MP* 2012-2013

Une année de MAPLE et SCILAB en MP*

Guillaume CONNAN

SOMMAIRE

TP N°

1

X 2012

Un peu d'algo pour SI et INFO sur un sujet pas très bien posé mais qui permet un bon survol de la programmtion MAPLE.

« Primitives »

En informatique, on désigne par primitive d'un langage un objet qui n'est pas construit à partir du langage lui-même. Par exemple en CAML, il existe des primitives écrites en C.
Ici, nous allons construire les fonctions évoquées dans le préambule du sujet pour nous « remettre dans le bain » onctueux de MAPLE...

allouer

La fonction `allouer(m,c)` renvoie un tableau de taille *m* rempli de *c*. Comme il s'agit de tableaux d'une ligne, nous allons plutôt nous tourner sur des objets de type *liste*, facilement manipulables en MAPLE, mais il faut bien faire la différence entre les listes qui sont définies dynamiquement et les tableaux qui sont définis statiquement. Le jour du concours, le jury accepte que vous employiez des listes au lieu de tableaux `array` si vous le précisez et le motivez.

```
> allouer := proc(m::nonnegint,c)
    RETURN([c$m])
  end:
```

Il est optionnel de préciser le type des arguments des fonctions mais cela sert de garde-fou. Notez l'utilisation du `$`

taille

Il existe une fonction qui compte le nombre d'opérandes d'une liste : `nops`.

Partie entière

Il existe deux fonctions relatives à la partie entière d'un réel : `floor` et `ceil`. Votre niveau d'anglais vous permettra de comprendre ce qu'elles représentent...

Exemple

Nous utiliserons le tableau du sujet comme exemple de référence pour nos tests :

```
> tab11 := [3,2,5,8,1,34,21,6,9,14,8]:
```

Question 1

Exceptionnellement, je vous donne une proposition pour vous rappeler quelques éléments de syntaxe MAPLE sur les fonctions :

```
> calculeIndicMaximum := proc(tab::list,a,b::nonnegint)
   local m,i,j;
   i:=a;
   m:=tab[a];
   for j from a+1 to b do
     if tab[j] > m then
        i:=j;
        m:=tab[j];
     fi;
   od;
   RETURN(i);
  end:
```

Question 3

C'est là que ça commence à être pénible... La supposition entre parenthèses ne correspond pas à une expérimentation. En tant que mathématicien(ne)s, nous sommes plus à l'aise avec la manipulation de fonctions et non pas de procédures avec des variables globales et des changements en place. Nous allons donc créer une fonction :
`partition := proc(tab,a,b,indicePivot)` qui va renvoyer le nouvel indice évoqué dans l'énoncé ainsi que la liste réordonnée, en deuxième opérande.

Par exemple, on obtiendra :

```
> partition(tab11,1,11,4);
              6, [3, 2, 5, 1, 6, 8, 8, 34, 21, 9, 14]
```

Le sujet n'évoque pas le traitement des erreurs. Montrons cependant comment cela peut être traité avec MAPLE :

```
if (indicePivot < a) then ERROR('indice trop petit')
```

Question 4

L'algorithme est donné dans le texte mais de manière étrange. Notez son côté récursif. Transformez-le un peu pour l'adapter à notre fonction `partition`.
Vous créerez également une fonction `median(tab)` qui renvoie la médiane d'un tableau.

Danger

Il y a un problème dans l'énoncé, au niveau de la huitième ligne de la page 3...

```
> median(tab11[1..11]);
                                8
```

Question 5

Combien de comparaisons effectue-t-on avec la fonction `partition` sur un tableau de taille n ?

Question 6

On va décomposer le problème. Créez d'abord une fonction `decoupe5 := proc(tab)` qui découpe une liste en paquets de 5 en s'arrangeant avec les morceaux restant...

```
> decoupe5(tab11[2..9]);
                   [[2, 5, 8, 1, 34], [21, 6, 9]]
> decoupe5(tab11[2..4]);
                            [[2, 5, 8]]
```

Pour la fonction `choixPivot`, elle encore récursive, on utilisera la fonction `median` préalablement mise au point.

ÉCOLE POLYTECHNIQUE – ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET DE CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2012 **FILIÈRE MP HORS SPÉCIALITÉ INFO**
FILIÈRE PC

COMPOSITION D'INFORMATIQUE – B – (XEC)

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Sandwich au jambon

Le problème dit du *sandwich au jambon* ou bien encore appelé théorème de *Stone-Tukey* s'énonce de la manière suivante : un ensemble de n points en dimension d peut toujours être séparé en deux parties de cardinal au plus $\lfloor n/2 \rfloor$ par un hyperplan de dimension $d-1$ (certains points peuvent être dans l'hyperplan), où $\lfloor n/2 \rfloor$ désigne la partie entière de $n/2$. De manière concrète, un ensemble de points dans l'espace peut être séparé en deux parties quasi-égales par un plan. De même un ensemble de points dans le plan peut être séparé en deux par une droite et même en 4 à l'aide de deux droites. Ce sujet porte sur la résolution algorithmique de ce problème et de problèmes connexes selon différentes méthodes.

Dans tout le problème, les tableaux sont indicés à partir de 1. L'accès à la i-ème case d'un tableau *tab* est noté $tab[i]$. Quel que soit le langage utilisé, on suppose que les tableaux peuvent être passés comme arguments des fonctions. En outre, il existe une primitive $\mathsf{allouer}(m, c)$ pour créer un tableau de taille m dont chaque case contient c à l'origine, ainsi qu'une primitive $\mathsf{taille}(t)$ qui renvoie la taille d'un tableau t. Enfin, on disposera d'une fonction `floor`(x) qui renvoie la partie entière $\lfloor x \rfloor$ pour tout réel $x \geq 0$.

La complexité, ou le temps d'exécution, d'un programme P (fonction ou procédure) est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc...) nécessaires à l'exécution de P. Lorsque cette complexité dépend d'un paramètre n, on dira que P a une complexité en $\mathcal{O}(f(n))$, s'il existe $K > 0$ tel que la complexité de P est au plus $Kf(n)$, pour tout n. Lorsqu'il est demandé de garantir une certaine complexité, le candidat devra justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Partie I. Grand, petit et médian

Dans cette partie, nous supposerons donné un tableau `tab` contenant n nombres réels. Les indices du tableau vont de 1 à n. Nous dénoterons par `tab[a..b]` le tableau pris entre les indices a et b c'est-à-dire les cellules $tab[a], tab[a+1], \ldots, tab[b-1], tab[b]$. Nous supposerons dans l'énoncé que $a \leq b$.

Nous utiliserons le tableau de taille 11 suivant pour nos exemples :

3	2	5	8	1	34	21	6	9	14	8

Question 1 Écrire une fonction `calculeIndiceMaximum`(tab, a, b) qui renvoie l'indice d'une case où se trouve le plus grand réel de $tab[a..b]$. Sur le tableau précédent avec $a = 1$ et $b = 11$, la fonction renverra 6 car la case 6 contient la valeur 34.

Question 2 Écrire une fonction `nombrePlusPetit`(tab, a, b, val) qui renvoie le nombre d'éléments dans le tableau `tab[a..b]` dont la valeur est plus petite ou égale à val. Sur le tableau exemple, pour une valeur de val égale à 5, et $a = 1, b = 11$, la fonction devra renvoyer la valeur 4 car seuls les nombres $3, 2, 5, 1$ sont inférieurs ou égaux à 5.

Nous allons maintenant calculer un médian d'un tableau. Rappelons qu'une valeur médiane m d'un ensemble E de nombres est un élément de E tel que les deux ensembles $E_{<m}$ (les nombres de E strictement plus petits que m) et $E_{>m}$ (les nombres de E strictement plus grands que m) vérifient $|E_{<m}| \leq \lfloor n/2 \rfloor$ et $|E_{>m}| \leq \lfloor n/2 \rfloor$. Notez que le problème du médian est une reformulation de problème dit du *sandwich au jambon* pris en dimension 1. Une méthode naïve consiste donc à parcourir les éléments de l'ensemble et à calculer pour chacun d'eux les valeurs de $|E_{<m}|$ et $|E_{>m}|$ mais il est possible de faire mieux comme nous allons le voir dans la partie suivante.

Partie II. Un tri pour accélérer

Une méthode plus efficace serait de trier le tableau par ordre croissant tout en prenant la cellule du milieu dans le tableau trié. Cette méthode certes rapide requiert $\mathcal{O}(n \ln n)$ opérations. Il existe une méthode optimale en temps linéaire $\mathcal{O}(n)$ pour trouver le médian d'un ensemble de n éléments. Cette partie a pour but d'en proposer une implémentation.

Une fonction annexe nécessaire pour cet algorithme consiste à savoir séparer en deux un ensemble de valeurs. Soit un tableau tab et un réel appelé pivot $p = tab[i]$, il s'agit de réordonner les éléments du tableau en mettant en premier les éléments strictement plus petits que le pivot $tab_{<p}$, puis les éléments égaux au pivot p, et en dernier les éléments strictement plus grands $tab_{>p}$. Sur le tableau exemple, en prenant comme valeur de pivot 8 on obtiendra le tableau résultat suivant :

3, 2, 5, 1, 6	**8, 8**	21, 34, 9, 14

Notez que dans le résultat les nombres plus petits que le pivot $3, 2, 5, 1, 6$ peuvent être dans n'importe quel ordre les uns par rapport aux autres.

Question 3 Écrire une fonction `partition`$(tab, a, b, indicePivot)$ qui prend en paramètre un tableau d'entiers $tab[a..b]$ ainsi qu'un entier $a \leq indicePivot \leq b$. Soit $p = tab[indicePivot]$. La fonction devra réordonner les éléments de $tab[a..b]$ comme expliqué précédemment en prenant comme pivot le nombre p. La fonction retournera le nouvel indice de la case ou se trouve la valeur p.
Dans cette question, on suppose que les modifications effectuées par la fonction sur le tableau tab sont persistantes, même après l'appel de la fonction.

Remarquons que le $\lfloor n/2 \rfloor$-ème élément dans l'ordre croissant d'un tableau de taille n est un élément médian du tableau considéré. Nous allons donc non pas programmer une méthode pour trouver le médian mais plus généralement pour trouver le k-ème élément d'un ensemble. Nous allons utiliser l'algorithme suivant :

On cherche le k-ème élément du tableau $tab[a..b]$.
- Si $k = 1$ et $a = b$ alors renvoyer $tab[a]$
- Sinon, soit $p = tab[a]$. Partitionner le tableau $tab[a..b]$ en utilisant le pivot p en mettant en premier les éléments plus petits que p. Soit i l'indice de p dans le tableau résultant.
  - Si $i - a + 1 > k$ chercher le k-ème élément dans $tab[a..i-1]$ et renvoyer cet élément.
  - Si $i - a + 1 = k$ renvoyer le pivot.
  - Si $i - a + 1 < k$ chercher le $(k - i + a - 1)$-ème élément dans $tab[i+1..b]$ et renvoyer cet élément.

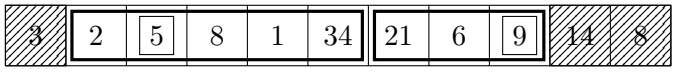
Question 4 Écrire une fonction `elementK`(tab, a, b, k) qui réalise l'algorithme de sélection du k-ème élément dans le tableau $tab[a..b]$ décrit précédemment et renvoie cet élément.

Question 5 Supposons que dans l'algorithme précédent nous voulions rechercher le premier élément mais qu'à chaque étape le pivot choisi est le plus grand élément, quel est un ordre de grandeur du nombre d'opérations réalisées par votre fonction ?

L'algorithme précédent ne semble donc pas améliorer le calcul du médian. Le problème vient du fait que le pivot choisi peut être mauvais c'est-à-dire qu'à chaque étape un seul élément du tableau a été éliminé. En fait, si l'on peut choisir un pivot p dans $tab[a..b]$ tel qu'il y ait au moins $(b-a)/5$ éléments plus petits et $(b-a)/5$ plus grands alors on peut montrer que l'algorithme précédent fonctionne optimalement en temps $\mathcal{O}(n)$.

Pour choisir un tel élément dans $tab[a..b]$, on réalise l'algorithme `choixPivot` suivant où chaque étape sera illustrée en utilisant le tableau donné en introduction en prenant $a = 2$ et $b = 9$.

- On découpe le tableau en paquets de 5 éléments plus éventuellement un paquet plus petit. On calcule l'élément médian de chaque paquet.



  S'il n'y a qu'un paquet on renvoie son médian. Sinon on place ces éléments médians au début du tableau.

– On réalise `choixPivot` sur les médians précédents. Dans notre exemple on recommence donc les étapes précédentes en prenant $a = 2$ et $b = 3$.

3	5	9	2	8	1	34	21	6	14	8

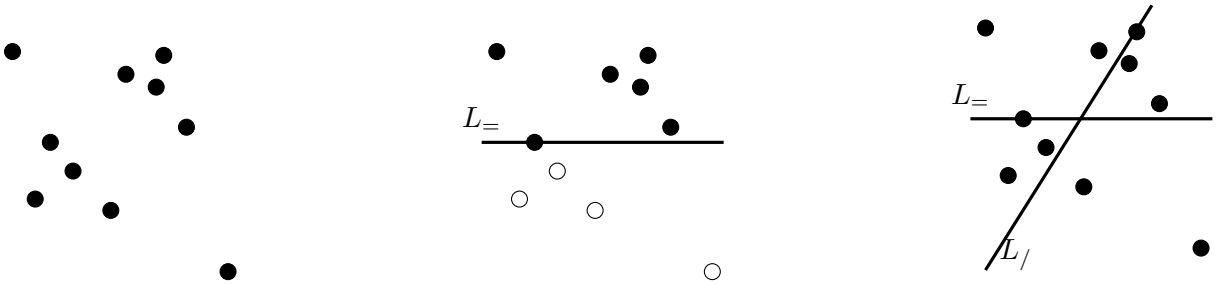
Question 6 Écrire la fonction `choixPivot`(tab, a, b) qui réalise l'algorithme précédent et renvoie la valeur du pivot.

Partie III. De la 1D vers la 2D, des nombres aux points.

Pour un réel $x \geq 0$, on note dans cette partie $\lceil x \rceil$ la partie entière supérieure de x, c'est-à-dire, le plus petit entier qui est plus grand ou égal à x : $\lceil x \rceil - 1 < x \leq \lceil x \rceil$. On supposera disposer d'une fonction `ceil`(x) qui renvoie la partie entière supérieure $\lceil x \rceil$ pour tout réel $x \geq 0$.

Dans la partie précédente, nous avons étudié le problème du médian en dimension 1. On supposera donc que vous disposez maintenant d'une fonction `indiceMedian`(tab, a, b) qui calcule un élément médian du tableau $tab[a..b]$ et renvoie l'indice de cet élément. Vous supposerez de plus que cette fonction ne modifie pas l'ordre des éléments du tableau tab.

Dans cette partie, nous généralisons l'algorithme de manière à trouver deux droites dans le plan séparant un ensemble de n points en 4 parties de cardinal plus petit que $\lceil n/4 \rceil$, Soit E un ensemble de n points tel qu'il n'existe pas trois points alignés. Nous allons chercher deux lignes $L_=$ et $L_/$ séparant cet ensemble de points en 4 parties comme le montre la troisième figure ci-dessous. En effet, dans cette figure chaque partie est composée d'exactement 2 points, les points situés sur les lignes n'étant pas pris en compte. Nous supposerons donnés dans cette partie deux tableaux $tabX$ et $tabY$ de taille n contenant les coordonnées des n points. Ainsi le point i a comme abscisse $tabX[i]$ et comme ordonnée $tabY[i]$.

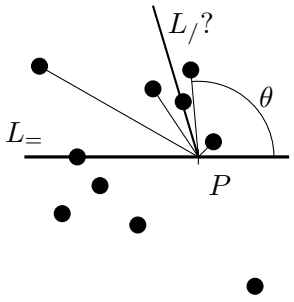


La première étape est de séparer les points en deux ensembles de même cardinal. Il suffit de remarquer que l'on peut toujours effectuer cette séparation par une ligne horizontale notée $L_=$ passant par un point d'ordonnée médiane comme le montre le second schéma ci-dessus.

Question 7 Écrire une fonction `coupeY`$(tabX, tabY)$ qui renvoie l'ordonnée d'une ligne horizontale séparant les points en deux parties de cardinal au plus $\lfloor n/2 \rfloor$.

La seconde ligne $L_/$ est plus difficile à trouver. Nous allons en réalité trouver le point d'intersection des deux lignes $L_=$ et $L_/$.

Soit P un point sur la droite horizontale $L_=$ précédente de coordonnées (x, y). On veut vérifier si ce point peut être le point d'intersection des deux lignes $L_=$ et $L_/$. Nous allons trouver dans un premier temps l'angle entre $L_=$ et $L_/$ en utilisant le fait que $L_/$ doit séparer en deux parties de cardinal proche les points au dessus de $L_=$. Ensuite nous allons vérifier si la droite $L_/$ ainsi devinée sépare les points en dessous de $L_=$ en deux parties presque égales.



Concrètement on considère les demi-droites partant de $P = (x, y)$ et joignant les k points de E au dessus strictement de $L_=$ comme indiqué sur le schéma ci-dessus. On calcule ensuite les angles θ entre $L_=$ et ces demi-droites. Remarquez alors que toute demi-droite d'angle médian partage en deux parties de cardinal $\leq \lfloor k/2 \rfloor$ les points au dessus de $L_=$.

Nous supposerons donnée une fonction `angle`$(x, y, x2, y2)$ qui calcule et renvoie l'angle formé par une demi-droite horizontale partant de (x, y) et allant vers la droite et le segment $(x, y) - (x2, y2)$. La valeur retournée sera comprise dans l'intervalle $[0, 2\pi[$.

Question 8 Écrire une fonction `demiDroiteMedianeSup`$(tabX, tabY, x, y)$ qui calcule et renvoie un angle médian entre $L_=$ et les segments reliant $P = (x, y)$ avec les points de E dont l'ordonnée est supérieure à y.

Pour un point P donné, nous avons déterminé l'angle que doit prendre $L_/$ pour couper les points au dessus de $L_=$ en 2 parties de taille au plus moitié. Il faut maintenant vérifier que cette droite $L_/$ coupe aussi les points en dessous de $L_=$ en 2 parties quasi-égales. Il suffit de vérifier que l'angle de $L_/$ partitionne les angles formés entre P et les ℓ points en dessous de $L_=$ en deux parties de cardinal $\leq \lceil \ell/2 \rceil$.

Question 9 Écrire une fonction `verifieAngleSecondeDroite`$(tabX, tabY, x, y, theta)$ qui calcule les ℓ angles formés entre $L_=$ et les points strictement au dessous de $L_=$. Votre fonction devra renvoyer :

- $\rightarrow$ 0 si theta est une médiane des ℓ angles.
- $\rightarrow$ -1 si le nombre d'angles strictement plus petits que theta est $> \lceil \ell/2 \rceil$
- $\rightarrow$ 1 si le nombre d'angles strictement plus grands que theta est $> \lceil \ell/2 \rceil$

Si $xmin$ est l'abscisse minimale des points de E et $xmax$ l'abscisse maximale, alors il est évident que l'intersection entre $L_=$ et $L_/$ ait une abscisse entre $xmin$ et $xmax$. Nous allons donc rechercher cette abscisse en utilisant l'algorithme suivant :

1. Trouver $L_=$ d'ordonnée y.
2. Soit $\alpha = xmin$ et $\beta = xmax$.
3. On calcule P le point au milieu de (α, y) et (β, y).
4. On calcule l'angle possible de la droite $L_/$ par la fonction `demiDroiteMedianeSup`.
5. Soit d la valeur donnée par la fonction $verifieAngleSecondeDroite$ avec l'angle trouvé précédemment. Si $d = 0$ alors on a trouvé $L_/$. Si $d = -1$ on recommence à partir du point 3 en prenant l'abscisse de P à la place de β. Si $d = 1$ on recommence mais en prenant l'abscisse de P à la place de α.

Question 10 Écrire la fonction `secondeMediane`$(tabX, tabY, y)$ qui à partir d'un ensemble E de points donnés par leurs coordonnées et de l'ordonnée de la droite $L_=$ calcule et renvoie un tableau contenant dans la première case l'abscisse x du point d'intersection de $L_=$ et de $L_/$ ainsi que l'angle de $L_/$ dans la seconde case.

* *
*

TP N°

2

Scilab for dummies

Scilab est un logiciel libre, multiplateforme de calcul numérique orienté informatique industrielle et ingénierie. Certains d'entre vous l'utiliseront peut-être dans leurs TIPE...

Nous l'utiliserons pour notre part en analyse numérique et en probabilités.

1 Avant de commencer

SCILAB est téléchargeable rapidement à l'adresse suivante :

http ://www.scilab.org/

Le logiciel est accompagné d'une documentation très complète. Vous pouvez compléter votre apprentissage en parcourant « Le guide du calcul avec des logiciels libres » paru chez Dunod en 2008 :-)
De nombreux TP sont présents sur le site de Scilab référencé ci-dessus.
Il faut savoir que scilab, comme matlab, est basé sur un noyau de fonctions en Fortran ou en C déjà compilées donc rapides d'utilisation. En revanche, tout ce que vous créerez vous-même sera interprété ce qui ralentira l'exécution par rapport à d'autres langages. De plus, la gestion des nombres n'est pas toujours optimale car ils sont codés en complexes double précision ce qui peut être lourd.

2 Scilab comme super-calculatrice

2.1 Les opérations de base

Scilab peut fonctionner de manière classique :

```
--> 3+2
```

Si on ajoute un « ; » à la fin de la ligne, celle-ci est lue par SCILAB mais le résultat n'est pas affiché : c'est le mode silencieux.
Mais Scilab fait du calcul numérique ; les résultats sont donc donnés sous forme de nombres à virgule flottante :

```
--> 1+sqrt(2)
```

sachant que `sqrt` désigne la racine carrée (SQare RooT en anglais...).

Remarque

Navigation et historique
On ne peut pas recommencer un calcul en plaçant le curseur de la souris sur une ligne précédente déjà évaluée. En revanche, avec les touches Page ↑ et Page ↓, on peut naviguer dans l'historique et réafficher des commandes précédentes.

Par défaut, les résultats sont affichés avec 10 caractères. Pour avoir plus de précision, on utilise `format` :

```
--> format(20);  1+sqrt(2)
```

```
 2.41421356237309492
```

On peut vouloir utiliser l'écriture scientifique d'un nombre : il faut préciser l'option `'e'` :

```
--> format('e',20); 1+sqrt(2)
```

```
 2.4142135623731D+00
```

Le `D+00` signifie mathématiquement $\times 10^0$.
On peut revenir à la configuration par défaut avec l'option `'v'` :

```
--> format('v',10)
```

Certaines constantes utiles sont déjà implémentées comme π :

```
--> %pi
```

```
 3.1415927
```

et exp(1) :

```
--> %e
```

```
 2.7182818
```

Pour les fonctions usuelles, la syntaxe est assez standard. Veuillez toutefois noter que le logarithme népérien se note `log` :

```
--> log(%e^2)
```

```
 2
```

Notez une particularité des logiciels de calcul numérique :

```
--> sin(%pi)
```

```
 1.225D-16
```

C'est le « zéro » fixé par Scilab qui est affiché...
On peut faire malgré tout des calculs avec des nombres plus petits :

```
--> 1D-25 - 1D-26
```

```
 9.000D-26
```

La valeur du « zéro » de Scilab est donnée par **`%eps`** :

```
--> %eps
```

```
 2.220D-16
```

Cela peut jouer des tours pour l'addition :

```
--> N=1D30; P=1D10;
```

```
--> N-N+D
```

```
--> N+D-N
```

Il faudra donc faire attention au moment de faire des tests d'égalité. Vous pouvez étudier la norme IEEE 754 plus en détail ou même envisager un TIPE sur l'arithmétique des flottants en informatique...
On peut travailler avec des nombre complexes, le nombre de carré -1 se notant **`%i`** :

```
--> (3+%i)^2
```

```
--> abs(1+%i)
```

2.2 Un brin d'informatique

2.2.a Types

Scilab travaille avec des objets de différents types et ne les traite pas de la même manière. Il faut distinguer :

- les réels et les complexes ;
- les booléens : **`%T`** pour vrai (True) et **`%F`** pour faux (False) ;
- les polynômes ;
- les chaînes de caractères entre apostrophes ;
- les listes ;
- les fonctions ;
- et bien d'autres encore...

On obtient le type d'un objet avec `typeof` :

```
--> typeof(%e)
```

```
 constant
```

```
--> typeof('%e')
```

```
 string
```

2.2.b Opérateurs

Scilab possède les opérateurs arithmétiques habituels `(+)`, `(-)`, `(*)`, `(/)` mais aussi une division inversée `(\)` et les opérateurs pointés dont nous parlerons plus tard.
Il sera également important d'utiliser les opérateurs booléens qui testent l'égalité `(==)`, la non-égalité `(<>)`, les inégalités strictes et larges `(<)`, `(>)`, `(<=)`, `(>=)`. Le résultat est un booléen (`%T` ou `%F`) :

```
--> 1+%eps/2==1

 T
```

Ce dernier résultat s'explique du fait que la différence entre les deux termes est inférieure à `%eps`
Enfin il existe des opérateurs logiques qui seront également utiles dans les tests : la négation (`~`), l'opérateur ET (`&`) et l'opérateur OU (`|`). Par exemple :

```
--> ~(3<3) == (3>=3)

 T
```

La négation de $3 < 3$ est bien $3 \geqslant 3$.

2.2.c Affectation

Pour donner un nom à une case mémoire où nous voulons stocker un objet, on utilise `(=)` :

```
--> a=3;
```

```
--> a^2
```

```
--> ans^2
```

La commande `ans` reprend en effet le résultat de la commande précédente.
La commande `who` permet de rappeler quelles sont les variables affectées.
Pour supprimer une variable affectée, on utilise `clear` :

```
--> clear a
```

La commande `clear` utilisée toute seule effacera toutes les variables affectées.

2.3 Définir une fonction

On peut définir toutes sortes de fonctions, d'une ou plusieurs variables, numériques ou non à l'aide de la syntaxe suivante :

```
--> function y=truc(x)
    y=x*log(x)
  endfunction
```

On utilise ensuite la fonction créée de manière naturelle :

```
--> truc(2)

 1.3862944
```

3 L'objet de base de Scilab : la matrice

3.1 Utilisation d'une matrice ligne

Même si on ne s'en aperçoit pas tout de suite, tout est matrice pour Scilab.
Occupons-nous d'abord des matrices-ligne.
Par exemple, nous voulons obtenir les valeurs prises par une fonction en certaines valeurs :

```
--> va = [1 3 74 100*%pi]
```

```
 1.    3.    74.    314.15927
```

Si on veut que ces valeurs soient incrémentées régulièrement :

```
--> x = [0:2:11]
```

```
 0.    2.    4.    6.    8.    10.
```

On obtient ainsi une suite arithmétique de nombres de premier terme 0, de raison 2 et inférieurs à 11.
Par défaut, l'incrément est de 1 :

```
--> y = [0:9]
```

```
 0.    1.    2.    3.    4.    5.    6.    7.    8.    9.
```

L'incrément peut être négatif et non entier :

```
--> X = [0:-2.5:-13]
```

```
 0.  - 2.5  - 5.  - 7.5  - 10.  - 12.5
```

On a facilement les valeurs de quelques suites obtenues à partir des précédentes :

```
-->  x^2
```

```
 0.    4.    16.    36.    64.    100.
```

On peut extraire une valeur de ces matrices :

```
-->  x(3)
```

```
 4.
```

ou une partie de ces valeurs :

```
-->  x(1:3)
```

```
 0.    2.    4.
```

Le dernier élément d'une matrice est obtenu à l'aide de $:

```
--> x($)
```

On peut extraire des composantes vérifiant une condition avec la commande `find` qui renvoie les rangs répondant à la demande :

```
-->  find(x>5)
```

```
 4.    5.    6.
```

On peut ajouter une valeur à une matrice :

```
-->  Xnew=[X,32]
```

```
 0.  - 2.5  - 5.  - 7.5  - 10.  - 12.5  32.
```

ou concaténer deux matrices :

```
-->  GrandX=[x,[1 2]]
```

```
 0.    2.    4.    6.    8.    10.    1.    2.
```

On peut effectuer des opérations entre ces vecteurs. Par exemple, essayons de les multiplier terme à terme :

```
--> x*X
```

Attention ! Les opérateurs arithmétiques (`*`), (`/`) et (`\`) effectuent des opérations sur des matrices. Souvenez-vous que ces opérations n'ont presque rien à voir avec celles sur les entiers. En particulier, $a*b \neq b*a$ en général.
Pour effectuer ces dernières opérations « terme à terme », il faut utiliser les opérateurs pointés sans oublier de laisser une espace entre le premier terme et le point :

```
--> x .*X
```

$$0 \quad -5 \quad -20 \quad -45 \quad -80 \quad -125$$

Des matrices particulières peuvent être utiles : **`zeros(m,n)`** et **`ones(m,n)`** renvoient des matrices remplies respectivement de 0 et de 1.

Pour extraire des lignes ou des colonnes ou des éléments particuliers :

```
A=[1,2,3;
   4,5,6;
   7,8,9]
A(1,3)
A(1,:)
A(:,1)
A($,:)
```

4 Les graphiques

On commence par créer un vecteur de 10000 valeurs régulièrement espacées entre 0 et 100 :

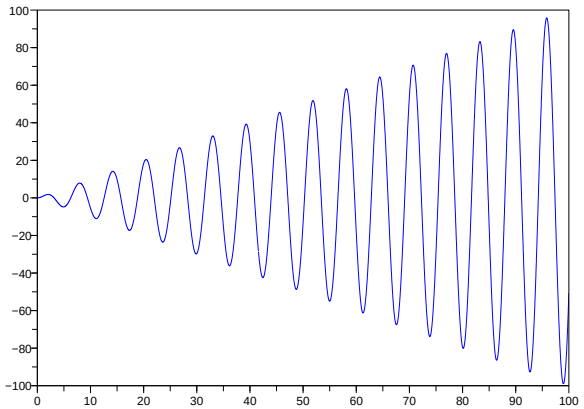
```
--> x=linspace(0,100,10000);
```

On aurait pu aussi bien poser :

```
--> x=0:.01:100
```

Nous voulons obtenir la représentation graphique de la fonction $x \mapsto x \cdot \sin(x)$ sur $[0;100]$:

```
--> plot(x,x .*sin(x))
```



Pour rajouter d'autres courbes sur la figure, nous avons plusieurs possibilités :

– demander une autre courbe :

```
--> plot(x,x)
```

```
--> plot(x,-x)
```

– soit tout demander en même temps. Pour éviter les superpositions, il faut donc effacer la figure précédente grâce à la commande `clf` :

```
--> clf; plot(x,x .*sin(x),x,x,-x)
```

On peut modifier le style des courbes à partir des menus de la fenêtre graphique ou en ligne de commandes :

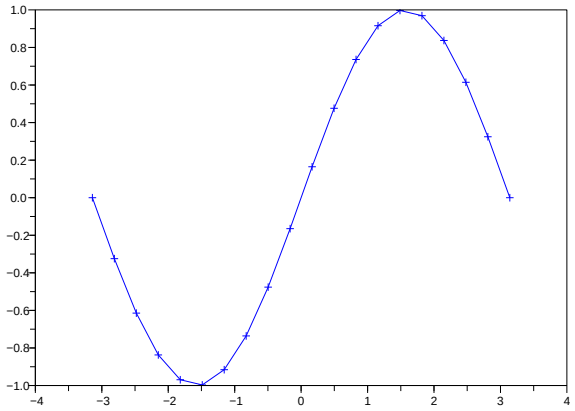
```
--> X=linspace(-%pi,%pi,20);
```

ou bien

```
--> X=-%pi:%pi/10:%pi
```

```
--> clf; plot(X,sin(X),"+-b")
```

L'option **+-b** signifie qu'on représente chaque point par une croix (+), qu'on les relie par un trait continu (-), en bleu (b).



Il existe d'autres raccourcis :

Couleur	jaune	magenta	cyan	rouge	vert	bleu	blanc	noir
Symbole	y	m	c	r	g	b	w	k

Ligne	plein	tirets	pointillés	mixte
Symbole	–	–	:	–.

Point	plus	rond	astérique	point	croix
Symbole	+	o	*	.	x

N'hésitez pas à explorer l'aide sur `gce` et `gda` pour plus de « configurabilité ».

5 La programmation

5.1 Les tests

5.1.a Si...alors...sinon

Construisons par exemple une fonction donnant la valeur absolue d'un nombre [a]

a. Cette fonction existe déjà sur Scilab (`abs`) mais il nous faut un exemple simple...

```
--> function y = ABS(x)
      if x >= 0 then
        y=x
      else
        y=-x
      end
    endfunction
```

En indentant son programme pour mettre en valeur les différentes structures créées, on le rend plus lisible.

5 1 b Éditeur de programme

Il existe un éditeur de texte dédié à Scilab (onglet `Editor`) qui permet de travailler à part sur son programme avant de l'éxécuter.

On clique alors sur `Execute -> Load into scialb` ou on appuie sur Ctrl + L pour exécuter le programme et l'envoyer vers Scilab. Si une erreur est décelée, le « débuggeur » est là pour vous aider à la déceler.
De retour dans Scilab, on appuis sur Entrée et on peut utiliser la fonction ABS comme tout à l'heure. Cela ne sera vraiment utile que pour de longs programmes.
On peut également, depuis l'éditeur, enregistrer le programme puis utiliser la commande `exec le_chemin_vers_le_f`

```
--> exec ./ABS.sci
```

5 1 c Emploi de return et de elseif

Dans l'exemple précédent il n'y avait qu'une seule alternative : x ou $-x$.
Intéressons-nous maintenant à des programmes dont les structures conditionnelles sont imbriquées.
Par exemple, construisons une fonction `Sol` qui donne les solutions d'une équations du second degré de la forme $ax^2+bx+c=0$. On pourrait construire notre programme comme précédemment :

```
function y=Sol(a,b,c)
d=b^2-4*a*c
  if d>0 then
    y=[(-b-sqrt(d))/(2*a) (-b+sqrt(d))/(2*a)]
  else
    if d<0 then
      y=[(-b-%i*sqrt(-d))/(2*a) (-b+%i*sqrt(-d))/(2*a)]
    else
      y=-b/(2*a)
    end
  end
endfunction
```

Cependant, il faut être très prudent et ne pas oublier de `end` et bien savoir se retrouver dans l'imbrication de conditions.
Il existe un moyen plus efficace qui permet de rester dans le même test : la commande `elseif` :

```
function y=Sol(a,b,c)
d=b^2-4*a*c
  if d>0 then
    y=[(-b-sqrt(d))/(2*a) (-b+sqrt(d))/(2*a)]
  elseif d<0 then
      y=[(-b-%i*sqrt(-d))/(2*a) (-b+%i*sqrt(-d))/(2*a)]
  else
```

```
    y=-b/(2*a)
  end
endfunction
```

5 2 Boucles for

Par exemple, calculons la somme S des 25 premiers entiers. Il faut commencer par initialiser S :

```
--> S=0;
```

```
--> for i=1:25 do S=S+i; end
```

Rappelez-vous que `1:25` crée la liste des entiers de 1 à 25. L'emploi du mode silencieux après le `S=S+i` permet d'éviter d'afficher tous les résultats intermédiaires.
On peut aussi boucler sur les éléments d'une matrice.

5 3 Boucles while

Reprenons le calcul de la somme des 25 premiers entiers :

```
--> S=0; i=1;
```

```
--> while i<=25 do S=S+i; i=i+1; end
```

5 4 Récursion

Il est également possible de créer des fonctions récursives. Du fait des divers appels de la même fonction, Scilab va vous envoyer sans cesse ce message :

```
Attention: redéfinition de la fonction: truc.
Utilisez funcprot(0) pour éviter ce message
```

On lui obéit... Que fait cette fonction :

```
function y = h(p,x)
  if p==[] then y = 0
  else y = p(1) + x*h(p(2:$),x)
  end
endfunction
```

sachant qu'elle s'utilise ainsi :

```
-->h([1 2 3],2)
 ans  =

    17.
```

Que pensez-vous alors de ces deux-là :

```
function y = dh(p,t,acc)
  if p == [] then y = acc
  else y = dh(p(1:$-1),t,p($) + t * acc)
  end
endfunction

function y = hb(p,t)
  y = dh(p,t,0)
endfunction
```

TP N°

3

Interpolations

Premier TP d'analyse numérique (très algébrique...) assisté par Scilab autour des différentes méthodes d'interpolation polynomiale.

1 Schéma de Horner-Ruffini-Holdred-Newton-Al-Tusi-Liu-Hui...

1 1 Un peu d'histoire...

La méthode que nous allons voir porte le nom du britannique William George HORNER (1786 - 1837) mais en fait elle fut publiée presque 10 ans auparavant par un horloger londonien, Theophilius HOLDRED et simultanément par l'italien Paolo RUFFINI (1765 - 1822) mais fut déjà utilisée par NEWTON 150 ans auparavant et par le chinois ZHU SHIJE cinq siècles plus tôt (vers 1300) et avant lui par le Persan SHARAF AL-DIN AL-MUZAFFAR IBN MUHAMMAD IBN AL-MUZAFFAR AL-TUSI vers (1100) et avant lui par le Chinois LIU HUI (vers 200) révisant un des résultats présent dans *Les Neuf Chapitres sur l'art mathématique* publié avant la naissance de JC...

Il faut cependant noter que RUFFINI l'avait employée en fait comme un moyen de calculer rapidement le quotient et le reste d'un polynôme par $(X-\alpha)$. C'est ce que nous allons (re)découvrir aujourd'hui...

1 2 Complexité

Dans toute la suite, un polynôme de dégré n sera représenté par le vecteur de ses coefficients. Par exemple, `[1 2 3]` correspond au polynôme $1+2x+3x^2$. N'oubliez pas enfin que Scilab commence à compter à partir de 1 :

```
--> p = [1 2 3];
--> p(1)
 ans  =
    1.
```

Nous avons vu le schéma de HORNER au TP précédent. Comparez la complexité au pire du calcul de $P(t)$ pour $t \in \mathbb{K}$. Vous prendrez comme « unité de complexité » les opérations arithmétiques de base : `+ - *`.

1 3 Dérivées successives

On note $P_n = a_0 + a_1 X + \cdots + a_n X^n$ un polynôme de degré n.

Déterminez le lien entre l'algorithme de HORNER et la division euclidienne de P_n par $(X-t)$. On utilisera par la suite la notation suivante :

$$P_n = (X-t)(b_n X^{n-1} + b_{n-1} X^{n-2} + \cdots + b_1) + b_0$$

Comment calculer les dérivées successives $\widetilde{P_n^{(j)}}(t)$ en utilisant uniquement des schémas de HORNER ? Quel est l'intérêt informatique ? Voici un peu d'aide...

Considérons le développement de P_n selon les puissances croissantes de $(X-t)$:

$$P_n = t_n (X-t)^n + t_{n-1}(X-t)^{n-1} + \cdots + t_0$$

Montrez que $\widetilde{P_n^{(j)}}(t)\, j! t_j$. Quel est le lien entre les t_j et nos schémas de HORNER ?

1 3 a Version récursive

Déterminez une fonction récursive sur Scilab `der_j(poly,j,t)` qui calcule $\widetilde{P_n^{(j)}}(t)$ en utilisant le schéma de HORNER.

Pour cela, on pourra commencer par modifier légèrement la fonction `horner` pour récupérer le vecteur $[b_0 b_1 ... b_n]$.

Créez ensuite `function y = quotient_horner(poly,t)` qui renvoie le quotient de `poly` par `(X-t)` puis `function y = reste_horner(poly,t)`.

Déduisez-en une fonction récursive `function y = jeme_poly(poly,j,t)` qui renvoie $\widetilde{P_n^{(j)}}(t)$.

1 3 b Version itérative

Prenons par exemple le polynôme $P_4 = [6,-7,1,-5,2]$. Essayez de trouver un algorithme simple pour remplir le tableau suivant :

Restes	0	1	2	3	4	j	
/	6	-7	1	-5	2	a_j	P_4
-12	-9	-1	-1	2	/	b_j	Q_3
						c_j	Q_2
						d_j	Q_1
						e_j	Q_0

En déduire l'écriture de P_4 sous la forme $R_4+R_3(X-2)+R_2(X-2)^2+R_1(X-2)^3+R_0(X-2)^4$ et faire la lien avec les dérivées successives.
Écrivez ensuite une fonction qui renvoie un tableau similaire puis transformez-la un peu pour obtenir les divisions successives en t.

```
-->jeme_poly_mat([6 -7 1 -5 2],5,2)
 ans  =
    0.     6.   - 7.     1.   - 5.     2.
  - 12.   - 9.   - 1.   - 1.     2.     0.
    1.     5.     3.     2.     0.     0.
    19.    7.     2.     0.     0.     0.
    11.    2.     0.     0.     0.     0.
    2.     0.     0.     0.     0.     0.
```

2 Interpolation de Lagrange

Edward WARING
(1736 - 1798)

Pour la petite histoire, même si LAGRANGE publia en 1795 la formule dans « *Leçons Élémentaires sur les Mathématiques Données a l'École Normale* », elle avait été publiée 16 ans plus tôt par le britannique WARING dans « *Problems Concerning Interpolations* » mais c'est l'inévitable GAUSS qui en fit le tour et la lia avec des travaux précédents de NEWTON et EULER, sans nommer personne, bien sûr, dans sa « *Theoria Interpolationis Methodo Nova Tractata* ».
On travaillera dans cette section avec des polynômes de $\mathbb{R}_n[X]$, une fonction numérique réelle f définie sur un intervalle $[a,b]$ et une famille $\{x_0,x_1,...,x_n\}$ de $n+1$ points distincts de $[a,b]$.

Théorème 3 - 1

Il existe un polynôme P_n de $\mathbb{R}_n[X]$ et un seul tel que :

$$\forall i \in \{0,1,...,n\} \qquad \widetilde{P_n}(x_i) = f(x_i)$$

Ce polynôme s'écrit :

$$P_n(X) = \sum_{i=0}^{n} f(x_i)\mathcal{L}_i(X)$$

avec :

$$\mathcal{L}_i(X) = \prod_{j=0, j\neq i}^{n} \frac{X-x_j}{x_i-x_j}$$

La preuve est une formalité pour de valeureux MP* (une des démonstrations fait le lien avec un déterminant de VANDERMONDE : comment ?)...
D'un point de vue algorithmique, que se passe-t-il lorsqu'on rajoute un point pour améliorer l'interpolation ?

3 Interpolation de Newton

3 1 L'idée

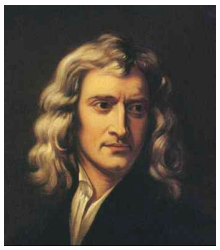
Pour remédier à ce problème, nous allons procéder autrement : appelons $f[x_1,x_2,...,x_n]$ le coefficient de x^n dans le polynôme interpolateur de LAGRANGE (attention à ne pas le confondre avec le polynôme de LAGRANGE $\mathcal{L}$).

On peut montrer par récurrence que ces coefficients sont liés par la relation :

$$f[x_1, x_2, ..., x_{n+1}] = \frac{f[x_2, ..., x_{n+1}] - f[x_1, x_2, ..., x_n]}{x_{n+1} - x_1}$$

avec $f[x_i] = f(x_i)$. On visualise la situation à l'aide d'un tableau du type :

3 2 Un exemple avec un pas constant



Isaac NEWTON
(1642 - 1727)

Supposons que nous connaissions quelques valeurs du logarithme décimal (par exemple par la méthode de BRIGGS) et que nous voulions connaître des valeurs intermédiaires. Nous allons pour cela procéder par interpolation : étant données quatre valeurs, nous allons chercher un polynôme P de degré 3 tel que $P(x) = \log(x)$ pour ces 4 valeurs.
Pour cela, nous allons utiliser la méthode présentée par Isaac NEWTON en 1676.
Avec des notations modernes, posons $P(x) = a + bx + cx^2 + dx^3$. On connaît les valeurs du logarithmes pour 1, 2, 3 et 4.

$x_1 = 1$	$p(x_1) = a + b + c + d = f(x_1)$
$x_2 = 2$	$p(x_2) = a + 2b + 4c + 8d = f(x_2)$
$x_3 = 3$	$p(x_3) = a + 3b + 9c + 27d = f(x_3)$
$x_4 = 4$	$p(x_4) = a + 4b + 16c + 64d = f(x_4)$

En soustrayant les lignes deux par deux, on fait disparaître a :

$$\begin{array}{l} f(x_2) - f(x_1) = b + 3c + 7d = f[x_1, x_2] \\ f(x_3) - f(x_2) = b + 5c + 19d = f[x_2, x_3] \\ f(x_4) - f(x_3) = b + 7c + 37d = f[x_3, x_4] \end{array}$$

En soustrayant les lignes deux par deux, on fait disparaître b :

$$\begin{array}{l} f[x_2, x_3] - f[x_1, x_2] = 2c + 12d = f[x_1, x_2, x_3] \\ f[x_3, x_4] - f[x_2, x_3] = 2c + 18d = f[x_2, x_3, x_4] \end{array}$$

En soustrayant les lignes deux par deux, on fait disparaître c :
$f[x_2, x_3, x_4] - f[x_1, x_2, x_3] = 6d = f[x_1, x_2, x_3, x_4]$
ce qui donne :
$d = \frac{1}{6} f[x_1, x_2, x_3, x_4]$
$c = \frac{1}{2} f[x_1, x_2, x_3] - f[x_1, x_2, x_3, x_4]$
$b = f[x_1, x_2] - \frac{3}{2} f[x_1, x_2, x_3] + 3f[x_1, x_2, x_3, x_4] - \frac{7}{6} f[x_1, x_2, x_3, x_4] = f[x_1, x_2] - \frac{3}{2} f[x_1, x_2, x_3] + \frac{11}{6} f[x_1, x_2, x_3, x_4]$
$a = f[x_1] - f[x_1, x_2] + \frac{3}{2} f[x_1, x_2, x_3] - \frac{11}{6} f[x_1, x_2, x_3, x_4] - \frac{1}{2} f[x_1, x_2, x_3] + \Delta^3 y_1 - \frac{1}{6} f[x_1, x_2, x_3, x_4] = f[x_1] - f[x_1, x_2] + f[x_1, x_2, x_3] - f[x_1, x_2, x_3, x_4]$
Finalement :

$$P(x) = f[x_1] + (x-1) f[x_1, x_2] + \frac{1}{2}(x-1)(x-2) f[x_1, x_2, x_3] + \frac{1}{6}(x-1)(x-2)(x-3) f[x_1, x_2, x_3, x_4]$$

NEWTON avait l'habitude de réunir les résultats dans un schéma de cette forme :

$\boldsymbol{f[x_1]}$	$\boldsymbol{f[x_1, x_2]}$	$\boldsymbol{f[x_1, x_2, x_3]}$	$\boldsymbol{f[x_1, x_2, x_3, x_4]}$
$f[x_2]$	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$	
$f[x_3]$	$f[x_3, x_4]$		
$f[x_4]$			

avec $f[x_1, x_2, ..., x_i] = \frac{f[x_2, x_3, ..., x_i] - f[x_1, x_2, ..., x_{i-1}]}{x_i - x_1} = f[x_2, x_3, ..., x_i] - f[x_1, x_2, ..., x_{i-1}]$

3 3 Cas général

On travaillera dans cette section avec des polynômes de $\mathbb{R}[X]$, une fonction numérique réelle f définie sur un intervalle $[a, b]$ et une famille $\{x_1, ..., x_n\}$ de n points distincts de $[a, b]$.
On peut montrer que le polynôme interpolateur est défini par :

$$P_n(t) = f[x_1] + \sum_{k=2}^{n} f[x_1, ..., x_n](x - x_1)(x - x_2) \cdots (x - x_{k+1})$$

Vous construirez une fonction **function y = newton(X,Y,t)** qui renvoie l'image de **t** par la fonction polynomiale obtenue par l'interpolation de Newton sur les points dont les abscisses et les ordonnées sont les éléments des matrices X et Y.
Le plus pratique est peut-être de considérer un tableau : déterminez une relation calculant `T(i,j)` en fonction de `T(i+1,j-1)`, `T(i,j-1)`, et des éléments de X...

```
-->p = newton([0 1 2 3 5],[0 1 8 27 125],4)
 p  =
    64.
```

Il peut s'avérer plus utile d'obtenir le polynôme lui-même plutôt que l'image d'un réel particulier par ce polynôme.
Modifiez alors **newton** pour obtenir une fonction **function y = newton_poly(X,Y)** qui renvoie le polynôme d'interpolation lui-même en utilisant `prod_mono` et `add_poly` :

```
-->newton_poly([0 1 2 3],[0 1 8 27])
 ans  =
    0.    0.    0.    1.
```

Pour construire une fonction Scilab associée au problème, nous aurons besoin de deux fonctions intermédiaires :

- **function y = prod_mono(poly,t)** qui calcule le produit de **poly** par le monôme de racine **t** ;

  ```
  -->prod_mono([1 3 3 1],-1)
   ans  =
      1.    4.    6.    4.    1.
  ```

- **y = add_poly(p1,p2)** qui calcule la somme des polynômes (de degrés quelconques...) **p1** et **p2**.

  ```
  -->add_poly([1 2 3] , [0 2 4 6 8 10 12])
   ans  =
      1.    4.    7.    6.    8.    10.    12.
  ```

Est-ce efficace ?
Observons avec trois points de contrôle (approximation parabolique) :

```
-->p = newton_poly([42 43 44],[log(42) log(43) log(44)])
 p  =
    2.2608845    0.0465221  - 0.0002705

-->horner(p,43.25)
 ans  =
    3.7669982

-->log(43.25)
 ans  =
    3.7669972
```

Observez en des lieux « moins plats ». Avec plus de points ? Tout ceci ne constitue que des observations.
Occupons-nous de l'erreur d'interpolation. Soit P_n un polynôme interpolateur en x_1, x_2,...,x_n points distincts de $[a,b]$ d'une fonction f définie sur $[a,b]$.
Soit $t \in [a,b]$ distinct des x_i (sinon...). Posons $e_n(t) = P_n(t) - f(t)$ et P_{n+1} le polynôme qui interpole f en x_1,...,x_n, t.
En pensant à la diagonale descendante de NEWTON, on obtient :

$$P_{n+1}(t) = f(t) = P_n(t) + f[x_1, x_2, ..., x_n, t] \prod_{i=0}^{n} (t - x_i)$$

donc

$$e_n(t) = f[x_1, x_2, ..., x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Une n^e formule de CAUCHY montre que si f est de classe $\mathcal{C}^{n+1}$ sur $[a,b]$, il existe $\xi_t \in]a,b[$ tel que :

$$e_n(t) = \frac{f^{(n+1)}(\xi_t)}{(n+1)!}\prod_{i=0}^{n}(t-x_i) \quad \text{soit} \quad |e_n(t)| \leqslant \max_{\xi\in]a,b[}\left|\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(t-x_i)\right|$$

Le meilleur choix des abscisses pour minimiser l'erreur est donc :

$$\min_{(x_i)}\left(\max_{t\in[a,b]}\left|\prod_{i=0}^{n}(t-x_i)\right|\right)$$

4 Observation graphique

Un peu d'observation maintenant. Pour cela nous aurons besoin d'une fonction **`function y = poly2x(poly,x)`** qui renvoie l'image d'un vecteur **`x`** par une fonction polynomiale **`poly`**.

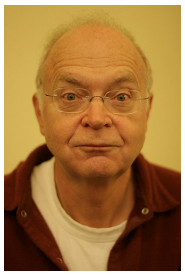
```
-->poly2x([1 2 1],x)
 ans  =

    1.    4.    9.
```

Que fait la boucle suivante :

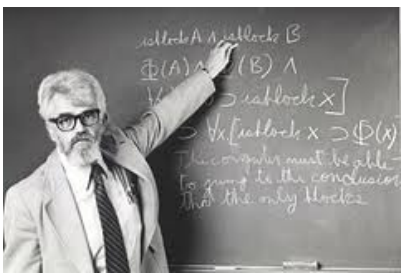
```
for k = [0.1:0.1:1] do
  x = [0:k:2];
  p = newton_poly(x,1/(1+x^2));
  plot(x,poly2x(p,x))
end
```

Commentaires ? La vie n'est pas si simple qu'elle n'en a l'air. Heureusement le cours de MP⋆ de M. SAUVAGEOT va vous permettre de résoudre ce genre de problème de manière efficace. Vous y croiserez alors TCHEBYCHEV, BERNSTEIN, FOURIER, WEIERSTRASS...

5 Quelques idées pour la frise...



Donald KNUTH (prix TURING 1974) :« *There are ways to amuse yourself while doing things and that's how I look at efficency* ».



John MAC CARTHY (prix TURING 1971) :« *He who refuses to do arithmetic is doomed to talk nonsense* ».

Edsger DIJKSTRA (prix TURING 1972) :« *The question « What is Mathematics ? » is as anavoidable and as unanswerable as the question « What is Life ? ». In actual fact I think it's almost the same question* ».

TP N°

4

Dérivation et intégration numérique

Il est très rare de pouvoir déterminer la dérivée ou une primitive formelle d'une fonction donnée. Il faut recourir à des méthodes numériques dont l'utilisation entraîne de nouveaux problèmes : il faut pouvoir contrôler l'approximation de manière théorique et ne pas oublier que cela sera fait la plupart du temps par une machine qui elle aussi apporte son lot d'imprécision.

Werner Romberg (1909 - 2003) né à Berlin, s'installe en Norvège en 1938. Dix-sept ans plus tard, il publie un article se basant sur l'extrapolation de Lewis Richardson (1881 - 1953) et qui accélère grandement l'efficacité de la méthode dite des trapèzes utilisée depuis longtemps pour calculer des approximations d'intégrales. C'est cette méthode que nous étudierons aujourd'hui en la comparant aux méthodes de Newton-Cotes, Gauss-Legendre, Kuncir après une introduction sur les méthodes de dérivation numérique.

1 Dérivation numérique

Notons $\tau_x(h) = \frac{f(x+h)-f(x)}{h}$ l'habituel taux d'accroissement.

Recherche

Expliquez pourquoi $\left|\tau_x(h) - f'(x)\right| \leqslant \frac{h}{2}\left\|f''\right\|_\infty$

À cela s'ajoute les problèmes d'arrondi de la machine. Notons ε la variable `%eps` de Scilab :

```
-->%eps
 %eps  =
    2.220D-16
```

Recherche

Expliquez pourquoi $\left|\tau_x(h) - \widetilde{\tau_x(h)}\right| \leqslant 2\varepsilon\frac{|f(x)|}{h}$ en appelant $\widetilde{\tau_x(h)}$ la valeur effectivement calculée par la machine.
En déduire que la fonction $h \mapsto \left|f'(x) - \widetilde{\tau_x(h)}\right|$ admet un minimum absolu sur $\mathbb{R}_+$ en $2\sqrt{\varepsilon}\frac{|f(x)|}{\|f''\|_\infty}$.
Quel ordre de grandeur va-t-on donc choisir pour h ?

Nous allons visualiser ce phénomène avec Scilab :

```
// on affiche suffisamment de chiffres significatifs
format(16)

// taux de variation pour différents h
function L = diff_avant(f,x,d)
L = []
h = 1
for n = 1:d do
  h = h/10;
  L = [L;[h , (f(x+h)-f(x))/h]];
end
endfunction

// x -> sqrt(1+x)
function y = sqa(x)
  y = sqrt(1+x)
endfunction

// on observe
diff_avant(sqa,0,15)
```

Recherche

On peut facilement améliorer la précision : déterminez les coefficients a, b, c et d tels que :

$$f'(x) = \frac{a\cdot f(x+2h) + b\cdot f(x+h) + c\cdot f(x) + d\cdot f(x-h)}{h} + \mathrm{O}(h^3)$$

Appliquez cette nouvelle formule en adaptant les fonctions Scilab précédentes.
Remplacez les puissances de 10 par des puissances de 2 : que remarquez-vous ? Pourquoi ? Comment s'écrit $\frac{1}{10}$ en base 2 ? $\frac{1}{2}$?

2 Intégration numérique

2 1 Méthode de quadrature simplifiée déterministe

On étudiera des formules d'approximation reposant sur des formules du type :

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=1}^{n} w_{i,n} f(x_i) + \mathrm{E}_n(f)$$

où les w_i sont les *poids*, les x_i sont les pivots et $\mathrm{E}(f)$ l'erreur commise.
Une méthode de quadrature est associée à un ordre :

Définition 4 - 1

Une méthode de quadrature est d'ordre m si elle est exacte pour tous les polynômes de $\mathbb{R}_m[X]$.

Nous admettrons le théorème suivant qui donne une CNS de convergence d'une méthode de quadrature :

Théorème 4 - 1

Pour toute fonction continue sur $]a,b[$, $\lim\limits_{n\to+\infty} \mathrm{E}_n(f)=0$ si, et seulement si,

- $(\exists M\in\mathbb{R}_+^\star)(\forall n\in\mathbb{N})(\sum\limits_{i=0}^{n}|w_{i,n}|\leqslant M)$
- $(\forall k\in\mathbb{N})(\lim\limits_{n\to+\infty}\mathrm{E}_n(x\mapsto x^k)=0)$

On peut se contenter, pour la condition suffisante, d'avoir chaque $w_{i,n}$ positif et f continue par morceaux.
Ce qui nous intéressera au plus haut point est le théorème suivant qui donne une estimation de l'erreur commise :

Théorème 4 - 2

On suppose que la méthode est d'ordre m et $f\in\mathcal{C}^{(m+1)}([a,b],\mathbb{R})$. Alors

$$\mathrm{E}(f)=\frac{1}{m!}\int_a^b \mathrm{K}_m(x)f^{(m+1)}\,\mathrm{d}x$$

avec $\mathrm{K}_n(x)=\mathrm{E}(x\mapsto\max((x-t)^n,0))$. La famille $(\mathrm{K}_n)_{n\in\mathbb{N}}$ s'appelle la famille des noyaux de PEANO associée à la méthode considérée.

On utilise la formule de TAYLOR intégrale puis le fait que la méthode est d'ordre m.
On en déduit le corollaire suivant :

Théorème 4 - 3

Si K_m est de signe constant sur $[a,b]$:

$$(\exists\xi\in]a,b[)(\mathrm{E}(f)=\frac{1}{(m+1)!}f^{(m+1)}(\xi)\mathrm{E}(x\mapsto x^{m+1}))$$

On utilise la deuxième formule de la moyenne.

2 2 Méthodes de Newton-Cotes composées

Pour donner une valeur approchée de l'intégrale I, une première idée est de remplacer la fonction f par un polynôme P qui interpole f en plusieurs points. Cependant, dès qu'on augmente le nombre de pivots, on risque fort de se retrouver confronté au phénomène de RUNGE.
On modifie alors l'idée de départ ainsi : on commence par subdiviser régulièrement l'intervalle d'intégration en n sous-intervalles $[x_j,x_{j+1}]$ où

$$x_0=a,\quad x_n=b,\quad\text{et}\quad\forall j\in[\![0,n]\!],\quad x_j=a+jh,\quad\text{avec } h=\frac{b-a}{n}$$

Ensuite, on remplace f par un polynôme P_j qui interpole f sur chacun des petits segments $[x_j,x_{j+1}]$.

- Si P_j interpole f au point x_j, alors le graphe de P_j est une droite horizontale : c'est la méthode des rectangles.
- Si P_j interpole f aux points x_j et x_{j+1}, alors le graphe de P_j est une droite affine : c'est la méthode des trapèzes.
- Si P_j interpole f aux points x_j, $\frac{x_j+x_{j+1}}{2}$ et x_{j+1}, alors le graphe de P_j est une parabole : c'est la méthode de SIMPSON 1/3.

Enfin, on somme les intégrales de chaque polynôme P_j sur $[x_j,x_{j+1}]$, et on obtient une valeur approchée de I :

$$\mathrm{I}=\sum_{j=0}^{n-1}\int_{x_j}^{x_{j+1}}f(x)\,\mathrm{d}x=\sum_{j=0}^{n-1}\left(\int_{x_j}^{x_{j+1}}\mathrm{P}_j(x)\,\mathrm{d}x+\mathrm{E}_j\right)\approx\sum_{j=0}^{n-1}\int_{x_j}^{x_{j+1}}\mathrm{P}_j(x)\,\mathrm{d}x$$

On se place sur l'intervalle $[x_j,x_{j+1}]$ et on note $\xi_j=\frac{x_j+x_{j+1}}{2}$; alors d'après les formules des différences divisées (cf. TP précédent...), le polynôme d'interpolation de f aux points (x_j,ξ_j,x_{j+1}) est donné par

$$\mathrm{P}_j(x)=f[x_j]+f[x_j,\xi_j](x-x_j)+f[x_j,\xi_j,x_{j+1}](x-x_j)(x-\xi_j)$$

On en déduit, après calculs,

$$\int_{x_j}^{x_{j+1}} \mathrm{P}_j(x)\,\mathrm{d}x = \frac{h}{6}\left(f(x_j)+4f(\xi_j)+f(x_{j+1})\right)$$

Par sommation on obtient

$$\begin{aligned}
\mathrm{I} &\approx \frac{b-a}{6n}\left(f(x_0)+4f\left(\tfrac{x_0+x_1}{2}\right)+2f(x_1)+4f\left(\tfrac{x_1+x_2}{2}\right)+\cdots+2f(x_{n-1})+4f\left(\tfrac{x_{n-1}+x_n}{2}\right)+f(x_n)\right)\\
&\approx \frac{b-a}{6n}\left(f(x_0)+2\sum_{j=1}^{n-1}f(x_j)+4\sum_{j=0}^{n-1}f\left(\tfrac{x_j+x_{j+1}}{2}\right)+f(x_n)\right)\\
&\approx \frac{b-a}{6n}\sum_{k=0}^{2n} w_k f(a_k) \quad \text{avec} \quad a_k = a+k\cdot\frac{b-a}{2n} \text{ et } w_k = \begin{cases} 1 \text{ si } k=0 \text{ ou } 2n \\ 4 \text{ si } 0<k<2n \text{ et } k \text{ impair} \\ 2 \text{ si } 0<k<2n \text{ et } k \text{ pair} \end{cases}
\end{aligned}$$

On peut montrer que l'erreur commise est majorée par

$$|\mathrm{E}| \leqslant \frac{1}{2880}\cdot h^4 \cdot \left\|f^{(4)}\right|_\infty \cdot (b-a) \qquad (**)$$

On dit alors que la méthode est d'ordre 4 ; noter que la méthode est exacte pour tout polynôme de degré au plus 3.

Recherche

Programmez les trois méthodes mentionnées : **Rectangles**, **Trapezes** et **Simpson**.
On utilisera les facilités offertes par Scilab pour manipuler des vecteurs et on fera attention aux difficultés de Scilab avec les évaluations de fonctions...
Voici un exemple pour les rectangles :

```
// rectangles
function R = Rectangles(f,a,b,N)
  h = (b-a)/N
  pivots = a + h*[1:N]
  R = h*sum(feval(pivots,f))
endfunction
```

On introduira la fonction $x \mapsto \frac{4}{1+x^2}$ et on comparera son intégrale sur [0,1] avec π en utilisant les différentes méthodes.

En pratique, il est difficile d'avoir un majorant de la dérivée quatrième d'une fonction. On préfère donc en pratique appliquer la méthode avec deux pas différents (h et $2h$ pour minimiser les évaluations de f) et on utilise la différence des deux approximations numériques comme estimation de l'erreur du moins bon résultat.
En prenant 5 points d'interpolation, on construit la méthode dite de BOOLE-VILLARCEAU et avec 7 celle de WEDDLE-HARDY. On ne va pas plus loin car sinon on obtient des poids négatifs qui amplifient les erreurs d'arrondi.

2 3 La méthode de Romberg

La méthode de ROMBERG utilise le procédé d'extrapolation de RICHARDSON pour accélérer la convergence de la méthode des trapèzes composée.
Soit f une fonction de classe $\mathcal{C}^\infty$ sur le segment $[a,b]$. On décompose le segment $[a,b]$ en n sous-intervalles égaux. Alors l'approximation de l'intégrale I de f sur ce segment par la méthode des trapèzes vaut

$$\mathrm{T}(h) = h\left(\frac{1}{2}f(x_0)+\sum_{j=1}^{n-1}f(x_j)+\frac{1}{2}f(x_n)\right) \quad \text{où} \quad \forall j\in[\![0,n]\!], \quad x_j = a+jh, \quad \text{et} \quad h=\frac{b-a}{n}$$

On peut alors montrer à l'aide de la formule d'EULER-MACLAURIN que l'erreur commise dans la méthode des trapèzes possède un développement limité à tout ordre de la forme

$$\mathrm{E}(h) = a_1h^2+a_2h^4+\cdots+a_{k-1}h^{2k-2}+\mathrm{O}(h^{2k}) \quad \text{où} \quad \mathrm{E}(h) = \mathrm{T}(h)-\int_a^b f(t)\,\mathrm{d}t$$

De plus, les a_i ne dépendent ni de n, ni de h.

L'idée est alors de faire une combinaison linéaire de $\mathrm{T}(h)$ et $\mathrm{T}(h/2)$ pour annuler le premier terme du développement de E :

$$\begin{aligned} \mathrm{T}(h) &= \int_a^b f(t)\,\mathrm{d}t + a_1 h^2 + a_2 h^4 + \cdots + a_{k-1} h^{2k-2} + \mathrm{O}(h^{2k}) \\ \mathrm{T}\left(\frac{h}{2}\right) &= \int_a^b f(t)\,\mathrm{d}t + a_1 \frac{h^2}{2^2} + a_2 \frac{h^4}{2^4} + \cdots + a_{k-1} \frac{h^{2k-2}}{2^{2k-2}} + \mathrm{O}(h^{2k}) \\ \hline \frac{1}{3}\left(4\mathrm{T}\left(\frac{h}{2}\right) - \mathrm{T}(h)\right) &= \int_a^b f(t)\,\mathrm{d}t + \underbrace{b_2 h^4 + \cdots + b_{k-1} h^{2k-2} + \mathrm{O}(h^{2k})}_{\mathrm{O}(h^4)} \end{aligned}$$

On remarque qu'ainsi, on a gagné un ordre dans le développement de l'erreur. C'est un bon exercice de vérifier que la nouvelle expression obtenue correspond à la méthode de SIMPSON.
Le procédé précédent se généralise aisément : on utilise des dichotomies successives avec des pas de la forme $h = \frac{b-a}{2^m}$ et on remplit un tableau triangulaire comprenant les nombres :

$$\mathrm{A}_{m,0} = \mathrm{T}\left(\frac{b-a}{2^m}\right) \quad \text{et} \quad \forall n \in [\![1, m]\!],\ \mathrm{A}_{m,n} = \frac{4^n \mathrm{A}_{m,n-1} - \mathrm{A}_{m-1,n-1}}{4^n - 1}$$

étape	$t=0$	$t=1$	$t=2$	$\cdots$	$t=m$
tableau[0]	$\mathrm{A}_{0,0}$				
tableau[1]	$\mathrm{A}_{1,0}$	$\mathrm{A}_{1,1}$			
tableau[2]	$\mathrm{A}_{2,0}$	$\mathrm{A}_{2,1}$	$\mathrm{A}_{2,2}$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
tableau[n]	$\mathrm{A}_{m,0}$	$\mathrm{A}_{m,1}$	$\mathrm{A}_{m,2}$	$\cdots$	$\mathrm{A}_{m,m}$

Le dernier élément de la diagonale principale $\mathrm{A}_{m,m}$ est alors la meilleure approximation de l'intégrale I que l'on puisse obtenir en évaluant f en 2^m points.
En pratique, on commence par remplir la première colonne en se servant de la relation de récurrence :

$$\mathrm{A}_{k,0} = \frac{1}{2}\mathrm{A}_{k-1,0} + \mathrm{A}'_{k,0} \quad \text{où} \quad \mathrm{A}'_{k,0} = \frac{b-a}{2^k} \sum_{j=1}^{2^{k-1}} f\left(a + (2j+1)\frac{b-a}{2^k}\right)$$

En effet, en notant $h = \frac{b-a}{2^k}$, on a

$$\begin{aligned} \mathrm{A}_{k,0} &= h\left(\frac{1}{2}f(a) + \sum_{j=1}^{2^k-1} f\left(a+jh\right) + \frac{1}{2}f(b)\right) \\ &= h\left(\frac{1}{2}f(a) + \sum_{p=1}^{2^{k-1}-1} f\left(a+2ph\right) + \sum_{p=0}^{2^{k-1}-1} f\left(a+(2p+1)h\right) + \frac{1}{2}f(b)\right) \\ &= \frac{1}{2}\mathrm{A}_{k-1,0} + \underbrace{h \sum_{j=0}^{2^{k-1}-1} f\left(a+(2j+1)h\right)}_{\mathrm{A}'_{k,0}} \end{aligned}$$

Ceci permet de calculer une fois et une fois seulement les valeurs de f nécessaires.
Ensuite, on remplit les autres colonnes en utilisant la formule

$$\forall n \in [\![1, m]\!],\ \mathrm{A}_{m,n} = \frac{4^n \mathrm{A}_{m,n-1} - \mathrm{A}_{m-1,n-1}}{4^n - 1}$$

On montre alors par récurrence que :

$$\mathrm{A}_{m,k} = \int_a^b f(x)\,\mathrm{d}x + \mathrm{O}\left(\frac{1}{2^{m(2k+2)}}\right)$$

Recherche

Programmez Scilab pour obtenir $\mathrm{A}_{m,m}$.
On estime l'erreur comme d'habitude à $\left|\mathrm{A}_{m,m} - \mathrm{A}_{m-1,m-1}\right|$.

2 4 Méthode de Monte-Carlo

3 Lectures

Mathématiques pour l'informatique - PICHAT/WOLF/DI CRESCENZO - Armand Colin 1971
Modélisation à l'oral de l'Agrégation - DUMAS - Ellipses 1999
Analyse numérique et équations différentielles - DEMAILLY - EDP Sciences 1996
Programmation en Python pour les mathématiques - CASAMAYOU/CHAUVIN/CONNAN - Dunod 2012

TP N°

5

Discrétisation d'équations différentielles

La résolution formelle d'équations différentielles s'avère très compliquée et limitée : la plupart des problèmes ne peuvent être qu'approchés.

Le très prolifique mathématicien suisse Leonard Euler mit au point au XVIII^{e} siècle une méthode de discrétisation des équations différentielles que généralisèrent et améliorèrent les Allemands Carl Runge (1856-1927) dont vous pouvez admirer le portrait, et Martin Kutta (1867-1944). Nous allons donc étudier ces méthodes de Runge-Kutta très utiles pour l'ingénieur.

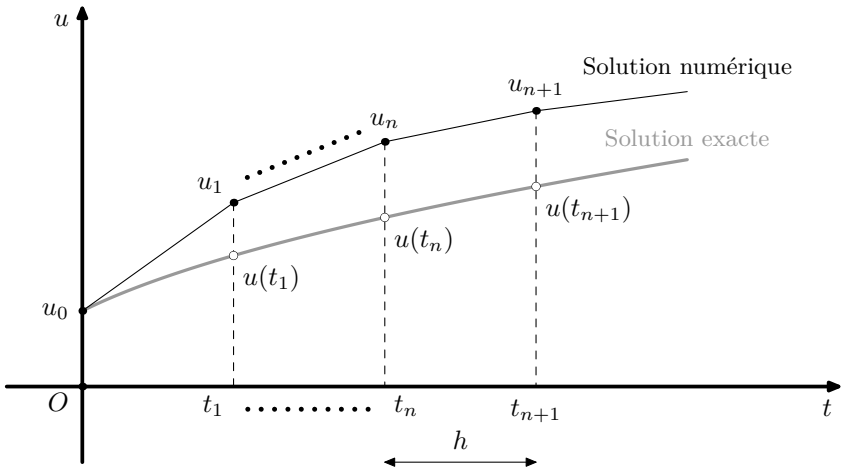
Il est à noté que Runge mit également en évidence des fonctions assez régulières faisant diverger la méthode d'interpolation polynomiale de Lagrange et donna son nom à ce phénomène.

1 Principe

On considère une équation différentielle ordinaire (EDO pour les intimes) $u'(t) = f(t, u(t))$. Lorsqu'on ne connait pas de solution exacte à cette EDO, on essaye d'en avoir une bonne approximation par des méthodes numériques.

Nous allons par exemple « observer » u à intervalle de temps régulier : notons h cette période d'échantillonnage.

Nous allons essayer d'obtenir une suite d'approximations u_n de $u(t_n)$ et d'évaluer la pertinence de cette approximation, le principe étant résumé sur la figure suivante :



Nous considérerons des fonctions f K-lipschitziennes en chacune des variables et des fonctions u de classe $\mathcal{C}^1$ d'un intervalle I dans $\mathbb{R}$ avec une condition initiale $u(t_0) = y_0$.

Pour une étude mathématique complète des phénomènes de convergence, on pourra étudier l'article http://www.math.ens.fr/~debarre/Exponentielle-Log.pdf, lire le chapitre 5 de Modélisation à l'oral de l'agrégation de Laurent DUMAS et les chapitres 4 et 5 de l'ouvrage Analyse numérique des équations différentielles par Michel CROUZEIX et Alain MIGNOT et surtout « Solving Ordinary Diffrential Equations » de HAIRER, NORSETT et WANNER chez Springer.

On notera I_n l'intervalle $[t_n, t_{n+1}]$ avec $t_{n+1} = t_n + h$, la suite (t_n) constituant une subdivision régulière de l'intervalle I.

On cherche donc à construire une suite finie (u_n) telle que $u_0 = y_0$ et $u_{n+1} = u_n + h\varphi(t_n, u_n)$ (format explicite à un pas) ou $u_{n+1} = u_n + h\varphi(t_{n+1}, u_{n+1})$ (format implicite à un pas).

2 Méthode d'Euler explicite

Intégrons l'équation différentielle $u'(t) = f(t, u(t))$ sur l'intervalle I_n :

$$u(t_{n+1}) - u(t_n) = \int_{t_n}^{t_{n+1}} f(s, u(s))\,\mathrm{d}s$$

L'utilisation de méthodes de quadratures vues au TP précédent vont nous permettre de calculer u_{n+1} connaissant u_n.

La méthode des rectangles à gauche donne le schéma d'EULER explicite.

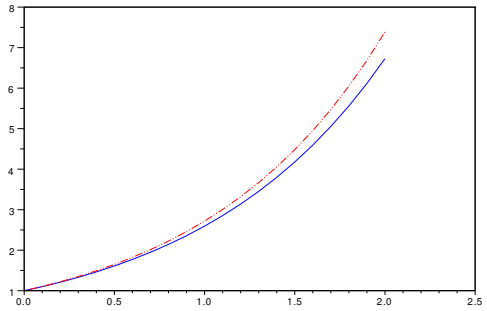
Déterminer une fonction Scilab qui renvoie la liste des approximations des abscisses et des ordonnées de la série de points de la solution d'une équation différentielle par le schéma d'Euler explicite.

On pourra créer par exemple **`function euler_imp(f,a,b,N,yo)`**.

On pourra comparer avec des cas connus. Par exemple

```
e = euler_imp(deff('[u] = f(x,y)', 'u = y') ,0,2,20,1);
plot(e(1,:),e(2,:)) , plot(e(1,:),exp(e(1,:)),":r")
```

devrait donner :



Commentez la fonction :

```
function  M = mystere(f,x,y,xmax,h,liste)
  if x > xmax then M = liste
  else M = mystere(f,x+h,y+h*f(x,y),xmax,h,[liste,y+h*f(x,y)])
  end
endfunction
```

Comment l'utiliser pour obtenir le même tracé que précédemment ?
Commentez :

```
max(abs((e(2,:) - exp(e(1,:)))))
```

3 Méthode d'Euler modifiée

Nous avons vu au TP précédent que la méthode d'intégration du point milieu nous faisait gagner un ordre de précision.
Appliquez alors cette méthode à notre problème.
On pourra alors remarquer qu'avec N = 4096 la méthode d'EULER a une précision de l'ordre de :

```
 ans  =
    0.0036059
```

et avec la méthode d'EULER modifiée :

```
 ans  =
    0.0000006
```

4 Méthode RK4

On utilise cette fois la méthode de SIMPSON. Notons :

$$S = \frac{h}{6}\left(f(t_n, u(t_n)) + 4f(t_n + h/2, u(t_n + h/2)) + f(t_n + h, u(t_n + h))\right)$$

Alors faisons les approximations suivantes :

$$u(t_n) \to u_n = U_0$$

puis :

$$u(t_n + \frac{h}{2}) \to u_n + \frac{h}{2} f(t_n, u_n) = U_1$$

en utilisant le rectangle gauche, ou bien :

$$u(t_n + \frac{h}{2}) \to u_n + \frac{h}{2} f\left(t_n + \frac{h}{2}, u(t_n + \frac{h}{2})\right) \to u_n + \frac{h}{2} f\left(t_n + \frac{h}{2}, U_1\right) = U_2$$

en utilisant le rectangle droit.

On remplace alors $f(t_n + h/2, u(t_n + h/2))$ par la moyenne de U_1 et U_2
Enfin :

$$u(t_n + h) \to u_n + \frac{h}{2}\left(t_n + \frac{h}{2}, u(t_n + \frac{h}{2})\right) \to u_n + \frac{h}{2}(t_n + h/2, U_2) = U_3$$

Alors :

$$6S \to hf(t_n, u_n) + 4h\frac{f(t_n + h/2, U_1) + f(t_n + h/2, U_2)}{2} + hf(t_n + h, U_3)$$

On note conventionnellement :
$k_1 = hf(t_n, u_n)$
$k_2 = hf(t_n + h/2, u_n + k_1/2)$
$k_3 = hf(t_n + h/2, u_n + k_2/2)$
$k_4 = hf(t_n + h, u_n + k_3)$
Alors

$$u_{n+1} = u_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

Programmez cette nouvelle méthode. Quelle est alors la précision pour la fonction habituelle avec N = 4096 ?

5 La boîte de Pandore des approximations

Encore une fois, dès que vous tentez d'approcher une solution, de nombreux facteurs peuvent en fait vous en éloigner : une étude mathématique sophistiquée doit alors être envisagée (vous verrez ça en master...)
Voici quelques exemples apparemment anodins tirés de la littérature.

5.1 Problème mal posé mathématiquement

Que pensez-vous du problème :

$$\begin{cases} u'(t) = 2\sqrt{|u(t)|} \\ u(0) = 0 \end{cases}$$

5.2 Problème mal posé numériquement

Considérons le problème :

$$\begin{cases} u'(t) = 5u(t) - 5, \quad t \in [0, 50] \\ u(0) = 1 \end{cases}$$

et imaginons une petite perturbation de la valeur initiale : $u(0) = 1 + \varepsilon$. Estimez $\widetilde{u}(50) - u(50)$ pour $\varepsilon = 10^{-17}$ (de l'ordre de l'epsilon de Scilab).

5.3 Problème mal conditionné

Cette fois, le problème est le suivant :

$$\begin{cases} u'(t) = -150u(t) - 30 \\ u(0) = 1/5 \end{cases}$$

Vérifiez qu'il est bien posé mathématiquement et numériquement.
Approchons-le maintenant à l'aide de la méthode d'EULER explicite.
Vérifiez qu'on obtient

$$u_n = \frac{1}{5} + \left(1 - \frac{150}{N}\right)^n \left(u_0 - \frac{1}{5}\right)$$

Qu'obtenez-vous pour N = 50 et $u_0 = \frac{1}{5} + \varepsilon$?
Observez maintenant avec nos fonctions Scilab avec les différentes méthodes vues précédemment.

5.4 Problèmes raides

Il existe des problèmes qui résistent à toutes les méthodes usuelles. On trouve par exemple ces phénomènes en chimie où deux réactions ont des échelles de temps très différentes.
Considérez par exemple :

$$\begin{cases} u'(t) = -1000(u(t) - \exp(-t)) - \exp(-t) \\ u(0) = 0 \end{cases}$$

et testez vos fonctions sur $[0,1]$.
Cependant, vous pouvez utiliser la fonction **ode** de Scilab qui utilise au mieux les méthodes précédentes en adaptant de plus les pas ce qui permet parfois d'obtenir de bons résultats.
La doc nous dit en effet :

> *Le solveur lsoda du package ODEPACK est utilisé par défaut. Il choisit automatiquement entre un schéma prédicteur-correcteur d'Adams et un schéma adapté au systèmes raides (stiff) de type « Backward Differentiation Formula » (BDF).*
>
> *Initialement le schéma adapté aux systèmes non raides est choisi puis la méthode adaptée est ensuite choisie dynamiquement.*

Pour le problème mal conditionné du paragraphe précédent :

```
t = 0:0.02:1;
uo = 1/5+1D-15;
to = 0;
u = ode(uo,to,t,f);
plot(t,u)
```

6 Systèmes différentiels et ordre 2

6.1 Systèmes

Comme Scilab travaille sur des vecteurs, on peut considérer des fonctions dans $\mathbb{R}^2$ de manière naturelle.
Supposons que nous voulions résoudre le système :

$$\begin{cases} u_1'(t) = 3u_1(t) - 4u_2(t) \\ u_2'(t) = u_1(t) + 3u_2(t) \\ u_1(0) = 2 \\ u_2(0) = 0 \end{cases}$$

On utilise la même syntaxe que précédemment...mais avec une fonction $u : \mathbb{R} \to \mathbb{R}^2$:

```
deff('du=f(t,u)' , 'du(1) = 3*u(1) - 4*u(2) ; du(2) = u(1) + 3*u(2)');
t0 = 0 ; u0 = [2 ; 0] ; t = 0: 0.1 : 2 ;
z = ode(u0,t0,t,f) ;
plot(t,z(1,:),"ro") // donne u1
```

Comparez avec la solution attendue.

6.2 Ordre 2

On considère à présent l'équation bien connue du pendule pesant amorti : $y'' + y' + \sin(y) = 0$ avec $y(0) = 0$ et $y'(0) = 1$.
Posez $u = {}^t(y, y')$ et essayez de revenir à un système différentiel que vous résoudrez sur $[0,10]$.
Vous tracerez le portrait de phase.

TP N° 6

Dessine-moi une matrice...

Aujourd'hui, des mathématiques concrètes vont nous permettre de réduire, agrandir, assombrir, éclaircir, compresser, bruiter, quantifier,... une photo. Pour cela, il existe des méthodes provenant de la théorie du signal et des mathématiques continues. Nous nous pencherons plutôt sur des méthodes plus légères basées sur l'algèbre linéaire et l'analyse matricielle. Une image sera pour nous une matrice carrée de taille 2^9 à coefficients dans $[\![0, 2^9 - 1]\!]$. Cela manque de charme ? C'est sans compter sur Lena qui depuis quarante ans rend les maths sexy (la population hantant les laboratoires mathématiques et surtout informatiques est plutôt masculine ...).

Nous étudierons pour cela la « Décomposition en Valeurs Singulières » qui apparaît de nos jours comme un couteau suisse des problèmes linéaires : en traitement de l'image et de tout signal en général, en reconnaissance des formes, en robotique, en statistique, en étude du langage naturel, en géologie, en météorologie, en dynamique des structures, en....

Dans quel domaine travaille-t-on alors : algèbre linéaire, analyse, probabilités, topologie,... ? Un peu de tout cela et d'autres choses encore : cela s'appelle la mathématique.

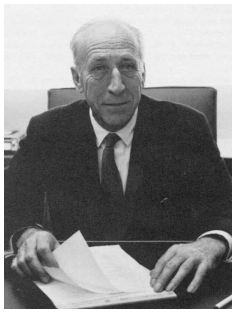
1 Lena

W. KAHAN (NÉ EN 1933)

Nous allons travailler avec des images qui sont des matrices de niveaux de gris. Notre belle amie Léna sera représentée par une matrice carrée de taille 2^9 ce qui permet de reproduire Léna à l'aide de $2^{18} = 262\,144$ pixels. Léna prend alors beaucoup de place. Nous allons tenter de compresser la pauvre Léna sans pour cela qu'elle ne perde sa qualité graphique. Une des méthodes les plus abordables est d'utiliser la décomposition d'une matrice en valeurs singulières.
C'est un sujet extrêmement riche qui a de nombreuses applications. L'algorithme que nous utiliserons (mais que nous ne détaillerons pas) a été mis au point par deux très éminents chercheurs en 1965 (Gene GOLUB, états-unien et William KAHAN, canadien, père de la norme IEEE-754). Il s'agit donc de mathématiques assez récentes, au moins en comparaison avec votre programme...
La petite histoire dit que des chercheurs américains de l'University of Southern California étaient pressés de trouver une image de taille 2^{18} pixels pour leur conférence. Passe alors un de leurs collègues avec, en bon informaticien, le dernier Playboy sous le bras. Ils décidèrent alors d'utiliser le poster central de la Playmate comme support...

2 La SVD

2 1 Le théorème



C. ECKART (1902-1973)

Un théorème démontré officiellement en 1936 par Carl ECKART et Gale YOUNG affirme alors que toute matrice rectangulaire A se décompose sous la forme :

$$A = U \times S \times {}^tV$$

avec U et V des matrices orthogonales et S une matrice nulle partout sauf sur sa diagonale principale qui contient les valeurs singulières de A rangées dans l'ordre décroissant. SYLVESTER s'y était déjà intéressé pour des matrices carrées réelles en 1889 mais ce résultat a semblé n'intéresser personne pendant des années.
Le court article de ECKART et YOUNG est disponible à l'adresse suivante : http ://projecteuclid.org/DPubS ?service=U
Ce qui est remarquable, c'est que n'importe quelle matrice admet une telle décomposition, alors que la décomposition en valeurs propres (la diagonalisation d'une matrice) n'est pas toujours possible. Et quand on dit n'importe quelle matrice, on ne rigole pas : toute matrice, même rectangulaire. C'est vraiment très puissant et à la fois simple mais cette décomposition n'a trouvé d'applications importantes qu'assez récemment (ce qui veut dire après 1899 à l'échelle de votre programme de prépa...) ce qui explique peut-être qu'on en parle si peu en premier cycle.
Cependant, la décomposition en éléments singuliers utilise la décomposition en éléments propres.
Notons r le rang de A et U_i et V_i les vecteurs colonnes de U et V. La décomposition s'écrit :

$$A = \begin{pmatrix} U_1 & U_2 & \cdots & U_r & \cdots & U_m \end{pmatrix} \times \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & \ddots & & \\ & & & & 0 & \end{pmatrix} \times \begin{pmatrix} {}^tV_1 \\ \vdots \\ {}^tV_r \\ \vdots \\ {}^tV_n \end{pmatrix}$$

ou formulé autrement :

$$\begin{aligned} A &= \sigma_1 \cdot U_1 \times {}^tV_1 + \sigma_2 \cdot U_2 \times {}^tV_2 + \cdots + \sigma_r \cdot U_r \times {}^tV_r + 0 \cdot U_{r+1} \times {}^tV_{r+1} + \cdots \\ &= \sigma_1 \cdot U_1 \times {}^tV_1 + \sigma_2 \cdot U_2 \times {}^tV_2 + \cdots + \sigma_r \cdot U_r \times {}^tV_r \end{aligned}$$

Il faut ensuite se souvenir que les σ_i sont classés dans l'ordre décroissant ce qui va avoir des applications importantes en informatique.

2 2 Interprétation géométrique

Comment interpréter géométriquement ce résultat ? Considérez une sphère de rayon 1 : que devient-elle transformée par A, i.e. par $U \times S \times {}^tV$? Faites un petit dessin en dimension 2. Que vaut $\|A\|_2 = \max_{\|x\|=1} \|Ax\|_2$?

Afin de répondre, un peu de vocabulaire d'abord. On a $AV_i = \sigma_i U_i$: on dit alors que U_i est un vecteur singulier à gauche pour la valeur singulière σ_i.
De même ${}^tAU_i = \sigma_i V_i$ et V_i est un vecteur singulier à droite pour la valeur singulière σ_i. Les bases (u_i) et (V_i) sont orthonormées : si l'on exprime la matrice de l'application linéaire associée à A dans ces bases, que pouvez-vous dire géométriquement ?

2.3 Un exemple simple

On suppose que la SVD d'une matrice A de taille $m \times n$ s'écrit $U \times S \times {}^tV$.

1. Quelles sont les dimensions de U et V ? Que pensez-vous de $A \times {}^tA$? Que représentent les colonnes de U pour $A \times {}^tA$?. Et les colonnes de V pour ${}^tA \times A$.
2. Calculez (à la main...) la SVD de $A = \begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix}$.
3. Shoot again avec $\begin{pmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{pmatrix}$

Quels liens existent entre la SVD et la recherche des éléments propres d'une matrice ? Avec le théorème spectral ? Vaut-il mieux utiliser $A \times {}^tA$ ou ${}^tA \times A$?

2.4 Approximation de rang minimum d'une matrice

Dans de nombreux domaines, on doit travailler avec de grosses matrices. Il est alors très important de déterminer une matrice $\widetilde{A_s}$ de même taille qu'une matrice A mais de rang s inférieur au rang r de A et qui minimise l'erreur commise pour une certaine norme (qui est le plus souvent la norme de FROBENIUS, i.e. la racine carrée de la somme des carrés des coefficients). Le théorème d'ECKART-YOUNG permet alors de dire que :

$$\min_{\text{rg}(X)\leqslant s} \|A - X\|_F = \|A - \widetilde{A_s}\|_F = \sqrt{\sum_{j=s+1}^{r} \sigma_j^2(A)}$$

avec $\widetilde{A_s} = \sigma_1 \cdot U_1 \times {}^tV_1 + \sigma_2 \cdot U_2 \times {}^tV_2 + \cdots + \sigma_s \cdot U_s \times {}^tV_s$: on considère que les éléments diagonaux de A sont nuls en-dessous d'un certain seuil. Nous allons l'illustrer informatiquement à défaut de le démonter. On voit tout de suite des conséquences pratiques, que nous allons également illustrer informatiquement dans le cas de la compression des images.

3 Manipulation d'images

3.1 Scilab

Nous utiliserons la fonction **`grayplot(x,y,z)`**, **`x`** étant un vecteur de taille n, **`y`** un vecteur de taille m et **`z`** une matrice de taille $n \times m$, qui permet d'afficher les points de coordonnées (**`x(i)`**,**`y(j)`**) avec la couleur **`z(i,j)`** selon une « **`colormap`** », i.e. une fonction qui à un entier associe une couleur selon le codage du langage.
Nous utiliserons une matrice carrée de taille 2^9 donnant la coloration des pixels selon une échelle de gris graduée de 0 à 255. Il suffit de récupérer le fichier **`lena.csv`** à l'adresse suivante : http://download.tuxfamily.org/tehes
Le préambule de notre fichier Scilab sera donc :

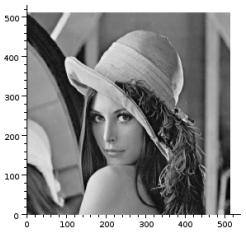
```
funcprot(0);
l = read('lena.csv',512,512); // on charge lena.csv placé dans le répertoire de travail
lena = l'; // on met lena à l'endroit en transposant la matrice
x = [1:512]; // la liste des abscisses
y = [512:-1:1]; // la liste des ordonnées inversée pour avoir l'origine en bas à gauche
xset('colormap',graycolormap(256)); // on choisit 256 niveaux de gris
isoview(0,512,0,512); // on choisit une vue orthonormée
```

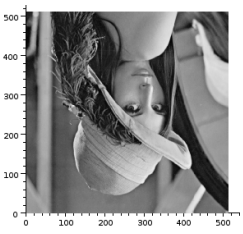
Il ne reste plus qu'à faire apparaître Lena...

```
grayplot(x,y,lena)
```

3.2 Manipulations basiques

Comment obtenir les images suivantes :

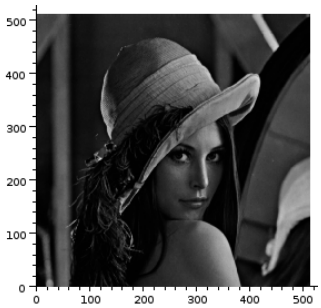
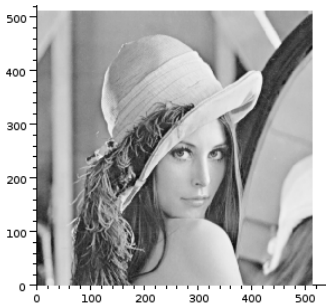
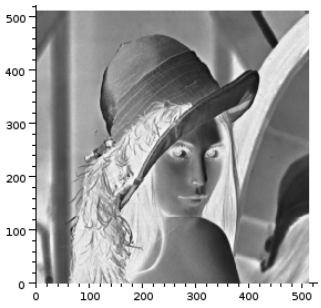




Sachant que l'échelle de gris est celle-ci :

0	18	36	54	72	90	108	128	146	164	182	200	218	236	255

comment obtenir les images suivantes :

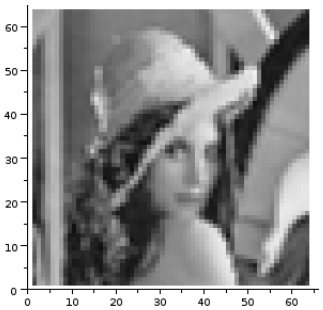
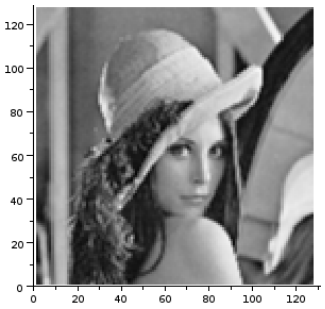
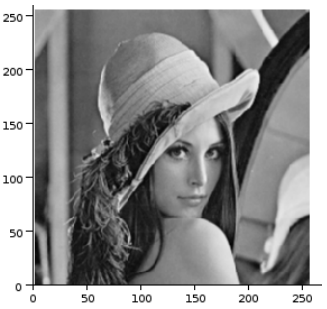




3 3 Gagner de la place

Une matrice de taille 2^9 contient 2^{18} entiers codés entre 0 et 2^8-1 ce qui prend pas mal de place en mémoire. Peut-on être plus économique sans pour cela diminuer la qualité esthétique de la photo ?

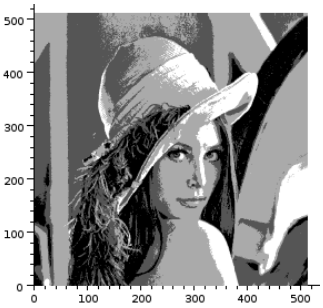
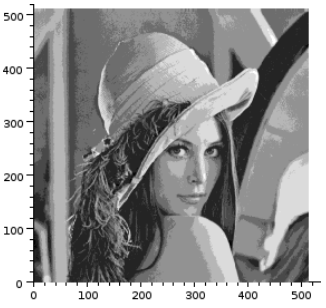
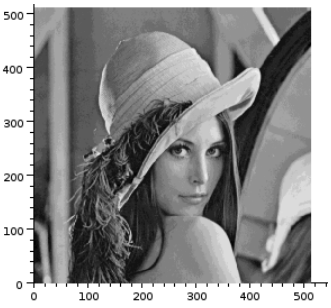
3 3 a Résolution

La première idée est de prendre de plus gros pixels, c'est-à-dire une matrice plus petite : on extrait par exemple régulièrement un pixel sur k (en choisissant k parmi une puissance de 2 inférieure à 2^9).
Comment obtenir par exemple ces images ?

3 3 b Quantification

On peut également réduire le nombre de niveaux de gris en regroupant par exemple tous les niveaux entre 0 et 63 en un seul niveau 0, puis 64 à 127 en 64, 128 à 191 en 128, 192 à 256 en 192.
Par exemple, avec 4, 8 puis 16 niveaux :

3 3 c Compression et SVD

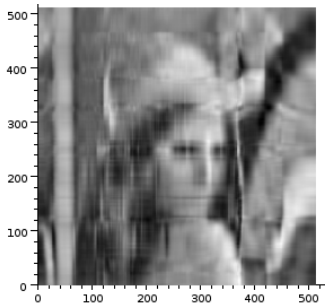
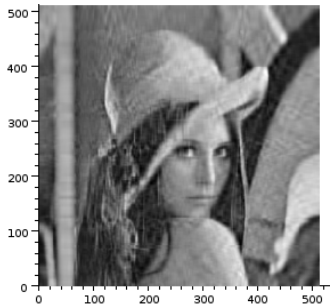
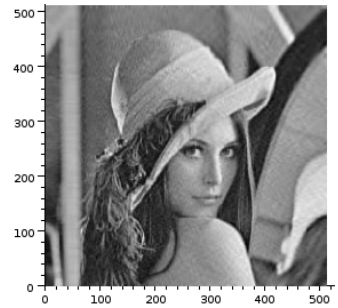
Tout langage orienté calcul comprend des fonctions liées à la décomposition en éléments simples. Par exemple, sur Scilab, la fonction **spec(mat)** renvoie le spectre de la matrice **mat**.
Scilab a aussi sa fonction **svd** qui renvoie la décomposition en valeurs singulières sous la forme du triplet U, S et V.
Commentez les résultats des commandes suivantes :

```
gsort(sqrt(spec(lena*lena')))
[U,S,V] = svd(lena)
diag(S)
```

et également la figure suivante :

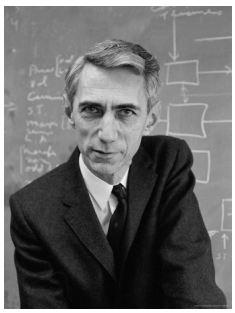
```
plot(x,diag(S)/S(1,1))
```

En vous inspirant des commentaires faits dans les paragraphes précédents sur la SVD, expliquez comment cette décomposition permet de compresser une image. Quantifiez également le niveau de compression.
Voici trois niveaux de compression :

3 4 Enlever le bruit

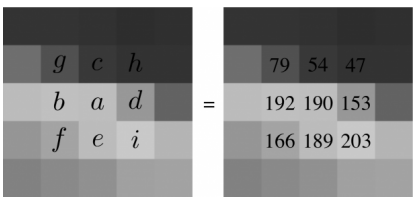
La transmission d'informations a toujours posé des problèmes : voleurs de grands chemins, poteaux télégraphiques sciés, attaques d'indiens, etc.


C.E. SHANNON (1916-2001)

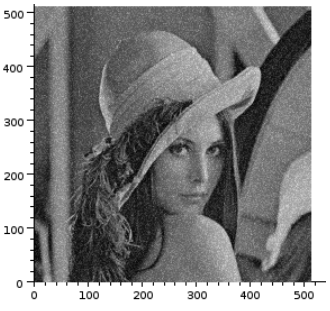
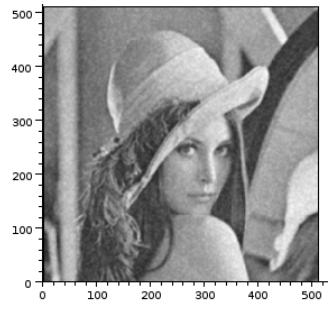
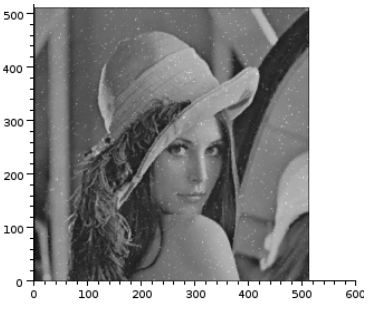
Claude Elwood SHANNON (1916 - 2001) est un mathématicien-inventeur-jongleur américain qui, suite à son article « *A mathematical theory of communications* » paru en 1948, est considéré comme le fondateur de la *théorie de l'information* qui est bien sûr une des bases de...l'*informatique*.
L'idée est d'étudier et de quantifier l'« information » émise et reçue : quelle est la compression maximale de données digitales ? Quel débit choisir pour transmettre un message dans un canal « bruité » ? Quel est le niveau de sûreté d'un chiffrement ?...
La théorie de l'information de SHANNON est fondée sur des modèles probabilistes : leur étude est donc un préalable à l'étude de problèmes de réseaux, d'intelligence artificielle, de systèmes complexes.
Par exemple, dans le schéma de communication présenté par SHANNON, la source et le destinataire d'une information étant séparés, des perturbations peuvent créer une différence entre le message émis et le message reçu. Ces perturbations (bruit de fond thermique ou accoustique, erreurs d'écriture ou de lecture, etc.) sont de nature *aléatoire* : il n'est pas possible de prévoir leur effet. De plus, le message source est par nature *imprévisible* du point de vue du destinataire (sinon, à quoi bon le transmettre).
SHANNON a également emprunté à la physique la notion d'entropie pour mesurer le désordre de cette transmission d'information, cette même notion d'entropie qui a inspiré notre héros national, Cédric VILLANI...
Mais revenons à Lena. Nous allons simuler une Lena bruitée :

```
function B = bruit(m,ratio)
  n = length(m(1,:));
  B = m + 75*grand(n,n,'bin',1,ratio/100)
endfunction

grayplot(x,y,bruit(lena,20))
```

Il existe de nombreuses méthodes, certaines très élaborées, permettant de minimiser ce bruit. Une des plus simples est de remplacer chaque pixel par la moyenne de lui-même et de ses 8 autres voisins directs, voire aussi ses 24 voisins directs et indirect, voire plus, ou de la médiane de ces séries de « cercles » concentriques de pixels, comme on le voit sur cette figure (extraite du site `Images des mathématiques`) :



Créez des fonctions Scilab qui feront le boulot. Vous utiliserez `mean(mat)` et `median(mat)` pour obtenir par exemple :

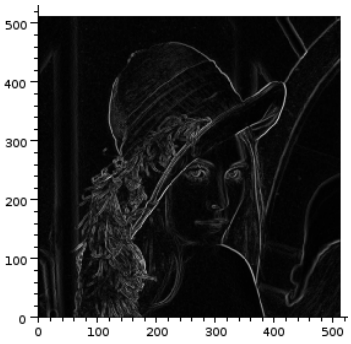
où l'on trouve de gauche à droite l'image bruitée originale puis sa correction par moyenne sur 25 pixels et par médiane sur 9 pixels.

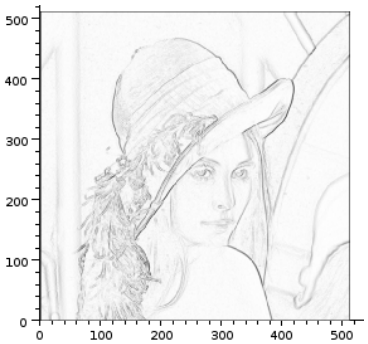
3 5 Détection de bords

Pour sélectionner des objets sur une image, on peut essayer de détecter leurs bords, i.e. des zones de brusque changement de niveaux de gris.
On remplace alors un pixel par une mesure des écarts des voisins immédiats donnée par exemple par :

$$\sqrt{\left(m_{i,j+1}-m_{i,j-1}\right)^2+\left(m_{i+1,j}-m_{i-1,j}\right)^2}$$

On obtient alors au choix :





4 MAPLE et les matrices

6 - 1 Centrale, Oral 2010

Si $n \geq 2$, on note $= (\mathbf{e}_1, ..., \mathbf{e}_n)$ la base canonique de $\mathbb{R}^n$.
Soit $f_n \in \mathcal{L}(\mathbb{R}^n)$ tel que $f_n(\mathbf{e}_i) = \mathbf{e}_{i+1}$ si $i \in [\![1, n-1]\!]$ et $f_n(\mathbf{e}_n) = \mathbf{e}_1$.

1. Écrire une procédure permettant d'obtenir la matrice A_n de f_n dans la base $\mathbf{e}$.
2. Dans cette question, $n = 6$.
   (a) Calculer le polynôme caractéristique de A_6. Factoriser ce polynôme sur $\mathbb{R}[X]$.
   On écrit $\chi = \prod_{1\leq k\leq p} R_k$ où les R_k sont irréductibles dans $\mathbb{R}[X]$.
   (b) Déterminer $\ker R_k(f_6)$ pour $k \in [\![1, p]\!]$. Montrer que $\mathbb{R}^6 = \ker R_1(f_6) \oplus \cdots \oplus \ker R_p(f_6)$.
   (c) Donner la matrice de f_6 dans une base adaptée à cette décomposition.
3. Si $n \geqslant 2$, calculer le polynôme caractéristique χ de A_n. Le factoriser sur $\mathbb{R}[X]$.

Voyons quelques commandes utiles issues de la bibliothèque `LinearAlgebra` :

```
with(LinearAlgebra);
# a)
A := proc(n);
    M := Matrix(n);
    for i to n-1 do
        M[i+1, i] := 1;
        end do;
    M[1, n] := 1;
    return M
    end proc;

M := A(6);
I6 := IdentityMatrix(6);
latex(M);

# b)
#    i)
chi := CharacteristicPolynomial(M, x);
chi := factor(chi);
R := op(chi);
```

```
#    ii)
# Détermination des sev Ker(Rk(f))
F1 := NullSpace(M - I6);
F2 := NullSpace(M + I6);
F3 := NullSpace(M^2 + M + I6);
F4 := NullSpace(M^2 - M + I6);

P := <F1[1]|F2[1]|F3[1]|F3[2]|F4[1]|F4[2]>;
# Trois manières de vérifier que P est inversible :
Determinant(P);
NullSpace(P);
ColumnSpace(P);

#    iii)
# Matrice de f_6 dans un base adaptée :
P^(-1).M.P;

# c)
for n from 2 to 10 do
    CharacteristicPolynomial(A(n), x),\
    factor(CharacteristicPolynomial(A(n), x));
    end do;
```

6 - 2 Centrale, Oral 2010

1. Démontrer que, si deux endomorphismes u et v d'un espace vectoriel E commutent, alors, les sous-espaces propres de u et l'image de u sont stables par v.
   Dans les deux cas suivants :

$$A = \begin{pmatrix} 20 & 12 & -4 & 12 \\ -4 & -3 & 9 & -5 \\ -4 & 1 & 5 & -5 \\ -8 & -10 & 6 & -2 \end{pmatrix} \quad \text{et} \quad A = \begin{pmatrix} -12 & -16 & -8 & -4 \\ 4 & 13 & 1 & -1 \\ 4 & 5 & 9 & -1 \\ 8 & 10 & 2 & 6 \end{pmatrix}$$

2. Préciser les matrices qui commutent avec A (structure, dimension, base éventuelle).
3. Etudier dans $\mathcal{M}_4(\mathbb{R})$, puis dans $\mathcal{M}_4(\mathbb{C})$, l'équation

$$X^2 = A$$

   (nombre de solutions, un exemple de solution quand il y en a, somme et produit des solutions quand elles sont en nombre fini).

```
with(LinearAlgebra):
# a)
A := <<20|12|-4|12>, <-4|-3|9|-5>, <-4|1|5|-5>, <-8|-10|6|-2>>;
(vpA, P) := Eigenvectors(A);
P^(-1).A.P;

B := <<-12|-16|-8|-4>, <4|13|1|-1>, <4|5|9|-1>, <8|10|2|6>>;
(vpB, Q) := Eigenvectors(B);
Q^(-1).B.Q;

# c)
Determinant(A);
Determinant(B);
```

6 - 3 Centrale, Oral 2010

Soit $(\mathcal{E})$ la surface d'équation $x^2 + y^2 + 4z^2 = 1$.

1. Représenter $(\mathcal{E})$.

2. Soit R_t la rotation d'axe dirigé et orienté par $\mathbf{u} = \begin{pmatrix} 4 \\ 3 \\ 0 \end{pmatrix}$ et d'angle t.
   Soit (x', y', z') l'image de (x, y, z) par R_t. Exprimer (x', y', z') en fonction de (x, y, z).
3. Déterminer une équation de $(\mathcal{E}_t)$ l'image de $(\mathcal{E})$ par R_t.

Pour le 3), un petit Joker...

```
# Avec LinearAlgebra

M := <x, y, z>;
Mt := Transpose(Rt).M;
eq := x^2 + y^2 + 4*z^2 - 1;
eqt := simplify(subs({x=Mt[1], y=Mt[2], z=Mt[3]}, eq));

# Avec linalg

M := matrix(3,1,[x, y, z]);
Mt := evalm(transpose(Rt) &* M);
eq := x^2 + y^2 + 4*z^2 - 1;
eqt := simplify(subs({x=Mt[1], y=Mt[2], z=Mt[3]}, eq));
```

6 - 4 Centrale, écrit PC 2012

On suppose $a = 0$. Écrire en Maple ou en Mathematica une fonction f de six variables telle que, pour tous a, b, c, d, e réels et tout p entier naturel, $f(a, b, c, d, e, p)$ soit le terme d'ordre p, c'est-à-dire u_p , de la suite récurrente (u_n) définie par

$$\begin{cases} u_0 = d \\ u_1 = e \\ \forall n \in \mathbb{N},\ au_{n+2} + bu_{n+1} + cu_n = 0 \end{cases}$$

6 - 5 Centrale, écrit PC 2011

On considère l'ensemble des suites (x_k) de réels vérifiant :

$$x_0 > 0 \quad \text{et} \quad x_{k+1} = x_k \cdot \frac{x_k^2 + 3}{3x_k^2 + 1} \quad \text{pour } k \in \mathbb{N}$$

Écrire une instruction en Maple ou Mathematica permettant de calculer les trente premiers termes de la suite (x_k) telle que $x_0 = 0,1$.

6 - 6 Centrale, écrit MP 2011

Pour $n \in \mathbb{N}^*$, $A \in \mathcal{S}_n(\mathbb{R})$ et $i \in [\![1; n]\!]$, on note $A^{(i)}$ la matrice carrée d'ordre i extraite de A, constituée par les i premières lignes et les i premières colonnes de A.
Écrire une procédure, dans le langage Maple ou Mathematica, qui prend en entrée une matrice $M \in \mathcal{S}_n(\mathbb{R})$ et qui, en utilisant la caractérisation

$$A \text{ est définie positive } \Leftrightarrow \forall i \in [\![1; n]\!], \det(A^{(i)}) > 0$$

renvoie « true » si la matrice M est définie positive, et « false » dans le cas contraire.

CONCOURS CENTRALE•SUPÉLEC

Mathématiques 2

Oral

MP

On considère le système différentiel S suivant :

$$S : \begin{cases} x'(t) = x(t)\,(10 - x(t) - y(t)) \\ y'(t) = y(t)\,(-6 + x(t) - y(t)) \end{cases}$$

Dans tout l'exercice, il est fortement conseillé d'utiliser le logiciel de calcul formel pour les calculs.

1. Déterminer l'unique solution (x,y) de S avec x et y fonctions constantes non nulles.

Dans la suite, on notera α, β les valeurs respectives de ces fonctions constantes x et y.

2. À l'aide du logiciel de calcul formel, déterminer l'allure des solutions vérifiant respectivement les conditions initiales suivantes :

$$(x(0), y(0)) \in \{(10,10),\ (1,1),\ (2,6)\}$$

   Que remarquez-vous ?

Soit maintenant $X = (x,y)$ une solution maximale de S. On va justifier l'existence d'un voisinage U de $\Omega = (\alpha,\beta)$, dans $\mathbb{R}^2$ tel que, s'il existe t_0 pour lequel $X(t_0)$ appartient à U, alors la solution X est définie sur $[t_0, +\infty[$ et converge vers Ω à vitesse exponentielle.

3. Calculer la matrice jacobienne de l'application $f : (x,y) \mapsto ((x(10 - x - y),\ y(-6 + x - y))$ au point Ω et déterminer ses valeurs propres.

4. Soit $A \in \mathcal{M}_2(\mathbb{R})$ dont les valeurs propres sont complexes non réelles et qu'on note $a + ib$ et $a - ib$. Justifier que A est semblable à

$$\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

   Montrer qu'il existe un produit scalaire sur $\mathbb{R}^2$ noté $(\cdot \mid \cdot)$ tel que :

$$\forall X \in \mathbb{R}^2, \quad (AX \mid X) = a\|X\|^2$$

   où $\|\cdot\|$ désigne la norme euclidienne associée à ce produit scalaire.

5. Justifier l'existence d'un réel $r > 0$ tel que

$$\forall X \in B(\Omega,r), \quad (f(X) - f(\Omega) \mid X - \Omega) \leqslant -\|X - \Omega\|^2$$

   Conclure à l'aide de l'application $t \mapsto \|X(t) - \Omega\|^2$

Dans cet exercice, on confondra polynôme et fonction polynomiale.

1. Soit $n \in \mathbb{N}$. Montrer qu'il existe un unique polynôme $P_n \in \mathbb{R}_n[X]$ tel que, pour tout polynôme $Q \in \mathbb{R}_n[X]$:

$$\int_0^1 P_n(t)Q_n(t)dt = Q(0)$$

2. Cette question est à traiter avec le logiciel de calcul formel.

   a. Calculer les polynômes P_n pour $n = 0, \ldots, 5$. On pourra déterminer chaque P_n en résolvant un système de $n+1$ équations.

   b. Sur un même graphique, représenter $P_0, \ldots, P_5$ sur l'intervalle [0,1].

3. Montrer que P_n s'annule n fois sur l'intervalle]0,1[. Que peut-on en conclure pour P_n ?

4.

   a. Pour quelles valeurs du réel x l'intégrale

$$\int_0^1 P_n(t)t^x dt$$

   est-elle définie ?

   b. À l'aide du logiciel de calcul formel, obtenir une expression de

$$\int_0^1 P_n(t)t^x dt$$

   pour $n = 0, \ldots, 3$ sous forme d'une fraction rationnelle factorisée.

   Déterminer alors la valeur de cette intégrale pour tout n.

   c. Calculer $P_n(0)$.

   d. Déduire de ce qui précède la valeur de

$$\max\{|Q(0)|/Q \in \mathbb{R}_n[X], \int_0^1 Q^2(t)dt = 1\}$$

5.

   a. Déterminer un produit scalaire sur $\mathbb{R}[X]$ et une propriété P convenable faisant de la famille (P_n) l'unique famille de polynômes échelonnée en degré, orthogonale pour ce produit scalaire et vérifiant la propriété P.

   b. Retrouver ainsi, par une autre méthode, les polynômes $P_0, \ldots, P_5$ de la question 2.

TP N°

7

Systèmes dynamiques

Henri Poincaré s'est intéressé à la fin du XIX^{e} siècle à l'évolution des systèmes dynamiques en partant d'un problème de mécanique céleste : le soleil est beaucoup plus massif que les planètes, c'est pourquoi les planètes décrivent des trajectoires elliptiques autour de lui. Cependant, chaque planète est également influencée par ses congénères mais dans une mesure bien moindre : les trajectoires elliptiques sont certes déformées, mais de manière extrêmement lente. Que va-t-il se passer dans le long terme : ces micro-changements vont-ils avoir une grande influence ou bien rester négligeable ? L'étude des systèmes dynamiques était née : on étudie un système très simple a priori, décrit par un petit nombre de paramètres dont la loi dévolution est déterminée, et on étudie son évolution à long terme : la simplicité des systèmes cache parfois une incroyable complexité de l'évolution à long terme.

Cette théorie a été très développée au cours su XX^{e} siècle et popularisée par la médiatisation de certains résultats de la théorie du chaos, de l'étude des fractales.

Elle pose le problème de la prédictibilité : le futur peut-il être déduit du présent ? C'est un domaine passionnant, riche mathématiquement et en lien avec la physique, la biologie, les sciences de l'ingénieur, l'informatique...

C'est enfin un domaine mathématique très vivant : le grand favori pour la médaille Fields 2014 est en effet le brésilien Artur Avila dont le domaine de recherche est justement lié aux systèmes dynamiques.

1 Logistique

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 $R \geqslant 1$ telle que :

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

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

$$u_{n+1} = Ru_n(1-u_n)$$

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.

Recherche

Déterminez une fonction `s = logistic(x,l,n)` qui donne les n premiers éléments de l'orbite de x avec $f(x) = \ell(1-x)x$.
Observez les orbites de différents points pour $\ell = 2$ ou $\ell = 3$.

Voici une petite fonction à commenter :

```
function escargot(u0,n,l)
 function y = f(x)
   y = l*(1-x)*x
 endfunction
 u = u0
 lx = []
 ly = []
 for k = 1:n do
   ly = [ly,u,f(u)]
   lx = [lx,u,u]
   u = f(u)
 end
 clf()
 x = 0:0.01:1
 plot(lx,ly,"r");plot(x,x,"k");plot(x,l*(1-x).*x,"b")
endfunction
```

Voici un peu de vocabulaire :

– un point périodique est son propre itéré : il existe un entier n tel que $f^n(x) = x$;
– le plus petit de ces entiers est la période de x.

Recherche

Montrez que si f est une fonction continue et monotone de $[0,1]$ dans lui-même, alors toute orbite converge vers un point fixe ou un point périodique de période 2.

 - 3 février 2013