Correction de TP 1

Exercices 1 à 3 de la feuille de TP no 1

```
//programme TP1.sce
clear
v=[1 2 3 4 5];
A=v'*v;
l1=A(1,:)
l2=A(5,:)
l1*l2'
tril(A)
triu(A)
B=rand(5,5)
A+B
A-B
A*B
inv(B)*A
B\A
trace(A)
rank(A)
spec(A)
//Exercice 1, f)
n=5;
C=diag(2*ones(n,1),0)-diag(ones(n-1,1),-1)-diag(ones(n-1,1),1)

//Exercice 2
u=poly([1 -2 1 -2],'z','coeffs')
v=poly([1 -2 3 -6],'z','coeffs')
u*v
u/v
w=derivat(u/v)
horner(w,4)

//Exercice 3
M=[u v;u*v u/v]
horner(M,1)
S=['ceci' 'est';'une' 'matrice']
v=[1 2 3 4 5];
A=v'*v;
A1=modulo(A,2)==0
A2=bool2s(A1)
A.*A2
A(A1)
and(A<>0)
```

Exercices 4 et 6 de la feuille de TP no 1

```
//Programme "TP1suite.sce"
clear
clf
//Exercice 4
help integrate
integrate('1/sqrt(1-x^2)','x',0,1)
integrate('(4*x.^4+4*x.^3-2*x.^2-10*x+6)./(x.^5+7*x.^4+16*x.^3+10*x.^2)','x',1,10)
integrate('exp(-x)*log(x)','x',0,100)
//NB si on replace 100 par %inf un message d'erreur s'affiche


deff('y=f(x)','y=1./sqrt(1-x.^2)')
intg(0,1,f)
deff('y=f(x)','y=(4*x.^4+4*x.^3-2*x.^2-10*x+6)./(x.^5+7*x.^4+16*x.^3+10*x.^2)')
intg(1,10,f)
deff('y=f(x)','y=exp(-x)*log(x)')
intg(0,100,f)

//Exercice 6
deff('y=f(x)','y=sin(x)+x')
x=[0:0.01:%pi];
plot2d(x,f(x))
xtitle('premiers dessins','x','y')
plot2d(x,cos(x),-1)
halt()
clf
plot2d(x',[f(x') cos(x')],leg='f@cos')
halt()
clf
x=[0:0.05:2*%pi];
y=[0:0.025:2*%pi];
z=cos(x')*sin(y);
plot3d(x,y,z)
```

Solution de l'exercice 2

```
//Programme "pluspetitreel.sce"
//calcul du plus petit reel et plus grand reel >0
clear
x=1;
while x>0
      y=x;
      x=x/2;
end
'le plus petit reel strictement positif est '
```

```
y  //y est le plus petit reel >0
N1=log2(y)

x=1;
while x<%inf
      y=x;
      x=2*x;
end
'le plus grand reel est'
y
N1=log2(y)
```

Remarquer que si N est le nombre de chiffres dans la mantisse, et P le nombre de chiffre dans l'exposant (avec la convention que l'exposant p prend les valeurs entre $+2^{P-1}$ et $-(2^{P-1}-1)$, le plus petit réel $x > 0$ normalisé représentable est

$$x = 1,0 \cdot 2^{-(2^{P-1}-1)},$$

et le plus grand réel représentable est

$$X = (2 - 2^{-N}) * 2^{2^{P-1}}.$$

En fait, pour des raisons de calcul et d'arrondis, la norme IEEE 754 autorise également des nombres non normalisés de la forme (en double précision)

$$0, a_1 a_2 \cdots a_N 2^{-1023},$$

de sorte que le plus petit réel > 0 représentable est obtenu pour $a_1 = a_2 = \cdots = a_{N-1} = 0$ et devient

$$x = 2^{-N} \cdot 2^{-2^{P-1}-1}.$$

Ainsi,

$$\begin{cases} log_2(x) = -N - (2^{P-1} - 1), \\ log_2(X) = 2^{P-1} + 1, \end{cases}$$

donc la connaissance de x et X permet de calculer N et P. Dans notre cas, on trouve $log2(x) = -1074 = -51 - 1023$, et $log_2(X) = 1023$. Comme $2^{10} = 1024$, on vérifie que cela correspond bien à $N = 52$ et $P = 11$ (à une unité près, qui dépend des règles d'arrondis du logiciel et du test d'arrêt dans notre algorithme).

Solution des exercices 3 et 4

```
//Programme condition.sce
//Quelques essais de calcul
clear
A=[1 10;0 1] //remarquer que A n'est pas symetrique
//deux manieres de calculer la condition en norme 2
'condition 2 = '
cond(A)
//pour comparer
Lambda=spec(A'*A);
sqrt(max(Lambda)/min(Lambda))
```

```
//condition en norme 1 (deux manieres)
'condition 1 = '
1/rcond(A)
norm(A,1)*norm(inv(A),1)
//condition en norme infinie (trois manieres)
'condition inf = "
1/rcond(A')
norm(A,'inf')*norm(inv(A),'inf')
norm(A',1)*norm(inv(A'),1)

//condition de la matrice du laplacien
clear
x=[];
y2=[];//tableau du conditionnement en norme 2
y1=[];//tableau du conditionnement en norme 1
yinf=[];//tableau du conditionnement en norme infinie
kmax=6;
for k=2:kmax
    N=2^k
    A=2*diag(ones(N,1),0)-diag(ones(N-1,1),-1)-diag(ones(N-1,1),1);
    x=[x N];
    y2=[y2 cond(A)];
    y1=[y1 1/rcond(A)];
    yinf=[yinf 1/rcond(A')];
end
clf
plot2d(x',[y2',y1',yinf'],logflag="ll",leg="K2@K1@Kinf")
```

correction de l'exercice 5 (matrice de Van der Monde) SCILAB

```
//condition de la matrice de Van der Monde
TabN=[];//tableau des abscisses
y2=[];//tableau des ordonnees (condition en norme 2)
y1=[];//idem en norme 1
yinf=[];//idem en norme infinie
Nmax=40;
for N=1:Nmax
    x=[0:1/N:1];//construction des x_i, points equirepartis
    V=zeros(N+1,N+1);
    //construction de la matrice de Van der Monde
    for i=0:N
        for j=0:N
             V(i+1,j+1)=x(i+1)^j;
        end
    end //NB : la numerotation des tableaux commence toujours a 1 en Scilab
    TabN=[TabN N];
    y2=[y2 cond(V)];
    y1=[y1 1/rcond(V)];
    yinf=[yinf 1/rcond(V')];
```

```
end
clf
plot2d(TabN',[y2',y1',yinf'],logflag="nl",leg="K2@K1@Kinf")
xtitle('condition de la matrice de Van der Monde','N','Kappa')
```

NB : on n'a traité que le cas des points équidistants sur $[0, 1]$.

Le dessin affiché par ce programme est donne dans la Figure 1. On constate que le conditionnement croit de maniere exponentielle en fonction de N, puisque le logarithme de K_N, où $K_N = cond(V_N)$, est une fonction affine de N : $log_{10}(K_N) \approx aN + b$, ou encore, $K_N \approx 10^{aN+b} = C\delta^N$, avec $C = 10^b$ et $\delta = 10^a > 1$.

On remarque également que lorsque le conditionnement est de l'ordre de 10^{16}, les calculs ne sont plus fiables (les résultats sont de l'ordre de 10^{16}, de manière aléatoire). Cela est directement lié à la précision des calculs dans SCILAB ($\epsilon \approx 10^{-16}$). La matrice de Van der Monde est connue pour avoir un très mauvais conditionnement : pour $N \geq 25$ (ce qui n'est pas très grand pour une taille de matrice), on dépasse la précision de l'ordinateur.

Si on compare avec l'exemple précédent où le conditionnement est en $O(N^2)$, il faudrait une taille de matrice $N \approx 10^8$ pour arriver à la limite de précision.

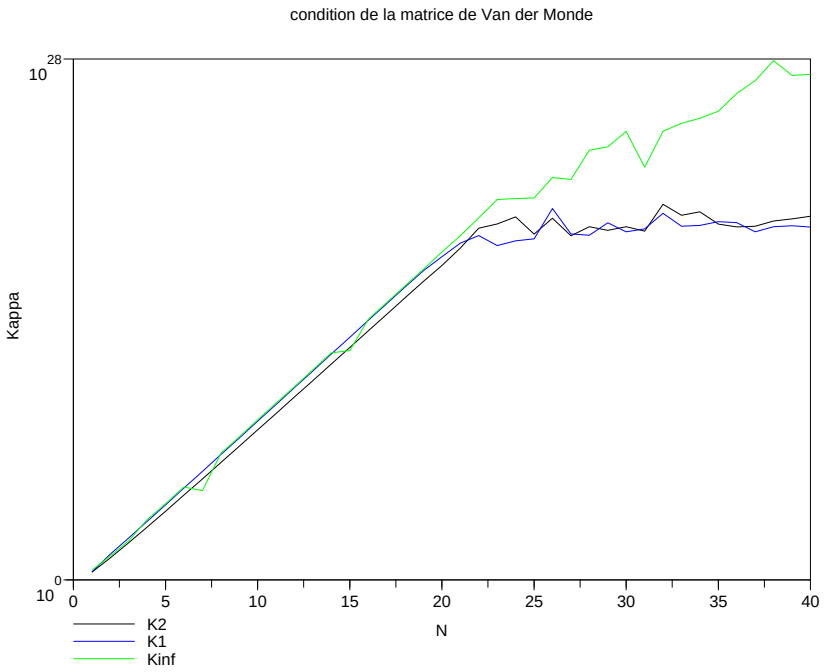

Figure 1: Condition de la matrice de Van der Monde