



scilab

III - Boucles « pour », graphes

Lycée

Auteur : Raymond Moché

Mots-clefs : Boucle « pour », « sum », « \cdot^2 », « `afficher(A,B)` », « `afficher('A ='+string(A))` », « `input` », boucles « pour » imbriquées, « `trier` », « `size(A,'r')` », « `size(A,'c')` », « `A(j,a:b)` », « `A(j,:)` », « `clf` », « `clear` », « `plot` », résolution graphique d'une équation, zoom, annulation du zoom, « `subplot` », « `zeros` », « `ones` », « `tirage_entier` », « `taille` », « `frequence` ».

Référence :

✓ Livret de présentation de « scilab pour les lycées » (2010) et Mise à jour et compléments (mars 2011) :

<http://www.scilab.org/education/lycee/docs>

✓ Aide scilab 5.3.2

http://help.scilab.org/docs/5.3.2/fr_FR/index.html

Remarque : Nous appelons souvent les tableaux « matrice » ou « vecteur-ligne » ou « vecteur-colonne » suivant le cas. Mais nous ne ferons jamais de calcul matriciel autre que des opérations composante par composante. Il n'y a pas d'inconvénient à se contenter du terme « tableau ».

Syntaxe de la boucle « pour »

for x =vecteur, instruction; end

L'instruction est exécutée, x prenant successivement toutes les valeurs du vecteur.

Tracés élémentaires : commande « plot »

- ✓ Pour tracer le graphe d'une fonction, « scilab » calcule les coordonnées d'un certain nombre de points de ce graphe et relie les points consécutifs par un segment de droite.
- ✓ Les abscisses de ces points sont collectées dans le vecteur-ligne X , les ordonnées correspondantes dans le vecteur-ligne Y .
- ✓ « scilab » se charge de tout, mais on peut lui imposer de nombreuses options. Par exemple la commande « `orthonorme` » oblige le repère à être orthonormé.
- ✓ Très souvent, Y est calculé à l'aide d'une boucle « pour ».

Liste des exercices :

Énoncé n°1 : [*] Sommes de nombres entiers.

Énoncé n°2 : [*] Deux boucles « pour ».

Énoncé n°3 : [*] Boucle « pour » dans une boucle « pour ».

Énoncé n°4 : [*] Tracé d'un triangle.

Énoncé n°5 : [*] Graphe de la fonction tangente entre 0 et π .

Énoncé n°6 : [*] Sauts de puce.

Énoncé n°7 : [**] Résolution graphique d'une équation.

Énoncé n°8 : [**] Matrice Zorro.

Énoncé n°9 : [**] Aiguilles de l'horloge.

Énoncé n°10 : [**] Calculs d'effectifs et de fréquences.

Énoncé n°11 : [***] Matrice-croix.

Énoncé n°12 : [***] Tracés de polygones réguliers inscrits dans un cercle.

Énoncé 1 [*] Sommes de nombres entiers

- 1 - Écrire la suite des nombres entiers de 1 à 500.
- 2 - En déduire la somme de ces nombres.
- 3 - Calculer la somme des carrés des nombres entiers de 1 à 500.

Mots-clefs : Boucle « pour », « sum », « .^ 2 », « afficher(A,B) », « afficher('A =' +string(A)) »

Le listing ci-dessous contient 3 solutions, de plus en plus compactes, de cet exercice.

Listing 1 – Sommes de nombres entiers

```
// Liste X des 500 premiers entiers, leur somme S.
X=[]; // La liste est vide au début. On va la remplir.
S=0;
for i=1:500
    X=[X, i];
    S=S+i;
end;
// Somme T des carrés des 500 premiers entiers.
T=0;
for i=1:500
    T=T+i^2;
end;
// Voici le même algorithme avec des commandes plus sophistiquées :
Xbis=1:500;
Sbis=sum(X);
Ybis=Xbis.^2; // Si A est un tableau, A.^2 est le tableau des carrés.
Tbis=sum(Ybis);
// Si on veut seulement les deux sommes, on tapera
Ster=sum(1:500);
Tter=sum([1:500]^2);
```

Ce script est téléchargeable (« Sommes.sce »). Il suffit de le charger dans « SciNotes ».

```
Console Scilab
-->exec('/Users/raymondmoche/303-Sommes.sce', -1)
-->T
T =
    41791750.
-->Tter
Tter =
    41791750.
-->
```

On aurait pu inclure les commandes d'affichage dans le script, par exemple :

Listing 2 – source scilab

```
// Affichage  
afficher (S, Sbis, Ster, T, Tbis, Tter);
```

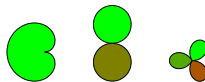
On rappelle que dans ce cas, « scilab » affiche les résultats à l’envers. Dans certains cas, il vaut mieux choisir l’affichage des items un par un :

Listing 3 – source scilab

```
afficher ('S=□'+string(S));  
afficher ('Sbis=□'+string(Sbis));  
afficher ('Ster=□'+string(Ster));  
afficher ('T=□'+string(T));  
afficher ('Tbis=□'+string(Tbis));  
afficher ('Tter=□'+string(Tter));
```

En effet, on peut alors arrêter séparément les affichages trop longs (à la demande de « scilab »). Cela aurait été utile si on avait demandé aussi l’affichage de X et de Y .

```
Console Scilab  
-->exec('/Users/raymondmoche/303-Sommes.sce', -1)  
  
S= 125250  
  
Sbis= 125250  
  
Ster= 125250  
  
T= 41791750  
  
Tbis= 41791750  
  
Tter= 41791750  
  
-->
```



Énoncé 2 [*] Deux boucles « pour »

Ranger un tableau numérique dans l'ordre décroissant.

Mots-clefs : commandes « trier », « size(A,'r') », « size(A,'c') », « A(j,a : b) », « A(j, :) ».

- ✓ La commande « trier » trie les vecteurs, autrement dit des tableaux à une seule ligne, dans l'ordre croissant ou décroissant suivant l'option choisie. Comment l'utiliser pour trier un tableau à n lignes et m colonnes?
- ✓ Dans le script ci-dessous, on écrit en ligne le tableau donné, on le trie et on le reconstitue.
- ✓ En exécutant le script (voir l'image de la console « scilab » au-dessous), on a saisi à la main une matrice T . C'est long.
- ✓ Habituellement, on engendre une matrice de nombres aléatoires, voir la commande « rand ».
- ✓ « size(A,'r') » est le nombre de lignes de la matrice A .
- ✓ « A(j, :) » et « A(j,a : b) » sont respectivement la $j^{\text{ème}}$ ligne de A et la partie de cette ligne située entre les colonnes a et b (comprises).

Listing 4 – Ranger une matrice

```
// Ranger un tableau numérique dans l'ordre décroissant.
T=input('T=');
n=size(T,'r'); // Nombre de lignes de T.
m=size(T,'c'); // Nombre de colonnes de T.
Ligne=[]; // On se prépare à écrire T en ligne.
for j=1:n
    Ligne=[Ligne,T(j,:)]; // T(j,:) : jième ligne de T ; noter la virgule.
end
Ligne=trier(Ligne,'<'); // Ligne est maintenant dans l'ordre décroissant.
T=Ligne(1,1:n);
for j=2:n
    T=[T;Ligne(1,(j-1)*n+1:j*n)];
end
afficher(T);
```

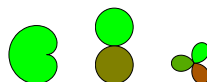
L'exécution de ce script produit :

Console Scilab

```
-->exec('/Users/raymondmoche/Magasin_scilab/304/304-Ranger-un-tableau.sce', -1)
T=[1,4,5;-7,2,-4;-1,3,-8];

    5.    4.    3.
    2.    1.   -1.
   -4.   -7.   -8.

-->
```



Énoncé 3 [*] Boucle « pour » dans une boucle « pour »

1 - Écrire la matrice (n,n) (tableau carré à n lignes et n colonnes) dont les éléments sont $1, 2, \dots, n^2$ écrits de la gauche vers la droite et de haut en bas.
2 - Cette matrice étant notée M , expliquer comment se font les calculs suivants :
 $N=M+M$, $Q=3*M$, $R=M.^2$.

Mots-clefs : commandes « input », boucles « pour » imbriquées.

Voici un script solution :

Listing 5 – Boucles « pour » imbriquées

```
// 2 boucles "pour" imbriquées
n=input('n='); // On pourra choisir n à l'exécution du script.
M=[]; // On va remplir M.
for i=1:n
    Mbis=[]; // On va remplir la ième ligne de M.
    for j=1:n
        Mbis=[Mbis,(i-1)*n+j]; // ième ligne de M.
    end
    M=[M;Mbis]; // On ajoute à M sa ième ligne.
end
afficher(M);
```

Commentaires : $Mbis = [Mbis, (i-1)*n+j]$ signifie que l'on adjoint à la matrice-ligne $Mbis$ une composante supplémentaire $((i-1)*n+j)$, ce qui est indiqué par « , » et que la nouvelle valeur de $Mbis$ est cette matrice-ligne allongée. Au contraire, dans $M = [M; Mbis]$, le « ; » indique que l'on ajoute une nouvelle ligne. Le script ci-dessus donne après exécution, pour $n = 7$:

```
Console Scilab
-->exec('/Users/raymondmoche/Desktop/303-Boucles-pour-imbriquees.sce', -1)
n=7

    1.    2.    3.    4.    5.    6.    7.
    8.    9.   10.   11.   12.   13.   14.
   15.   16.   17.   18.   19.   20.   21.
   22.   23.   24.   25.   26.   27.   28.
   29.   30.   31.   32.   33.   34.   35.
   36.   37.   38.   39.   40.   41.   42.
   43.   44.   45.   46.   47.   48.   49.

-->
```

Bien sûr, pour que l'affichage soit lisible, il vaut mieux choisir de petites valeurs de n . Pour se rendre compte de la puissance et de la rapidité de « scilab », le lecteur pourra tester de plus grandes valeurs, par exemple $n = 1000$. Dans ce cas, ce qui est long, c'est l'affichage, que l'on interrompra.

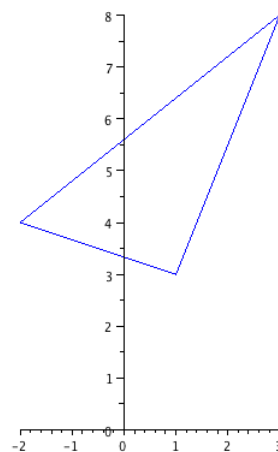
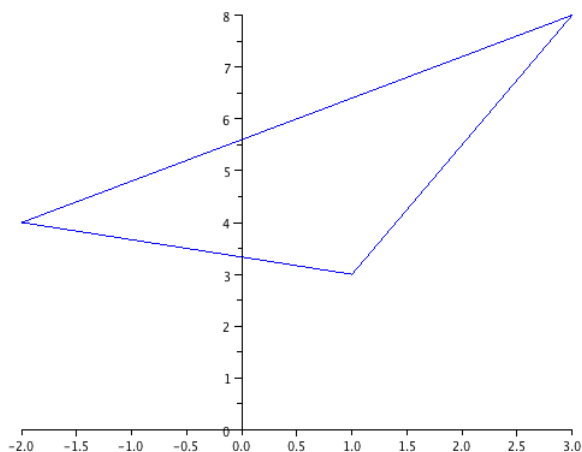
Les calculs de la deuxième question se font composante par composante.

Énoncé 4 [*] Tracé d'un triangle

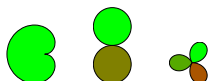
Tracer le triangle ABC , ces points ayant respectivement pour coordonnées $(1, 3)$, $(-2, 4)$ et $(3, 8)$.

Listing 6 – Triangle

```
// Tracer de triangle
clf
X=[1, -2, 3, 1];
Y=[3, 4, 8, 3];
plot(X,Y);
```



- ✓ On remarque que les triangles ont été fermés en répétant les coordonnées du premier sommet. Le repère librement choisi par « scilab » (à gauche) n'est pas orthonormé.
- ✓ On pourrait se demander si le triangle est isocèle ou rectangle. Pour cela, dans la figure de droite qui représente le même triangle, on a imposé au repère d'être orthonormé à l'aide de la commande « orthonorme ».
- ✓ La commande « clf » nettoie la fenêtre graphique. Elle est indispensable si on a déjà fait des tracés dans la même session. Sans elle, les tracés se superposeraient.



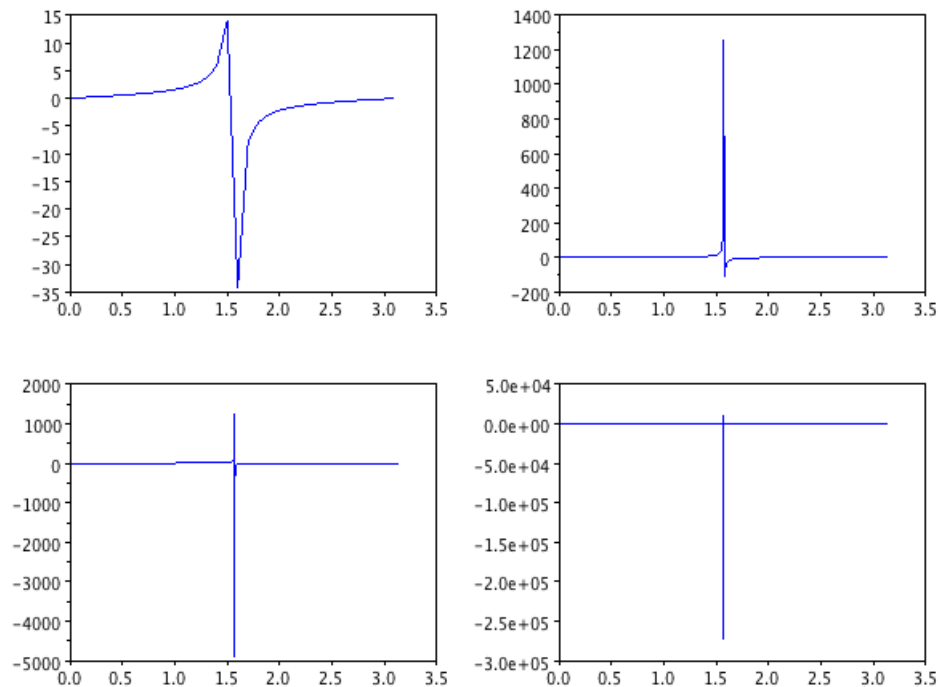
Énoncé 5 [*] Graphe de la fonction tangente entre 0 et π

Tracer le graphe de la fonction tangente entre 0 et π , la variable variant avec un pas de 0.1, puis 0.01, puis 0.001 et enfin 0.0001.

Mots-clefs : « subplot ». « subplot(2,2,i) » signifie que le tracé que l'on va faire est le $i^{\text{ème}}$ d'un tableau de 4 graphes répartis en 2 lignes et 2 colonnes, la numérotation se faisant de gauche à droite et de haut en bas.

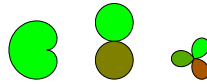
Listing 7 – Graphes de la fonction tangente

```
// Graphe de tangente
clf
A=[1,2,3,4];
for i=1:4
    X=0:10^(-i):%pi;
    Y=tan(X);
    subplot(2,2,i);
    plot(X,Y);
end
```



- ✓ La variable évitant la valeur $\frac{\pi}{2}$, on obtient des lignes polygonales continues : c'est le cas de tous les graphes « scilab ».
- ✓ Évidemment, on n'obtient pas l'asymptote verticale en $\frac{\pi}{2}$.

- ✓ Lorsque le pas vaut 0.001 ou 0.0001, on s'approche trop de $\frac{\pi}{2}$. « Tangente » prend des valeurs très grandes en valeur absolue. « scilab » adapte l'échelle sur l'axe des ordonnées et les variations de la fonction sont écrasées. C'est très net sur le dernier graphe.
- ✓ Cet exercice montre bien ce que l'on peut attendre de « scilab » et ce qui est hors de sa portée, en matière de tracés.
- ✓ Avec « scilab », on peut tracer de nombreuses courbes (cartésiennes, paramétriques, notamment polaires) et de nombreuses surfaces et acquérir ainsi une bonne connaissance pratique des fonctions.
- ✓ Mais on ne peut pas démontrer qu'une fonction est continue ou dérivable (par exemple).
- ✓ Entre deux points calculés consécutifs d'un graphe, on ne sait pas ce qui se passe.
- ✓ La théorie mathématique reste indispensable.



Énoncé 6 [*] Sauts de puce

Une puce fait des bonds successifs indépendants les uns des autres en ligne droite. La longueur de chaque bond est aléatoire. On considère que c'est un nombre choisi au hasard dans l'intervalle $[0, 5[$, l'unité de longueur étant le cm. On note la distance totale parcourue au bout de m bonds. On répète cette expérience n fois. Calculer la distance moyenne parcourue au cours de ces n expériences.

Mots-clefs : Commande « rand ».

Cet exercice ne pose pas de problème d'algorithmique. Dans les deux solutions proposées, l'algorithme demandera de saisir la valeur de n , puis de m . La commande « rand(1,N) » où N désigne un entier naturel quelconque, produit N nombres qui peuvent être considérés comme les valeurs prises par N variables aléatoires indépendantes prenant leurs valeurs dans l'intervalle $[0, 1[$ en suivant la loi uniforme.

Dans le premier algorithme, on stocke les distances parcourues au cours des n expériences dans S à l'aide d'une boucle « pour ».

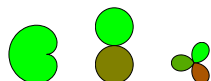
Listing 8 – Algorithme avec une boucle « pour »

```
n=input(" valeur de n : ");
m=input(" valeur de m : ");
S=[];
for j=1:n
    sauts=5*rand(1,m);
    S=[S,sum(sauts)];
end
moyenne=sum(S)/n;
```

Dans le second algorithme, on utilise une propriété de la moyenne : il suffit d'additionner les longueurs des $n \times m$ sauts et de diviser par n .

Listing 9 – Algorithme avec une boucle « pour »

```
n=input(" valeur de n : ")
m=input(" valeur de m : ")
moyenne=sum(5*rand(1,n*m))/n;
afficher('La distance moyenne parcourue est égale à '+string(moyenne));
```



Énoncé 7 [**] Résolution graphique d'une équation

- 1 - Calculer les valeurs de $f(x) = \frac{2x^2 + 1}{x^4 - \sqrt{3} \cdot x^2 + 127}$ lorsque la variable x prend successivement les valeurs $-1, -0.9, \dots, 0.9, 1$.
- 2 - Tracer le graphe de la fonction f quand x varie entre -1 et 1 en utilisant seulement les valeurs de $f(x)$ calculées précédemment.
- 3 - Apparemment, l'équation $f(x) = 0$ a une solution α entre 0 et 1 . En utilisant le zoom, proposer une valeur approchée de α .
- 4 - Tracer de nouveau le graphe de f entre -1 et 1 en faisant varier x avec un pas de 0.001 .
- 5 - Ce nouveau graphe amène-t-il à corriger la valeur de α proposée ?

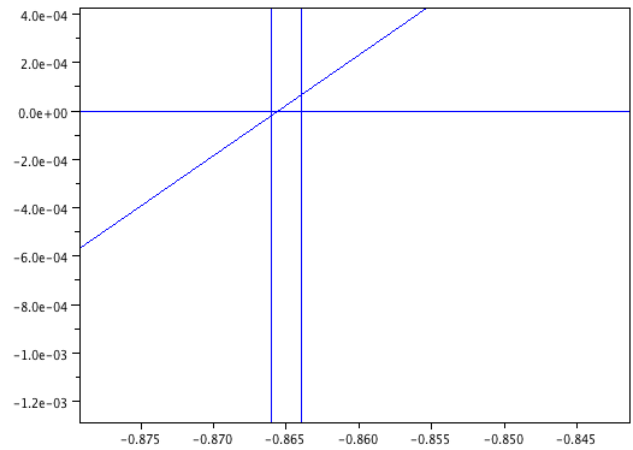
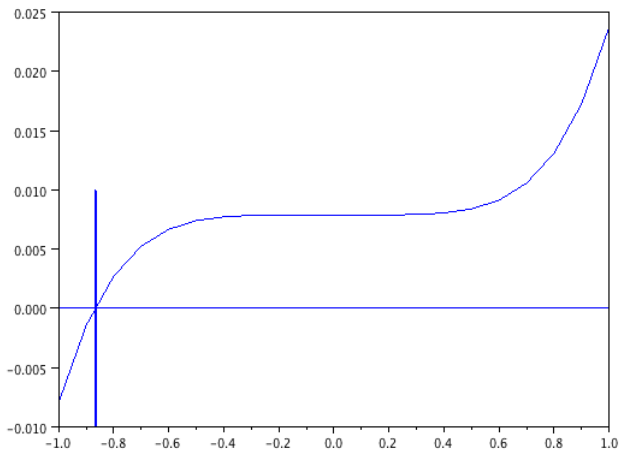
Mots-clefs : commandes « clf », « clear », « plot », résolution graphique d'une équation, zoom, annulation du zoom.

« clf » nettoie la fenêtre graphique, « clear » supprime toutes les variables non protégées, « clear a » supprime la variable « a » si elle n'est pas protégée.

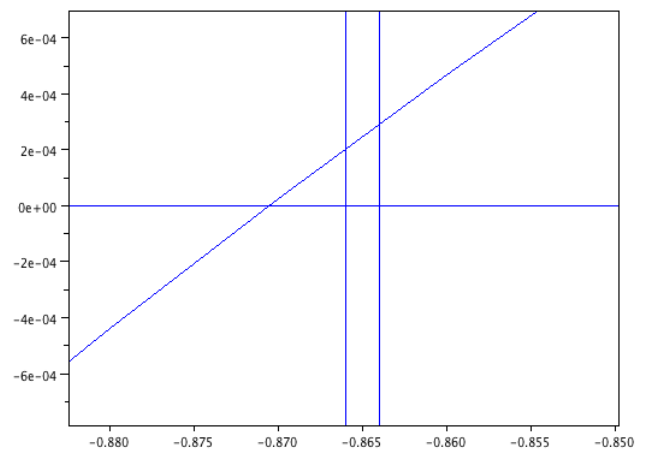
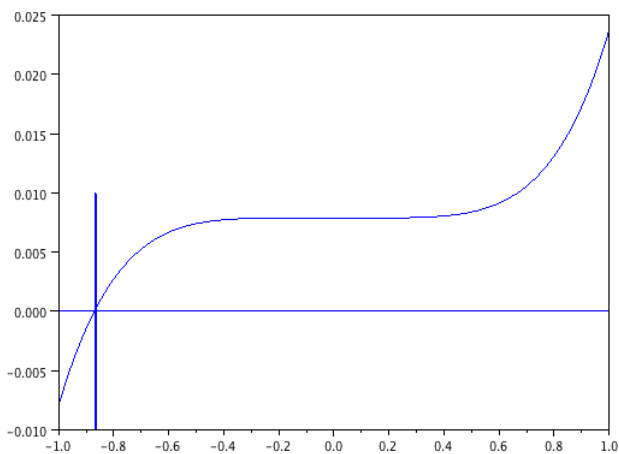
Listing 10 – Résolution graphique d'une équation

```
function y=effede(x);  
    y=(2*x^5+1)/(x^4-sqrt(3)*x^2+127);  
endfunction  
pas=input('pas=');  
X=-1:pas:1;  
Y=[];  
for i=1:size(X,'c')  
    Y=[Y,effede(X(1,i))];  
end;  
clf;  
plot(X,Y);  
plot([-1,1],[0,0]);
```

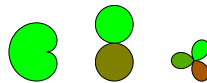
On appliquera deux fois le script précédent en donnant au pas la valeur 0.1 puis la valeur 0.001 .



Avec le premier pas, en zoomant deux fois autour du point $(\alpha, 0)$, on voit bien que le graphe est une ligne polygonale. On a tracé pour plus de commodité la droite $y = 0$ et des segments verticaux d'abscisse -0.864 et -0.866 . On peut proposer $\alpha \approx -0.8655$. Avec le pas 0.001 ,



on est amené à proposer $\alpha \approx -0.8708$. Le zoom et l'annulation du zoom se trouvent dans la barre d'outils de la fenêtre graphique.



Énoncé 8 [**] Matrice Zorro

Écrire la matrice Zorro, matrice carrée de taille n comprenant des 1 sur la première ligne, sur la deuxième diagonale et sur la dernière ligne et des 0 partout ailleurs.

Mots-clefs : « ones », « zeros ».

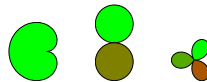
La solution ci-dessous comprend un maniement de matrices typique de « scilab ». Z est définie comme un assemblage de 3 matrices : la matrice « ones(1,n) », matrice à 1 ligne et n colonnes ne comportant que des 1, utilisée deux fois et la matrice « zeros(n-2,n) », matrice à $n - 2$ lignes et n colonnes ne comportant que des zéros. Les « ; » qui séparent ces matrices indiquent qu'elles sont disposées les unes en dessous des autres. Elles forment alors la matrice Z à n lignes et n colonnes. Si on remplaçait les « ; » par des « , », elles auraient été disposées en ligne, ce qui, dans le cas présent, aurait provoqué un message d'erreur à l'exécution du script car on ne peut aligner ainsi que des matrices ayant le même nombre de lignes (de même, on ne peut empiler que des matrices ayant le même nombre de colonnes).

Listing 11 – Matrice Zorro

```
// Matrice Zorro
n=input('n=');
Z=[ones(1,n);zeros(n-2,n);ones(1,n)];
for i=2:n-1
    Z(n-i+1,i)=1;
end
afficher(Z);
```

Voici ce que l'on obtient pour $n = 7$:

```
1.    1.    1.    1.    1.    1.    1.
0.    0.    0.    0.    0.    1.    0.
0.    0.    0.    0.    1.    0.    0.
0.    0.    0.    1.    0.    0.    0.
0.    0.    1.    0.    0.    0.    0.
0.    1.    0.    0.    0.    0.    0.
1.    1.    1.    1.    1.    1.    1.
```



Énoncé 9 [**] Aiguilles de l'horloge.

1 - Combien de fois l'aiguille des minutes tourne-t-elle plus vite que l'aiguille des heures ?

2 - Quand l'aiguille des heures a tourné de α degrés à partir de midi, l'extrémité de l'aiguille des minutes est repérée sur le cadran de l'horloge par l'angle

$$\beta = 12 \cdot \alpha - 360 \cdot \left[\frac{12 \cdot \alpha}{360} \right]$$

Tracer le graphe de β en fonction de α quand celui-ci varie de 0 à 360 degrés.

3 - Combien y a-t-il de superpositions des aiguilles entre midi et minuit ? Calculer les angles correspondants.

4 - Que se passe-t-il à partir de minuit ?

où $[]$ désigne la fonction partie entière.

Mots-clefs : boucle « pour », « plot ».

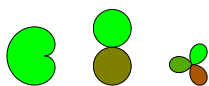
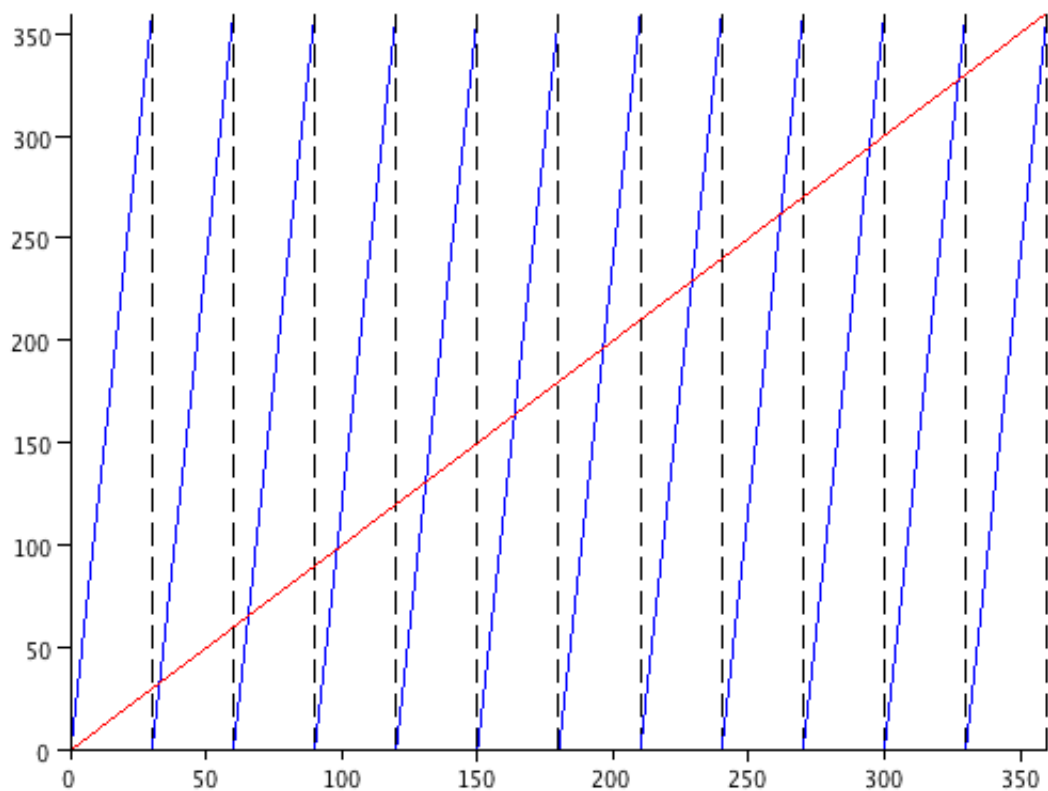
- ✓ **Première heure** : Quand l'aiguille des heures tourne de 30 degrés, l'aiguille des minutes fait un tour complet. Elle tourne 12 fois plus vite (si l'aiguille des heures tourne de α degrés, l'aiguille des minutes tourne de $\beta = 12 \cdot \alpha$). Il y a une superposition des aiguilles, évidemment à midi.
- ✓ **Deuxième heure** : Quand l'aiguille des heures a tourné de α degrés à partir de midi, l'extrémité de l'aiguille des minutes est repérée par l'angle $\beta = 12 \cdot \alpha - 360$. Elle refait un tour complet pendant cette heure. Il y a donc une nouvelle superposition, solution de l'équation $\alpha = 12 \cdot \alpha - 360$.
- ✓ **10 heures suivantes** : analogue, voir le script ci-dessous.

Listing 12 – Aiguilles de l'horloge

```
clf;
A=[];
for k=0:11
    X=[k*30,(k+1)*30];
    Y=[0,360];
    plot(X,Y,'b'); // graphe de beta quand alpha varie de 30k degrés
                    // à 30k+30 degrés, en bleu.
    plot([(k+1)*30,(k+1)*30],[0,360],'k-'); // pour améliorer
                    // la lisibilité du graphe, en pointillés noirs.
    A=[A,360*k/11]; // kième angle de superposition.
end;
plot([0,360],[0,360],'r'); // 'Première diagonale', en rouge
afficher('Les angles en degrés où les aiguilles se superposent sont');
afficher(A);
```

- ✓ Le graphe de β comme fonction de α se compose de 12 segments de droite, définis par leurs extrémités.
 - ✓ Cela entraîne que des positions de l'aiguille des heures sont prises en compte 2 fois. En fait, on aurait dû faire varier α sur les intervalles $[0, 30[, \dots, [330, 360[$. On aurait trouvé 11 superpositions entre midi (compris) et minuit (non compris).
-

- ✓ À minuit, nous sommes revenus dans la configuration de midi. Tout se répète.
- ✓ Les lignes pointillées ne font pas partie du graphe. La commande de tracé utilise l'option « 'k-' » (k avec 2 tirets), qui signifie que la ligne sera pointillée en noir (k).
- ✓ On pourrait faire une résolution graphique des intersections du graphe de β avec la ligne $y = \alpha$. C'est ici sans intérêt.



Énoncé 10 [******] Calculs d'effectifs et de fréquences

Une suite S de nombres entiers étant donnée, calculer les effectifs dans cette suite des entiers n , $\min(S) \leq n \leq \max(S)$ ainsi que leurs fréquences.

Mots-clefs : Commandes « input », « min » et « max », « taille », « tirage_entier », « frequence », matrice « zeros ».

Dans l'algorithme ci-dessous, on calcule les minimum et maximum de S notés a et b . Il y a donc au plus $b - a + 1$ entiers différents dans la suite S . On stockera leurs effectifs dans un vecteur C de longueur $b - a + 1$ que l'on initialise en le remplissant de zéros. Quand on s'occupe du $i^{\text{ème}}$ nombre de S , on regarde sa valeur $S(i)$ et on augmente de 1 la composante de rang $S(i)$ du vecteur C . À la fin de la boucle « pour », C est le vecteur des effectifs.

Listing 13 – Calcul d'effectifs

```
S=input('S='); // S est une suite de nombres entiers.
a=min(S);
b=max(S);
C=zeros(1,b-a+1);
for i=1:taille(S);
    C(S(i)-a+1)=C(S(i)-a+1)+1;
end;
afficher(C);
```

On exécute cet algorithme écrit dans « SciNotes » en le chargeant dans « scilab » (Exécuter ?> ... fichier sans écho). En réponse à la question du logiciel, nous choisissons pour S une suite de 2011 nombres entiers tirés au hasard entre 1 et 6 (ce qui simule le lancer d'un dé 2011 fois de suite). On obtient :

```
S=tirage_entier(2011,1,6);
    356.    325.    315.    334.    334.    347.
-->
```

On ajoute les commandes qui permettent de calculer les fréquences de 1, 2, ..., 6 dans S , ce qui donne finalement :

```
-->N=taille(S);F=C/N; afficher(F);

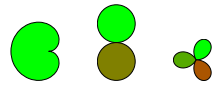
    column 1 to 3
0.1770263550472    0.1616111387369    0.1566384883143
    column 4 to 6
0.1660865241174    0.1660865241174    0.1725509696668
-->
```

On peut retrouver immédiatement ces fréquences à l'aide de la commande « frequence » de « scilab » (consulter l'aide). Pour 1, cela donne :

```
-->f1=frequence(1,S);afficher(f1);
```

```
0.1770263550472
```

```
-->
```



Énoncé 11 [******] Matrice-croix

Écrire la matrice carrée de taille $2 * n + 1$ comportant des 1 sur la $n + 1$ ^{ième} ligne et la $n + 1$ ^{ième} colonne et des 0 ailleurs.

Mots-clefs : Commandes « zeros(p,q) », « ones(p,q) ».

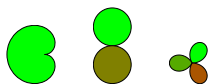
Le script qui suit est particulièrement compact (pour le plaisir). On peut faire moins bien !

Listing 14 – Matrice croix

```
// Matrice croix
input( 'n=' );
C=[zeros(n,n), ones(n,1), zeros(n,n)]; ones(1,2*n+1);
zeros(n,n), ones(n,1), zeros(n,n)];
afficher(C);
```

Voici ce que l'on obtient pour $n = 3$:

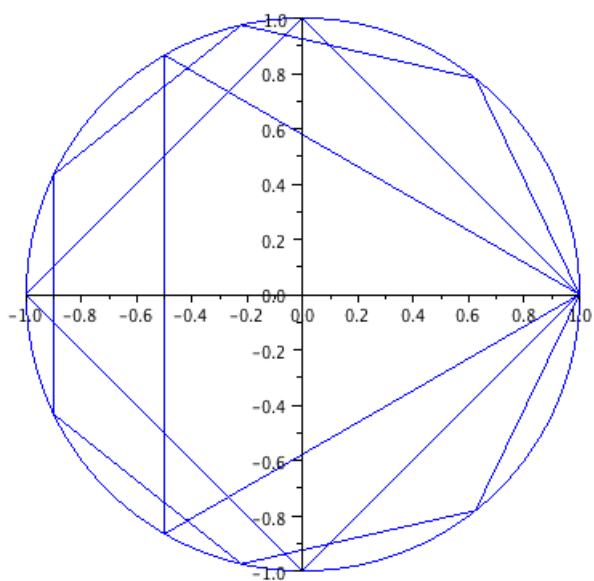
```
0.    0.    0.    1.    0.    0.    0.
0.    0.    0.    1.    0.    0.    0.
0.    0.    0.    1.    0.    0.    0.
1.    1.    1.    1.    1.    1.    1.
0.    0.    0.    1.    0.    0.    0.
0.    0.    0.    1.    0.    0.    0.
0.    0.    0.    1.    0.    0.    0.
```



Énoncé 12 [***] Tracés de polygones réguliers inscrits dans un cercle

Tracer dans le même repère orthonormé le cercle de centre O et de rayon 1 ainsi que le triangle équilatéral, le carré et l'heptagone régulier inscrits dans ce cercle et dont un sommet est le point $(1, 0)$.

Script téléchargeable (« Polygones.sce »). Bien sûr, ce script est assez compact. On peut détailler davantage. On obtient le tracé suivant :



Remarque : Pour faire de la géométrie, il vaut mieux utiliser un logiciel de géométrie !

