MAPLE

MPSI 2 2010-2011

Une année de MAPLE en MPSI

Guillaume CONNAN

une année de maple en mpsi

tp maple en mpsi

TABLE DES MATIÈRES

AU PROGRAMME

Voici quelques extraits du programme officiel :

L'informatique en classes préparatoires a pour principaux objectifs d'offrir dans le tronc commun :

- *Une familiarisation avec l'utilisation d'outils informatiques évolués (logiciel de calcul formel et numérique, logiciels d'acquisition et de traitement de données, logiciels de modélisation, logiciels de simulation...) en vue de permettre l'approfondissement des disciplines scientifiques et techniques ;*
- *Une introduction à l'informatique en tant que discipline, par une initiation élémentaire au traitement automatique de l'information, à l'algorithmique et à la programmation structurée (illustrée à l'aide du langage du logiciel de calcul formel retenu).*

[...]
On habituera les élèves à se servir de logiciels, qui fournissent un support au raisonnement par la confrontation rapide et commode des hypothèses et résultats, et permettent :

- *D'enrichir la compréhension des phénomènes mathématiques et des modèles physiques par la simulation de leurs comportements en fonction de divers paramètres ;*
- *De mieux cerner la notion de domaine de validité d'une hypothèse ou d'une méthode par l'étude de cas limites ;*
- *D'étudier certains problèmes par la mise en oeuvre de modèles dont la résolution numérique manuelle serait trop lourde ou trop complexe ;*
- *D'alléger la part de calcul systématique au profit de l'intuition mathématique ou du sens physique.*

[...]
La mise en œuvre de la programmation n'est pas séparée de l'utilisation du logiciel de calcul formel en tant qu'outil et s'effectue à l'occasion des séances de travaux pratiques, appliquées à la résolution de problèmes de mathématiques, de physique, de chimie, de mécanique et automatique.
[...]
L'outil informatique n'est pas une fin en soi mais un moyen efficace pour faire des mathématiques, des sciences physiques ou des sciences industrielles.

À CENTRALE

Voici extrait du rapport du jury 2010 concernant l'oral de mathématiques II en série MP :

Présentation de l'épreuve

L'épreuve de Maths II est une épreuve de mathématiques assistée par ordinateur, en l'occurrence par l'usage d'un logiciel de calcul formel. Elle comporte un seul exercice qui est préparé pendant 30 minutes avec accès libre à l'ordinateur ; puis le candidat vient présenter pendant 30 minutes ses résultats et poursuivre la résolution de l'exercice au tableau. Une fois au tableau, il est souhaitable d'indiquer succinctement les questions qui ont été élucidées pendant la préparation et ensuite de ne pas trop perdre de temps sur les questions élémentaires, pour arriver au coeur du sujet. De façon générale, le candidat doit être très attentif aux conseils et a fortiori aux indications données oralement, et surtout de ne pas s'enfermer dans une tentative de résolution si l'examinateur lui indique qu'elle risque de conduire à une impasse : on aboutit à un échec donc à une perte de temps qui empêchera la résolution d'autres questions. Il est essentiel que l'oral reste un dialogue, et une très bonne note peut être attribuée à un candidat qui, sans avoir résolu l'exercice lors de la préparation, aura montré une bonne réactivité aux indications proposées. Une des difficultés de cette épreuve est de savoir gérer l'aide que peut apporter le logiciel pour résoudre la question mathématique posée. Les exercices comportent pour la plupart au moins une question à résoudre avec l'outil informatique. Il est souvent attendu de pouvoir émettre une conjecture, qui sera démontrée dans la suite de l'exercice. Parfois, l'énoncé conseille d'utiliser le logiciel à bon escient au cours de l'exercice, sans qu'il soit imposé pour une question précise : il revient alors au candidat d'évaluer les questions où l'ordinateur lui apportera une aide précieuse. C'est par exemple le cas pour des calculs auxiliaires de développements limités ou d'intégrales élémentaires (coefficients de Fourier ...), ou des résolutions d'équations lors d'un exercice de géométrie : le logiciel permet d'alléger les calculs et d'éviter les erreurs liées au stress de l'épreuve. Il faut par contre savoir être circonspect sur certaines réponses du logiciel : que penser d'un candidat qui ne réagit pas face à l'affichage de valeurs propres « complexes » pour une matrice symétrique réelle, alors qu'une demande de valeurs approchées montre que leurs parties imaginaires sont des `10^(-6)*I` ? Il ne s'agit pas là de savoir comment fonctionne le logiciel mais d'avoir un minimum de recul ou de prudence sur l'affichage de certains résultats.

[...]

Commentaires et conseils concernant l'usage du logiciel de calcul formel

L'impression générale est ici un peu mitigée. Et il semble utile de redonner une liste de « savoir-faire » déjà publiée dans le rapport 2009 et qui figure en Annexe. Le jury a vu avec plaisir environ un quart des candidats très bien préparés, très à l'aise avec cet aspect de l'épreuve. Une majorité a montré l'habitude de côtoyer le logiciel et la connaissance des commandes usuelles. Signalons qu'il n'est d'ailleurs attendu ni dextérité ni connaissance savante des options possibles d'une fonction prédéfinie ; et il est normal et raisonnable qu'un candidat s'assure du bon emploi d'une fonction du logiciel en ayant recours à l'aide en ligne (encore faut-il en connaître le nom... : alors pourquoi se priver très souvent de l'affichage à l'écran des fonctions résidant dans une librairie, par un `with(...):` au lieu d'un `with(...);` du logiciel Maple ?). Par contre, il faut constater et regretter qu'un trop grand nombre –même si c'est une minorité– continue à espérer découvrir les fonctions de base avec un usage fébrile de l'aide en ligne pendant la préparation : cela se traduit par une lourde perte de temps, et le bilan est en général catastrophique : aucun résultat ne sort d'une succession de lignes de code dont la plupart ont été rejetées par le logiciel.... À la moindre sollicitation de l'examinateur pour apporter les premières modifications permettant l'obtention d'au moins un résultat partiel (faute de temps, il ne peut être question de corriger les fautes accumulées), la réponse « je n'ai pas pratiqué le logiciel cette année » ne peut constituer une excuse. L'usage d'un logiciel de calcul formel figure pour tous, au programme des deux années de classes préparatoires, et les candidats savent qu'ils en auront besoin lors de l'épreuve de maths II du concours Centrale. Le jury sanctionnera plus nettement cette attitude désinvolte : on ne peut pas arriver à cette épreuve si on ne sait pas utiliser efficacement le logiciel. Le peu de programmation attendue ne dépasse pas l'écriture de boucles avec d'éventuelles instructions conditionnelles ! Il ne s'agit pas d'un exercice d'algorithmique. Relevons quelques pratiques maladroites. On regrette d'abord un recours trop systématique à l'écriture de nombreuses et inutiles procédures, quelle que soit la complexité de 32 Épreuves orales Rapport du jury 2010 - Filière MP la question posée : cette démarche peut sembler tout à fait honorable, mais conduit trop souvent hélas à un échec, et donc à l'absence de résultats effectifs ; or c'est le but attendu. Et puis c'est une mauvaise compréhension de l'intérêt d'un logiciel de calcul formel : les fonctions prédéfinies sont là pour gagner du temps, et écrire une procédure pour définir la fonction « factorielle » (exemple caricatural mais vu cette année) n'a aucun intérêt ici. En fait, il est rarement indispensable d'écrire une procédure lors des questions proposées, ce qui ne signifie pas qu'elles sont parfois bienvenues ; mais l'écriture directe d'une boucle est souvent suffisante. Il est nécessaire de savoir distinguer la manipulation des « fonctions » et des « expressions », et d'estimer quand l'usage des unes ou des autres est plus favorable. Signalons que l'usage des expressions est souvent plus souple lorsqu'il doit se doubler de la création d'un opérateur mathématique qui manipule ces expressions. Certains ne savent d'ailleurs pas créer une suite, ou une fonction ; l'usage des crochets ou des parenthèses est mal compris : u[n] ou u(n) ? Et plus gênant encore est de voir écrire des tentatives du type `u(n):=` ... ou `f(x):=` ... pour fabriquer

une fonction, et des candidats qui sont surpris que le logiciel ne réponde pas à leur attente !! Le logiciel met à disposition des outils commodes pour créer des séquences de résultats. Combien de fois on a vu le recours à des copier-coller quand l'énoncé demandait une séquence d'une dizaine ou d'une vingtaine de résultats (nécessaires à l'ébauche d'une conjecture « fiable ») ? Il faut enfin savoir indiquer au logiciel qu'une variable est par exemple entière, réelle positive, etc... Et connaître quelques commandes qui simplifient ou convertissent ou transforment des expressions sous une forme souhaitée.
Mais voici quelques points encore trop mal maîtrisés :

- la construction de matrices de taille variable. Il faut savoir fabriquer une fonction « définissante » des coefficients. On a ainsi vu régulièrement des candidats contraints d'écrire une matrice 10*10 en tapant les 100 coefficients (dont beaucoup étaient nuls heureusement) !
- savoir obtenir des valeurs approchées des racines d'une équation, savoir que l'affichage d'un seul résultat numérique ne se traduit pas nécessairement par l'unicité d'une solution... ;
- pour le graphisme, il faut savoir superposer sur un même schéma divers types de graphes ;
- dans le cas particulier des équations différentielles, beaucoup ne savent pas visualiser le graphe d'une solution, lorsque le logiciel n'en donne pas une expression exacte.
- proscrire l'ouverture et l'usage simultanés des librairies Maple `linalg` ET `LinearAlgebra` ;
- connaître les inconvénients ou avantages respectifs des commandes « sum » et « add » ...

[...]

Liste de savoir-faire conseillés pour l'épreuve assistée par un logiciel de calcul formel

Calcul algébrique (entiers, polynômes, équations)

- savoir calculer le quotient, le reste dans une division euclidienne dans $\mathbf{Z}$, dans $\mathbf{Q}[X]$;
- savoir tester qu'un entier est premier, savoir travailler modulo n ;
- savoir factoriser (dans $\mathbf{Q}[X]$ et éventuellement dans une extension simple suggérée par l'énoncé), développer, ordonner un polynôme ;
- savoir obtenir tous les coefficients, ou des coefficients précis d'un polynôme ;
- savoir calculer le pgcd de deux entiers, de deux polynômes ;
- savoir obtenir un couple donnant la relation de Bézout ;
- savoir déterminer les racines d'une équation (algébrique ou non) de façon exacte, de façon approchée ; savoir déterminer une valeur approchée d'une racine localisée dans un intervalle ;
- savoir décomposer une fraction rationnelle en éléments simples dans $\mathbf{Q}(X)$ (éventuellement dans une extension simple de $\mathbf{Q}$ suggérée par l'énoncé).

Calcul matriciel

- savoir construire une matrice dont les coefficients sont donnés par une formule fonction du couple (i, j), et dont la taille peut être variable (il ne peut être question de se limiter à savoir entrer une matrice 3x3 par ses neuf coefficients) ;
- savoir calculer des produits matriciels, créer une matrice diagonale et a fortiori la matrice identité, former la transposée ;
- savoir calculer le rang, le noyau ou l'image (en obtenant une base de ces sous-espaces) ;
- savoir calculer le déterminant, éventuellement l'inverse, la comatrice (ou sa transposée) d'une matrice carrée ;
- savoir calculer le polynôme caractéristique d'une matrice carrée, ses valeurs propres, ses vecteurs propres ;
- savoir résoudre une équation d'inconnue matricielle (après l'avoir transformée en un ensemble d'équations scalaires d'inconnues les coefficients) ;
- savoir calculer le produit scalaire, le produit vectoriel de deux vecteurs de $\mathbf{R}^3$.

Fonctions d'une ou plusieurs variables réelles, calcul différentiel, calcul intégral

- savoir composer des fonctions (ou des opérateurs), calculer des dérivées d'ordre supérieur à un ;
- savoir calculer un développement limité, savoir extraire la partie régulière d'un tel développement ;
- savoir calculer une intégrale de façon exacte, de façon approchée, faire un changement de variable ou une intégration par parties ;
- comprendre pourquoi le logiciel n'affiche pas toujours une limite explicite, ou le résultat d'un calcul d'intégrale, par manque d'information sur la nature d'un paramètre introduit : savoir préciser à quelle partie de $\mathbf{R}$ il appartient (entier, réel positif...)

Suites et séries numériques, suites et séries de fonctions

- savoir expliciter les premiers termes (de façon exacte ou approchée) d'une suite numérique ou d'une suite de fonctions, en particulier lorsqu'elle est définie par récurrence ;
- savoir obtenir un développement asymptotique d'une suite (fonction explicite de n) ;
- savoir calculer les coefficients de Fourier d'une fonction périodique ;
- savoir visualiser sur un même schéma les premiers termes d'une suite de fonctions.

Équations différentielles

- savoir résoudre une équation différentielle, un système d'équations différentielles, avec ou sans conditions initiales ;
- savoir récupérer une fonction solution et la tracer ;
- savoir tracer directement le graphe d'une solution obtenue par résolution numérique.

Graphisme

On a déjà évoqué le tracé de graphes de fonctions d'une variable réelle, de solutions d'une équation différentielle.

- savoir tracer une courbe du plan, définie par une équation cartésienne (de façon implicite), ou par un paramétrage, peut-être en coordonnées polaires, et gérer les discontinuités ;
- savoir tracer une courbe paramétrée de l'espace ;
- savoir tracer une surface définie par un paramétrage, ou par une équation cartésienne ;
- savoir visualiser un ensemble de points, sous forme d'une ligne polygonale ou non.

CHAPITRE

1

Premiers pas en Maple

1 Avant de commencer

1 1 Entrée des instructions

Maple 12 offre pas mal de fioritures d'écriture qui peuvent être perturbantes.
Nous choisiront le mode le plus basique qui nous permettra plus facilement de repérer nos erreurs. Pour cela, il faut aller dans le menu **Tool/Options** onglet **Display** option **Input display** où on choisira **Maple Input**. On peut aussi basculer en mode « Text » avec la touche F5 .
Toutes les instructions sont tapées devant un « prompt » (*invite* en french) rouge : **>**. Elles s'affichent en rouge. On termine son instruction obligatoirement par un point-virgule (;) si l'on veut que la réponse de MAPLE soit affichée, ou par deux points (:) si l'on veut que MAPLE exécute notre instruction sans l'afficher.
Toutes les instructions entre deux prompts représentent une cellule.

Remarque

Changer de ligne sans valider
On presse en même temps sur ⇧ et Entrée .

Pour faire exécuter l'ensemble d'une expression, on positionne le curseur sur n'importe quelle ligne de la cellule et on presse la touche Entrée .
MAPLE répond en bleu.
Pour utiliser le dernier résultat renvoyé par Maple (au sens *chronologique*), on utilise **%**, pour le pénultième, **%%**, pour l'antépénultième **%%%**, etc.
Par exemple, que répond la dernière instruction ?

```
> 2*3; %*5; %%-2; %%-2;
```

Remarque

Vous avez perdu le curseur
Réduisez votre fenêtre puis maximisez-la à nouveau.

1 2 Aide en ligne

Pour obtenir de l'aide, il y a trois possibilités :

- Tapez la commande qui vous intéresse, par exemple **plot**, puis selectionnez-la et tapez F1 ;
- Tapez **?plot;** puis Entrée ;
- Cliquez sur `Topic search` du menu `Help`.

1 3 Affectation

Pour stocker une valeur (un nombre, une fonction, un graphique, etc.) en mémoire, on l'affecte à un nom de variable (une étiquette) en utilisant le symbole **:=**
Par exemple, si l'on veut affecter la valeur 2 à la variable « a », on tape :

```
> a:=2;
```

et on obtient

Réponse du logiciel

```
a:=2
```

Pour ne pas afficher le résultat, on utilise « : » au lieu de « ; ».
On peut évaluer toute fonction de la variable « a » :

```
> a; a^2; cos(a); a/2;
```

On peut être amené à libérer une variable de son contenu.
On utilise **unassign('variable')** :

```
> unassign('a');
> a; a^2;
```

Pour libérer toutes les variables, on utilise **restart**.
Que se passe-t-il ici :

```
> x:=1; y:=2;
> x:=x+y;
> y:=x-y;
> x:=x-y;
```

et là :

```
> x:=100000; y:=0.0000002;
> x:=x+y;
> y:=x-y;
> x:=x-y;
```

1 4 Dangers de la modification d'une cellule

Avec Maple, vous pouvez revenir sur une cellule précédente et la modifier.

```
> a:=2;
> b:=a+3;
> c:=b^2;
> a+1;
```

Ah ! Vous vous êtes trompé sur la valeur de a qui vaut en fait -1 : vous la modifiez puis validez avec ↵. Qu'observez-vous ?
Conclusion : l'ordre chronologique ne correspond pas toujours au sens de lecture de haut en bas de l'écran.

1 5 Différents types de nombres

Comparez :

```
> 1+1/2+1/3;
> 1.+1/2+1/3;
> 1+1/2.+1/3;
> 1.+sqrt(2);
> 1+sqrt(2.);
```

Il n'y a rien de très rigoureux là-dedans : c'est du 100% subjectif made in Maple mais il faut savoir comment Maple choisit de traiter les nombres.
Pour plus de sûreté, on utilisera **evalf** :

```
> evalf(sqrt(2));
> evalf[100](sqrt(2));
```

1 6 Fonction et expression

On rentre une fonction comme on l'écrit. On utilise `-` et `>` pour faire `->`

```
> f:=x->2*x+5;
> f(5);
> (f(a+h)-f(a))/h;
```

Il ne faudra pas confondre avec l'expression associée.

```
> restart;
> f:=2*x+1;
> f(5);
> (f(a+h)-f(a))/h;
```

Pour transformer une expression en fonction, on utilise :

`unapply(expression,variable)`

```
>  f:=2*x+1;
>  f(5);
>  f:=unapply(f,x);
>  f(5);
```

1 7 Opérations mathématiques de base

La multiplication s'obtient avec `*`, la division avec `/`, les puissances avec `^`, la racine carrée (SQuare RooT en anglais...) avec **`sqrt(nombre)`**.
On peut de plus résoudre des équations avec **`solve(équation,inconnue)`**, calculer des limites avec **`limit(expression,x=a,direction)`** avec a un réel ou l'infini, **`direction`** prenant les valeurs **`left`** ou **`right`**.
On calcule des dérivées avec **`diff(f(x),x)`**, des intégrales avec **`int(g(x),x)`** ou bien avec **`int(g(x),x=-a..b)`**.

Avec les entiers, on utilisera **`irem(a,b)`** pour obtenir le reste entier. On obtient le même résultat avec **`a mod b`**. Pour le quotient, c'est **`iquo(a,b)`** :

```
> irem(1258478957,12547);
```

On aura parfois besoin de simplifier une expression renvoyée par Maple. On utilisera alors **`simplify`** ou bien parfois **`normal`** s'il s'agit d'une fraction rationnelle.

```
> g:=x->1/(1+x);
> e:=(g@@7)(x);
> normal(e);
```

L'opérateur de composition itérée est `@@`

1 8 Résolutions d'équations

La commande magique est **`solve(equation,variable)`** :

```
>  solve(x^2-2*x+2,x);
>  solve(x^2+x+1,x);
>  solve(x^2+b*x+c,x);
>  solve(x^2+b*x+c,b);
```

Il peut cependant y avoir quelques problèmes :

```
> solve(x^5+x^3-5*x+1,x);
> solve(ln(x)=cos(exp(x)),x);
```

On peut cependant demander une valeur approchée des solutions avec **fsolve**.

Pour résoudre une équation dans $\mathbb{Z}$, on utilise **isolve** (**i** comme *integer*) :

```
> isolve(51*x+44*y=1)
```

Pour les systèmes, on utilise les mêmes commandes mais avec une syntaxe adaptée :

```
> solve({x+y=1,x-y=2},{x,y});
```

La commande **allvalues** permet parfois de remplacer **RootOf** par sa valeur :

```
> e1:=x+y+z=1;
> e2:=x-y+z=2;
> e3:=x^2+1/y-1/z=2;
> sa:=fsolve({e1,e2,e3},{x,y,z});
> se:=solve({e1,e2,e3},{x,y,z});
> sv:=allvalues(se);
```

2 Les graphes

2.1 En dimension 2

La commande à utiliser est **plot(expression,xmin..xmax,options)**. Les options sont nombreuses...

```
>  plot(sin(x),x=-4*Pi..5*Pi);
>  plot(sin(x),x=-4*Pi..5*Pi,y=-2..5,color=wheat,labels=['abscisse','
    ordonnee'],title='le beau dessin');
```

Remarque

L'accent grave
L'apostrophe utilisée s'obtient avec Alt Gr + 7 + Espace

Modifier la fenêtre peut être très utile. Par exemple, comment remédier à ce problème :

```
>  plot(tan(x),x=-2*Pi..2*Pi);
```

On peut superposer des graphes :

```
>  plot([sin(x),cos(x)],x=-3*Pi..2*Pi);
>  plot([sin(x),cos(x)],x=-3*Pi..2*Pi,color=[navy,wheat],linestyle
    =[2,3]);
```

pour plus de précisions, allez voir l'aide de **plot**.
Pour des fonctions définies par morceaux, on utilise **piecewise**. Par exemple :

```
restart:
> f:=x->piecewise(x<2,x*sin(x),x>=2,3*x^2+1);
> plot(f(x),x=-4..4,y=-10..30,discont=true);
```

Certaines courbes sont définies implicitement par une équation qui ne correspond pas à une fonction. Par exemple, pour un cercle d'équation $x^2+y^2+x-3=0$:

```
> with(plots):
> implicitplot(x^2+y^2+x-3=0,x=-5..5,y=-2..2,color=blue);
```

On peut définir des fonctions à l'aide de procédures (que nous étudierons bientôt...)

```
> F:=proc(x)
    if x<=0 then x+1
    else sin(x)/x
    fi
  end:
```

On peut alors obtenir le graphe avec au choix :

```
> plot(F,-3..3)
```

ou

```
> plot(evaln(F(x)),x=-3..3)
```

Notez la différence...**`evaln`** signifiant « evaluate as a name »
Pour les courbes en polaire, on utilise l'option **`coords=polar`** :

```
plot(3*cos(t)-4*sin(t),t=0..2*Pi,coords=polar)
```

2.2 Animations

On peut créer une animation en créant une séquence de graphes dépendant d'un paramètre.
On va utiliser **`seq(expression dépendant de a, valeurs prises par a)`**
Par exemple :

```
> g:=seq(plot(sin(a*x),x=-5..5),a=[-1,0,1,2,3,4,5]):
```

Pour afficher l'animation, on doit aller chercher la commande **`display`** dans la bibliothèque **`plots`** qui n'est pas chargée par défaut avec la commande **`with`** et lui indique l'option **`insequence=true`** pour ne pas afficher tous les graphes en même temps :

```
> with(plots):
> display(g,insequence=true);
```

Vous cliquez sur le graphe et une barre d'icône apparaît pour lancer l'animation.
Il n'y a pas assez de valeurs de a. On peut en créer toute une série avec **`seq`** :

```
> g:=seq(plot(sin(a*x),x=-5..5),a=[seq(-1+0.01*h,h=0..100)]):
```

3 Quelques erreurs types

Voici une série d'erreurs que l'on retrouve souvent : il faut savoir les corriger seul(e)...
Partons d'un code correct :

```
> f:=x->sin(x)/x;
> a:=20;
> plot(f(x),x=-a..a);
```

Ça marche.
Mais

```
>  restart;
>  f:=x->sin(x)/x;
>  a=20;
>  plot(f(x),x=-a..a);
```

ne marche plus : pourquoi ?
Et pour ceci :

```
>  restart;
>  f=x->sin(x)/x;
>  a:=20;
>  plot(f(x),x=-a..a);
```

Le message d'erreur est différent mais c'est le même type d'erreur.
La plus fréquente des erreurs :

```
> a:=20
```

à égalité avec celle-ci :

```
> restart;
> f:=x->sin(x)/x;
> a:=20;
> plt(f(x),x=-a..a);
```

Sans oublier celle-ci :

```
> restart;
> f:=x->sin(x)/x;
> a:=20;
> plot(,x=-a..a);
```

EXERCICES

1 - 1

Explorez les fonctions

$$f : x \mapsto x^4 - 2x^2 + 3$$

$$g : x \mapsto \sqrt{\frac{x^2 - 2x + 2}{x^2 + 4}}$$

$$h : x \mapsto \arcsin(x) - \ln(x)$$

(limites, dérivées, signe de la dérivée, graphe...)
On peut procéder ainsi :

```
> f:=x->1/(x-1/2):
>   valx:=[1,2,3,5,10,15,20,100]:
>   valfx:=seq([x,f(x)],x=valx):
>   titre:=['x','f(x)']:
>   array([titre,valfx]);
```

sachant que **array(liste)** crée un tableau.
Définissez également un tableau de valeurs pour les trois fonctions.

1 - 2

La position d'un mobile en mouvement harmonique amorti est donné par l'équation :

$$x(t) = \mathrm{e}^{-0,1t}\left(\frac{2}{3}\sin(10t) + \frac{4}{5}\cos(10t)\right)$$

Déterminez les trois premiers instants où la vitesse du corps est nulle.

1 - 3

Trouvez les constantes a et b qui rendent la fonction suivante continue sur $\mathbb{R}$ à l'aide de MAPLE :

$$f(x) = \begin{cases} x^2 + bx + 1 \text{ si } x < 5 \\ 8 \text{ si } x = 5 \\ ax + 3 \text{ si } x > 5 \end{cases}$$

Vous utiliserez **piecewise, solve, limit**. Pour avoir la limite à gauche en **xo** on entre **limit(f(x),x=xo,left)**.

1 - 4

Trouvez les constantes a, b et c telles que le graphe de $f : x \mapsto 3x^4 + ax^3 + bx^2 + cx + d$ ait des tangentes horizontales en $(2\,;-3)$ et $(0\,;7)$. Tracez le graphe associé. Il y a un troisième point où la tangente est horizontale : trouvez-le ! Vérifiez ensuite s'il s'agit de maxima ou de minima relatifs ou ni l'un ni l'autre.

1 - 5

Parmi les exercices d'analyse faits en classe, reprenez ceux où MAPLE aurait pu vous aider.

1 - 6

On considère la fonction $f : x \mapsto x^3 - 2x^2 + 1$. Créez une animation permettant de comparer différentes sécantes passant par le point de coordonnées $(2, f(2))$ et la tangente à $\mathscr{C}_f$ en ce même point.

1 - 7

Trouvez le dernier chiffre de l'écriture décimale de

$$2007^{2011^{2013}}$$

1 - 8

Vous savez que Giralomo CARDANO a établi en 1547 qu'une solution de l'équation $x^3 + px + q = 0$ est :

$$\sqrt[3]{\frac{-q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{\frac{-q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

Sans utiliser la touche **solve**, résolvez les équations :

a. $(\mathrm{E}_1) : x^3 - 36x - 91 = 0$

b. $(\mathrm{E}_2) : x^3 - 15x - 4 = 0$

Vous pouvez essayer de prouver la formule en posant $x = u + v$ et en résolvant un système d'équations d'inconnues u et v.

1 - 9

Dérivez $x^{(x^x)}$ deux fois puis intégrez deux fois.
Faites de même avec $(x^x)^x$.

1 - 10

À l'aide de la fonction **sum**, vérifiez que

$$\sqrt{2} = \frac{7}{5}\left(1 + \frac{1}{100} + \frac{13}{100200} + \frac{135}{100200300} + \cdots\right)$$

1 - 11

La cycloïde est la courbe paramétrée définie par $\begin{cases} x(t) = t - \sin(t) \\ y(t) = 1 - \cos(t) \end{cases}$.

a. Tracer cette courbe pour $t \in [0, \mathrm{T}]$, avec $\mathrm{T} = 15$.

b. Construire une animation de paramètre $t \in [0, \mathrm{T}]$ dont le graphique au temps t est constitué du cercle de rayon 1 et de centre $(t, 1)$.

c. Construire une animation de paramètre $t \in [0, \mathrm{T}]$ dont le graphique au temps t est constitué du cercle de rayon $\varepsilon = 0.08$ et de centre $(x(t), y(t))$.

d. Construire une animation de paramètre $t \in [0, T]$ dont le graphique au temps t est constitué du morceau de cycloïde délimité par les points $(0,1)$ et $(x(t), y(t))$.

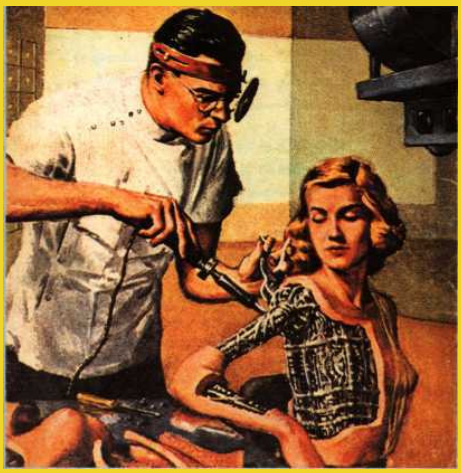
e. Afficher les trois animations sur un même graphique (à l'aide de la fonction `display`).

f. Comment interpréter la cycloïde en terme de trajectoire ?

Merci à Magali HILLAIRET *pour ce bel exercice...*

CHAPITRE 2

Programmer avec Maple



1 Quelques rappels

Quelques rappels avant de travailler. Commencez chaque nouvel exercice par un

```
> restart;
```

pour « désaffecter » toutes les variables éventuellement utilisées dans un calcul précédent. Si vous avez lancé un calcul qui a l'air de ne pas vouloir se terminer, cliquez sur l'icône **STOP** qui n'est active que lorsqu'un calcul est en cours.
Chacune de vos entrées débute après un *prompt* >, se termine par un ; [a] ou un : [b] puis appuyez sur la touche Entrée. Si vous voulez passer à la ligne sans valider la ligne précédente, tapez ⇧ + Entrée. Pour revenir plus haut, utilisez la souris ou les flèches de déplacement.

2 Listes et séquences

2 1 Listes

Les listes sont un type de variable particulier que peut manipuler MAPLE. Une liste se présente sous la forme d'une suite d'objets séparés par des virgules entre crochets :

```
> L:=[1,2,a,g,maman,ln(5),3/2,x->3*x+2,[7,9]]
```

Cette suite est ORDONNÉE. Ses éléments sont donc numérotés. On peut en extraire un élément particulier :

```
> L[5]
```

```
> L[9]
```

```
> L[9][1]
```

ou bien tous les éléments de la liste :

```
op(L)
```

ou bien une sous-liste :

```
> L[3..6];
```

On peut connaître la taille d'une liste :

```
> nops(L);
```

On peut ajouter un élément à une liste :

```
> L:=[op(L),nouveau]
```

On peut *concaténer* deux listes :

a. si vous voulez que le résultat soit affiché.
b. si vous voulez que MAPLE exécute sans afficher.

```
> L1:=[1,2,3,4];
> L2:=[a,b,c];
> L:=[op(L1),op(L2)]
```

La liste vide se note :

```
> V:=[];
> nops(V);
```

On peut effectuer des opérations sur une liste car c'est un type de variable comme un autre :

```
> S:=L1+[op(L2),d];
> P:=3*L1;
```

2.2 Les séquences

Elles ressemblent aux listes mais sans les crochets. La différence majeure est pourtant qu'on ne peut pas créer des séquences de séquences alors qu'on peut créer des listes de listes. Plus accessoirement, on peut modifier un élément d'une liste à l'aide d'une affectation mais ce n'est pas possible pour une séquence :

```
> S:=a,b,c,d,e,f;
> S[3];
> S[3]:=4;
> L:=[S];
> L[3]:=4;
> L;
```

En général, elles servent à fabriquer des listes...

```
> S1:=10,20,30,40,50,60;
> S2:=seq(10*k,k=3..10);
> S3:=10*k $ k=2..9;
> S4:= m $ 5;
```

3 Algorithme et programme

Un algorithme est une description rigoureuse en un nombre fini d'étapes de la résolution d'un problème.
La notion d'algorithme existe depuis bien avant l'existence des ordinateurs et même bien avant la naissance du mot lui-même puisqu'il est tiré du nom du célèbre mathématicien persan ىسوم نب دمحم يبلبرطقلا يسوجملا يمزراوخلا ou si vous préférez Abu 'Abdallah Muhammad ibn Musa al-Khwarizmi qui vécut au IX^{e} siècle de notre ère, c'est-à-dire près de 3000 ans après les premiers algorithmes de calcul connus mis au point en Mésopotamie.
Vous utilisez depuis longtemps des algorithmes, à la manière de M. Jourdain, sans toujours le savoir, par exemple lorsque vous lisez une recette de cuisine, lorsque vous recherchez un mot dans le dictionnaire, lorsque vous effectuez une addition posée.
Cependant, cette année, nous nous contenterons d'algorithmes destinés à être utilisés par un ordinateur. Nous introduirons donc un langage algorithmique le moins ambiguë pos-

sible et le plus standard afin de pouvoir mettre en œuvre ces algorithmes sur n'importe quelle machine.
Même si nos algorithmes doivent être indépendants de tout langage de programmation, il sera enrichissant de les tester « en vrai » sur machine. Nous serons donc amenés à parler à la machine. Pour cela nous utiliserons un langage servant d'interface entre la machine et nous car tout être humain normalement constitué ne peut décemment entamer une longue conversation avec un ordinateur en langage machine.
C'est bien sûr Maple qui sera notre langage. Ceux d'entre vous qui choisirons l'option info utiliseront plutôt le langage Pascal.

4 Quelques types prédéfinis

Les variables, les constantes, les arguments introduits dans les algorithmes peuvent être de types différents. Tous les langages ont des types prédéfinis à peu près communs.
Sur Maple, les principaux types sont `float`, `integer`, `negint`, `nonnegint`, `fraction`, `list`, etc.
Consultez l'aide : `?type;`

4.1 Les entiers

Un entier naturel est un entier positif ou nul. Le choix à faire (c'est-à-dire le nombre de bits[c] à utiliser) dépend de la fourchette des nombres que l'on désire utiliser. Pour coder des nombres entiers naturels compris entre 0 et 255, il nous suffira de 8 bits (un octet) car $2^8 = 256$. D'une manière générale un codage sur n bits pourra permettre de représenter des nombres entiers naturels compris entre 0 et 2^{n-1}. Les processeurs disponibles actuellement proposent des codages sur 32 ou 64 bits. Cependant, les logiciels ne tiennent pas toujours compte des 64 bits et vous avez beau disposer d'un processeur 64 bits, il sera utilisé comme s'il était de type 32 bits (notamment si le système d'exploitation est microsoft-windows...).
Pour représenter un nombre entier naturel après avoir défini le nombre de bits sur lequel on le code, il suffit de ranger chaque bit dans la cellule binaire correspondant à son poids binaire de la droite vers la gauche, puis on « remplit » les bits non utilisés par des zéros.

Pour les entiers relatifs, le bit situé à l'extrême gauche (le bit de poids fort), représentera le signe : 0 pour un nombre positif, 1 pour un nombre négatif. Cela restreint donc l'étendue des nombres disponibles.
Un entier relatif positif est donc codé comme un entier naturel mais avec un 0 à gauche.
Pour les entiers négatifs, on utilise une petite ruse : le complément à 2 que nous étudierons à titre d'exercice.
Ainsi, sur les ordinateurs qui sont à notre disposition, tout introduction d'un entier dépassant $2^{31} - 1 = 2\,147\,483\,647$ engendrera une erreur.

4.2 Les flottants

La norme IEEE définit la façon de coder un nombre réel. Cette norme se propose de coder le nombre sur 32 bits et définit trois composantes :

- le signe est représenté par un seul bit, le bit de poids fort (celui le plus à gauche) ;
- l'exposant est codé sur les 8 bits consécutifs au signe ;
- la mantisse, c'est-à-dire le nombre « sans la virgule », sur les 23 bits restants.

La précision des nombres réels est approchée. Elle dépend du nombre de positions décimales, suivant le type de réel elle sera au moins :

c. Le mot « bit » est la contraction des mots anglais binary digit, qui signifient « chiffre binaire », avec un jeu de mot sur bit, « morceau ». Il ne faut pas confondre avec le mot « byte », en français « multiplet », qui désigne un assemblage de bits, en général 8 (un octet).

– de 6 chiffres après la virgule pour un système 32 bits ;
– de 15 chiffres après la virgule pour un système 64 bits.

Ces limitations peuvent causer quelques surprises. Par exemple, vous pouvez tester sur Maple :

```
> 100000+0.0000002;
```

Réponse du logiciel

```
100000
```

Remarque

Ces problèmes d'implémentation des entiers et des flottants ne concernent pas l'algorithmique mais la programmation dans un certain langage, sur un certain système.

4.3 Les caractères et chaînes de caractères

Le type « caractère » permet de stocker le code d'un caractère, c'est-à-dire un nombre entier. Une chaîne de caractères est une suite ordonnée de caractères. La représentation d'une chaîne de caractères dépend d'un système à un autre.

4.4 Les booléens

Avant d'étudier les structures conditionnelles, il nous faut nous habituer aux tests, aux booléens et à leurs opérateurs, qui ne sont pas d'un usage aussi naturel que les types flottants et entiers.

– Une variable de type booléen n'a que deux valeurs possibles : VRAI ou FAUX.
– Ces booléens sont en particulier les valeurs retournées par les opérateurs de comparaison : **=**, **<**, **<=**, **>**, **>=**, **<>**.
– Ils ont en général 4 opérateurs : **ET**, **OU**, **OUEX**, **NON**.

Recherche

Par exemple, quelle sera la réponse à :

– $3 > 2$ ET $2 > 2$?
– $3 > 2$ ET $2 \geqslant 2$?
– $3 > 2$ OU $2 > 2$?
– $3 > 2$ OU $2 \geqslant 2$?
– $3 > 2$ OUEX $2 \geqslant 2$?
– $3 > 2$ OUEX $2 > 2$?
– NON $2 > 2$?
– NON $3 > 2$?

Sur Maple, les opérateurs logiques sont **and** et **or**. Il y a aussi **not** qui donne la négation.

Recherche

Que dire du **or** de Maple ?

Il existe une bibliothèque dédiée à la logique qui logiquement s'appelle **logic**.
Quand une expression ne contient pas **and**, **or** ou **not**, il faut utiliser **evalb** ou **is** :

```
> evalb(3>2);
```

Danger

type de nombre
On ne peut comparer que des nombres de type **numeric** (i.e. **integer**, **float** et **fraction**) :

```
> evalb(3>2.);
> evalb(3>7/2);
> evalb(sqrt(2)>1);
> evalb(sqrt(2.)>1);
```

Cependant, cela fonctionne avec **is** mais cela ne fonctionnera pas avec les tests de boucles et de structures conditionnelles.

5 Les structures conditionnelles

Elles sont communes aux deux types de programmation que nous étudierons. C'est le fameux SI...ALORS...SINON...
Voici un bel algorithme :

```
Fonction zorg(a,b,c : entier) : chaîne
Début
    Si a<>b Alors
        Si a+1=b Alors
            Retourner 'aarrgh'
        Sinon
            Si c<>7 Alors
                Retourner 'glurp'
            Sinon
                Retourner 'pas glop'
            FinSi
        FinSi
    Sinon
        Retourner 'eviv bulgroz'
    FinSi
Fin
```

Recherche

Que répond
- **zorg(4,5,7)**?
- **zorg(4,5,8)**?
- **zorg(4,6,7)**?

Sur Maple, on utilise la valeur d'un test pour exécuter une instruction conditionnelle dont la syntaxe est :

if condition **then** instruction1 **else** instruction2 **fi**;

Par exemple :

```
> if (3>2) then print('bonjour') else print('salut') fi;
> if (3<2) then print('bonjour') else print('salut') fi;
```

On peut écrire des instructions conditionnelles en cascade :

if condition1 **then** instruction1 **elif** condition2 **then** instruction2 **else** instruction3 **fi** ;

Par exemple :

```
> a:=1; b:=1; c:=1;

> if (b^2-4*a*c>0)
     then print(‘2 solutions reelles‘)
  elif (b^2-4*a*c=0)
     then print(‘1 solution reelle‘)
  else print(‘2 solutions complexes conjuguees‘)
  fi;
```

6 Procédures

Une procédure est une fonction à la syntaxe précise dépendant de la donnée d'une série d'arguments et d'une série d'instructions.
La syntaxe générale est :

```
> nom:=proc(argument1::type1,argument2::type2,.....)
  local var1, var2,... ;
  instructions
  end:
```

Les variables locales sont celles introduites dans la procédure et dont les noms seront « désaffectés » à la fin de la procédure.

Danger

première ligne
Pas de point-virgule sur la première ligne contenant **`proc`** !

Par exemple :

```
> Table:=proc(n::posint)
  local T,k;
  T:=[seq(n*k,k=1..10)];
  printf("La table de %a est %a",n,T)
  end:
```

Vous pourrez regarder la syntaxe de **`printf`**. L'usage des **`%a`** indique qu'on va les remplacer dans l'ordre par les variables formelles qui sont listées après les guillemets. Pour avoir des nombres flottants on utilise **`%f`**.
Par exemple, pour obtenir la table de 5 on tape :

```
> Table(5);
```

On a souvent besoin du résultat d'une procédure et la petite phrase « pour faire joli » peut être gênante.
On préférera une procédure de cette forme :

```
> Table:=proc(n::posint)
  local k;
  RETURN([seq(n*k,k=1..10)])
  end:
```

On peut ainsi afficher le deuxième élément de la table de 5 :

```
> Table(5)[2];
```

Recherche Que donne `Table(-5)` ?

On peut utiliser une structure conditionnelle :

```
> parite:=proc(n::integer)
   if (irem(n,2)=0)
     then RETURN(‘pair‘)
     else RETURN(‘impair‘)
   fi
  end:
```

Recherche À votre avis, que renvoie `irem(a,b)` ?

7 La programmation impérative : au plus près de la machine

Un algorithme impératif va modifier chronologiquement l'état de la mémoire (voir la section 1.1.3 page 12). Les valeurs des variables vont sans cesse évoluer. Cependant, la plupart du temps, les notations des structures itératives utilisées n'expriment pas clairement l'état des variables. Il faudra donc, pour bien maîtriser l'algorithme, le commenter pour décrire les états parcourus par le système.

Danger

On veillera à ne pas modifier les arguments d'une fonction avec quelque chose du style :

```
> non:=proc(a,b,c)
  a:=a+1;
  RETURN(a);
  end:
```

Il faut introduire une variable locale prenant au départ la valeur de l'argument puis la modifier :

```
> oui:=proc(a,b,c)
  local temp;
  temp:=a;
  temp:=temp+1;
  RETURN(temp);
  end:
```

7 1 Structures itératives

Exemple

Soit deux entiers naturels n et N tels que $n \leqslant \mathrm{N}$. On voudrait calculer la somme des entiers de n à N.

Les notations mathématiques peuvent ici nous guider. En effet, cette somme $\mathrm{S}(n,\mathrm{N})$ s'écrit :

$$\mathrm{S}(n,\mathrm{N}) = \sum_{k=n}^{k=\mathrm{N}} k = n + (n+1) + (n+2) + \cdots + (\mathrm{N}-1) + \mathrm{N}$$

7 1 a Tant que

```
Fonction somme(n,N : entier) : entier
{ On doit avoir n<=N. On obtient la somme des entiers successifs de n à N compris }
{ Nous allons créer une variable provisoire qui contiendra la somme en construction ainsi que l'entier provisoire qui va être itéré. }
Variable
Stemp,ktemp :
Début
  Stemp← 0
  ktemp ← n
  { On choisit d'initialiser la somme à 0 et l'indice d'itération à n. }
  TantQue ktemp <= N Faire
    { On entre dans la boucle jusqu'à la valeur N comprise }
    Stemp← Stemp+ktemp
    { À l'état initial on commence donc par rajouter n à 0 }
    ktemp← ktemp+1
    { ktemp est incrémenté après la somme. }
    {  À la dernière étape, ktemp vaut N, }
    {  est rajouté à Stemp puis ktemp reçoit N+1 }
    {  et on sort donc de la boucle. }
    {  Dans l'état final on a donc la somme des entiers de n à N. }
  FinTantQue
  Retourner Stemp
Fin
```

Avec Maple :

```
> Somme:=proc(n,N)
   local Stemp,ktemp;
   Stemp:=0;
   ktemp:=n;

   while ktemp<=N do
     Stemp:=Stemp+ktemp;
     ktemp:=ktemp+1;
   od;

   RETURN(Stemp)
  end:
```

Vérifions que notre algorithme fait bien ce qu'on attend de lui.
La propriété $\mathscr{P}$(**ktemp**) : « **Stemp**$= \mathrm{S}(n,$**ktemp**$)$ et **ktemp**$\leqslant \mathrm{N}$ » est l'*invariant de boucle.*

Au départ, on a bien $\mathscr{P}(n)$: **Stemp**$= S(n,n) = n$ et $n \leqslant N$.
À chaque fin d'occurrence de la boucle, cette propriété reste vraie ainsi qu'en sortie.

Recherche

On modifie légèrement l'ordre des instructions de cet algorithme :

```
ktemp← ktemp+1
Stemp← Stemp+ktemp
```

Y a-t-il d'autres choses à modifier ?

7.1.b Pour

En anglais, c'est « for » :

```
for var from debut to fin by increment do
    instructions;
od;
```

Par exemple :

```
> Somme:=proc(n::posint)
  local S,k;
  S:=0;
   for k from 1 to n by 1 do
     S:=S+k;
   od;
 RETURN(S)
 end:
```

On peut arrêter l'itération dès qu'une condition a été réalisée (et se passer du **to**) :

```
> Somme:=proc(n::posint)
  local S,k;
  S:=0;
   for k from 1 while P<=n do
     S:=S+k;
   od;
 RETURN(S)
 end:
```

Expliquez....

8 Au plus près de l'humain : la programmation récursive

8.1 Qu'est-ce que c'est ?

La récursion est un mécanisme puissant permettant, comme l'itération, d'exprimer la répétition des opérations, mais de manière plus concise et en ayant la possibilité de se passer d'affectation et ainsi éviter les problèmes de surveillance de l'état de la mémoire.
Il ne se limite pas au cas des suites numériques définies par une relation de récurrence.
Par exemple, définissons un caractère comme un élément de l'ensemble constitué des 26 lettres de l'alphabet latin : $\{'a', 'b', \cdots, 'z'\}$
On peut donner une définition récursive d'un mot :

Un mot est soit un caractère, soit un caractère suivi d'un mot.
Sur Maple, reprenons un exemple du paragraphe précédent :

```
> Somme_r:=proc(n::posint)
    if n=1 then 1
    else Somme_r(n-1)+n;
    fi;
end:
```

Il faut connaître un cas simple et un moyen de simplifier un cas compliqué...
On peut obtenir les résultats intermédiaires avec l'option **remember** et en demandant le quatrième opérande de la procédure :

```
> Somme_r:=proc(n::posint)
    option remember;
    if n=1 then 1
    else Somme_r(n-1)+n;
    fi;
end:
```

Puis :

```
Somme_r(9);
op(4,eval(Somme_r));
```

Cette option est malgré tout plus intéressante dans le cas de récursions multiples : les calculs intermédiaires étant stockés, ils n'ont pas à être recalculés et cela gagne du temps.
Par exemple, pour la suite de Fibonacci $u_{n+2} = u_{n+1} + u_n$ avec $u_0 = u_1 = 1$:

```
> fibo:=proc(n::nonnegint)
   option remember;
   if n=0 then RETURN(1);
     elif n=1 then RETURN(1);
     else RETURN(fibo(n-1)+fibo(n-2));
   fi
  end:
```

Sans l'option `remember`, **`fibo(30)`** prend pas mal de temps mais est instantané quand on l'ajoute : à ne pas oublier...

EXERCICES

2 - 1 Valeur absolue

Écrivez un algorithme qui renvoie la valeur absolue d'un réel x.

2 - 2 Système

Soit un système $\begin{cases} a_{11}x + a_{12}y = b_1 \\ a_{21}x + a_{22}y = b_2 \end{cases}$.
Déterminez un algorithme qui renvoie la solution de ce système si elle est unique.

2 - 3 Années bissextiles

Une année bissextile est une année dont le millésime, supérieur à 1583, est divisible par 4, sauf les limites de siècles qui ne sont pas multiples de 400.
Ainsi 2 000 et 2 008 sont bissextiles mais 2 100 et 2 011 ne le sont pas.
Définissez alors un algorithme qui reconnaît si un millésime est celui d'une année bissextile. Vous pourrez utiliser la fonction `irem` qui donne le reste de la division euclidienne.

2 - 4 Conversion

Déterminez un algorithme qui reçoit un nombre entier de secondes et qui renvoie quatre entiers correspondant à sa conversion en jours, heures, minutes et secondes. On pourra utiliser une fonction « quo » qui renvoie le quotient entier de la division de deux entiers.

2 - 5 Partie entière

Donnez un algorithme impératif et récursif renvoyant la partie entière d'un réel x.

2 - 6 Division euclidienne

Déterminer un algorithme impératif et récursif qui renvoie le quotient de deux entiers naturels non nuls a et b.
Déterminer un algorithme qui renvoie le reste entier de a et b.

2 - 7 Intégration numérique

Déterminez un algorithme qui donne une valeur approchée de $\int_{t=a}^{t=b} f(t)\,\mathrm{d}t$ pour une certaine fonction f continue sur un intervalle $[a;b]$ par la méthode des rectangles puis celle des trapèzes.

2 - 8 Suite

On veut étudier la suite de terme général $u_n = \frac{1}{n^2}$ ainsi que la suite $v_n = \sum_{k=1}^{n} u_k$, avec $n \in \mathbb{N}^*$.
Proposez un algorithme qui affiche u_n et v_n pour une valeur de n donnée.

2 - 9 Maximum

Donner des procédures impératives et récursives donnant le maximum d'une liste de nombres réels quelconques.

2 - 10 Tête et queue

La tête d'une liste est son premier élément. La queue d'une liste est la sous-liste créée en lui coupant la tête. Donnez deux procédures `tete(liste)` et `queue(liste)`.

2 - 11 test de croissance

Donner des procédures impératives et récursives déterminant si une liste donnée est croissante. On pourra éventuellement utiliser `tete` et `queue`.

2 - 12 Dichotomie

Pour résoudre une équation du type $f(x) = 0$, on recherche graphiquement un intervalle $[a,b]$ où la fonction semble changer de signe.
On note ensuite m le milieu du segment $[a,b]$. On évalue le signe de $f(m)$.
Si c'est le même que celui de $f(a)$ on remplace a par m et on recommence. Sinon, c'est b qu'on remplace et on recommence jusqu'à obtenir la précision voulue.
Donner une procédure récursive puis une procédure impérative prenant comme arguments une fonction f, les bornes de l'intervalle d'étude a et b et une précision de calcul eps.

2 - 13 Héron

HÉRON d'Alexandrie a trouvé une méthode permettant de déterminer une approximation de la racine carrée d'un nombre positif vingt siècles avant l'apparition des ordinateurs.

Si x_n est une approximation strictement positive par défaut de $\sqrt{a}$, alors a/x_n est une approximation par excès de $\sqrt{a}$ (pourquoi ?) et vice-versa.
La moyenne arithmétique de ces deux approximations est $\frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$ et constitue une meilleure approximation que les deux précédentes.
On peut montrer c'est une approximation par excès (en développant $\left(x_n - \sqrt{a}\right)^2$ par exemple).
On obtient naturellement un algorithme... Donnez une version récursive puis impérative prenant en argument le nombre dont on cherche la racine carrée, une première approximation x_0 et une précision **eps**.

2 - 14 Décimales de e

Posons $e_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$
Alors on a aussi

$$e_n = 1 + 1 + \frac{1}{2}\left(1 + \frac{1}{3}\left(1 + \frac{1}{4}\left(\cdots\left(\frac{1}{n-1}\left(1 + \frac{1}{n}\right)\right)\right)\right)\right)$$

Un exercice classique montre que $e - e_n \leqslant \frac{n+2}{n+1}\frac{1}{(n+1)!}$.
Ainsi, pour $n = 167$, on obtiendra 300 bonnes décimales au moins : trouvez-les !

2 - 15 Somme de chiffres

Proposez un algorithme qui prend en argument un entier écrit en base 10 et qui calcule la somme des chiffres composant cet entier.
On pourra utiliser la fonction **irem**

2 - 16 Trapèzes

Soit f une fonction continue sur un intervalle $[a,b]$.
On suppose que cette fonction f est définie sur MAPLE.
Déterminez une procédure **trap:=proc(f,a,b,dx)** qui donne l'approximation de l'intégrale sur $[a,b]$ de f par la méthode des trapèzes en utilisant une subdivision de largeur **dx**.
Vous en donnerez une version récursive et une version impérative.

2 - 17 tests

On définit simultanément deux fonctions **truc** et **machin** :

```
truc(n):=n<>1 and (n=0 or machin(n-1))
and
machin(n):=n<>0 and (n=1 or truc(n-1))
```

Spécifiez au mieux ces deux fonctions.

2 - 18 Algorithme glouton

On dispose d'au moins deux types de pièces de valeurs premières entre elles et on veut rendre la monnaie en utilisant le moins de pièces possibles.
Une première idée est de répéter le choix de la pièce de plus grande valeur qui ne dépasse pas la somme restante puis de répéter le test avec le pièce de valeur inférieure : c'est un exemple d'algorithme glouton.
Écrivez une procédure MAPLE
piece_glouton:=proc(n,L)
qui donnera la décomposition de la somme d'argent **n** en utilisant les pièces de monnaies dont les montants appartiennent à la liste **L**.
Vous utiliserez des boucles **while**, les opérateurs **and**, **+**, **-** et **>**.
Ainsi, pour savoir comment former 39 centimes avec nos pièces européennes, on entrera sur MAPLE :

```
> piece_glouton
    (39,[1,2,5,10,20,50,100,200])
```

et on obtient **[20,10,5,2,2]**.
On rappelle que **nops(L)** renvoie le nombre d'éléments d'une liste **L**, et que **L[k]** renvoie son k-eme élément.

2 - 19 Base 2

Donnez une procédure MAPLE **base_deux:=proc(n)** qui renvoie la liste des chiffres de la décomposition de **n** en base 2.
On pourra utiliser les commandes **iquo(a,b)** et **irem(a,b)** qui donnent respectivement le quotient et le reste de la division euclidienne de l'entier **a** par l'entier **b**.
On rappelle qu'on rajoute un élément **e** à la fin d'une liste **L** en tapant **[op(L),e]**

CHAPITRE

3

Tableaux et matrices

1 Étoiles et espaces

La commande `array(1..n,1..m)` construit un tableau de `n` lignes et `m` colonnes.
Que fait cette procédure :

```
> mystere:=proc(n)
  local T,j,i;
  T:=array(1..n,1..n);
  for j from 1 to n do
    for i from 1 to n do
      if j>i then T[i,j]:=‘‘;
      fi;
    od;
  od;
  print(T);
  end:
```

Transformez-la un peu pour obtenir ceci avec $n = 10$ par exemple :

```
1
1    1
1    2    1
1    3    3    1
1    4    6    4    1
1    5    10   10   5    1
1    6    15   20   15   6    1
1    7    21   35   35   21   7    1
1    8    28   56   70   56   28   8    1
1    9    36   84   126  126  84   36   9    1
```

Complétez cette dernière procédure afin de remplacer chaque nombre pair par une espace `‘ ‘` et chaque nombre impair par une étoile `‘*‘`. Lancez-la pour $n = 48$: incroyable, non ?

2 Fabriquons nos outils

2 1 Création d'une matrice

Dans cette section, nous créerons nos matrices à l'aide de la commande `array(1..n,1..m)` vue lors du TD précédent ou directement comme une liste de listes représentant les lignes.

Par exemple, $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ sera rentré avec :

```
> M:=[[1,2],[3,4],[5,6]];
```

Comment faire calculer le nombre de lignes et le nombre de colonnes de `M` à MAPLE ?

2 2 Somme de matrices

Créez une procédure qui prend comme arguments deux matrices et renvoie leur somme. Elle commencera par :

```
> somm:=proc(A,B)
  local C,i,j,k;
  C:=[[0$nops(A[1])]$nops(A)];
              ?...?
  RETURN(C);
  end:
```

2 3 Produit de matrices

Soit $A = (a_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant i\leqslant m}}$ et $B = (b_{ij})_{\substack{1\leqslant i\leqslant m\\1\leqslant i\leqslant p}}$.
Alors $AB = C$ avec $C = (c_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant i\leqslant p}}$ et

$$\forall (i,j)\in \mathbb{N}_n^*\mathbb{N}_p^*,\quad c_{ij} = \sum_{k=1}^{m} a_{ik}\cdot b_{kj}$$

Créez alors une procédure qui prend comme arguments deux matrices et renvoie leur produit. Elle commencera par :

```
> prodm:=proc(A,B)
  local C,i,j,k;
  C:=[[0$nops(B[1])]$nops(A)];
              ?...?
  RETURN(C);
  end:
```

Vous n'utiliserez que des boucles (pas de fonctions de calcul formel !).

2 4 Matrice identité

Créer une procédure **`Id:=proc(n)`** qui crée la matrice identité d'ordre n.

2 5 Puissances d'une matrice

Tout est dans le titre...

```
> puim:=proc(A,n)
          ?...?
 end:
```

2 6 Trace d'une matrice

Tout est dans le titre...

```
> Tracem:=proc(A)
          ?...?
 end:
```

2 7 Transposée d'une matrice, matrices orthogonales

Tout est dans le titre...

```
> Transm:=proc(A)
          ?...?
 end:
```

On appelle matrice orthogonale une matrice carrée dont les vecteurs colonne sont de norme 1 et orthogonaux deux à deux.
Construisez une procédure testant si une matrice est orthogonale.
Calculez le produit d'une telle matrice par sa transposée.

3 Avec les outils MAPLE

Pour utiliser toute la puissance de calcul de MAPLE, on définit en fait une matrice par

```
Matrix(nb lignes, nb colonnes, [a11,a12,a13,...,a1n,a21,a22,a23,...
```

Par exemple

```
> A:=Matrix(2,2,[1,2,3,4]);
```

On peut additionner, multiplier, élever à une puissance, appliquer une fonction à une matrice selon la syntaxe habituelle, sauf pour le produit où il faudra taper · (le point). Il existe une centaine de fonctions dans la bibliothèque **LinearAlgebra** plus celles déjà présentes sur Maple : nous ne les explorerons pas toutes ! Nous allons les découvrir à travers des petits exercices.

EXERCICES

3 - 1 Noyau, image

Qu'est-ce que l'endomorphisme de $\mathbb{R}^2$ d'expression analytique canonique

$$\begin{cases} x' &= -\frac{1}{5}x - \frac{2}{5}y \\ y' &= \frac{3}{5}x + \frac{6}{5}y \end{cases}$$

Vous aurez besoin de déterminer la matrice A associée, le noyau de A grâce à NullSpace(A) qui en renvoie une base, l'image avec ColumnSpace(A). Ensuite, à vous d'essayer des petits calculs sur A pour deviner sa nature.

3 - 2 Puissances de matrices

1. Soit $A = \frac{1}{3}\begin{pmatrix} 0 & -2 & -2 \\ 2 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}$.

a. Calculer A^2, A^3 et A^4.

b. Montrez que $\{A, A^2, A^3, A^4\}$ est un groupe pour le produit matriciel de $\mathfrak{M}_3(\mathbb{R})$.

2. Pour chacune des matrices A suivantes, calculer A^n pour tout $n \in \mathbb{N}$.

a. $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$ avec $a \in \mathbb{R}$.

b. La matrice J_p de $\mathfrak{M}_p(\mathbb{R})$ dont tous les coefficients sont égaux à 1.

3 - 3 Matrice orthogonale

$$A = \frac{1}{5}\begin{pmatrix} 3 & 4 & c \\ a & -3 & 0 \\ 0 & b & d \end{pmatrix}$$

Déterminez a, b, c et d pour que A soit une matrice orthogonale ($A^{-1} = {}^t$).

Mini-chaînes de Markov...

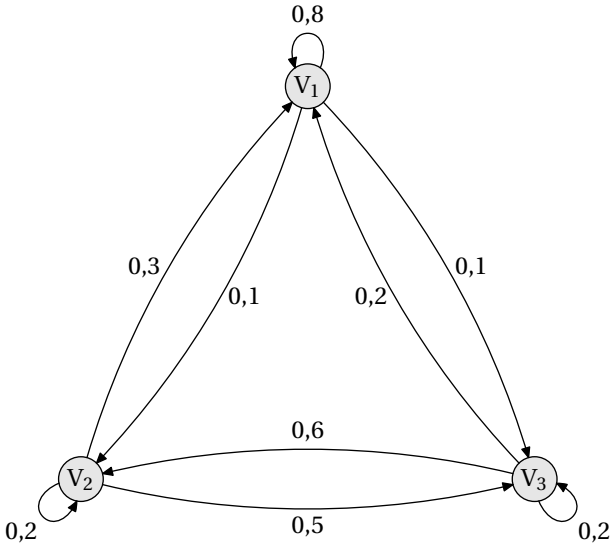
3 - 4 Aidons la mafia

Les chaînes de Markov sont issues de la théorie des probabilités et utilisent des outils d'algèbre linéaire qui nous intéressent aujourd'hui. Elles permettent de simuler des phénomènes aléatoires qui évoluent au cours du temps. Nous allons les découvrir à travers l'étude d'un exemple simple.

Zlot, Brzxxz et Morzgniouf sont trois villes situées respectivement en Syldavie, Bordurie et Bouzoukstan. Des trafiquants de photos dédicacées du groupe ABBA prennent leur marchandise le matin dans n'importe laquelle de ces villes pour l'apporter le soir dans n'importe quelle autre. On notera pour simplifier V_1, V_2 et V_3 ces villes et p_{ij} la probabilité qu'une marchandise prise le matin dans la ville V_i soit rendue le soir dans la ville V_j. La matrice $(p_{ij})_{\substack{1\leqslant i\leqslant 3\\ 1\leqslant j\leqslant 3}}$ est appelée *matrice de transition* de la chaîne de Markov. Que s'attend-on à observer sur les colonnes d'une matrice de transition ? Supposons que P soit connue et vaille

$$P = \begin{pmatrix} 0,8 & 0,3 & 0,2 \\ 0,1 & 0,2 & 0,6 \\ 0,1 & 0,5 & 0,2 \end{pmatrix}$$

Les trafiquants se promenant de ville en ville, il peut être utile de visualiser leurs déplacements par le *diagramme de transition* suivant



On notera $x_i^{(k)}$ la proportion de trafiquants qui se trouvent au matin du jour k dans la ville V_i. En probabilités, on appelle *vecteur d'état* tout élément $(x_1, \cdots, x_n)$ de $\mathbb{R}^{+n}$ tel que $x_1 + \cdots + x_n = 1$.

Ainsi, $x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$ est un vecteur d'état.

On montre que les vecteurs d'état de la chaîne sont liés par la relation

$$x^{(k)} = P \cdot x^{(k-1)}$$

et donc

$$x^{(k)} = \mathrm{P}^k \cdot x^{(0)}$$

Supposons que le chef de la mafia locale dispose de 1000 trafiquants qui partent tous le matin du jour 0 de la ville de Zlot. Quelle sera la proportion de trafiquants dans chacune des villes au bout d'une semaine ? d'un mois ? d'un an ?

Le parrain voudrait que la proportion moyenne de trafiquants soit stable d'un jour sur l'autre. Il recherche donc les vecteurs d'état x vérifiant l'équation $\mathrm{P} \cdot x = x$. Vous apprendrez après l'été à résoudre de manière systématique ce genre de problème. Nous allons pour l'heure nous débrouiller sans appui théorique mais avec Maple et `NullSpace` Comment procéder ? La matrice identité s'écrit `Matrix(3,3,shape=identity)`.

3 - 5 Météo

À Morzgniouf, les jours sont soit secs, soit pluvieux. On désigne par E_1 l'état sec et E_2 l'état pluvieux et par p_{ij} la probabilité qu'un jour soit dans l'état E_i sachant que le jour précédent était dans l'état E_j. Les scientifiques Bouzouks ayant observé les phénomènes météorologiques des trent-deux dernières années à Morzgniouf ont établi la matrice de transition suivante

$$\mathrm{P} = \begin{pmatrix} 0,750 & 0,338 \\ 0,250 & 0,662 \end{pmatrix}$$

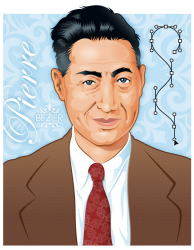
Sachant qu'il fait beau aujourd'hui, quelle est la probabilité qu'il pleuve dans dix jours ?

CHAPITRE

4

Courbes de Bézier

1 Renault et Citroën



Dans les années 60, les ingénieurs Pierre BÉZIER et Paul DE CASTELJAU travaillant respectivement chez Renault et Citroën, réfléchissent au moyen de définir de manière la plus concise possible la forme d'une carrosserie.
Le principe a été énoncé par BÉZIER mais l'algorithme de construction par son collègue de la marque aux chevrons qui n'a d'ailleurs été dévoilé que bien plus tard, la loi du secret industriel ayant primé sur le développement scientifique...
Pour la petite histoire, alors que Pierre BÉZIER (diplômé de l'ENSAM et de SUPÉLEC), à l'origine des premières machines à commandes numériques et de la CAO ce qui n'empêcha pas sa direction de le mettre à l'écart : il se consacra alors presque exclusivement aux mathématiques et à la modélisation des surfaces et obtint même un doctorat en 1977.
Paul DE CASTELJAU était lui un mathématicien d'origine, ancien élève de la Rue d'ULM, qui a un temps été employé par l'industriel automobile.
Aujourd'hui, les courbes de Bézier sont très utilisées en informatique.
Une Courbe de Bézier est une courbe paramétrique aux extrémités imposées avec des points de contrôle qui définissent les tangentes à cette courbe à des instants donnés.

2 Algorithme de Casteljau

2 1 Avec 3 points de contrôle

Soit t un paramètre de l'intervalle $[0,1]$ et P_1, P_2 et P_3 les trois points de contrôle.
On construit le point M_1 barycentre du système $\{(P_1,1-t),(P_2,t)\}$ et M_2 celui du système $\{(P_2,1-t),(P_3,t)\}$.
On construit ensuite le point M, barycentre du système $\{(M_1,1-t),(M_2,t)\}$.
Exprimez M comme barycentre des trois points P_1, P_2 et P_3.
Faites la construction à la main avec $t=1/3$ par exemple.
Commentez le programme suivant :

```
bezier1:=proc(p1,p2,p3)
local m,m1,m2,P,S,B,k,t;
P:=plot([p1,p2,p3],color=green);
B:=NULL;
for k from 0 to 1 by 0.01 do
   m1:=(1-k)*p1+k*p2;
   m2:=(1-k)*p2+k*p3;
   m:=(1-k)*m1+k*m2;
   B:=B,m;
od;
B:=plot([B],color=blue);
S:=seq(plots[display](B,P,plot([(1-t*0.01)*p1+t*0.01*p2,(1-t*0.01)*p2+t
    *0.01*p3],color=red)),t=0..100);

plots[display](S,insequence=true,axes=none,scaling=constrained);
end:
```

puis le résultat de son lancement :

```
bezier1([0,0],[3,2],[5,-2]);
```

Il s'agit d'une animation ! Il faut donc cliquer sur l'image et lancer l'animation à l'aide de l'icône appropriée.

2 2 Espace affine

Un espace affine c'est en gros une sorte d'espace vectoriel « pointé ». Si un point A et un vecteur u sont donnés, $B = A + u$ est un autre élément de l'espace affine. Plus clairement, dans le contexte du plan vectoriel vu au lycée, on écrit :

$$B = A + \overrightarrow{AB}$$

Cela nous permet de faire des calculs sur les points. D'ailleurs MAPLE calcule lui aussi de cette manière.
Ainsi $M_1(t) = (1-t)P_1 + tP_2$ (ou si vous préférez $\overrightarrow{OM_1}(t) = (1-t)\overrightarrow{OP_1} + t\overrightarrow{OP_2}$) et $M_2(t) = (1-t)P_2 + tP_3$ puis

$$M(t) = (1-t)M_1(t) + tM_2(t) = (1-t)^2P_1 + 2t(1-t)P_2 + t^2P_3$$

On peut dériver $M(t)$ par rapport à t (ou $\overrightarrow{OM}(t)$ par rapport à t) comme en physique (vous verrez ça de manière rigoureuse en maths plus tard) ce qui nous donnera le vecteur vitesse instantanée (c'est-à-dire un vecteur directeur de la tangente).
Vérifiez que $M'(t) = 2(M_2(t) - M_1(t))$. Comment l'interpréter ?

2 3 Avec 4 points de contrôle

Faites une étude similaire (« à la main ») avec 4 points de contrôle.
On pourra introduire les polynômes de Bernstein définis par :

$$B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j}$$

et utiliser une représentation similaire aux arbres de probabilité.
Créez une procédure **`bern:=proc(j,n,t)`** qui calcule $B_j^n(t)$.
Commentez le script suivant :

```
bezier3:=proc(p)
local Vi,Vf,B,k,j,M;
Vi:=plot([p[1],p[2]],color=green);
Vf:=plot([p[3],p[4]],color=green);
B:=p[1];
for k from 0.01 to 0.99 by 0.01 do
    M:=sum(bern(j,3,k)*p[j+1],j=0..3);
    B:=B,M;
od;
B:=plot([B,p[4]],color=blue);
plots[display](B,Vi,Vf,axes=none,scaling=constrained);
end:
```

et testez :

```
bezier3([[0,0],[1,0],[3,-3],[3,0]]);
```

Quel est le rôle des points de contrôle 2 et 3 ?

3 Avec un nombre quelconque de points de contrôle

Généralisez les procédures précédentes pour traiter un nombre quelconque de points de contrôle.
Testez avec $[[1,1],[2,3],[5,5],[6,2],[7,7],[10,5],[10,2],[8,0],[4,0],[5,1]]$.

4 Courbe de Bézier du 3e degré avec un nombre quelconque de points de contrôle

Il est pratique de travailler avec des polynômes de degré trois pour avoir droit à des points d'inflexion.
Augmenter le nombre de points de contrôle implique a priori une augmentation du degré de la fonction polynomiale.
Pour remédier à ce problème, on découpe une liste quelconque en liste de listes de 4 points.
Cependant, cela est insuffisant pour obtenir un raccordement de classe $\mathscr{C}^1$ (pourquoi ? pourquoi est-ce important d'avoir un raccordement de classe $\mathscr{C}^1$?)
Pour assurer la continuité tout court, il faut que le premier point d'un paquet soit le dernier du paquet précédent.
Le dernier « vecteur vitesse » de la liste $[P_1,P_2,P_3,P_4]$ est $\overrightarrow{P_3P_4}$. Il faut donc que ce soit le premier vecteur vitesse du paquet suivant pour assurer la continuité de la dérivée.
Appelons provisoirement le paquet suivant $[P'_1,P'_2,P'_3,P'_4]$. On a d'une part $P'_1 = P_4$ et d'autre part $\overrightarrow{P_3P_4} = \overrightarrow{P'_1P'_2}$, i.e. $P'_2 = P_4 + \overrightarrow{P_3P_4}$.
On a donc $P'_3 = P_5$ et $P'_4 = P_6$.
Connaissant **`bezier3`**, construire une procédure qui trace une courbe de Bézier cubique avec un nombre quelconque de points de contrôle (on prendra un nombre pair de points pour se simplifier la vie).
Testez avec $[[1,1],[2,3],[5,5],[6,2],[7,7],[10,5],[10,2],[8,0],[4,0],[5,1]]$.
Dessinez un cœur....

5 B-splines uniformes

5 1 Cas général

Tout ceci est très beau mais il y a un hic : en changeant un point de contrôle, on modifie grandement la figure.

On considère m nœuds $t_0, t_1, \cdots, t_m$ de l'intervalle $[0,1]$.
Introduisons une nouvelle fonction :

$$\mathbf{S}(t) = \sum_{i=0}^{m-n-1} \mathbf{P}_i b_{i,n}(t)\ ,\ t \in [0,1]$$

les P_i étant les points de contrôle et les fonctions $b_{j,n}$ étant définies récursivement par

$$b_{j,0}(t) = \begin{cases} 1 & \text{si} \quad t_j \leqslant t < t_{j+1} \\ 0 & \text{sinon} \end{cases}$$

et pour $n \geqslant 1$

$$b_{j,n}(t) = \frac{t - t_j}{t_{j+n} - t_j} b_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} b_{j+1,n-1}(t).$$

On ne considérera par la suite que des nœuds équidistants : ainsi on aura $t_k = \frac{k}{m}$.
On parle de B-splines uniformes et on peut simplifier la formule précédente en remarquant également des invariances par translation.

5 2 Fonctions B-splines d'ordre 3 avec 4 points de contrôle

À l'aide des formules précédentes, on peut prouver que dans le cas de 4 points de contrôles on obtient :

$$S(t) = \frac{1}{6}\Big((1-t)^3 P_0 + (3t^3 - 6t^2 + 4)P_1 + (-3t^3 + 3t^2 + 3t + 1)P_2 + t^3 P_3\Big)$$

Calculez $S(0)$, $S(1)$ puis $S'(0)$ et $S'(1)$: que peut-on en conclure ?
Reprenez l'étude faite avec les courbes de Bézier et comparez les résultats.

CHAPITRE 5

Tapis, arbres, brocolis et lapins

1 Polygones de Sierpinski

Waclaw SIERPINSKI (1882 - 1969) fut un mathématicien polonais qui travailla sur des domaines mathématiques très ardus : fondements des mathématiques, construction axiomatique des ensembles, hypothèse du continu, topologie, théorie des nombres...
Il a également travaillé sur les premiers objets fractals qu'étudiera plus tard Benoît MANDELBROT, mathématicien français d'origine polonaise connu pour ses travaux dans ce domaine.

1 1 Jouons aux dés

Prenez un dé à 6 faces, un triangle ABC et un point G quelconque à l'intérieur du triangle. Vous lancez le dé :

- si la face supérieure est 1 ou 2, vous faites une petite croix au niveau du milieu de G et de A ;
- si la face supérieure est 3 ou 4, vous faites une petite croix au niveau du milieu de G et de B ;
- si la face supérieure est 5 ou 6, vous faites une petite croix au niveau du milieu de G et de C ;

Ah, zut, on n'a pas de dé dans la salle d'info...mais on a Maple.

- `rand()` renvoie un entier aléatoirement choisi entre 0 et 999 999 999 999 ;
- `rand(n..p)` renvoie un entier aléatoirement choisi entre n et p inclus.

```
> dede:=proc(n)
  local P,r,t,d,G,L,k;
  P:=[[0,0],[1,0],[0.5,0.5*sqrt(3)]];
  r:=10.^(-12)*rand();
  t:=r();
  d:=rand(1..3);
  G:=t^2*P[1]+2*t*(1-t)*P[2]+(1-t)^2*P[3];

  ...

  end:
```

Expliquez le début de cette procédure, achevez-la et testez-la.

1 2 Tapis

Considérons les huit transformations de $\mathbb{C}$ dans $\mathbb{C}$ suivantes :

$$T_1(z)=\frac{1}{3}z \qquad T_2(z)=\frac{1}{3}z+\frac{1}{3} \qquad T_3(z)=\frac{1}{3}z+\frac{2}{3} \qquad T_4(z)=\frac{1}{3}z+\frac{2}{3}+\frac{1}{3}i$$

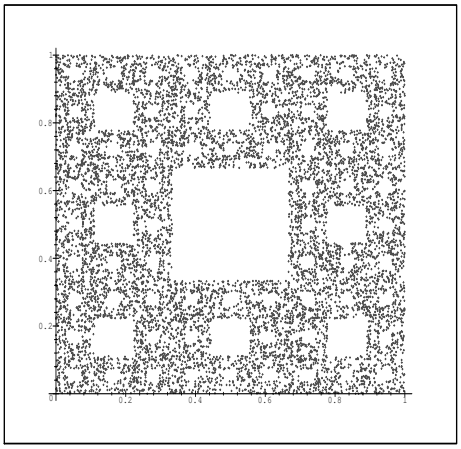
$$T_5(z)=\frac{1}{3}z+\frac{2}{3}+\frac{2}{3}i \qquad T_6(z)=\frac{1}{3}z+\frac{1}{3}+\frac{2}{3}i \qquad T_7(z)=\frac{1}{3}z+\frac{2}{3}i \qquad T_8(z)=\frac{1}{3}z+\frac{1}{3}i$$

Soit E_0 le carré unité.

- Dessinez E_0 ;
- Dessinez $E_1=T_1(E_0)\cup T_2(E_0)\cup\cdots\cup T_8(E_0)$;

– Identifiez les T_k.

Comment obtenir ce joli dessin avec MAPLE ?



Une possibilité est d'introduire une liste d'applications :

```
T:=[z->z/3,z->z/3+1/3,z->z/3+2/3,z->z/3+1/3+2*I/3,z->z/3+2/3+2*I/3,z->z
    /3+2/3+I/3,z->z/3+I/3,z->z/3+2*I/3];
```

13 Les mites de Sierpinski

Monsieur SIERPINSKI avait ramené d'un voyage en Orient un tapis carré de 1 mètre de côté dont il était très content. Jusqu'au jour où les mites s'introduisirent chez lui.

En 24 heures, elles dévorèrent dans le tapis un carré de côté trois fois plus petit, situé exactement au centre du tapis. En constatant les dégats, Monsieur SIERPINSKI entra dans une colère noire ! Puis il se consola en se disant qu'il lui restait huit petits carrés de tapis, chacun de la taille du carré disparu. Malheureusement, dans les 12 heures qui suivirent, les mites avaient attaqué les huit petits carrés restants : dans chacun, elles avaient mangé un carré central encore trois fois plus petit. Et dans les 6 heures suivantes elles grignotèrent encore le carré central de chacun des tout petits carrés restants. Et l'histoire se répéta, encore et encore ; à chaque étape, qui se déroulait dans un intervalle de temps deux fois plus petit que l'étape précédente, les mites faisaient des trous de taille trois fois plus petite...

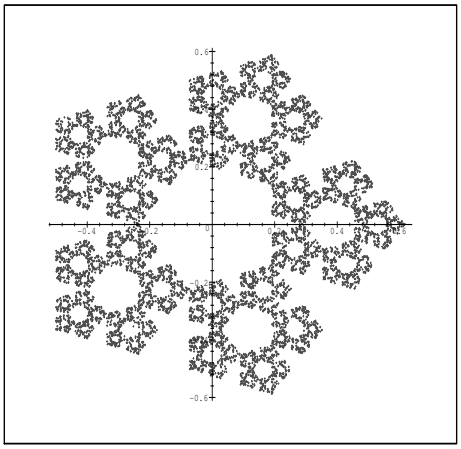
– Calculer le nombre total de trous dans le tapis de Monsieur SIERPINSKI après n étapes. Calculer la surface S_n de tapis qui n'a pas encore été mangée après n étapes. Trouver la limite de la suite $(S_n)_{n\geq 0}$. Que reste-t-il du tapis à la fin de l'histoire ?

– Calculer la durée totale du festin « mitique »...

Merci à Frédéric Le Roux

14 Dentelle de Varsovie

Nous voudrions obtenir le joli napperon en dentelle suivant :



Déterminez le rapport des similitudes qui transforment le grand pentagone en un des pentagones plus petit.

En vous inspirant de ce qui a été fait pour le triangle de SIERPINSKI, faites tracer à MAPLE ce joli napperon.

En fait on peut montrer que pour un nombre $c \geqslant 5$ de côtés, le rapport est de $\dfrac{1}{2\sum_{k=0}^{\left[\frac{c}{4}\right]} \cos\left(\frac{2k\pi}{c}\right)}$

2 Végétation récursive

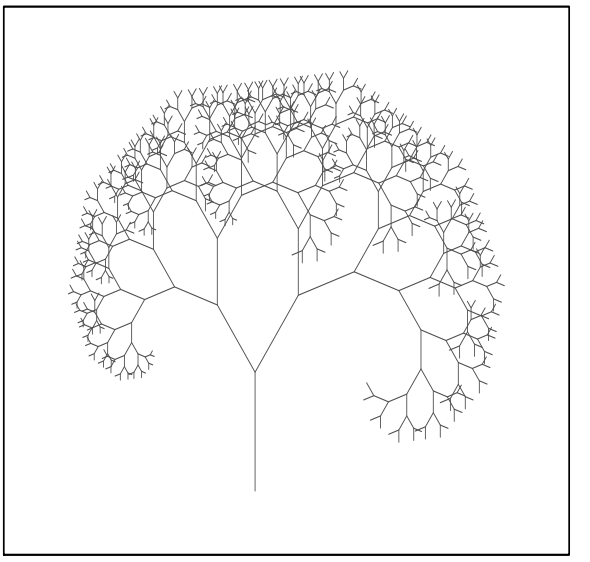
Analysez cet algorithme :

```
arbre:=proc(A,B,Rap,Ang,n)
   if n>0
     then [[Re(A),Im(A)],[Re(B),Im(B)]],seq(arbre(B,B+Rap[q]*exp(Ang[q
         ]*I)*(B-A),Rap,Ang,n-1),q=1..nops(Rap))
     else [[Re(A),Im(A)],[Re(B),Im(B)]]
   fi
end:
```

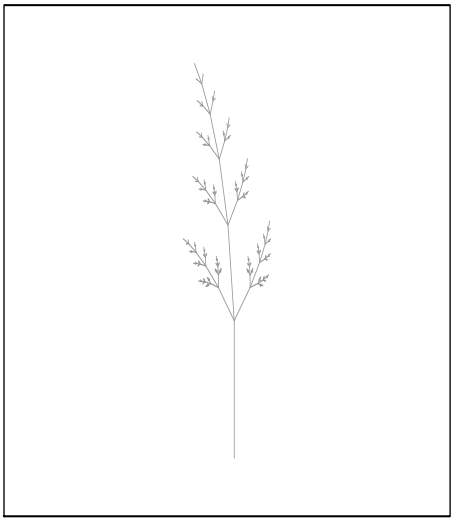
sachant que

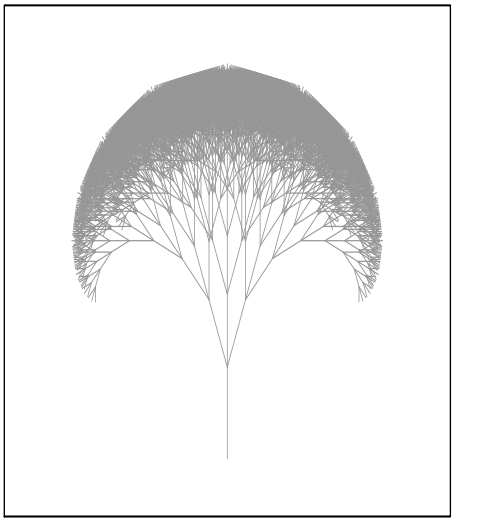
```
plot([arbre(0,I,[0.7,0.8],[Pi/5,-Pi/5],9)],axes=none,scaling=
    constrained,color=green);
```

donne

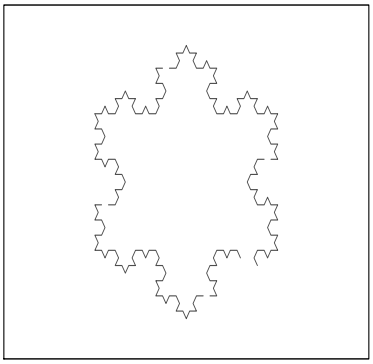


Comment obtenir cette fougère et ce brocolis ?





Ce n'est plus un arbre mais un beau flocon (le flocon de Helge VON KOCH(1870-1924)) : comment le générer avec MAPLE ?



3 Ensembles de Julia et de Mandelbrot

Gaston JULIA (1893 - 1978) est un des plus grands mathématiciens français. Ses travaux ont été largement utilisé par Benoît MANDELBROT (né en 1924) au début des années 1970. Nous allons survoler certains de leurs résultats les plus populaires.
Nous allons étudier les suites (z_n) définies par z_0 et $z_{n+1} = z_n^2 + c$ pour tout entier naturel n avec c un nombre complexe quelconque.
Nous avons déjà exploré une suite réelle semblable lors de notre incursion dans la dynamique des populations.
Le cas $c = 0$ est assez simple à étudier : faites-le !
Comment ce que vous avez obtenu est en lien avec cette procédure MAPLE et son utilisation ?

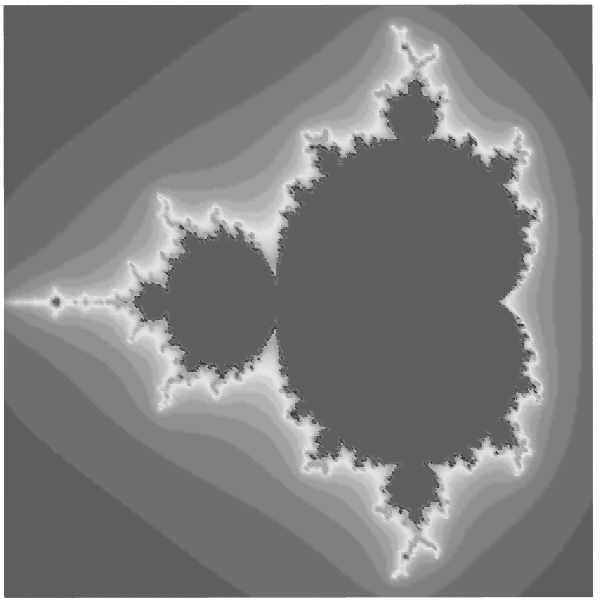
```
> julia:=proc(x,y)
    global c;
    local z,j;
    z:=evalf(x+I*y);
     for j from 0 to 50 while abs(z)<4 do
       z:=z^2+c
     od;
    j
  end:

> c:=0:

> plot3d(0,-2..2,-1.2..1.2,orientation=[-90,0],grid=[150,150],style=
    patchnogrid,scaling=constrained,color=julia);
```

Je vous dois une petite explication au sujet de la dernière instruction. Afin de gagner du temps, on utilise une petite ruse pour pallier aux carences de MAPLE en géométrie.
On utilise **plot3d** avec la fonction $f : (x, y) \mapsto 0$ afin de tracer le plan d'équation $z = 0$. Ses points sont colorés selon la procédure **julia**. L'option **grid** indique la taille de la grille tracée. On précise également qu'on veut un repère orthonormé, pas de maillage apparent et on place les axes de coordonnées dans l'orientation usuelle.
Testez différentes valeurs de c : -1, $0,32 + 0,043\text{i}$, $-0,122561 + 0,744862\text{i}$.
Quelles sont vos interprétations ?

Plutôt que de testez les valeurs de c une par une, Benoît MANDELBROT a cherché à représenter las valeurs de c qui faisaient converger la suite avec $z_0 = 0$: c'est l'ensemble de MANDELBROT.
Tracez-le !



CHAPITRE 6

Logistique dynamique

J'ai deux heures pour vous dire que MAPLE, c'est de la dynamique

1 Présentation du problème

Le mathématicien Belge Pierre-François VERHULST (28 octobre 1804 - 15 février 1849) proposa en 1838 un modèle d'évolution des populations animales qui porte son nom et qui rompt avec l'habituelle croissance exponentielle. On suppose qu'une population vaut p_n à un certain instant et qu'il existe une valeur d'équilibre e telle que la population tend à y revenir avec autant de force qu'elle s'en écarte. Il existe donc un coefficient positif k tel que :

$$\frac{p_{n+1}-p_n}{p_n} = -k(p_n - e)$$

Recherche

Déduisez-en qu'il existe une constante $\mathrm{R} \geqslant 1$ telle que :

$$p_{n+1} = \mathrm{R}p_n(1 - \frac{k}{\mathrm{R}}p_n)$$

puis qu'il existe une valeur maximum de la population $p_{\max}$ et que :

$$u_{n+1} = \mathrm{R}u_n(1-u_n)$$

en notant u_n le rapport $\frac{p_n}{p_{\max}}$.

Depuis VERHULST on désigne par *suite logistique* ce type de suite. Il faut bien sûr considérer la plus ancienne des définition du mot :

> **LOGISTIQUE** n.f. **1.** (1611) Anc. nom de la partie de l'algèbre qui traite des quatre règles.

car nous n'utiliserons que les quatre opérations arithmétiques de base mais n'étudierons pas les problème de ravitaillement des armées.

Recherche

Dans quel intervalle varie R ?

Étudier la suite (u_n), c'est étudier le *système dynamique* défini par la fonction f. L'ensemble des x, $f(x)$, $f(f(x))$,...,$f^n(x)$,... est appelé l'*orbite* de x.

2 Exploration au petit bonheur

Peut-on se contenter d'observer le comportement d'une suite ? L'outil informatique permet-il de se passer d'une exploration théorique ?
Commençons donc par explorer au petit bonheur le comportement de ces suites à l'aide de Maple. Il faudrait tout d'abord nous construire un petit outil qui affiche les fameux escargots et autres escaliers à l'aide de la première bissectrice. Ce programme sera très utile pour l'oral de Centrale...

```
> escargot:=proc(R,uo,n)
  local u,fu,k,e,g,c;
  u:=uo;
  e:=NULL;
```

```
for k from 1 to n do
 fu:=R*u*(1-u);
 e:=e,[u,u],[u,fu];
 u:=fu;
od;
e:=plot([e],color=pink,title=cat('R=',convert(R,string), ' et uo=',
    convert(uo,string)));
g:=plot(t,t=0..1,color=green);
c:=plot(R*t*(1-t),t=0..1,color=blue);
plots[display]({e,g,c});
end:
```

Recherche

Que fait `escargot` ?
Commentez alors les animations suivantes :

```
> A:=seq(escargot(0.01*R,0.2,50),R=100..400):
> plots[display](A,insequence=true);
> B:=seq(escargot(4,0.01*uo,50),uo=1..99):
> plots[display](B,insequence=true);
```

3 Convergence et points fixes

Vous connaissez parfaitement votre cours sur le rapport entre la convergence des suites définies par une relation $u_{n+1} = f(u_n)$ et les points fixes de f.

Malheureusement, on ne parle pas assez du caractère attractif ou répulsif des points fixes.

À l'aide du théorème des accroissements finis, on peut montrer que, ℓ étant un point fixe de f :

- si $|f'(\ell)| < 1$, le point fixe est attractif et (u_n) converge vers ℓ avec $|u_n - \ell| \leqslant k^n |u_0 - \ell|$ pour un certain $k \in]0;1[$;
- si $f'(\ell) = 0$, le point est dit *super attractif* et la suite converge très rapidement ;
- si $|f'(\ell)| > 1$, le point fixe est répulsif et (u_n) ne converge que si elle est constante et égale à ℓ à partir d'un certain rang ;
- si $|f'(\ell)| = 1$, le cas est litigieux.

Recherche

Mais au fait, ces points fixes, quels sont-ils ?
Discuter selon les valeurs de R et commencer à expliquer grossièrement certains comportements observés. Étudier en particulier « l'attractivité » des points fixes.

4 Étude de la convergence dans le cas 1<R<3

4 1 Cas 1<R<=2

Recherche

Traiter le cas $R = 2$.
Que peut-on dire du point fixe non nul dans les autres cas ? Dans quel intervalle se trouve-t-il ?
Discuter alors selon la position de u_0 par rapport à ce point fixe et à $\frac{1}{R}$

4 2 Cas 2<R<3

Les dessins obtenus avec Maple peuvent donner des idées.

Recherche

Montrer que $I = \left[\frac{1}{2}; \frac{R}{4}\right]$ est stable par f.
En déduire que si $u_0 \in I$, alors (u_{2n}) puis (u_n) converge vers le point fixe non nul.
Étudier ensuite le cas $u_0 \in \left]0; \frac{1}{2}\right[$ puis le cas $u_0 \in \left]\frac{R}{4}; 1\right[$ et montrer qu'on peut se ramener aux cas précédents.

5 Théorème de Coppel et conséquences

On note $f^n = f \circ f \circ f \circ \cdots \circ f$.
Soit $f : I \to I$ une fonction continue. Soit $x \in I$. Si x est un point fixe de f^n mais n'est pas un point fixe de f^k pour tout $0 < k < n$, on dit que x est ***n*-période**.
L'ensemble $\{x, f(x), \cdots, f^{n-1}(x)\}$ est un ***n*-cycle** pour f.
Soit p_0, p_1,...,p_{n-1} un n-cycle On dit qu'il est **attractif** si $\left|(f^n)'(p_i)\right| < 1$ pour tout entier naturel i strictement inférieur à n. Dans ce cas, si u_0 est suffisamment proche de l'un des éléments du cycle, la suite admet les éléments du cycle comme valeurs d'adhérence.

Théorème 6 - 1

Théorème de Coppel
Soit $f : [a,b] \to [a,b]$ une fonction continue. Si f n'admet pas de 2-cycle, alors, pour tout $u_0 \in I$, la suite définie par $u_{n+1} = f(u_n)$ converge.

La démonstration est assez longue et nous ne nous en occuperons pas aujourd'hui.
Nous sommes pourtant en mesure de démontrer l'utile lemme suivant :

Lemme 6 - 1

Soit $f : [a,b] \to [a,b]$ une fonction de classe $\mathscr{C}^1$. Si f admet deux points fixes répulsifs consécutifs distincts α et β avec $\alpha < \beta$, alors il existe $p_2 \in]\alpha,\beta[$ tel que p_2 soit un point fixe de f^2.

Pour démontrer ce lemme, il suffit d'étudier la fonction définie sur $[a,b]$ par

$$g(x) = f^2(x) - x$$

en précisant en particulier le signe de $g'(\alpha)$ et $g'(\beta)$.

Recherche

Démontrer le lemme.

5 1 Cas R=3

C'est un cas ambigu car dans ce cas $\left|f'(p)\right| = |2-3| = 1$.
Cependant, on peut utiliser le théorème de COPPEL pour montrer que la suite converge.

Recherche

Avec l'aide éventuelle du `solve` de Maple, démontrer la convergence de la suite (u_n) dans le cas R = 3.

5 2 Cas R>3

C'est maintenant que ça devient amusant...

5 2 a Cycles d'ordre 2

Les points fixes sont maintenant répulsifs : que peut-on en déduire ?
Utiliser éventuellement Maple pour déterminer les éventuels points fixes de f^2.
Que se passe-t-il lorsque R tend vers 3 par valeurs supérieures ?
Qu'illustre le script suivant :

Recherche

```
> IR:=[seq(0.01*k,k=100..400)]:
> C:=seq(plot([f(t),f(f(t)),t],t=0..1,color=[red,blue,pink],title=
    cat('R=',convert(R,string))),R=IR):
> plots[display](C,insequence=true);
```

5 2 b Attractivité des cycles d'ordre 2

Soit p_2 un point fixe de f^2 autre que 0 et p. Il faudrait maintenant savoir quand est-ce que le 2-cycle $(p_2, f(p_2))$ est attractif.

Montrer que $(f^2)'(p_2) = -p_2^2 + 2p_2 + 4$. Pour cela, montrer que $(f^2)'(p_2) = (f)'(p_2) \cdot f'(f(p_2))$ et utiliser le fait que p_2 et $f(p_2)$ sont des racines d'un polynôme du second degré introduit au paragraphe précédent.
Commenter alors les deux graphiques définis par :

Recherche

```
> escargot(evalf(0.99+sqrt(6)),0.5,200);
> escargot(evalf(1.01+sqrt(6)),0.5,200);
```

5 2 c Cycles d'ordre 4

Le 2-cycle cesse d'être attractif pour $R > 1 + \sqrt{6}$. D'après le théorème de COPPEL, on en déduit que f^2 admet un 2-cycle, c'est-à-dire que f admet un 4-cycle : jusqu'à quelle valeur de R ?
C'est plus difficile à déterminer. On peut regarder graphiquement, par affinements successifs :

```
> IR:=[seq(0.0001*k,k=35440..35450)]:
> G:=seq(plot([(f@@4)(t)-t,(f@@8)(t)-t],t=0.52..0.526,numpoints=1000,
    color=[pink,purple],title=cat('R=',convert(R,string))),R=IR):
> plots[display](G,insequence=true,view
    =[0.52..0.526,-0.00001..0.00001]);
```

Recherche

Quel renseignement nous donne ce graphique ?

6 Théorème de Feigenbaum

6 1 Diagramme de Feigenbaum

```
> Feig:=proc(r1,r2,pas)
  local k,r,tmp,ligne,res,t,etendue;
  etendue:=floor((r2-r1)/pas)+1;
  res:=[0$etendue]:
  ligne:=[0$20]:
  tmp:=0.2:
  for r from r1 to r2 by pas do
    for k from 1 to 100 do
      tmp:=r*tmp*(1-tmp);
    od;
    for k from 1 to 20 do
      ligne:=subsop(k=[r,tmp],ligne);
      tmp:=r*tmp*(1-tmp);
    od:
    t:=floor((r-r1)/pas)+1;
    res:=subsop(t=ligne,res):
  od:
  plot(res,x=r1..r2,style=point,color=black);
  end:
```

Recherche

Interpréter le dessin obtenu.

6 2 Le théorème

On doit au physicien Mitchell FEIGENBAUM des conjectures qui furent prouvées par la suite par des mathématiciens, des vrais, mais c'est le nom du physicien qui reste attaché aux bifurcations. Voici un condensé des résultats proposé par Daniel PERRIN

Théorème 6 - 2

Soit R_n la borne inférieure des R tel que f admette un cycle d'ordre 2^n.

- on a $R_0 = 0$, $R_1 = 3$, $R_2 = 1+\sqrt{6}$, $R_3 \approx 0,544090$, $R_4 \approx 3,564407$;
- la suite (R_n) est strictement croissante, majorée par 4. Elle converge vers un réel $R_\infty \approx 3,5699456$;
- Pour $R_n < R < R_\infty$, f_R admet un unique 2^n-cycle qui est attractif tant que R est strictement inférieur à R_{n+1} ;
- pour $R < R_\infty$, f_R n'a pas de cycle d'ordre p si p n'est pas une puissance de 2.

Voici qui confirme nos intuitions.

7 Exposant de Lyapounov

Pour mesurer la sensibilité d'un système dynamique aux conditions initiales, on mesure l'exposant de LYAPOUNOV introduit par le mathématicien russe Alexandre LYAPOUNOV à la fin du XIX^{e} siècle.

On considère une suite définie par la relation $u_{n+1} = f(u_n)$. Quelle est l'influence d'un écart e_0 sur u_0 pour la suite des itérés ?

Après une itération, l'écart absolu vérifie $|e_1| = |f(u_0 + e_0) - f(u_0)|$ et l'écart relatif vaut $\frac{|e_1|}{|e_0|} = \frac{|f(u_0+e_0)-f(u_0)|}{|e_0|} \approx f'(u_0)$ pour $|e_0|$ suffisamment petit.

Après n itérations, l'écart relatif vaut

$$\frac{|e_n|}{|e_0|} = \frac{|e_1|}{|e_0|}\frac{|e_2|}{|e_1|}\cdots\frac{|e_n|}{|e_{n-1}|} = \prod_{k=1}^{n} f'(u_{k-1})$$

Si un des écarts devient nul, notre étude est sans intérêt. Comme effectuer un produit est une opération coûteuse, informatiquement parlant, nous allons pouvoir considérer le logarithme de ce produit.

Notre problème est de savoir si les écarts s'amplifient et donc le produit est supérieur à 1, ou bien si le système est stable et donc le produit est inférieur à 1.

$$\prod_{k=1}^{n} f'(u_{k-1}) < 1 \Longleftrightarrow \sum_{k=1}^{n} \ln\left(f'(u_{k-1})\right) < 0$$

Pour relativiser le rôle du choix de n, nous allons normer cette somme en la divisant par n.

On définit alors l'exposant de LYAPOUNOV :

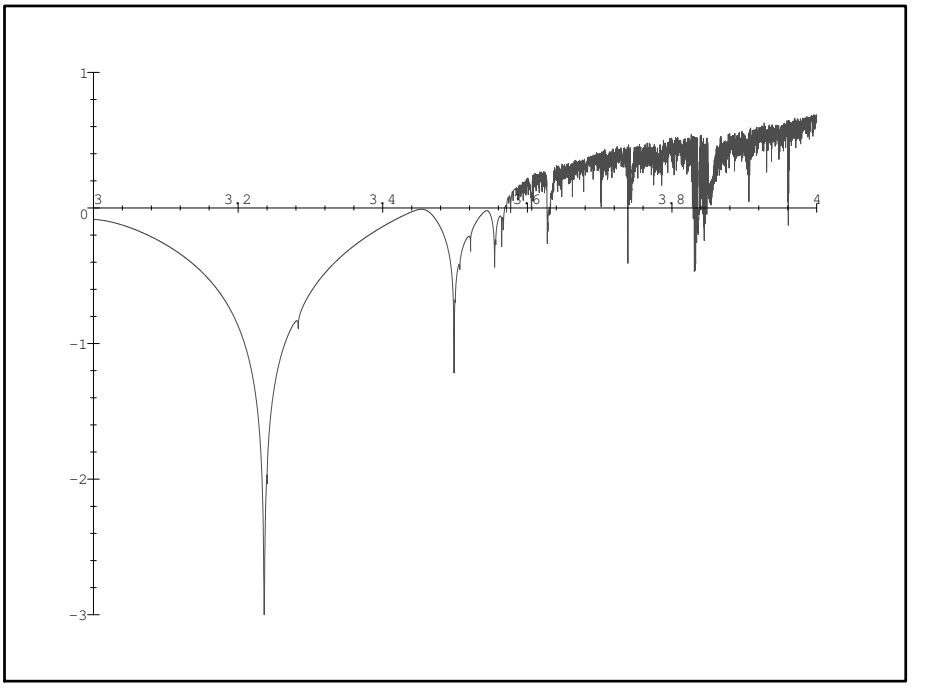
$$\lambda = \lim_{n\to+\infty} \frac{1}{n} \sum_{k=1}^{n} \ln\left(f'(u_{k-1})\right)$$

Recherche

Écrire une procédure **`lyapounov:=proc(u0,R,n)`** qui calcule une approximation numérique de l'exposant de Lyapounov qui dépend de la donnée de u_0, R et le nombre n d'itérations.
Ensuite, l'utiliser pour représenter l'exposant en fonction de R.

Par exemple, avec $u_0 = 0,45$ et 50 itérations :

```
plot([seq([r*0.001,lyapounov(0.45,r*0.001,50)],r=3000..4000)],view
    =[-3..4,-3..1]);
```



Recherche Comment interpréter ce graphique ?

8 Prolongements

Pour savoir ce qui se passe au-delà de R_∞, étudiez ce merveilleux document :
http ://www.math.u-psud.fr/ perrin/Conferences/logistiqueDP.pdf
écrit en 2008 par Daniel PERRIN de l'Université d'Orsay. Dans la note de première page, vous retrouverez le nom de deux professeurs bien connus...