la vie, de la nature et de la santé *(F. & J.P. Bertrandias)* • Approximation hilbertienne. Splines, ondelettes, fractales *(M. Attéia & J. Gaches)* • Mathématiques pour l'étudiant scientifique, T. 1 et 2 *(Ph.J. Haug)* • Analyse statistique des données expérimentales *(K. Protassov)* • Nombres et algèbre *(J.Y. Mérindol)* • Analyse numérique et équations différentielles *(J.P. Demailly)* • Outils mathématiques à l'usage des scientifiques et ingénieurs *(E. Belorizky)*

Bactéries et environnement. Adaptations physiologiques *(J. Pelmont)* • Enzymes. Catalyseurs du monde vivant *(J. Pelmont)* • Endocrinologie et communications cellulaires *(S. Idelman & J. Verdetti)* • Eléments de biologie à l'usage d'autres disciplines *(P. Tracqui & J. Demongeot)* • Bioénergétique *(B. Guérin)* • Cinétique enzymatique *(A. Cornish-Bowden, M. Jamin & V. Saks)* • Biodégradations et métabolismes. Les bactéries pour les technologies de l'environnement *(J. Pelmont)* • Enzymologie moléculaire et cellulaire, T. 1 et 2 *(J. Yon-Kahn & G. Hervé)* • Glossaire de biochimie environnementale *(J. Pelmont)*

L'Asie, source de sciences et de techniques *(M. Soutif)* • La biologie, des origines à nos jours *(P. Vignais)* • Naissance de la physique. De la Sicile à la Chine *(M. Soutif)* • Science expérimentale et connaissance du vivant. La méthode et les concepts *(P. Vignais, avec la collaboration de P. Vignais)* • Histoire de la science des protéines *(J. Yon-Kahn)*

La plongée sous-marine à l'air. L'adaptation de l'organisme et ses limites *(Ph. Foster)* • Le régime oméga 3. Le programme alimentaire pour sauver notre santé *(A. Simopoulos, J. Robinson, M. de Lorgeril & P. Salen)* • Gestes et mouvements justes. Guide de l'ergomotricité pour tous *(M. Gendrier)*

Listening Comprehension for Scientific English *(J. Upjohn)* • Speaking Skills in Scientific English *(J. Upjohn, M.H. Fries & D. Amadis)* • Minimum Competence in Scientific English *(S. Blattes, V. Jans & J. Upjohn)* • Minimum Competence in Medical English *(J. Upjohn, J. Hay, P.E. Colle, J. Hibbert & A. Depierre)*

Grenoble Sciences - Rencontres Scientifiques

Radiopharmaceutiques. Chimie des radiotraceurs et applications biologiques *(sous la direction de M. Comet & M. Vidal)* • Turbulence et déterminisme *(sous la direction de M. Lesieur)* • Méthodes et techniques de la chimie organique *(sous la direction de D. Astruc)* • L'énergie de demain. Techniques, environnement, économie *(sous la direction de J.L. Bobin, E. Huffer & H. Nifenecker)* • Physique et biologie. Une interdisciplinarité complexe *(sous la direction de B. Jacrot)*

Avant-propos

Qu'est-ce que l'analyse numérique ? C'est un ensemble d'outils qui permet d'obtenir une solution numérique approchée d'un problème mathématique, lui-même modèle d'une question technique ou scientifique.

Pourquoi étudier (et enseigner) l'analyse numérique conçue de cette manière ? N'est-il pas suffisant d'appuyer sur la touche « solve » d'une calculette pour résoudre une équation algébrique ? Si l'on veut vraiment utiliser un logiciel, pourquoi faire plus que d'appeler, à l'intérieur d'un logiciel de haut niveau, la fonction « solve » ? En réalité, il est toujours profitable de connaître le principe de fonctionnement des outils que l'on utilise afin de les employer au mieux et pour être conscient de leurs limites. De plus, il peut arriver qu'un programme, bien qu'immédiatement disponible, ne soit pas parfaitement adapté à l'usage prévu ; seul l'utilisateur bien informé pourra le modifier en connaissance de cause et étendre son domaine de validité. Enfin, la curiosité est une qualité légitime chez l'ingénieur ou le scientifique et ce livre aide à soulever les couvercles des « boîtes noires » que sont les algorithmes numériques.

C'est dans cette optique qu'a été conçu l'ouvrage que vous avez entre les mains : le principe de chaque algorithme important est présenté simplement, puis son fonctionnement est illustré par des exemples et les limites sont précisées.

Le livre est fait pour des scientifiques de niveau L2, L3 ou M1 en physique et physique appliquée. L'ouvrage est donc destiné aux étudiants et élèves ingénieurs, mais aussi à tous les utilisateurs de l'outil numérique, en particulier ceux qui ont peu de goût ou de temps pour les démonstrations rigoureuses et qui souhaitent aborder rapidement les applications concrètes.

Le texte est orienté vers la « physique numérique ». Il y a quinze ans, ce terme (ou plutôt ses équivalents anglais, « numerical physics » ou « computational physics ») suscitait une centaine de réponses sur un moteur de recherche. Il en évoque plusieurs millions aujourd'hui. Bien que cet ouvrage ne soit pas, au sens strict, un livre de physique numérique, il s'en approche, par l'intermédiaire de certains exemples, exercices et projets.

Par rapport aux programmes habituels de mathématiques, sont inclus des chapitres qui ne font pas traditionnellement partie de l'analyse numérique comme les polynômes orthogonaux et un rappel de calcul des probabilités. Certains sujets qui, à l'expérience, semblent rébarbatifs, ont été omis, traités sommairement ou introduits assez tard dans le développement. D'autres, au contraire, qui paraissent plus motivants, ont été placés au début.

On trouve d'excellents livres d'analyse numérique en français. Ils s'adressent en général à un public de mathématiciens ou de futurs mathématiciens et sont aussi d'un niveau assez élevé. Nous renvoyons systématiquement à ces ouvrages (Crouzeix et Mignot, Schatzmann, Demailly, Allaire et Kaber, etc.) pour la plupart des démonstrations. Les aspects élémentaires, tels qu'on peut les assimiler pendant les deux premières années post-baccalauréat, ont été privilégiés.

Chaque fois que l'on traite de méthodes numériques, se pose la question du langage de programmation à utiliser. Si l'ouvrage contient quelques exemples de code rédigés en C/C++, en Java, en Python et en Maple, l'essentiel des exemples présentés est écrit en Scilab. Ce logiciel gratuit, puissant et facile à installer, intègre des fonctions graphiques de qualité raisonnable. Il permet la programmation à plusieurs niveaux : niveau global (fonctions telles que « fsolve » pour résoudre une équation non-linéaire ou « ode » pour intégrer une équation différentielle avec conditions initiales) jusqu'au niveau des opérations élémentaires. Cette souplesse est pédagogiquement utile et permet une vérification commode des programmes. Un autre avantage de Scilab est que la syntaxe en est fort simple et facilement assimilable par toute personne formée, même sommairement, à Fortran, C ou Java. Sa ressemblance avec Matlab, un logiciel très répandu, constitue un autre facteur positif.

Chaque chapitre est accompagné d'exercices qui ont été expérimentés par de nombreuses promotions d'étudiants et qui peuvent illustrer utilement le sujet traité. Sont également proposés des énoncés de projets en textes « ouverts » : les données peuvent être incomplètes, la méthode à suivre est parfois décrite assez sommairement. La réalisation du projet demandera donc un investissement personnel certain mais, en contrepartie, permettra au lecteur de se familiariser avec des applications de l'analyse numérique plus proches du monde réel. La liste des questions proposées pour chaque projet n'est pas limitative et peut être complétée selon les intérêts de chacun.

Les chapitres 1 à 3 constituent une sorte d'introduction ou de pré-requis avant d'appliquer une quelconque méthode numérique. Visualiser une fonction est la meilleure façon de découvrir ses propriétés. La programmation détaillée du calcul d'une fonction, même élémentaire, est un exercice profitable, tant du point de vue de l'algorithmique que du point de vue de l'analyse numérique. Enfin, la construction de variables sans dimension est une étape obligée de toute simulation numérique.

L'analyse numérique proprement dite apparaît au chapitre 4 avec l'interpolation. Si la pratique de l'interpolation a beaucoup diminué depuis l'apparition des ordinateurs, son rôle théorique, comme fondement des méthodes d'intégration et de résolution des équations différentielles, est resté. Les méthodes les plus simples de résolution des équations non-linéaires sont présentées (chapitre 5), en consacrant un paragraphe spécial aux polynômes. Puis suit une brève description des familles de polynômes orthogonaux et des leurs principales propriétés (chapitre 7). Bien que ces connaissances soient appliquées au chapitre suivant, lors de la construction de l'algorithme de Gauss-Legendre, cette partie peut être omise en première lecture. Les chapitres 8 et 9 sont consacrés au calcul numérique de dérivées et d'intégrales. Les algorithmes classiques sont passés en revue, y compris l'algorithme de Cooley–Tuckey pour la transformation de Fourier rapide.

L'algèbre linéaire est à l'honneur dans les chapitres 6 et 10. Tout d'abord, la résolution de systèmes d'équations linéaires (chapitre 6), y compris les systèmes surdéterminés, tels qu'on les rencontre lors de l'application de la méthode des moindres carrés. Ce chapitre est accompagné d'une annexe rassemblant quelques définitions et théorèmes d'algèbre linéaire. Le chapitre 10 traite du calcul des valeurs propres et des vecteurs propres.

Sont ensuite abordés les problèmes différentiels : équations différentielles ordinaires avec conditions initiales (chapitre 11), avec conditions aux limites (chapitre 12), puis équations aux dérivées partielles (chapitre 13).

L'ouvrage se termine par deux chapitres consacrés à des aspects non déterministes : quelques rudiments de probabilité appliqués à la propagation des erreurs expérimentales et au lissage par moindres carrés (chapitre 14) puis une description des méthodes de Monte Carlo.

De nombreux sujets ont dû être omis, soit parce qu'ils semblaient trop difficiles ou demandaient un exposé trop long. C'est notamment le cas des méthodes de descente et de gradient conjugué pour la résolution de systèmes linéaires, des méthodes modernes de résolution numérique des équations aux dérivées partielles (collocation, méthodes spectrales, éléments finis) et de la décomposition d'une matrice en valeurs singulières. Les lecteurs pourront compléter leur information grâce aux références et aux mots-clés fournis en fin de chapitre.

REMERCIEMENTS

Pour la préparation du cours qui est l'ancêtre de ce livre, j'ai bénéficié des conseils de nombreux collègues, particulièrement de D. Mas et J.L. Rouet : je leur exprime ici ma reconnaissance. Ce livre n'existerait pas sans l'aide apportée par Grenoble Sciences. Les experts qui ont lu le manuscrit ont, non seulement corrigé un nombre incalculable de fautes de frappe et d'erreurs, mais ont aussi suggéré de nombreuses améliorations. Je rends ici hommage à leur dévouement, leur patience, leur esprit d'observation et leur rigueur mathématique. Le personnel de Grenoble Sciences a aussi droit à ma reconnaissance. Mmes Capolo et Bordage ont géré la genèse du livre avec une extrême bonne volonté, elles ont détecté d'autres erreurs et, avec patience et habileté, redessiné toutes mes figures malhabiles ; je les en remercie vivement.

This page intentionally left blank

Chapitre 1

Représentation graphique de fonctions

L'une des activités les plus fréquentes en informatique scientifique consiste à représenter l'allure d'une fonction à l'aide d'un dessin, et de très nombreux logiciels peuvent répondre à cette attente légitime. Nous présentons dans ce chapitre des exemples choisis dans trois catégories de logiciels : un sous-programme que l'on doit incorporer dans un programme principal pour produire un tracé « en ligne », une application autonome capable de représenter graphiquement des données produites par un autre programme et enfin des logiciels puissants capables de calculer et de tracer. On peut encore distinguer deux cas un peu différents en pratique : ou bien la fonction est définie par une ou des formules ou un programme (c'est une fonction « analytique »), ou bien elle est représentée par un tableau de valeurs créé à l'avance (fonction « numérique »). Dans le premier cas, il faudra programmer la formule correspondante ; dans le deuxième cas, il faudra faire lire un fichier de données par le logiciel considéré, à moins de devoir entrer toutes les valeurs au clavier. Nous expliquons la marche à suivre, pour quelques logiciels pratiques et faciles d'accès, dans les paragraphes qui suivent.

1.1. Les tableurs

Tous les tableurs comportent des outils graphiques puissants. Pour cet exemple, nous utiliserons le programme `Calc` de la collection « OpenOffice.org ». Pour représenter les variations de la fonction $y = \exp(-x/3)\cos(2x)$, nous installons dans la colonne A la suite des valeurs de x. Il suffit de donner les deux premières valeurs (0 dans A1 et 0.05 dans A2 par exemple), de sélectionner ces deux cellules puis de créer toutes les valeurs suivantes (jusqu'à une limite choisie par l'utilisateur) dans la même colonne à l'aide de la « poignée de recopie ». Le programme demande si nous voulons engendrer une progression arithmétique, ce que nous confirmons. La cellule B1 contiendra l'expression de la fonction ; comme il s'agit d'une formule, nous écrivons `=exp(-A1/3)*cos(2*A1)`. Utilisant encore la poignée de recopie, nous reproduisons cette expression dans toutes les cellules utiles de la colonne B, en indiquant dans la boite de dialogue que nous voulons dupliquer une formule. Il faut ensuite cliquer sur l'icône « insertion de diagramme », choisir avec la souris l'emplacement et la taille du dessin et répondre aux questions posées par le logiciel : où se trouvent les données, quelle est la présentation souhaitée.

S'agissant d'un fichier de données, le seul obstacle mineur est la lecture du fichier. Nous supposons que ce fichier (au format ASCII) contient des données produites par un autre programme : ce sont des nombres décimaux (écrits avec un **point décimal**) séparés par plusieurs blancs. Le nom du fichier se terminera de préférence par l'extension « .csv » (pour éviter qu'OpenOffice ne le considère comme un texte, destiné au traitement de texte, « Writer »). Nous ouvrons le fichier et nous indiquons, dans la boite de dialogue, que les valeurs sont séparées par des blancs, qu'il faut regrouper les séparateurs et que le modèle des nombres est anglo-saxon. Le tracé s'effectue comme précédemment. Si le fichier contient deux colonnes, celles-ci sont utilisées automatiquement. Sinon, il faut préciser, avec la souris, la colonne des abscisses et celle des ordonnées.

1.2. JAVA ET PTPLOT

PtPlot est une collection de méthodes (issues d'un gros projet baptisé « Ptolemy ») que l'on peut inclure dans un programme en Java pour tracer des graphes point par point. Le listing suivant permet d'afficher une sinusoïde amortie.

Listing 1.1 – Sinusoïde amortie en Java

```
import java.util.*;                                                    1
import ptolemy.plot.Plot;                                              2
import ptolemy.plot.PlotApplication;                                   3
public class graph1 {                                                  4
    public static void main(String[] args) {                          5
 Scanner in = new Scanner(System.in);                                  6
 double h, t, alfa, beta, x;                                           7
 Plot courbe = new Plot();                                             8
 System.out.print("duree:  ");                                        9
 double tmax = in.nextDouble();                                        10
 System.out.print("valeur du pas: ");                                 11
 h = in.nextDouble();                                                  12
 System.out.print("valeur de alpha: ");                               13
 alfa = in.nextDouble();                                               14
 System.out.print("valeur de beta: ");                                15
 beta = in.nextDouble();                                               16
 x = 1.0; t = 0;                                                       17
 courbe.addPoint(0,t,x,true);                                          18
 while(t <= tmax){                                                     19
    x = Math.exp(-alfa*t)*Math.cos(beta*t);// construction de          20
    courbe.addPoint(0,t,x,true);            // la courbe,              21
    t = t + h;                              // point par point         22
 }                                                                     23
 PlotApplication app = new PlotApplication(courbe);//affichage        24
    }                                                                  25
}                                                                      26
```

Les lignes 1 à 3 permettent l'insertion de méthodes d'entrée-sortie et graphiques. La lecture des données fait appel à la classe `Scanner` (disponible dans les versions de Java postérieures à 2004). Les lignes 6 à 8 déclarent la méthode `in` et l'objet `courbe` ainsi que les variables nécessaires. Les valeurs correspondantes sont lues par les instructions 9 à 16. L'objet `courbe` est initialisé lignes 17 et 18. La boucle `while` construit la courbe point par point ; celle-ci est affichée par l'instruction 24.

Nous vous rappelons les étapes d'une utilisation simplifiée de Java. On crée le programme à l'aide d'un éditeur de texte et on l'enregistre sous le nom `graph1.java` en prenant garde que le nom de la classe et le nom du fichier **coïncident exactement**. Sous Windows, dans une fenêtre « invite de commande », on compile ce programme par l'instruction `javac graph1.java`. Si tout se passe bien, on peut alors lancer l'exécution par `java graph1`. L'ordinateur doit savoir où se trouve le compilateur (il faut donc initialiser correctement la variable d'environnement `PATH`) et où se trouvent les classes PtPlot (initialiser `CLASSPATH`!).

La figure apparaît dans une nouvelle fenêtre interactive : certains paramètres du tracé, comme les échelles des axes, sont modifiables par l'utilisateur (menu « Edit »).

1.3. Python et Matplotlib

Vous pouvez considérer Java et Python comme des descendants de C++, en plus simples et plus commodes. Python (et toutes ses bibliothèques de sous-programmes spécialisés) est, à notre avis, plus facile d'emploi et plus commode que Java dans le domaine scientifique. Il souffre cependant d'un inconvénient : c'est un langage « interprété » assez lent. Il faut l'utiliser pour de petits programmes ou apprendre à l'interfacer avec des modules Fortran ou C qui exécuteront les gros calculs.

Listing 1.2 – Sinusoïde amortie en Python

```
from pylab import *                                          1
t = arange(0,2*pi,0.01)                                      2
alfa = float(raw_input("valeur de alfa: "))                  3
beta = float(raw_input("valeur de beta: "))                  4
ca = exp(-alfa*t)*cos(beta*t)                                5
plot(t,ca)                                                   6
show()                                                       7
```

Pour utiliser Python, il vous faut installer Python, les bibliothèques scientifiques Scipy et Numpy et la bibliothèque graphique Matplotlib ; l'environnement de programmation interactif IPython (et ses accessoires pour Windows) est commode. Si vous voulez lancer Python depuis un répertoire quelconque, vous devrez faire figurer son chemin d'accès dans `PATH`. L'utilisation est semblable à celle de Java : avec votre éditeur de texte favori, vous créez un programme (`graph1.py` par exemple) dont vous demandez l'interprétation et l'exécution (depuis une fenêtre « invite de commande ») par `python graph1.py`. Vous pouvez tout aussi bien taper `%run graph1.py` dans IPython.

Comme le montre l'exemple (lui-même réduit à l'essentiel), ce langage est compact et puissant. Une grande partie du travail est fait, en coulisse, par la ligne 1 qui « importe » de très nombreuses fonctions mathématiques et graphiques. La ligne 2 crée une liste de valeurs de `t`, les lignes 3 et 4 permettent d'entrer, au clavier, les valeurs de α et de β et la ligne 5 calcule, point par point, les valeurs de la fonction. Celle-ci est affichée (selon la syntaxe de Matlab, d'où le nom de la bibliothèque) par l'instruction 6. La ligne 7 empêche la disparition de votre beau graphique dès la fin de l'exécution du programme.

1.4. Gnuplot

Gnuplot est un logiciel gratuit et puissant, disponible pour tous les systèmes d'exploitation ; il a l'avantage d'être assez peu encombrant (2,5 Mo environ). Il possède un analyseur syntactique qui connaît la plupart des fonctions élémentaires. Il est très facile d'accomplir avec Gnuplot les deux tâches que nous nous sommes proposées. Pour tracer une fonction définie par une formule :

```
gnuplot>  alfa = 0.3; beta = 4;
gnuplot>  plot [0:10] exp(-alfa*x)*cos(beta*x);
```

La fonction `plot` admet comme premier paramètre l'intervalle de variation de la variable indépendante, laquelle doit s'appeler x, par convention.

Pour afficher un fonction définie comme une suite de valeurs (x, y), rangées en colonnes dans le fichier `ex142.dta`, il faut écrire :

```
gnuplot>  plot [-2.5:0] [-3.5:1.5] "C:/an_poly/ex142.dta"
```

Les obliques remplacent les contre-obliques de MS-DOS ; nous avons précisé l'intervalle de variation pour les abscisses (premières valeurs entre crochets) et pour les ordonnées (valeurs suivantes). Si le fichier comporte plusieurs colonnes, nous pourrons indiquer que les nombres de la colonne 2 devront servir d'abscisses et ceux de la colonne 5 d'ordonnées en ajoutant simplement `using 2:5` à la fin de l'instruction précédente.

Il existe un grand nombre d'options, pour modifier les caractéristiques du tracé (affichage de légendes, utilisation de symboles ou de traits continus, épaisseur et couleur du trait) et pour choisir le type de graphe (paramétrique, polaire). Vous pourrez les découvrir en lisant l'aide en ligne ou encore en sauvegardant votre travail dans un fichier sur disque (touche `SAVE`) ; en ouvrant ce fichier dans un éditeur de texte, vous verrez que Gnuplot enregistre un grand nombre d'options et de paramètres. Vous pourrez alors modifier progressivement ces valeurs et afficher le tracé enrichi par l'instruction `load`.

Spontanément, gnuplot utilise une fenêtre comme dispositif de sortie. Cette fenêtre est interactive et certains paramètres du dessin peuvent être modifiés en cliquant sur la rubrique « option » du menu déroulant. On peut aussi envoyer le tracé dans un fichier destiné à une imprimante, selon le format souhaité (PNG, GIF, HPGL et bien d'autres).

Le même programme peut représenter des surfaces en perspective ou des courbes de niveau (voir le fichier `all.dem`).

1.5. Maple

Le logiciel Maple est orienté vers le calcul algébrique ; il possède cependant des possibilités graphiques extrêmement puissantes, dont voici un tout petit aperçu. Le dialogue suivant permet de tracer une sinusoïde amortie.

```
> y := exp(-alpha*x)*cos(beta*x) ;
```

$$y := e^{(-\alpha\, x)} \cos(\beta\, x)$$

```
> y1 := subs(alpha = 0.3, beta = 4,y) ;
```

$$y1 := e^{(-0.3\, x)} \cos(4\, x)$$

```
> plot(y1,x = 0..10) ;
```

Nous aurions pu nous contenter de l'instruction unique

```
> plot( exp(-0.3*x)*cos(4*x),x = 0..10);
```

mais la version précédente nous permet de définir une quantité y dépendant de deux paramètres dont nous pouvons modifier la valeur aisément par l'instruction `subs`.

La lecture d'un tableau de valeurs externe se fait au moyen de l'instruction `readdata` qui admet deux paramètres obligatoires, le nom du fichier et le nombre de colonnes. On peut aussi préciser s'il s'agit d'entiers ou de nombres fractionnaires. La manoeuvre est simple dans le cas d'un tableau à deux colonnes, comme le montre l'exemple ci-dessous. Nous traçons le graphe correspondant à l'aide des instructions

```
>    M := readdata("C:/an_poly/ex122.dta",2);
```

$$M := [[-2.,\ -3.1],\ [-1.1,\ -0.99],\ [0.,\ 1.],\ [2.222,\ 3.1415],\ [4.,\ 5.]]$$

```
>    plot(M);
```

Les nombres entiers présents dans le fichier ont été transformés en nombres fractionnaires. Remarquez aussi que les contre-obliques du nom de fichier sont remplacées par des obliques sous Maple. Il est un peu plus compliqué d'extraire d'un fichier multi-colonnes une colonne d'abscisses et une colonne d'ordonnées. Il faut procéder comme ceci. Le fichier dont le nom complet est `D:\an_poly\graph02.dta` contient les valeurs :

```
-1.9  -1     -2.2
-1    -0.04  -1
0      0.97   0.35
0.9    1.87   2.43
2      3      4
3.21   3.96   4.43
```

On lit et on extrait les bonnes valeurs grâce aux instructions

```
> M := readdata("D:/an_001/graph02.dta",3);
```

$M :=$ [[−1.9, −1., −2.2], [−1., −0.04, −1.], [0., 0.97, 0.35], [0.9, 1.87, 2.43], [2., 3., 4.], [3.21, 3.96, 4.43]]

```
> points := [seq([M[i,1],M[i,3]],i=1..6)];
```

points := [[−1.9, −2.2], [−1., −1.], [0., 0.35], [0.9, 2.43], [2., 4.], [3.21, 4.43]]

```
> plot(points,style=POINT,symbol=BOX,symbolsize=15);
```

Maple considère M comme une liste de listes, chaque liste élémentaire représentant une ligne. Le code extrait les nombres de la première colonne (abscisses) et ceux de la troisième colonne (ordonnées), ligne par ligne, puis reporte les points sur un graphique (figure 1.1).

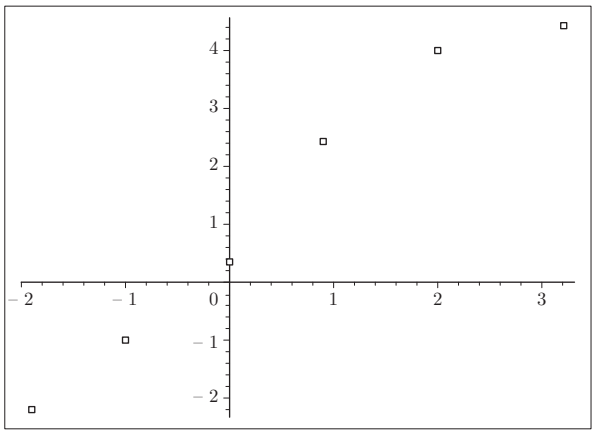


Figure 1.1 – Le tracé d'une suite de points par Maple.

1.6. SCILAB

Scilab est un logiciel gratuit, très bien adapté à l'algèbre linéaire et à la simulation des systèmes dynamiques. On peut l'utiliser de façon interactive (comme une calculette) ou préparer, à l'aide d'un éditeur de texte (par exemple celui qui est incorporé dans Scilab depuis la version 2.7), un programme que l'on fera exécuter ensuite. Nous utilisons, dans ce chapitre, essentiellement la partie graphique de Scilab.

Nous souhaitons encore tracer une sinusoïde amortie, définie par l'équation

$$x = \exp(-\alpha t)\cos(\beta t),$$

pour $\alpha = 0.3, \beta = 2$ et $0 \leq t \leq 10$. Les instructions suivantes répondent à notre souhait.

```
alfa = 0.3; beta = 2;
t = 0:0.1:10;
x = exp(-alfa*t).*cos(beta*t);
plot2d(t,x)
```

Le résultat apparaît sur la figure 1.2. Ce n'est pas ici le lieu de détailler la syntaxe de Scilab, qui est très bien expliquée dans l'aide en ligne, dans les manuels et sur divers sites (voir les références en fin de chapitre) ; nous nous contenterons de quelques indications.

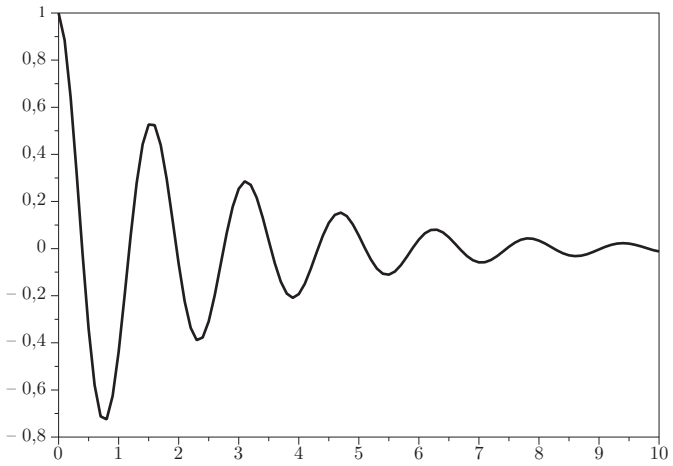


Figure 1.2 – Une sinusoïde amortie représentée par Scilab.

La première ligne définit et initialise les deux paramètres ; les points-virgules indiquent à Scilab qu'il ne doit pas afficher à l'écran les valeurs qui viennent d'être définies. La deuxième ligne initialise un vecteur ligne, t, dont les coordonnées sont $t_1 = 0, t_2 = 0.1, t_3 = 0.2, \ldots, t_{101} = 1.0$; les éléments de tableaux (vecteurs, matrices) sont toujours numérotés à partir de 1. La ligne suivante calcule le vecteur x. Vous pouvez constater qu'il se passe pas mal de choses « en douce ». Les composantes du vecteur t sont successivement utilisées comme arguments des fonctions exponentielle et cosinus. Ainsi, `exp(-alfa*t)` est un vecteur ligne de composantes $\exp(-\alpha * t_i)$. Les deux tableaux représentés par `exp(-alfa*t)` et `cos(beta*t)` sont ensuite multipliés **élément par élément** (c'est ce qu'indique la notation « **point-étoile** » du produit) pour former le résultat, un vecteur ligne de coordonnées $x_i = \exp(-\alpha t_i)\cos(\beta t_i)$. Le programme relie ensuite par des segments les points successifs de coordonnées (t_i, x_i), pour produire un tracé lisse.

La fonction `plot2d` peut recevoir des arguments supplémentaires, qui permettent de définir des axes, des graduations, de modifier l'épaisseur et la couleur des traits, d'introduire des légendes ou un titre.

En ajoutant la ligne `y = exp(-alfa*t).*sin(beta*t)` et en modifiant l'instruction de tracé, qui devient `plot2d(t,[x,y])`, nous obtenons, sur le même graphe, les

courbes représentant les deux fonctions x et y. Nous pouvons aussi considérer ces deux fonctions ensemble comme la représentation paramétrique d'une courbe ; celle-ci s'obtient facilement par l'instruction graphique `plot2d(x,y)`.

Comment procéder pour tracer la courbe correspondant à un fichier de valeurs numériques ? C'est encore plus simple.

Nous supposons que le fichier `C:/an_poly/ex112.dta` contient les données qui nous intéressent, à raison de trois par ligne, ces nombres étant séparés par des espaces ou une tabulation. Nous procédons alors comme suit

```
M = read("C:/an_poly/ex112.dta", -1,3);
plot2d(M(:,1),M(:,3),style = -3)
```

Scilab admet aussi bien les obliques que les contre-obliques dans les noms de fichiers. À la première ligne, on lit le contenu du fichier et on range les valeurs dans la matrice M. La valeur `-1` oblige Scilab à lire toutes les lignes de `C:/an_poly/ex112.dta` quel que soit leur nombre et le point-virgule est là pour l'empêcher d'afficher ces valeurs à l'écran. On représente ensuite graphiquement tous les nombres de la troisième colonne de M (ordonnées) en fonction des valeurs correspondantes de la première colonne (abscisses). Sauf indication contraire, Scilab relie les points par des segments. Pour éviter cela, nous avons indiqué un `style` de trait négatif (-3 ici) ; Scilab représente alors des points isolés à l'aide du symbole correspondant.

1.7. Grace

Les utilisateurs du système Linux ont à leur disposition de nombreux autres outils graphiques gratuits. Citons la collection de programmes « Plotutils » (qui se lancent depuis la ligne de commande), la bibliothèque « pgplot » conçue pour s'interfacer facilement avec des programmes en Fortran (ou C) et enfin le somptueux « xmgrace » interactif. Ce logiciel possède les mêmes fonctionnalités que gnuplot à l'exception des représentations de courbes ou surfaces à trois dimensions, mais il est complètement interactif. On peut entrer une formule dans une fenêtre de commande pour représenter une fonction analytique ou importer des données contenues dans un fichier. Toutes les options sont accessibles par des menus déroulants. On peut également sauvegarder un graphique sous forme de fichier texte (terminaison `.agr`). En relisant un fichier de ce type à l'aide de votre éditeur de texte favori, vous constaterez que Grace a enregistré de très nombreux paramètres du tracé, que vous pourrez modifier à loisir : ils seront disponibles lorsque vous relirez le fichier `.agr`.

1.8. Pour en savoir plus

– PtPlot :
`http://ptolemy.eecs.berkeley.edu`

– Python et cie :
`http://python.org`

http://scipy.org
http://ipython.scipy.org
http://matplotlib.sourceforge.net

– Scilab :
http://www.scilab.org
Distributions pour toutes plates-formes.
La page http://www.scilab.org/publications/ contient des références de livres et des liens vers de nombreux textes pédagogiques gratuits.

– Maple :
http://www.maplesoft.com/
http://lumimath.univ-mrs.fr/~jlm/cours/maple/maple.html
http://algo.inria.fr/ : voir la page personnelle de M. Dumas.
http://pagesperso-orange.fr/eddie.saudrais/index.html

– gnuplot :
http://www.gnuplot.info/
ftp://ftp.irisa.fr/pub/gnuplot/

– Grace :
http://plasma-gate.weizmann.ac.il/Grace/

– plotutils :
http://www.gnu.org/software/plotutils/

– pgplot :
http://www.astro.caltech.edu/~tjp/pgplot/

Il existe encore de très nombreux logiciels de visualisation ou de calcul que nous n'avons pas eu la possibilité de mentionner dans le texte. Voici les sites Internet relatifs aux plus connus d'entre eux.

– Mathematica : http://www.wolfram.com

– Mupad : http://www.mupad.de

– Matlab : http://www.mathworks.fr/

– O-Matrix : http://www.omatrix.com

– Origin : www.originlab.com/

– Octave : http://www.gnu.org/software/octave/

– R : http://www.r-project.org

– Root : http://root.cern.ch

– Sage : http://www.sagemath.org/

1.9. Exercices

Exercice 1

En utilisant le logiciel de votre choix, tracer (séparément ou sur un même graphique) les courbes représentant les deux fonctions

$$x = \exp(-\alpha t)\cos(\beta t) \quad ; \quad y = \exp(-\alpha t)\sin(\beta t)$$

avec $\alpha = 0.25, \beta = 6$, puis la courbe d'équations paramétriques $x(t), y(t)$. Reproduire enfin la figure 1.3 ci-dessous.

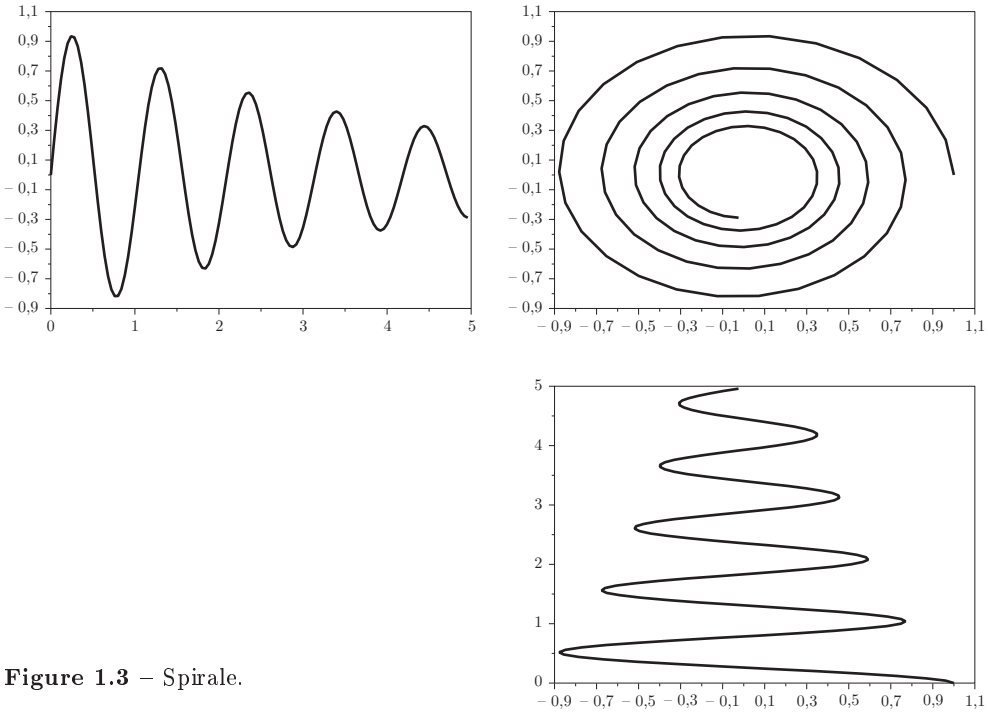


Figure 1.3 – Spirale.

Exercice 2

On a déterminé le nombre d'atomes radioactifs présents dans un échantillon au cours du temps, avec les résultats suivants.

t	0	1	2	3	4	5	10
$N(t)$	1000	370	130	50	17	8	1

Représenter graphiquement ces données, en coordonnées normales ou logarithmiques.

Exercice 3

Examiner, en fonction des valeurs des entiers L, M et N, l'aspect des courbes d'équations paramétriques

$$x = \sin(Lt)\cos(Mt); \quad y = \sin(Lt)\sin(Nt).$$