Ce stage permet de présenter les éléments d'algorithmiques présents dans les programmes de lycée :
- dans les constructions géométriques et les feuilles de calcul présentes dès la seconde
- dans les programmes de Terminale L pour des algorithmes qui ne sont pas nécessairement numériques.
- dans les programmes de Première S où les notions de boucles et de tests sont au programme.
- dans les premières STG (logique Algorithmique).
- dans les programmes des Terminales STI.

Ils peuvent prendre plusieurs formes :
- sur calculatrice pour les élèves et/ou sur calculatrice rétroprojectables ou simulateurs pour l'enseignant
- pour atténuer la différence entre calculatrice, il peut être fait le choix de travailler sur un logiciel permettant de travailler l'algorithmique. Le logiciel Execalgo (gratuit, source : http://ldif.education.gouv.fr/wws/d_read/eduscol.maths-l/Document%20accompagnement/) permet de le faire avec une syntaxe en français, un logiciel numérique comme Scilab (gratuit, en français : http://www.scilab.org/download/) permet de pratiquer l'algorithmique à égalité entre les élèves. Ce logiciel possède l'avantage d'être un traceur de courbes et d'avoir un langage semblable à celui employé sur TI et Casio.
- les constructions géométriques nécessitent parfois de pratiquer des notions d'algorithmique (répétition de construction) et ce, dès la seconde.
- les connaissances sur les tests se révèlent parfois nécessaires sur la constitution de feuilles calcul sur un tableur (simulation en seconde, feuilles de calcul en Première L, Epreuve pratique en TS) L'utilisation du copier/coller permet de pratiquer une répétition, les tableurs sont également capables d'itérations (boucles). Enfin, le tableur permet de visualiser ou non des résultats ou des constructions : ces affichages conditionnels sont pensés dans la construction des instructions ou bien dans le cadre de mises en forme conditionnelles.

Après une première partie de présentation des éléments d'algorithmique, nous pratiquerons cette notion sur des exemples simples qui résument les différents cas que l'on peut rencontrer. Les algorithmes sont essentiellement de deux formes, soit numériques soit permettant la répétition d'un mécanisme (recette de cuisine) ou d'une construction.
Nous reviendrons sur la programmation sur calculatrice même si ce n'est pas le but de ce stage. La traduction en tel ou tel langage n'est pas le but.
Nous verrons ensuite comment pratiquer la programmation sur un tableur, comme celui d'OpenOffice (nous pourrons visualiser également sur Excel la démarche).
Enfin, nous terminerons par la pratique de l'algorithmique sur une construction géométrique.

Ecriture des algorithmes

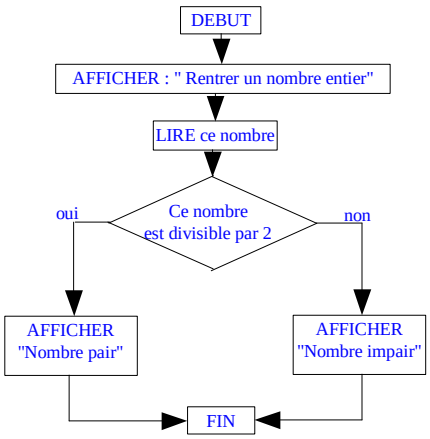
Un *algorithme* est une suite d'actions à effectuer pour obtenir, à partir de données initiales, la solution d'un problème. Comme il existe souvent plusieurs manières de résoudre un problème, on peut imaginer plusieurs algorithmes plus ou moins différents. Il doit pouvoir être effectué exactement, et dans un temps fini, par un homme utilisant des moyens manuels.

Dans l'expression d'un algorithme, il n'est pas nécessaire de faire appel à un langage de programmation, le langage courant suffit.
Ex : Détermination de la parité d'un nombre entier

- Demander à l'utilisateur de taper un nombre entier
- Lire ce nombre
- S'il est pair, afficher "Nombre Pair" sinon afficher "Nombre Impair".

Un organigramme constitue une expression graphique d'un algorithme. On y distingue trois types d'éléments :

- Les suites d'instructions, dites séquentielles, représentées par des rectangles.
- Les conditions ou test portant sur des expressions qui orientent la marche du programme, représentées par des losanges.
- L'ordre dans lequel le programme avance de l'un des éléments précédents à un autre est indiqué par une flèche.



Les variables

Ces données ainsi que les résultats des calculs intermédiaires ou finaux, sont rangés dans des "cases-mémoires" appelées *variables* que l'on repère par des *identificateurs* (que l'on choisira autant que possible significatifs).
Les contenus des variables sont de nature diverse, évoluent pendant l'exécution des algorithmes, mais une variable ne peut contenir au cours du traitement que des données de même nature :

Le *type* d'une variable est l'ensemble des valeurs possibles de son contenu. Le type définit la nature et le champ des valeurs successives de la variable, On précise ainsi l'intervalle ou l'ensemble de définition. On distingue :

Les types élémentaires :

- les types numériques : ENTIER et REEL. Distinction entre ces types car l'ordinateur ne réserve pas de la même place mémoire et vu la limitation de la mémoire, il ne les considère pas de la même manière : approximation pour les réels (0.99999999999 pour 1, même si à l'affichage, il indique 1) et valeurs exactes pour les entiers
- le type BOOLEEN (deux valeurs possibles : "vrai", "faux")
- le type CHAÎNE (ou chaîne de caractère)

Les types structurés :

- le type TABLEAU ou MATRICE (à une ou plusieurs dimensions)
- le type ENREGISTREMENT (ou type composé)

Dès le début du traitement, on indique (par exemple, dans un tableau), la liste des variables qui seront utilisées en précisant pour chacune d'elles *le nom*, *le type* ainsi que *le rôle* de cette variable dans l'algorithme.

Les instructions

Les instructions élémentaires

- La lecture au clavier du contenu d'une ou plusieurs variables :
  LIRE(variable) ; LIRE(A,B,C)

Remarques : la lecture au clavier s'achève dès que l'on presse la touche "entrée" (ou "retour chariot"). La donnée tapée doit être du même type que la variable qui la reçoit.

- L'affichage à l'écran (ou l'édition sur imprimante) d'un objet (nombre, chaîne, ...) du contenu d'une ou plusieurs variables, d'une expression, ...

ECRIRE('Prix de revient = ',P_Achat + Frais)

- L'affectation (donner une valeur au contenu d'une variable) :

Nom de Variable ← Expression (la flèche ← peut se lire reçoit)

ex : P_Vente ← P_Achat + Frais + Bénéfices

- L'appel d'une fonction (algorithme défini par ailleurs)

Les instructions composées

- Un bloc d'instructions est une suite d'instructions (élémentaires ou non) séparées par des points-virgules et encadrées des deux mots DEBUT et FIN. Dans la suite, "instruction" désignera soit une instruction élémentaire soit un bloc.

- Les instructions conditionnelles :

L'alternative : On effectue un test pour choisir entre deux instructions possibles :

```
SI <condition> ALORS instruction_1
                     SINON instruction_2;
```

La conditionnelle simple : même structure mais la deuxième instruction est vide :

```
SI <condition> ALORS instruction_1;
```

La conditionnelle multiple :

```
SELON NomVar
        Cas_1 : Instruction_1;
        Cas_2 : Instruction_2;
        ...
        Cas_n : Instruction_n;
FIN;
```

- Les instructions répétitives (ou boucles): une même séquence est répétée un certain nombre de fois.

La boucle POUR : on connaît exactement le nombre de répétitions à effectuer. On utilise un compteur de boucles :

```
POUR i VARIANT DE a A b EFFECTUER Instruction;
```

La boucle TANT_QUE : elle fonctionne comme la précédente, sauf que le test d'entrée dans la boucle est effectué avant l'exécution de l'instruction, on ne connaît pas toujours le nombre de répétition à effectuer :

```
TANT_QUE <condition> EFFECTUER instruction;
```

Remarque : Cette dernière boucle est la plus générale. Si la condition d'entrée est fausse dès le début, l'instruction n'est jamais effectuée, alors qu'elle est exécutée au moins une fois dans la boucle REPETER.

La boucle REPETER : elle est un peu moins générale que la précédente. Si la condition d'entrée est fausse dès le début, l'instruction est exécutée au moins une fois dans la boucle REPETER alors qu'elle n'est jamais effectuée, dans la boucle TANT QUE.

```
REPETER  instruction EFFECTUER <condition>;
```

Utilisations de la calculatrice

Exercice 1

1) Calculer, en utilisant la calculatrice (préciser le modèle de calculatrice utilisé)

$A = 123\,456^2 - 123\,455 \times 123\,457$

$B = 456\,789^2 - 456\,785 \times 456\,793$

$C = 123\,456\,789^2 - 123\,456\,787 \times 123\,456\,791$

2) Donner les résultats exacts. Pour cela, dans chaque expression :

- appeler x le premier nombre élevé au carré
- exprimer les deux autres en fonction de x
- écrire plus simplement l'expression en fonction de x, en développant

3) Calculer $D = 123\,456\,789\,010^2 - 123\,456\,789\,009 \times 123\,456\,789\,011$

Exercice 2

Calculer, en utilisant la calculatrice $a - b$ avec $a = 123456789123456789^2$
et $b = 12345678913245678 8 \times 123456789123456790$.

Ce résultat vous semble-t-il juste ?

Exercice 3

Les décimales cachées

1) Afficher le nombre π sur votre calculatrice.

Soustraire 3 et multiplier par 10.

Soustraire 1 et multiplier par 10.

Et ainsi de suite : soustraire la partie entière et multiplier par 10, plusieurs fois.

Finalement, combien votre calculatrice connaît-elle de décimales de π ?

Combien en affiche-t-elle ?

2) a) Faire afficher $\sqrt{2}$ à votre calculatrice.

Grâce à la méthode précédente, faire afficher les décimales cachées de votre calculatrice.

b) Faire afficher le quotient $\frac{941\,664}{665\,857}$ et ses décimales cachées.

A combien près cette fraction est-elle une valeur approchée de $\sqrt{2}$?

Comment peut-on être sûr de l'exactitude d'une certaine décimale ?

Exercices : Savoir exécuter un algorithme numérique élémentaire

I. Dans cet algorithme *a, b, c* désignent des variables numériques

```
début
    a←5;
    b←12;
    c←2*a-b;
    b←2*b-c*3;
    a←b-a*4+c*5;
    écrire('A=',a,' B=',b,' C=',c);
fin.
```

1) Exécuter cet algorithme
2) Le résultat constaté sur *a* est-il vrai quelles que soient les valeurs initiales des variables *a* et *b* ?

II. Quelle est l'action effectuée par l'algorithme suivant ?

```
début
    lire(a,b);
    a←a+b;
    b←a-b;
    a←a-b;
    écrire('A=',a,' B=',b);
fin.
```

Proposer une autre méthode permettant d'effectuer la même action.

III.1) Effectuer l'algorithme suivant pour les triplets (*a,b,c*) :
a) (2,-1,3) b) (-1,6,0) c) (7,4,3)

2) Que réalise cet algorithme ?

```
début
    lire(a,b,c); {a,b,c sont des entiers}
    si a>b
        alors si a>c
                alors si b>c
                        alors écrire (a,' ',b,' ',c)
                        sinon écrire (a,' ',c,' ',b)
                sinon écrire(c,' ',a,' ',b)
        sinon si a>c
                alors écrire(b,' ',a,' ',c)
                sinon si b>c
                        alors écrire (b,' ',c,' ',a)
                        sinon écrire (c,' ',b,' ',a);
fin.
```

IV. Exécuter le programme suivant pour *n*=5 puis 10. Que réalise-t-il ?

```
Début
    lire(n); {n,p,i sont des entiers}
    p←1;
    Pour i ← 1 à n faire p←p*i;
    écrire('P=',p);
fin.
```

V. Spécialité mathématiques, BAC L, session 2007
On considère l'algorithme suivant :

Entrée : *a* un entier naturel.

Initialisation :
L liste vide ;
Affecter la valeur a à *x*.

Traitement :
Tant que $x > 0$;
Effectuer la division euclidienne de *x* par 7 ;
Affecter son reste à *r* et son quotient à *q* ;
Mettre la valeur de *r* au début de la liste *L* ;
Affecter *q* à *x*.

Sortie : Afficher les éléments de la liste *L*.

Faire fonctionner cet algorithme pour $a = 486$. On reproduira sur la copie un tableau analogue à celui donné ci-dessous et on le complètera :

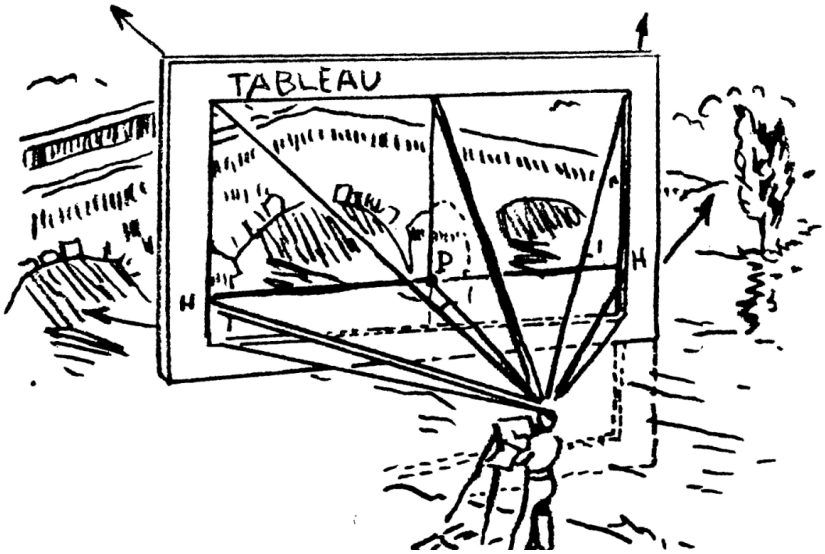
	r	*q*	*L*	*x*
Initialisation			vide	486
Fin étape 1				
Fin étape 2				
...				
...				
...				

VI. Extrait de la banque d'exercices pour la filière L

Compléter la figure ci-dessus, en respectant l'algorithme de construction suivant :

```
Tracer les droites (AC) et (BD)
Tant que la distance CD est supérieure à 1 cm,
    construire le milieu I du segment [CD]
    tracer le point E d'intersection des droites (AC) et (BI)
    tracer le point F d'intersection des droites (BD) et (AI)
    remplacer A par C, C par E, B par D et D par F.
```

Points de distance et champ visuel



(*HH'*) est la ligne d'horizon pour le spectateur sur le tableau. Dans le plan horizontal formé par l'œil du spectateur et cette ligne d'horizon, l'angle optique est d'environ 37°, c'est l'angle maximal qu'un individu peut isoler sans difficulté.

1) a) En considérant que le spectateur porte son regard perpendiculairement au tableau, au centre de celui-ci, déterminer à quelle distance il doit se placer vous visualiser l'ensemble du sujet décrit par le tableau (ou la vue derrière le cadre du tableau).

b) Sachant que dans le plan vertical, l'angle optique pour un être humain est d'environ 28°, établir, de même, la distance à laquelle le spectateur doit se placer devant le tableau.

2) On suppose dans la suite que l'on se place à environ 1,5l où l est la largeur du tableau.
On note P le point défini par l'observateur, S le point de fuite principal et D_1et D_2 les deux points de distance.
Montrer que si on respecte la règle des points de distance énoncée par Alberti, le triangle D_1PD_2 est rectangle en P et isocèle et les points D_1 et D_2 ne sont pas utilisable sur la tableau car en dehors de celui-ci.

3) Comment remédier à cette situation de façon à ce que le peintre d'un tableau puisse néanmoins utiliser des points de distance ?
Il s'agit de construire l'image perspective de la figure formée par une suite de rectangles dont le premier côté $[M_0N_0]$ prend la largeur du tableau et les rectangles sont placés les uns derrière les autres.

a) Construire la vue sur le tableau de 2 de ces rectangles $M_0N_0N_1M_1$ et $M_1N_1N_2M_2$ en utilisant les points de distance, sachant que l'utilisateur est placé de façon à ce que son angle optique sur le tableau soit de 37°.

b) On construit le point L_0 sur $[MN]$ tel que $L_0M_0 = \frac{1}{3}M_0N_0$.

Construire l'image perspective de $L_0N_0N_1L_1$, rectangle formé sur $[M_0N_0]$ et $[M_1N_1]$.

Tracer (L_0N_1). Cette droite coupe (D_1D_2). Que remarquez-vous ? Le démontrer.

En déduire comment le peintre peut construire l'image perspective de $M_0N_0N_1M_1$ sans utiliser les points de distance. Reprendre totalement la construction demandée dans le 3) a) sans utiliser les points de distance.

4) Donner l'algorithme de construction, sans utiliser les points de distance :

- d'un rectangle
- de n rectangles comme sur la figure décrite ci-dessus.

Simulation par la commande RANDOM de votre calculatrice

Toutes les calculatrices ont une commande RANDOM intégrée (*hasard*, en anglais).
Un tableur (Excel, OpenOffice) possède l'instruction ALEA().
Sur TI, il faut aller dans MATH puis PRB pour trouver l'instruction rand
Sur Casio, dans OPTN puis PROB pour trouver ran#.
Il suffit ensuite de taper plusieurs fois la touche ENTER ou EXE pour simuler ces nombres aléatoires entre 0 et 1 comportant 10 chiffres après la virgule.

1) Fréquence d'apparition d'un chiffre : le chiffre 7.

a) Créer cinq nombres à l'aide de la commande RANDOM de votre calculatrice (à défaut, prenez parmi les nombres ci-dessus) et noter le nombre de 7 qui apparaissent.
Recommencer avec cinq autres nombres.
Calculer la fréquence d'apparition du 7 sur ces dix nombres.
b) Comparer les fréquences obtenues avec les autres élèves de la classe.
Calculez la fréquence moyenne dans la classe.
c) On tire au sort un jeton parmi dix jetons de 0 à 9. *A priori*, quelles chances a-t-on de tirer un 7 ?

2) Tirage aléatoire d'un nombre entier

a) On utilise de nouveau la commande RANDOM :
sur TI : 10×rand et sur Casio : 10×ran#
Exécuter plusieurs fois cette commande. Quelle est la nature du nombre obtenu ?
b) On utilise la commande int(x) qui donne le plus grand entier inférieur au nombre x:
sur TI : int(10×rand) et sur Casio : int(10×ran#)
Exécuter plusieurs fois cette commande. Quelle est la nature du nombre obtenu ?

côté Face

3) Simulation du jeu de PILE ou FACE

a) En s'aidant du 2), créer, à l'aide de la calculatrice, une simulation d'une série de dix lancers de pièce possédant deux résultats possibles : PILE ou FACE.
b) Collecter les expériences faites dans la classe.
Donner le plus grand nombre de PILE obtenus pour dix lancers et le plus petit.
A priori, quelle est la fréquence théorique ?

côté Pile

4) Evolution des fréquences d'apparition du côté PILE

a) A l'aide de la calculatrice, lancer une pièce vingt fois de suite en complétant à chaque lancer le tableau suivant :

N° du lancer	1	2	3	4	… … … … … … … …	20
PILE (1) ou FACE (2)						
Nombre de PILE depuis le premier lancer						
Fréquence de PILE depuis le début (à 0,01 près)						

b) Dans un repère, placer les points dont l'abscisse est le numéro du lancer et l'ordonnée est la fréquence de PILE obtenus depuis le premier lancer.
Commenter, si besoin avec la simulation effectuée par le professeur au tableau.

5) Simulation sur calculatrice

Le programme ci-contre simule le lancer d'une pièce 300 fois de suite, calcule à chaque lancer la fréquence de PILE obtenus depuis le début, puis il place le point correspondant dans un

TI 80, 82(*), 83	Casio GRAPH	commentaires
:0➜Xmin : 300➜Xmax : 100➜Xscl :0.4➜Ymin : 0.6➜Ymax : 0.1➜Yscl :DrawF 0.5	ViewWindow 0, 300, 100, 0.4, 0.6, 0.1 ⌟ Graph Y=0.5 ⌟	Définition de la fenêtre d'affichage et des graduations sur les axes. Tracé de $y = 0{,}5$.
:0 ➜ P	0 ➜ P ⌟	0 dans le compteur de PILE .
:For (K , 1 , 300)	For 1 ➜ K To 300 ⌟	Début de boucle
:randInt(0 , 1) ➜ A	Int(2 * Ran#) ➜ A ⌟	Lancer de la pièce
:If A=1: P+1 ➜ P	if A=1 Then P+1 ➜ P ⌟	Si c'est PILE, ajouter 1 au compteur.
:P / K ➜ F	P / K ➜ ⌟	Calcul de la fréquence.
:Pt-On(K , F)	Plot K , F ⌟	Affichage du point (K ; F).
:End	Next	Fin de boucle.
:Disp F	F ◢	Sortie de la fréquence finale

repère, compare la fréquence à 0,5 et affiche la fréquence finale.

Exercices sur la récurrence

1) La suite (u_n) est définie par $u_0 = 0$ et $u_{n+1} = \frac{1}{2}u_n + 3$. Faire afficher les termes u_1 à u_n, n étant saisi au clavier.

2) Conjecture Syracuse (ou de Kollek, ou de Collatz) :
La suite (u_n) est définie par l'entier naturel u_0 et par la relation suivante :

si u_n est impair $u_{n+1} = 3u_n + 1$ et si u_n est pair $u_{n+1} = \frac{u_n}{2}$, u_0 étant saisi au clavier.

Faire afficher tous les termes de la suite jusqu'à ce que l'un d'eux soit égal à 1.

3) Algorithme Babylonien ou méthode de Newton pour calculer $\sqrt{a}$

En partant de a et u_0 saisis au clavier, avec la formule de récurrence $u_{n+1} = \frac{1}{2}(u_n + \frac{a}{u_n})$, donner le premier terme de la suite vérifiant : $\left|\frac{u_{n+1} - u_n}{u_n}\right| < 10^{-5}$

4) La suite de Fibonacci est définie par $u_0 = u_1 = 1$ et par la relation : $u_{n+2} = u_{n+1} + u_n$.
Faire afficher les termes u_2 à u_n, n étant saisi au clavier.

Exercices sur les sommes ou produits

5) Faire afficher la somme $S_1(n) = 1 + 2 + ... + n$, n étant saisi au clavier.
Reprendre l'exercice avec la somme $S_2(n)$ des carrés puis la somme $S_3(n)$ des cubes des entiers de 1 à n.

6) On démontre que la limite de la somme $S(n)$ suivante, lorsque n tend vers $+\infty$, est égale à $\frac{\pi}{4}$:

$$S(n) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ... + \frac{(-1)^n}{2n+1}$$

Faire afficher une valeur approchée de π en calculant $S(n)$, n étant saisi au clavier.

7) Un nombre est parfait s'il est égal à la somme de ses autres diviseurs. C'est le cas, par exemple, de 6 = 1 + 2 + 3 ou de 28 = 1 + 2 + 4 + 7 + 14.
Ecrire un programme testant si un nombre est parfait.
Modifier ce programme pour faire afficher la liste des nombres parfaits compris entre deux entiers saisis au clavier.

8) Un nombre est premier s'il admet exactement deux diviseurs : lui-même et l'unité.
Ecrire un programme testant si un nombre est premier.
Modifier le programme précédent pour faire afficher la liste des nombres premiers compris entre deux entiers saisis au clavier.

Chaînes de caractères

9) Sur les enveloppes utilisées par la Poste, les codes postaux écrits en chiffres par l'envoyeur sont codés par des bandes colorées imprimées dans la partie inférieure.
Les codages des cinq chiffres du code postal sont écrits de droite à gauche, jointivement. La table de conversion est la suivante :

0:	..||||	1:	.|.|||	2:	.||.||	3:	.|||.|	4:	|..|||
5:	|.|.||	6:	|.||.|	7:	||..||	8:	||.|.|	9:	|||..|

où le . représente un espace de code ASCII 32
et | est le caractère de code ASCII 124

On suppose que ce codage est contenu dans le tableau Tableau de taille 10 contenant les chaînes de la table de conversion.
Ainsi Tableau(4) est la chaîne '.|||.|' (en considérant un décalage de 1 entre la valeur et son repérage)
Ecrire la fonction permettant d'écrire le codage c correspondant au code postal donné par *n*.
En vue d'un décodage, on doit s'assurer tout d'abord de la validité de la lecture du code (par un scanner ou lecture optique). Ecrire la fonction retournant la valeur vraie si le code est possible et la valeur faux sinon sachant que le codage de chaque chiffre doit comprendre 4 barres et nécessairement une barre en première position (dans son écriture qui est de droite à gauche ! *cf* table de conversion).

Ecrire la fonction permettant, après s'être assuré de la fiabilité du codage c, de déterminer le code postal en l'entier n.

Tableau

10) a) Ecrire une fonction permettant de déterminer si l'entier *n* donné en paramètre est un nombre de Janus. Un nombre ou un mot de Janus peut se lire indifféremment dans les 2 sens, comme 13431 ou ressasser.
b) Ecrire un programme qui conserve dans un tableau tous les nombres de 1 à *N* , entier donné au clavier, dont le carré est un nombre de Janus.
Vous afficherez le tableau obtenu.

Le roi Janus
l'homme aux deux visages

Caricature représentant Louis XVI sous la forme de Janus aux deux visages. Qui d'un coté prête serment aux représentants de la Nation

"Je soutiendrais la Constitution"

et de l'autre affirme aux représentants de l'église

"Je détruirai la Constitution ".

Quelques palindromes sont bien connus :
- Esope reste ici et se repose.
- Elu par cette crapule.
- A man, a plan, a canal : Panama.
- Georges Pérec (1936 - 1982) est l'auteur d'un palindrome de 1 247 mots et plus de 76 000 caractères, qui débute ainsi :

Trace l'inégal palindrome. Neige. Bagatelle, dira Hercule. Le brut repentir, cet écrit né Pérec …
et se termine par :

… S'il porte, sépulcral, ce repentir, cet écrit ne perturbe le lucre : Haridelle, ta gabegie ne mord ni la plage, ni l'écart.

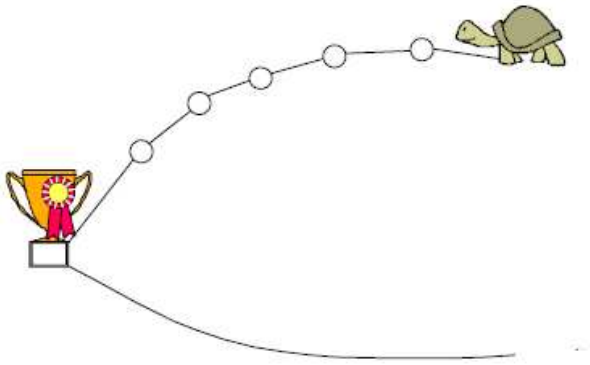
Divers

9) Le jeu se joue à deux : toi (l'élève) contre l'ordinateur (le prof). C'est chacun à son tour de jouer.
Le tapis comporte une seule rangée de 21 allumettes (on prendra la variable *n* pour le nombre d'allumettes).
Lorsque c'est à ton tour de jouer, tu dois enlever une, deux ou trois allumettes.
Celui qui retire la dernière allumette perd la partie.
Il s'agit de te laisser jouer le premier et de te faire perdre à tous les coups en construisant le programme dont l'algorithme est le suivant :
tant que la partie n'est pas finie :

- afficher le message "Il reste xxxx allumettes, vous devez en retirer entre 1 et 3" où xxxx représente le nombre d'allumettes restantes.
- demander à l'utilisateur un nombre entre 1 et 3. Redemander ce nombre s'il ne se trouve pas entre 1 et 3, et mettre à jour le nombre d'allumettes disponibles.
- l'ordinateur doit enlever à son tour des allumettes.
  La stratégie qu'il doit adopter est d'enlever le complément à 4 d'allumettes (C'est à dire : si vous enlevez 1 allumette, l'ordinateur en enlève 3, si vous en enlevez 2 l'ordinateur 2 et si vous en enlevez 3 l'ordinateur en enlève 1).
- S'il ne reste plus d'allumette après votre tour de jeu, afficher : « L'ordinateur a gagné ».

Si c'est après le tour de l'ordinateur (???), afficher : « L'ordinateur a perdu ».

10) On lance un dé.
Si le 6 sort, le lièvre gagne.
Sinon, la tortue avance d'une case.
On continue jusqu'à ce qu'il y ait un gagnant.
Quelle est la situation la plus enviable, celle du lièvre ou celle de la tortue ?



a) Expliquer pourquoi la formule tapée (ici sous Scilab) : *int*(*n*×rand()) + 1 permet d'obtenir un entier quelconque (et équiprobable) entre 1 et *n*.
b) Il s'agit dans cette question de simuler une seule partie en construisant une somme contenant les déplacements de la tortue : tant que le résultat n'est pas 6 ou que la tortue n'est pas arrivée, retirer un nombre quelconque entre 1 et 6.
Afficher alors le gagnant de la partie.
c) Modifier le programme précédent de manière à réaliser 100 simulations de parties.
Quelle est la situation la plus avantageuse, celle de la tortue ou celle du lièvre ?
Vous proposerez un affichage pour répondre à cette question.

11) **Conjecture d'Erdös**
Pour tout entier $n > 1$, il existe trois entiers x, y et z tels que :

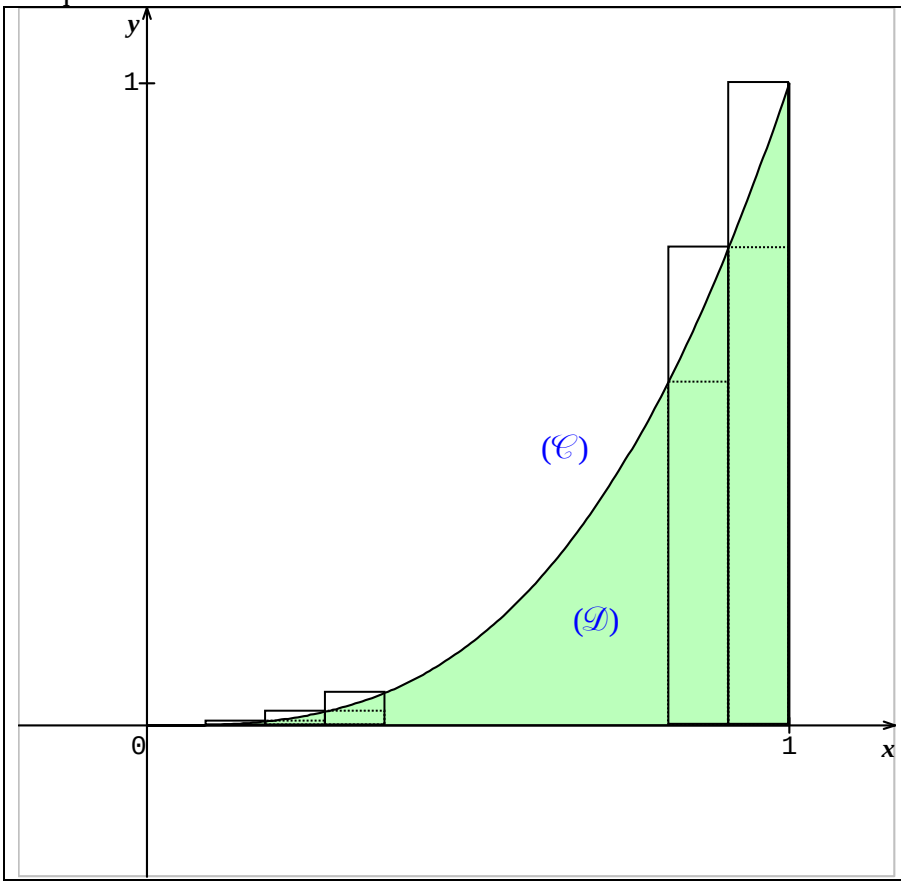
$$\frac{4}{n} = \frac{1}{x} + \frac{1}{y} + \frac{1}{z}$$

Construire un programme permettant de vérifier cette conjecture pour $n \in \{1; 2; \ldots; 100\}$, dans ce cas, vous supposerez que les entiers x, y et z sont à rechercher parmi les entiers entre 1 et n^2.
S'il existe un entier pour lequel l'égalité n'est pas vérifiée, vous ferez afficher "La conjecture est fausse" sinon vous ferez afficher "La conjecture semble vraie".

Remarque : *C'est un cas particulier de fractions égyptiennes où l'on cherche à écrire un rationnel comme somme d'un nombre donné d'inverses d'entiers, conjecture vérifiée pour* $n \leqslant 10^8$.
De même existe la conjecture de Sierpinski qui suppose que pour $n > 1$, *il existe trois entiers naturels a, b et c tels que* $\frac{5}{n} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c}$.

Calculs approchés d'intégrales par la méthode des rectangles

($\mathscr{C}$) est la courbe représentant la fonction f définie sur [0;1] par $f(x) = x^3$ dans un repère orthonormal. $\mathscr{S}$ est l'aire, en unités d'aire, du domaine ($\mathscr{D}$) délimité par la courbe ($\mathscr{C}$), l'axe des abscisses et la droite d'équation $x = 1$.



On subdivise l'intervalle [0;1] à l'aide des nombres $a_i = \frac{i}{n}$ avec $0 \leqslant i \leqslant n$.

Sur $[a_i;a_{i+1}]$, avec $0 \leqslant i \leqslant n-1$, on construit le rectangle de hauteur $f(a_i)$ et le rectangle de hauteur $f(a_{i+1})$.

On note $\mathscr{A}_n$ la somme des aires des rectangles contenus dans ($\mathscr{D}$) et $\mathscr{B}_n$ la somme des aires des rectangles qui contiennent ($\mathscr{D}$).

Vérifier que pour tout entier n tel que $n \geqslant 1$, $\mathscr{A}_n \leqslant \mathscr{S} \leqslant \mathscr{B}_n$, avec $\mathscr{A}_n = \frac{1}{n^4}\left(1^3 + 2^3 + \ldots + (n-1)^3\right)$ et $\mathscr{B}_n = \frac{1}{n^4}\left(1^3 + 2^3 + \ldots + n^3\right)$.

$\mathscr{A}_n$ et $\mathscr{B}_n$ sont appelées les *sommes de Riemann* de la fonction f sur [0;1] et sont respectivement des valeurs approchées par défaut et par excès de $\mathscr{S}$.

Partie A

Algorithme pour déterminer une valeur approchée de $\mathscr{S}$ par $\mathscr{A}_n$:

Entrer l'ordre de la subdivision : n
Abscisse du premier point : $x \leftarrow 0$
Initialisation de la somme : $S \leftarrow 0$
Boucle de calcul : Pour $k = 0$ à $n - 1$ faire
début
$S \leftarrow S + x^3 \times \frac{1}{n}$
$x \leftarrow x + \frac{1}{n}$
fin
Afficher : S

1) Traduire cet algorithme dans le langage employé par votre calculatrice. Créer également un programme permettant de calculer $\mathscr{B}_n$.

Utiliser ces programmes pour déterminer une valeur approchée de $\mathscr{S}$ à 10^{-3} près.

2) Reprendre l'algorithme proposé pour construire le programme permettant :
- de rechercher la première valeur de n telle que la différence relative entre les deux sommes $\mathscr{A}_n$ et $\mathscr{A}_{n+}1$ soit inférieure à 10^{-3}, c'est-à-dire telle que $\left|\frac{\mathscr{A}_{n+1} - \mathscr{A}_n}{\mathscr{A}_n}\right| \leqslant 10^{-3}$.
- d'afficher la valeur de $\mathscr{S}$ obtenue dans ce cas.

Partie B

Détermination de $\mathscr{S}$.

1) Démontrer que pour tout $n \in \mathbb{N}$:

$$1^3 + 2^3 + \ldots + n^3 = \left[\frac{n(n+1)}{2}\right]^2.$$

En déduire les expressions de $\mathscr{A}_n$ et $\mathscr{B}_n$ en fonction de n.

2) Démontrer que les suites $(\mathscr{A}_n)_{n \geqslant 1}$ et $(\mathscr{B}_n)_{n \geqslant 1}$ sont adjacentes.

3) Déterminer la valeur de $\mathscr{S}$.

La suite de Fibonacci

Partie A

1) On considère la suite de Fibonacci définie par $F_0 = 0$, $F_1 = 1$
et pour tout $n \geqslant 2$, $F_n = F_{n-1} + F_{n-2}$
Ecrire en une fonction ou un programme qui, pour un entier n donné, calcule la valeur du terme F_n de la suite de Fibonacci.

Partie B

On désire pouvoir calculer exactement, pour $2 \leqslant n \leqslant 100$, la valeur d'un terme F_n de la suite de Fibonacci.
La fonction précédente renvoie un résultat erroné à partir de $n = 79$.
Afin de calculer F_n, pour $79 \leqslant n \leqslant 100$, sans erreur de troncature ou d'arrondi, on définit l'algorithme suivant :
Cet entier est représenté par un tableau de taille 25 à raison d'un chiffre par élément. Si on note t une variable de type entier, alors $t(25)$ est le chiffre des unités de cet entier, $t(24)$ celui des dizaines, $t(23)$ celui des centaines, etc … Au delà du dernier chiffre de l'entier, les éléments du tableau sont nuls.
Ainsi $F_{47} = 2971215073$ est représenté par le tableau

0	0	0	0	…	0	2	9	7	1	2	1	5	0	7	3
1	2	3	4	…	15	16	17	18	19	20	21	22	23	24	25

Ce type permet donc de représenter tout entier naturel de l'intervalle $[0…(10^{26} - 1)]$.

2) Ecrire une fonction pour calculer la somme de deux nombres de type entier où la somme des entiers représenté par f1 et f2 est donné par la variable f3.
Exemple :

f1	0	0	0	0	0	…	0	8	1	7
+										
f2	0	0	0	0	0	…	1	4	6	4
=										
f	0	0	0	0	0	…	2	2	8	1

3) Ecrire une fonction qui construit le tableau t représentant le nombre de Fibonacci F_n.
4) Ecrire une fonction pour afficher à l'écran l'entier naturel représenté par un tableau t de type entier.
Par exemple, pour le tableau f de la question 2), cette procédure devra afficher 2281.

Partie C

5) Ecrire un programme permettant de saisir au clavier la valeur d'un entier n, si cet entier est inférieur à 79, d'utiliser la fonction Fibonacci de la partie A, si l'entier est entre 79 et 100, d'utiliser la fonction Fibonacci2 de la partie B et si l'entier est supérieur à 100, demander un autre entier.
Il suffira d'afficher la valeur du terme F_n ainsi obtenu.

La suite de Fibonacci : Correction

```
function [f]=fibonacci(n)
u=0;
v=1;
for i=2:n do
f=u+v;
u=v;
v=f;
end

function [c]=somme(a,b)
c=zeros(1,25);
for i=25:-1:1 do
c(i)=a(i)+b(i)+c(i);
if c(i)>9 then
c(i)=c(i)-10;
c(i-1)=1;
end
end

function afficher(f)
i=1;
while (f(i)==0)&(i<25) do i=i+1; end
c='';
for j=i:25 do c=c+string(f(j)); end
disp(c)


function [f]=fibonacci2(n)
u=zeros(1,25);
v=zeros(1,25);v(25)=1;
for i=2:n do
f=somme(u,v);
u=v;
v=f;
end
afficher(f);

function programme;
n=input('entrez un entier')
while n>100 do n=input('entrez un entier'); end
if n<79 then disp(fibonacci(n)); else fibonacci2(n); end
```

Cryptographie affine

A) Le codage affine

Pour transmettre un message secret, on peut utiliser la procédure :

- A toute lettre de l'alphabet, on associe le nombre lu dans le tableau ci-dessous :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	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

- Soit x le nombre obtenu à partir de la lettre de départ. On calcule $y = ax + b$.
  Le couple d'entiers $(a;b)$ où $a \neq 0$, s'appelle la *clé* du codage.
- On calcule ensuite $c(x)$, reste de la division de y par 26.
- Enfin, on associe à $c(x)$ la lettre correspondante par lecture inverse du tableau.

1) Cas $a = 1$

Dans ce cas, le codage se résume à un décalage. Pendant la guerre des Gaules, Jules César utilisait un tel procédé ($y = x + 3$) pour envoyer des messages chiffrés à Cicéron qui était resté en poste au Sénat à Rome.

Quel mathématicien se cache derrière **YXKETM** avec la clé (1,19) ?

2) Quelques exemples de clé

a) Coder votre prénom avec la clé (7,17).

b) Vérifier que si $a \equiv a'$ [26] et $b \equiv b'$ [26], les codes obtenus avec les clés (a,b) et (a',b') sont identiques.

c) De combien de clés dispose-t-on en prenant $1 \leqslant a \leqslant b$ et $0 \leqslant b \leqslant 25$?

d) On prend pour clé (2,13). Recopier et compléter le tableau suivant :

Mot Initial	**E**	**N**	**T**	**I**	**E**	**R**
Code x	4					
$2x + 13$	21					
$c(x)$	21					
Mot codé	*v*					

Le petit nombre de clés explique que cette méthode ne soit plus employée depuis longtemps.

Quel problème apparaît dans le codage ci-dessus ?

3) Ecrire un programme permettant de :

- Demander à l'utilisateur un texte constitué de lettres en majuscules.
- Parcourir chaque caractère du texte précédent, si c'est une lettre parmi les 26 lettres de l'alphabet, pratiquer le chiffrement affine sinon, laisser le caractère inchangé.
- Afficher le texte codé.

Sur TI, vous pourrez utiliser les instructions suivantes, présentes dans le menu Math puis String.
length(x) donnant la longueur de la chaîne c, mid(x,k,l) donnant la sous-chaîne contenue dans x définie entre les caractères de position k et l dans x, char(65) donne "A" et char(66) donne "B" sont les instructions permettant d'obtenir les codages mémoires des deux premières majuscules, les suivants étant consécutives, ord("A") donne 65 et ord("B") donne 66 : de même aux codes consécutifs partants de 65, on peut retrouver les lettres majuscules consécutives, "A"&"B" donne "AB" ; c'est la concaténation des chaînes de caractères.

B) Décodage

1) Dans le cas d'un codage affine de clé (7,17), chercher une lettre dont le codage final soit **B**.

a) A l'aide de l'algorithme d'Euclide, trouver deux entiers u et v tels que $7u - 26v = 1$. Justifier $7u \equiv 1$ [26].

b) Soit x le code initial de la lettre cherchée.
Démontrer que x vérifie (E) : $7x \equiv -16$ [26].
En déduire que $x \equiv -16u$ [26].
c) En déduire l'entier x compris entre 0 et 25 solution de (E) puis la lettre cherchée.

2) Expliquer pourquoi la méthode ci-dessus assure le décodage de n'importe quelle lettre dès qu'on choisit une clé (a,b) telle que a soit premier avec 26.

3) Avec la clé (5,13), quel mot se cache derrière **SUNOF** ?

4) Programme de calcul des coefficients dans la relation de Bezout.
a et b sont des entiers naturels, $a > b > 0$ et g est leur PGCD. Alors il existe u et v entiers relatifs tels que $au + bv = g$.
L'exemple ci-après et le programme permettent de trouver u et v.

On met en œuvre sur l'exemple ci-après l'algorithme d'Euclide et on calcule les restes successifs en fonction de a et b. On sait que le dernier reste non nul est le PGCD. On développe alors les calculs de façon à faire apparaître à chaque étape une écriture du reste de la forme $au + bv$.

Prenons $a = 47$ et $b = 35$.

$47 = 35\times1 + 12$ soit	$a = b + 12$	donc $12 = a - b$
$35 = 12\times2 + 11$	$b = (a - b)\times2 + 11$	$11 = -2a + 3b$
$12 = 11\times1 + 1$	$a - b = (-2a + 3b)\times1 + 1$	$1 = 3a - 4b$
$11 = 11\times1 + 0$		

Nous avons bien obtenu le PGCD comme combinaison linéaire de a et b.

Les programmes suivants, pour Casio et TI, permettent d'afficher les valeurs de U et V après avoir demandé à l'utilisateur les valeurs de A et B.

Famille Casio	**Famille TI**
"A=" : ? → R "B=" : ? → Y I → U : 0 → W : 0 → V : I → X While Y ≠ 0 Int(R÷Y) → Q U → Z : W → U : Z-Q*W → W V → Z : X → V : Z-Q*X → X R → Z : Y → R : Z-Q*Y → Y WhileEnd "U=" : U◢ : "V=" : V◢ "PGCD=" : R◢	Input "A=", R Input "B=", Y I→ U : 0 → W : 0 → V : I → X While Y ≠ 0 Int(R÷Y) → Q U → Z : W → U : Z-Q*W → W V → Z : X → V : Z-Q*X → X R → Z : Y → R : Z-Q*Y → Y End Disp "U=", U, "V=3, V Disp "PGCD=", R

Reprendre le programme précédent afin de :
- Demander à l'utilisateur un texte constitué de lettres en majuscules et la clé de codage.
- Déterminer la clé de décodage associée.
- Parcourir chaque caractère du texte précédent, si c'est une lettre parmi les 26 lettres de l'alphabet, pratiquer le déchiffrement affine sinon, laisser le caractère inchangé.
- Afficher le texte décodé.

Méthode de Monte Carlo

On appelle méthode de Monte-Carlo toute méthode visant à calculer une valeur numérique, et utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. Le nom de ces méthodes fait allusion aux jeux de hasard pratiqués à Monte-Carlo.
Les méthodes de Monte-Carlo sont particulièrement utilisées pour calculer des intégrales en dimensions plus grandes que 1 (en particulier, pour calculer des surfaces, des volumes, etc.)

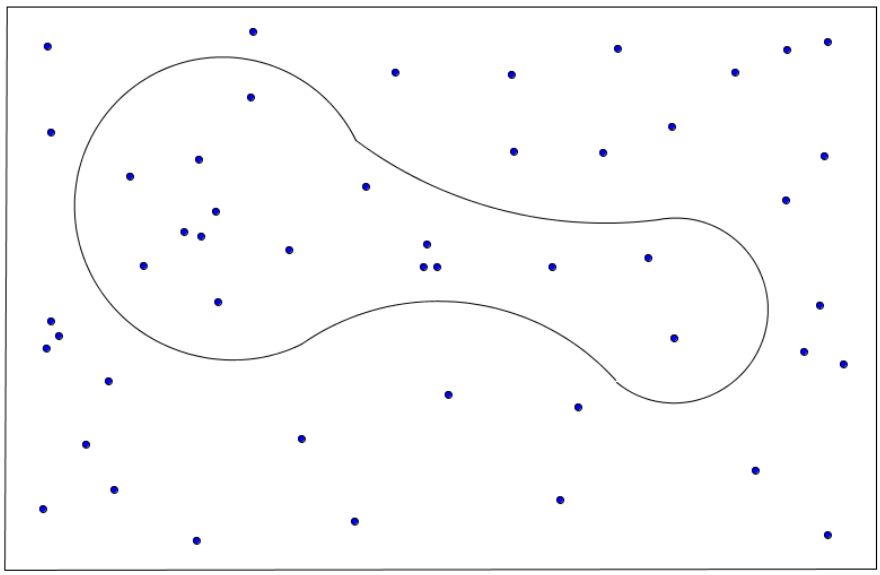
Soit une surface incluse dans un rectangle comme illustré sur la figure ci-dessous. Le problème ici consiste à évaluer l'aire de la surface connaissant l'aire du rectangle. Pour cela, on fait appel à la loi empirique des grands nombres. On suppose que nous soyons capables d'inscrire les points dans le rectangle de manière aléatoire (uniformément répartis). Par la loi empirique des grands nombres, nous avons :

$$\lim_{n \to \infty} \frac{n_S}{n} = \frac{\text{aire de la surface}}{\text{aire du rectangle}}$$

où n est le nombre de points total dans le rectangle et n_S le nombre de points sur la surface.

Ainsi, nous pouvons donner une bonne approximation de l'aire de la surface dès que le nombre de points n devient important, par la formule suivante :

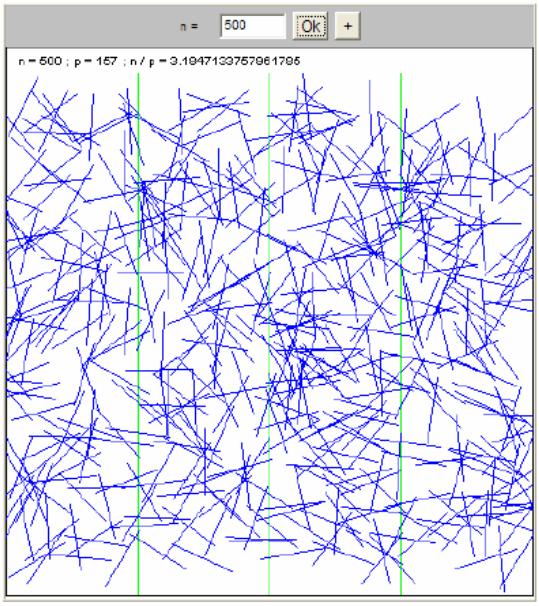
$$\text{aire de la surface} \approx \frac{n_S}{n} \times \text{aire du rectangle}$$



Cette méthode possède l'inconvénient de nécessiter un très grand nombre de points pour que la précision soit acceptable.

Remarque : ce problème est très proche de celui de Buffon (G. Leclerc) qui a écrit l'un des premiers traité de calcul des probabilités en liaison avec le calcul différentiel et intégral en 1733. Le plan est "pavé" par des droites distantes d'une distance égale à d, on lance au hasard un bâton de longueur $L < d$. LA probabilité que le bâton touche un trait est :

$$P = \frac{12L}{\pi\, d}$$



http://perso.wanadoo.fr/jpq/proba/montecarlo/buffon.htm

Cette méthode permet également de déterminer des valeurs approchées d'aires sous des courbes de fonctions dont on ne saurait pas déterminer des primitives.

Sur le principe de la méthode de Monte Carlo, déterminer la loi de probabilité de la variable aléatoire *X* simulée par les algorithmes suivants :

1) On note Int(*x*) la partie entière du nombre réel *x* et Random un nombre aléatoire entre 0 et 1.
N ← Int(Random × 5)
X ← Int(Random × N)

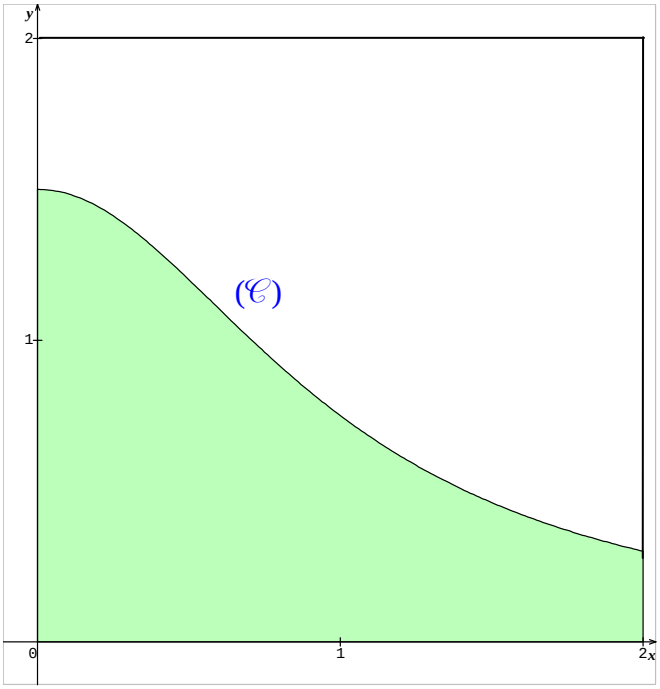
2) X ← 0 ; Y ← 1
Tantque Random < Y faire
    X ←X + 1 ; Y ← Y/2
fin

3) N ← 0
Répéter N fois
    Si Random < p1 faire n ← N + 1
    Fin
Fin
X ← 0
Répéter N fois
    Si Random < p2 faire x ← X + 1
    Fin
Fin

4) P ← p ; F ← P ; X ← 1
Tantque Random > F faire
    P ← P × (1 – p) ; F ←F + P ; X ← X + 1
Fin

Hasard et calcul d'aire

Une cible de fléchettes de base carrée est découpée en deux zones par la courbe ($\mathscr{C}$).
Dans le repère ayant pour origine le coint O du carré et pour unité graphique la moitié du côté, l'équation de ($\mathscr{C}$) est $y = f(x) = \dfrac{1{,}5}{1 + x^2}$.



On ne peut pas calculer simplement l'aire de la partie grisée, sous la courbe ($\mathscr{C}$) mais nous allons déterminer une estimation de cette aire en calculant la proportion entre cette aire et celle du carré.

Ce rapport peut être approché par une loi uniforme dans le plan :
Dans le lancer de la fléchette, la détermination du point d'impact ne dépend que du hasard. Le tirage au sort du point de coordonnées (x;y) se modélise par le couple (Y_1;Y_2), chacune de ces variables étant uniforme sur le segment [0;2]. Comme la loi uniforme sur un segment se traduit par des rapports de longueurs de segments, la loi uniforme dans le plan se traduit par des rapports d'aires.

Simulation avec la calculatrice
On simule le lancer d'une fléchette à l'aide de la fonction rand de la calculatrice. Les coordonnées x et y du point d'impact de la fléchette sont Y_1 et Y_2.
On suppose bien sûr qu'aucune fléchette ne rate sa cible…
1) Déterminer Y_1 et Y_2 à l'aide de la fonction rand de la calculatrice.

2) Exprimer en fonction de Y_1 et Y_2 les deux parties de la cible atteinte par la fléchette.

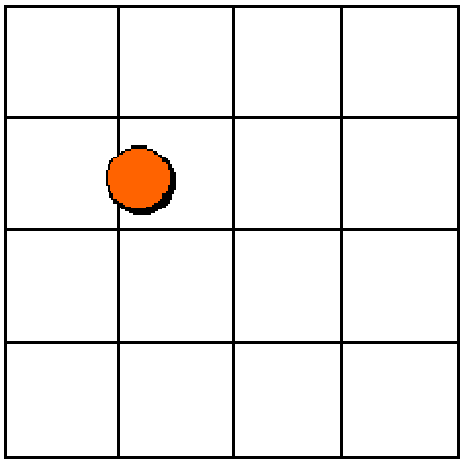
3) Créer un algorithme permettant de connaître le nombre d'impacts de 100 fléchettes lancées sur la cible. Déterminer alors la proportion des impacts de la zone grisée puis une estimation de son aire.

4) Traduire cet algorithme en un programme sur votre calculatrice permettant de simuler 100 lancers puis 1 000 lancers.

5) Modifier le programme précédent pour estimer la probabilité d'être sur la courbe ($\mathscr{C}$).
Ce résultat était-il prévisible ?

Le lancer de Palet[1]

Cet exercice permet de réutiliser dans un cadre concret, les résultats découverts dans les 2 parties précédentes
Sur un sol recouvert uniformément de carreaux de coté 10 cm, on fait glisser au hasard un palet ayant la forme d'un disque de diamètre d = 5 cm.
A chaque arrêt du palet, on regarde si celui-ci touche ou non une rainure.
On notera "A" l'événement : "le palet ne touche pas de rainure".



1) D'après vous, l'événement "A" a-t-il :

- moins d'une chance sur deux de se produire ?
- plus d'une chance sur deux de se produire ?
- environ une chance sur deux de se produire ?

On demande une réponse intuitive non justifiée .....

2) Méthode expérimentale (Correspondance entre la fréquence et la probabilité)
Trouvez une méthode qui permette de répondre à la question précédente :
a) En choisissant un protocole expérimental réaliste modélisant cette situation.
b) En utilisant un programme sur calculatrice permettant d'afficher votre réponse.

3) Méthode théorique (Proportionnalité entre Aire et Probabilité)
a) Où doit se situer le centre du palet pour que celui-ci ne touche pas de rainure ?
b) En vous inspirant de la méthode de Monte-Carlo, calculez la probabilité de "A".

[1] http://www.crdp.ac-grenoble.fr/imel/delahaye/td2/td2.html

Hörner

Remarque : Soit $P(x)$ un polynôme de degré $n=3$. $P(x)$ peut s'écrire sous la forme suivante :

$$\begin{aligned} P(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 \\ &= a_0 + x(a_1 + a_2x + a_3x^2) \\ &= a_0 + x(a_1 + x(\boldsymbol{a_2 + xa_3})) \end{aligned}$$

On peut utiliser cette forme pour calculer l'image d'un réel x_0 par P.
C'est la méthode de Hörner.

Algorithme :

Demander le degré n du polynôme
la valeur de x_0
les coefficients du polynômes

Affecter la valeur de a_n à A

Pour i allant de n à 1
affecter la valeur de $Ax + a_{i-1}$ à A

Afficher A.

Voici comment cet algorithme peut se traduire en langage machine :

Commentaires	TI	CASIO
	prompt N,X	"N" : ? $\rightarrow$ N "X" : ? $\rightarrow$ X
TI : Vide la liste L_1 du mode statistique.	clrlist L_1	
casio : Annonce la dimension de la liste (sur une GRAPH 35)		N+1 $\rightarrow$ Dim List1
Demande les coefficients du polynôme puis les met dans la liste 1. **TI** : L'utilisateur devra entrer les coefficients sous cette forme : $\{2;1;3\}$ **casio** : donner les coefficients dans l'ordre $a_0; a_1; a_2...$	Disp "$\{a_0; a_1; a_2; ...\}$" input L_1	for 1 $\rightarrow$ I to N+1 "coef ?" : ? $\rightarrow$ List1[I] next
remarque : les coefficients sont indicés de 0 à n mais les éléments de la liste sont numérotés de 1 à $n+1$.	L_1(N+1) $\rightarrow$ A	List1[N+1] $\rightarrow$ A
Lorsque I décrémente, il faut préciser le **pas : -1**	for (I, N+1, 2, **-1**) $A \times X + L_1(I-1) \rightarrow A$ end	for N+1 $\rightarrow$ I to 2 **step –1** $A \times X + \text{List1}[I-1] \rightarrow A$ Next
	Disp A	A

Remarque : Sur une GRAPH 25 il faut remplacer l'instruction N+1 $\rightarrow$ Dim List1 par :

seq(0,X,1,N+1,1) $\rightarrow$ List 1

(ce qui revient à remplir la liste 1 de N+1 zéros)

Questions :

1. Tester cet algorithme avec $P(x) = 2 - 4x + 3x^2 + 2x^3 - 0,5x^4$ et $x_0 = 7$.
2. Ecrire ce polynôme sous la forme présentée en introduction puis déterminer le nombre d'opérations effectuées (additions et multiplications) pour calculer P(7).
3. Compter le nombre d'opérations effectuées pour le calcul suivant :

$$P(7) = 2 - 4\times 7 + 3\times 7^2 + 2\times 7^3 - 0,5\times 7^4$$

4. Ecrire l'algorithme qui permet de calculer l'image de x_0 comme dans le calcul précédent.

TI	CASIO
prompt N,X clrlist L_1 Disp "$\{a_0; a_1; a_2; ...\}$" input L_1 $L_1(1) \to A$ for (I, 2,N+1) $L_1(I)\times X^{I-1} + A \to A$ end Disp A	"N" : ? $\to$ N "X" : ? $\to$ X N+1 $\to$ Dim List1 for 1 $\to$ I to N+1 "coef ?" : ? $\to$ List[1] next List1[1] $\to$ A for 2 $\to$ I to N+1 List1[I]$\times X^{I-1} + A \to A$ Next A

Pour **comparer les vitesses d'exécution** des deux programmes précédents (calcul de l'image d'un nombre par un polynôme) sur les calculatrices **TI**, , on pourra les faire tourner avec le polynôme de degré 100 et dont tous les coefficients sont 10 :

seq(10,K,1,101,1)

(Avec X=2 ; 4 secondes avec Hörner et 7 secondes avec l'autre !)

Pour information :

Pour un polynôme de degré n :

- Le nombre d'opérations avec l'écriture $a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1} + a_nx^n$ est de l'ordre de $\frac{n^2}{2}$ $\left(\text{plus précisément } \frac{n^2}{2} + \frac{3}{2}n\right)$

Le nombre d'opérations avec l'écriture $a_0 + x(a_1 + x(a_2 + x(a_3 + ... (a_{n-1} + xa_n) ...)$ est de l'ordre de $2n$.

Calculatrice TEXAS

Instructions de programmation possibles en français	Instruction correspondante dans le langage de programmation	Où trouver cette instruction ?
Afficher à l'écran : mot Afficher à l'écran : A	DISP " mot " DISP A	• Taper sur la touche PRGM, puis avec le curseur droit sélectionner I/O, puis **DISP**. • Les guillemets **"** s'obtiennent en tapant ALPHA, puis +.
La valeur donnée par l'utilisateur est stockée sous la variable A	INPUT A	Taper sur la touche PRGM, puis avec le curseur droit sélectionner I/O, puis **INPUT**.
Affecter cette valeur sous la variable A	→ A	Taper sur la touche STO> (qui est à côté du 1).
Si… Alors … Sinon …	IF … THEN … ELSE …	Taper sur la touche PRGM, puis sélectionner **IF**, **THEN** ou **ELSE**.
Placer l'étiquette N° n	LBL n	Taper sur la touche PRGM, puis sélectionner **Lbl**, en vous déplaçant en bas à l'aide du curseur pour le trouver.
Aller à l'étiquette N° n	GOTO n	Taper sur la touche PRGM, puis sélectionner **Goto**, en vous déplaçant en bas à l'aide du curseur pour le trouver.
Faire une pause dans l'affichage	PAUSE	Taper sur la touche PRGM, puis sélectionner **Pause**, en vous déplaçant en bas à l'aide du curseur pour le trouver.
Fin du programme	END	Taper sur la touche PRGM, puis sélectionner **End**, en vous déplaçant en bas à l'aide du curseur pour le trouver.
Prendre la partie entière de A	INT(A)	Taper sur la touche MATH, puis sélectionner NUM à l'aide du curseur droit, puis **Int(** .
Est différent de …	≠	Taper sur la touche TEST (2^{nd} puis MODE), puis sélectionner **≠**

➢ **Pour créer un nouveau programme :**
Taper sur la touche PRGM, puis sélectionner NEW.
Écrire ensuite le nom du programme que vous créez.

➢ **Pour rédiger le programme :**
- Taper les instructions, puis, après chaque instruction, taper sur ENTER (**:** vont alors apparaître en fin de ligne)
- Pour modifier le programme après en être sorti, taper sur la touche PRGM, puis sélectionner EDIT ainsi que le programme en question.

➢ **Pour exécuter le programme :**
Taper sur QUIT (2^{nd} puis MODE), puis taper sur la touche PRGM, sélectionner le programme, puis EXEC.

Calculatrice CASIO

Instructions de programmation possibles en français	Instruction correspondante dans le langage de programmation	Où trouver cette instruction ?
Afficher à l'écran : mot	" mot "	Les guillemets " s'obtiennent, selon votre modèle de calculatrice : - soit, en bas de votre écran, il y a SYBL en F6, le sélectionner, puis sélectionner les " . - S'il n'y a pas SYBL en bas de l'écran, taper sur la touche ▶ (qui est à droite de F4), puis sélectionner les " .
La valeur donnée par l'utilisateur est stockée sous la variable A	? → A	• Pour trouver le **?** : Taper sur la touche PRGM (SHIFT puis VARS). Soit il se trouvera en F4, soit, il faut aller le chercher en tapant sur la touche ▶ (qui est à droite de F4). • La flèche → est sur la touche → qui se situe juste au dessus de AC/ON.
Si… Alors … Sinon …	IF … THEN … ELSE …	Taper sur la touche PRGM, puis sélectionner COM en F1, puis **IF**, **THEN** ou **ELSE**.
Placer l'étiquette N° n	LBL n	Taper sur la touche PRGM, puis sélectionner JUMP en F3, puis **Lbl** .
Aller à l'étiquette N° n	GOTO n	Taper sur la touche PRGM, puis sélectionner JUMP en F3, puis **Goto** .
Faire une pause dans l'affichage	◢	Taper sur la touche PRGM, soit il se trouvera en F5, soit, il faut aller le chercher en tapant sur la touche ▶ (qui est à droite de F4).
Fin du programme	STOP	Taper sur la touche PRGM, puis sélectionner CTL en F2, puis **Stop** en F4 .
Prendre la partie entière de A	INT(A)	Taper sur la touche OPTN, puis chercher NUM à l'aide de la touche ▶ (qui est à droite de F4), puis sélectionner **Int(** .
Est différent de …	≠	Taper sur la touche PRGM, puis chercher REL à l'aide de la touche ▶ (qui est à droite de F4), puis sélectionner ≠

➢ **Pour créer un nouveau programme :**
Taper sur la touche MENU, puis sélectionner le menu PRGM, puis NEW.
Écrire ensuite le nom du programme que vous créez.

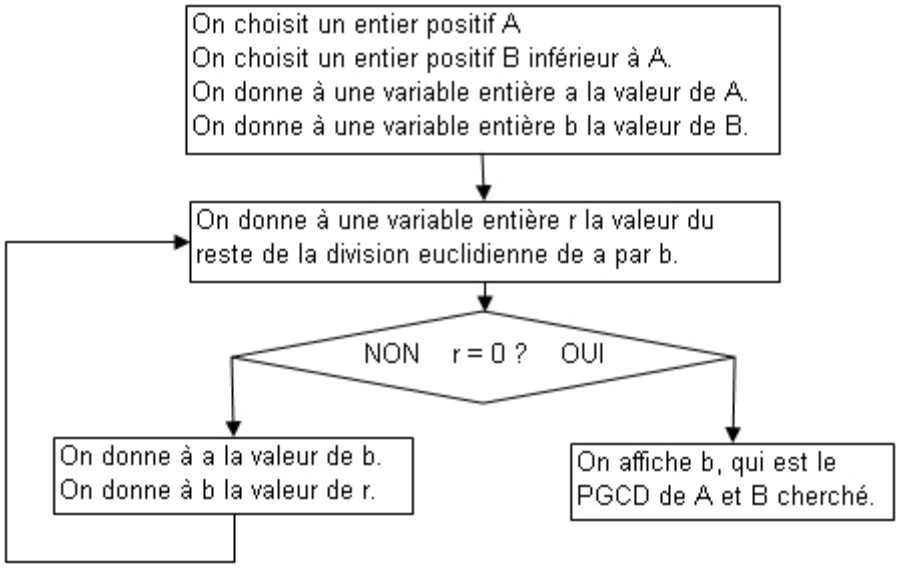
➢ **Pour rédiger le programme :**
- Taper les instructions, puis, après chaque instruction, taper sur ENTER (↵ va alors apparaître en fin de ligne)
- Pour modifier le programme après en être sorti, taper sur la touche EXIT, puis sélectionner EDIT ainsi que le programme en question.

➢ **Pour exécuter le programme :**
Taper sur EXIT, puis sélectionner EXE ainsi que le programme en question.

Réflexions sur les contenus

- On réalisera la programmation sur calculatrice ou tableur de l'algorithme d'Euclide pour la recherche du pgcd.
- L'ensemble des diviseurs communs à plusieurs entiers est l'ensemble des diviseurs de leur pgcd.

Ci-dessous, un extrait du document d'aide à l'utilisation du logiciel Execalgo.



Type d'instruction	Avec Execalgo	Sur Casio	Sur TI
Affectation	Donner à A la valeur 123456	123456 → A	123456 → A
Affectation	Donner à B la valeur 56745	56745 → B	56745 → B
Point de branchement	[Début de la boucle]	Lbl 1	Lbl 1
Affectation	Donner à R la valeur reste(A,B)	A - B*Int(A/B) → R	A - B*Int(A/B) → R
Branchement conditionnel	Aller à [Sortie] si R = 0	R = 0 ⇒ Goto 2	If R = 0 Goto 2
Affectation	Donner à A la valeur B	B → A	B → A
Affectation	Donner à B la valeur R	R → B	R → B
Branchement	Aller à [Début de la boucle]	Goto 1	Goto 1
Point de branchement	[Sortie]	Lbl 2	Lbl 2
Affichage	Afficher B	B ◢	Disp B

source : http://pedagogie.ac-toulouse.fr/math/stages2/interaca05/l2_arithmetique_serieL/livret.doc

Programmation sur tableur

Algorithme de Babylone

Équipe Acacadémique Mathématiques, Bordeaux, 2002

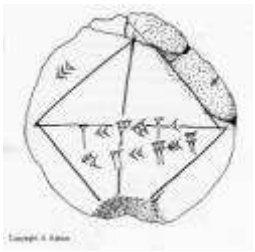
Destination
Professeurs

Niveau
Terminale L, ooption facultative, nouveaux programmes

Type
Papier et TICE (tableur). Fichiers babylone.123, babylone.xls

Commentaires
Sur la base d'un texte historique on introduit une méthode de calcul de la racine carrée d'un nombre A à l'aide de suites récurrentes. Cette activité peut être utilisée clé en main à l'aide des fichiers tableurs joints ou il peut être demandé aux élèves de construire les suites afin d'étudier leur convergence.

Problème



Il existe un très ancien document babylonien donnant une approximation de la racine de 2 sous la forme 1 24 51 10 en sexagésimal, c'est-à-dire, en décimal : 1,414 212 963, au lieu de 1,414 213 562. Cliquer ici pour trouver une image, avec des commentaires, sur un site en anglais

Les historiens des mathématiques se sont interrogés pour savoir comment les Babyloniens avaient obtenu cette excellente approximation. On trouvera une réponse possible dans une activité, sur ce site, ainsi qu'une autre activité basée sur un texte d'Euler.

Modélisation
Un rectangle R_1 d'aire A a pour dimensions x_1 et y_1.
On fabrique le rectangle R_2 de dimensions

$$x_2 = \frac{x_1 + y_1}{2} \text{ et } y_2 = \frac{1}{\frac{x_1 + y_1}{2}}$$

donc de même aire que le rectangle R_1.
En itérant le processus on va « *se rapprocher* » d'un carré d'aire A.
Créer à l'aide d'un tableur les suites des valeurs des nombres x_n et y_n.
Comparer les résultats obtenus à $\sqrt{3}$.

Programmer l'algorithme d'Euclide avec un tableur

Le tableur permet d'afficher clairement, si on le souhaite, les différentes étapes de l'algorithme. Le travail présenté ici se propose de bâtir une feuille de calcul « interactive », c'est-à-dire l'utilisateur et n'affiche que ce qui est utile.
Il est nécessaire de connaître certaines fonctions comme la fonction Si et sa syntaxe :
SI(valeur de test ; action si test positif ; action si test négatif).

Mise en œuvre
1) Commencer par mettre en forme le début de la feuille :

	A	B	C	D	E	F	G
1	PGCD	(		;		)=	
2	dividende	=	diviseur	×	quotient	+	reste

On entre les deux entiers non nuls en Cl et El.

2) En A3, on reportera la plus grande des deux valeurs absolues des entiers proposés et en C3 la plus petite. On fait en sorte que la feuille n'affiche rien si les deux cases Cl et El ne sont pas complètement renseignées.

=SI(ou(estvide(C1);estvide(E1));" ";max(abs(C1);abs(E1)))

Renvoie VRAI si au moins l'une des deux cases Cl ou El est vide

Valeur Absolue

• En C3, la même chose en remplaçant max par min. Ainsi l'utilisateur ne se préoccupe ni du signe, ni de l'ordre des valeurs.
• On calcule ensuite le quotient et le reste dans la division euclidienne de A3 par C3. En entre :

=Si(et(estnum(A3);estnum(C3));ent(A3/C3);" ")

• En G3 on entre :

=Si(et(estnum(A3);estnum(C3));mod(A3;C3);" ")

La fonction estnum est une fonction booléenne qui renvoie VRAI si la case testée contient un nombre.
La fonction ent(a/b) renvoie la partie entière de la division a/b.
La fonction mod(a;b), renvoie le reste dans la division euclidienne de a par b.
Les cases B3, D3 et F3 n'affichent rien si la case A3 n'est pas un nombre ; donc, on a :
=Si(estnum(A3);"=";" ") en B3,
et on remplace = par × en D3 et + en F3.

3) Il reste à écrire la ligne 4, car, à ce stade, une recopie des formules vers le bas entraînerait une erreur due à l'utilisation de la ligne 2 qui ne contient pas de nombre.
Pour remplir A4 et C4, on doit d'abord tester si A3 et C3 sont bien numériques, puis si G3 (le reste précédent) est non nul.
Si ces trois conditions sont remplies, on place C3 en A4 et G3 en C4.

- En A4, on a donc :

=SI(et(estnum(A3);estnum(C3);G3<>0);C3;" ")

Adapter la formule pour C4.

Les calculs du quotient et du reste sont les mêmes ; donc on peut, pour les colonnes B, D, E, F et G recopier les formules vers le bas jusqu'à la ligne 3.

4) On sélectionne la plage A4 : G4 et on copie vers le bas jusqu'à un nombre suffisant de lignes, 80 par exemple (nettement suffisant, il est rare d'avoir autant de lignes pour l'algorithme d'EUCLIDE !).
5) Il reste maintenant à calculer le PGCD. On va utiliser la colonne H.
Le PGCD est le dernier reste non nul, donc, c'est le diviseur dans la ligne où le reste est nul.

• En H3, on entre :

=Si(ou(estvide(A3);estvide(C3));" ";Si(G3=0;C3;" "))

puis on tire vers le bas jusqu'à la ligne 80.
• On masque ensuite cette colonne H en la sélectionnant, puis en utilisant le menu correspondant, par le clic droit de la souris.

6) Enfin en G1, on entre :

=Si(ou(estvide(C1);estvide(E1));" ";max(H3:H80))

On va ainsi rechercher la seule valeur non nulle de la colonne, c'est-à-dire le PGCD. On peut mettre cette case en gras et dans une taille supérieure au reste pour que le résultat saute aux yeux.

Applications
l) Donner à a et b les valeurs 45 212 et 30 148.

G1 — =SI(OU(ESTVIDE(C1);ESTVIDE(E1));" ";MAX(H3:H80))

	A	B	C	D	E	F	G	H	I
1	PGCD	(	45212	;	20148	)=	**4**		
2	dividende	=	diviseur	×	quotient	+	reste		
3	45212	=	20148	×	2	+	4916		
4	20148	=	4916	×	4	+	484		
5	4916	=	484	×	10	+	76		
6	484	=	76	×	6	+	28		
7	76	=	28	×	2	+	20		
8	28	=	20	×	1	+	8		
9	20	=	8	×	2	+	4		
10	8	=	4	×	2	+	0	4	
11									
12									

Pour plus de sécurité, on peut ensuite protéger la feuille contre toute mauvaise manipulation. Sélectionner la feuille, puis dans le menu format-cellule, verrouiller la protection.
Ensuite, sélectionner C1 et E1 et ôter leur protection.
Enfin, dans le menu outils-protection, sélectionner protéger la feuille (le mot de passe n'est pas utile).

2) On peut de même bâtir une feuille permettant, à l'aide de cet algorithme, de trouver un couple solution dans $\mathbb{Z}^2$ pour l'équation :

$au + bv = \text{PGCD}(a;b)$.

Algorithmique dans les constructions géométriques : Promenade aléatoire sur une droite

Sur une règle graduée de $-n$ à n ($n \in \mathbb{N}^*$), un point lumineux initialement positionné au point O de la règle, se déplace à chaque seconde d'une unité vers la droite ou vers la gauche avec des probabilités respectives p et $q = 1 - p$.
On appelle X la variable aléatoire égale à l'abscisse du point lumineux au bout des n déplacements.

Soit X_1 le nombre de déplacement vers la droite et X_2 vers la gauche.
X_1 suit la loi binomiale de paramètres n et p, X_2 de paramètres n et q.
On a, de plus,

$X_1 + X_2 = n$
$X = X_1 - X_2$

Donc

$X = X_1 - X_2 = 2X_1 - n$

Détermination des valeurs prises par X :
Comme $X_1(\Omega) = \{0;1;\ldots;n\}$, $2X_1(\Omega) = \{0;2;\ldots;2n\}$ et

$X(\Omega) = \{-n;-n+2;\ldots;n-2;n\}$

On remarque tout d'abord que $0 \in X(\Omega)$ lorsque n est pair.

Valeurs de la loi de probabilité :
$\forall k \in \{-n;-n+2;\ldots;n-2;n\}$,

$$P(X = k) = P(2X_1 - n = k) = P\left(X_1 = \frac{k+n}{2}\right) = \left(\frac{n}{\frac{k+n}{2}}\right) p^{\frac{k+n}{2}} (1-p)^{n-\frac{k+n}{2}} = \left(\frac{n}{\frac{k+n}{2}}\right) p^{\frac{k+n}{2}} q^{\frac{n-k}{2}}$$

Algorithme sous Scilab (transposable sur une calculatrice programmable, type TI ou Casio) :

```
x=1:200;
y=zeros(1,200);
Z=0;

for i=1:200 do
Z=Z+2*floor(2*rand())-1;
    //semblable à
    //x=rand()
    //if x<0.5 then z=z-1
    //        else z=z+1
    //end
y(i)=Z;
end

plot(x,y,'.')
```

---

fichier rayon_lu.txt
à exécuter par Scilab

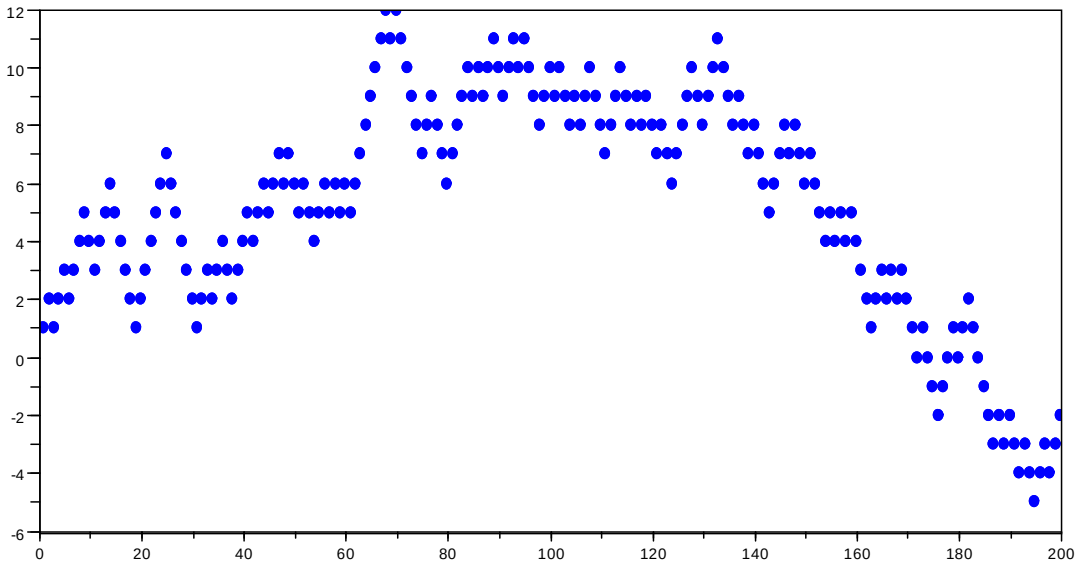
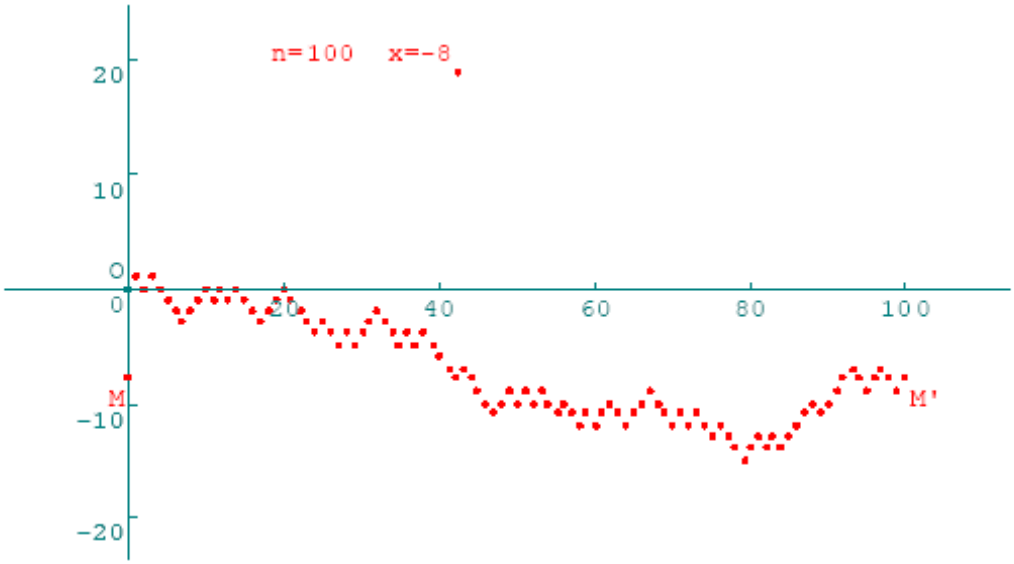


Illustration sous Geoplan ; réalisation de la simulation suivante :



n contient le nombre de déplacements, *x* l'abscisse finale.

Construction géoplan : http://www.ac-grenoble.fr/maths/perso/delahaye/SIMUGEOP.pdf
pour le déplacement sur la droite de M : pages 2 à 4, fichier rayon_lu.g2w

Algorithme employé dans le cadre de cette construction sous géoplan:

n et *x* : entier
b réel de [0;2]
Cm0 n=0
x=0
Cm3 pour i=1 à 100 faire
Cm1 b=2*rand (b est un réel quelconque de [0,2[)
a=2*int(b)-1
Cm2 x=x+a
n=n+1
finpour

pour la visualisation de la marche de *M'* : pages 4 à 5, fichier rayon_lv.g2w

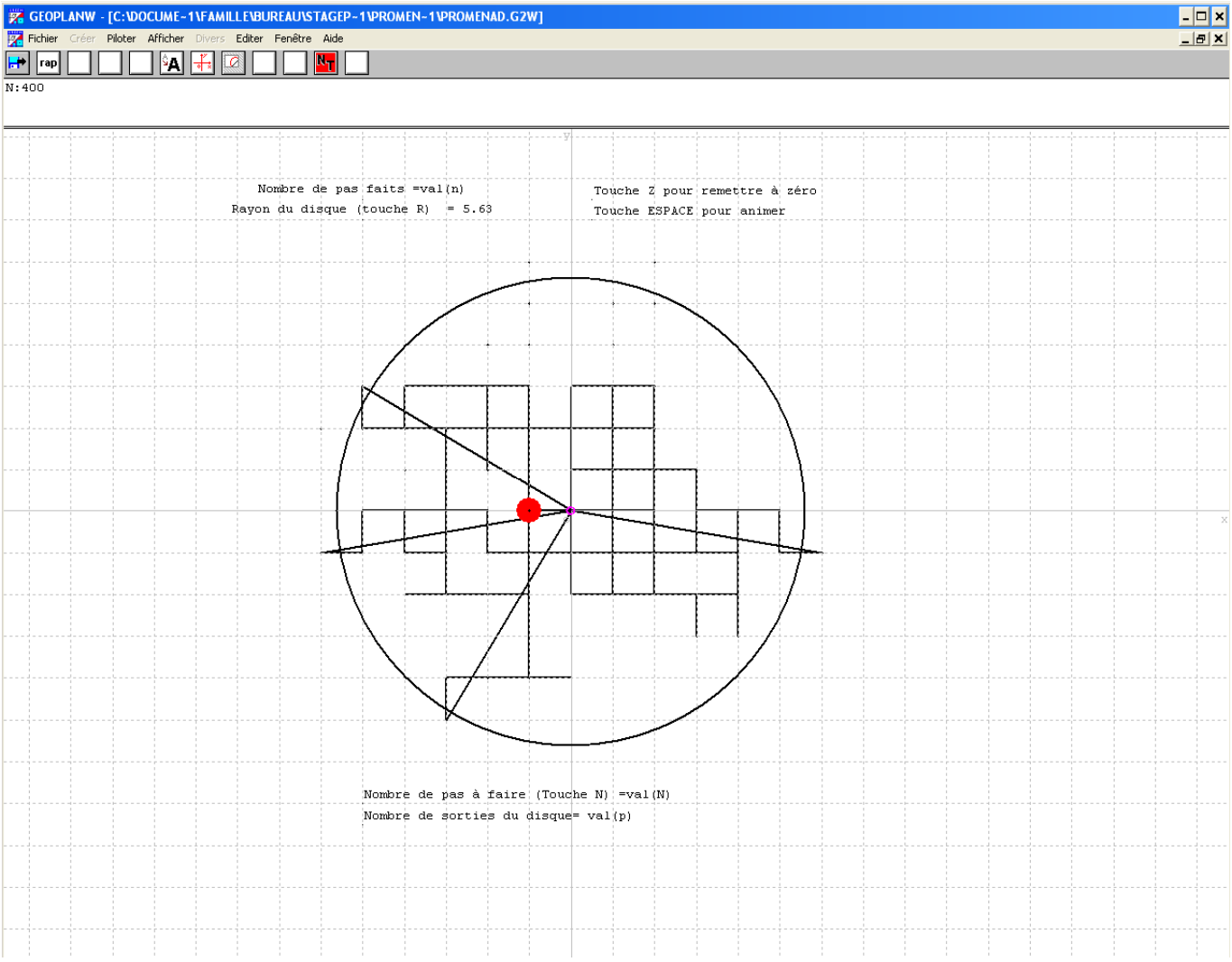
Une version équivalente de ces programmes serait de faire des simulations de la planche de Galton.
De nombreux programmes sous Scilab, via le tableur sont disponibles.

regarder le fichier galton.xls

promenades aléatoires dans le plan
Objet se déplaçant sur un réseau triangulaire ou carré ou hexagonal ou …

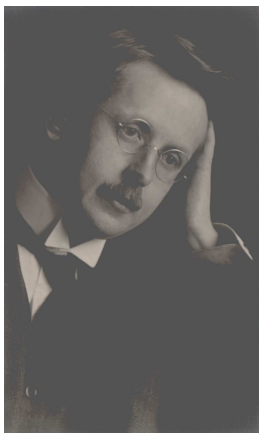
De nombreuses simulations existent également.
cette première animation permet de vérifier l'équiprobabilité des lieux.
fichier 1_4[1].swf récupéré sur http://www.calmette.net/maths/flash/articles/index1_4.htm

On peut envisager, en ne limitant plus la taille du réseau, de regarder également le comportement dans le plan : déplacements aléatoires sur un réseau maillé jusqu'à ce que le mobile s'éloigne d'une certaine distance de l'origine : fichier promenda.pdf et fichier promenad.g2w



Utilisation du principe de promenade aléatoire pour visualiser une conjecture[2] :

Si on considère la suite des puissances de $\frac{3}{2}$ et si on ne regarde que les chiffres après la virgule, on tombe sur un problème que les mathématiciens ne savent pas encore résoudre : cette suite de nombres est-elle aléatoire ?



Autre formulation du problème : **Le critère de Weyl**

En 1916, Hermann Weyl énonce le critère d'équirépartition modulo 1 : la suite (u_n) est équirépartie modulo 1 si et seulement si, pour tout entier h non nul, $\frac{1}{N}\sum_{n=1}^{N} \exp(2i\pi h u_n) \to 0$ lorsque $n \to +\infty$.

On peut donner une traduction graphique de ce critère en associant aux N premiers termes hu_n des points d'un cercle de rayon 1. Le critère de Weyl exprime que pour tout h, le barycentre de l'ensemble ainsi obtenu tend vers l'origine lorsque n tend vers l'infini.

Ce critère permet de montrer que la suite des multiples d'un nombre irrationnel est équirépartie modulo 1.

Afin d'illustrer cette question, vous devrez utiliser une méthode graphique :
Comment construire une promenade associée à une suite modulo 1 ?

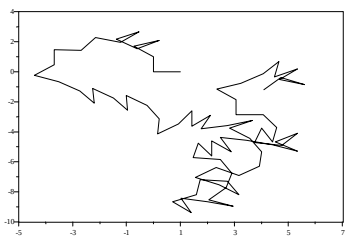
- Le point de départ est le centre d'un cercle de rayon 1.

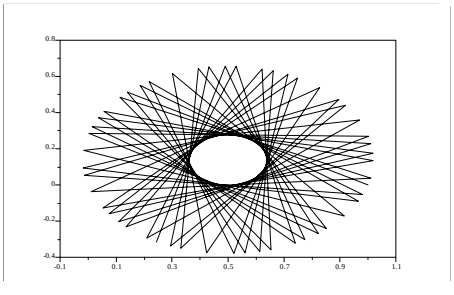
- A la première étape, on place le premier terme de la suite, suivant sa valeur, sur le cercle et l'on joint les deux points. L'angle permettant de repérer ce point étant proportionnel à sa valeur, modulo 1.

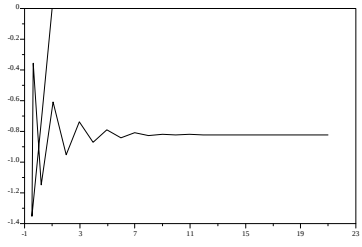
Exemple : si la première valeur est 3,258 968 … soit 0,258 968 … modulo 1, l'angle permettant de repérer ce point sur le cercle est 0,258 968 … $\times 2\pi$ rad

- Le dernier point dessiné devient le centre du nouveau cercle qui va servir à la deuxième étape et ainsi de suite…

Les schémas suivants présentent respectivement une promenade aléatoire des puissances de $\frac{3}{2}$, une autre sur les multiples de $\sqrt{2}$ et enfin des puissances de $\varphi = \frac{1+\sqrt{5}}{2}$, le "nombre d'or".







---

voir fichier promenades_aleatoires.txt

---

[2] D'après *La suite des puissances de 3/2*, François Dress et Michel Mendès France, La recherche, Octobre 2001.

Exercice 1 Un générateur de Charabia latin

Nous nous proposons de réaliser un générateur de charabia latin (jeu utilisé par les petits enfants de langue anglaise), c'est-à-dire un programme qui modifie un mot du français en un mot d'un charabia latin. Cette transformation s'effectue en plaçant la première lettre du mot à la fin, et en y ajoutant la lettre "a". Ainsi, le mot "tortue" devient "ortueta", "Scilab" devient "cilaba", et ainsi de suite.

Ecrivons maintenant un programme Scilab qui lira une phrase en français et écrira son équivalent en charabia latin. Pour des raisons de simplicité, nous ne tiendrons pas compte du problème des lettres majuscules et des signes de ponctuation.

L'écriture du programme traduisant le fonctionnement de ce jeu sera facilitée par la construction des modules suivants :

1) Ecrire la fonction **[ch]=Entrez_Texte()** permettant d'entrer le texte en français représenté par la chaîne de caractère **ch** (vérifiez que le texte comporte moins de 80 caractères).
2) Ecrire la fonction **[n]=Nombre_mots()** permettant de déterminer le nombre **n** de mots de la phrase **ch** en utilisant le nombre d'espaces.
3) Ecrire la fonction **[m]=recherche_mot(ch,n)** permettant d'affecter à la chaîne **m** le **n**$^{\text{ième}}$ mot de la phrase **ch**.
4) Ecrire la fonction **[ma]=Transf_Franc_Charabia(mf)** permettant de transformer le mot français de la chaîne **mf** en son équivalent en charabia latin dans la chaîne **ma**.
5) Ecrire le programme complet permettant de transformer une phrase complète en son équivalent en charabia latin.

Exercice 2 Le nombre d'Or

Le nombre d'Or, noté φ, représente une proportion idéale pour certains. Il vérifie, entre autres, la relation suivante :

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\dots}}}}}$$

on peut définir une suite (x_i) de valeurs approchées de φ, convergeant vers φ, de la façon suivante :

$x_1 = 1$

$x_2 = 1 + \frac{1}{1} = 2$

$x_3 = 1 + \cfrac{1}{1 + \frac{1}{1}} = 1,5$

$x_4 = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{1}}} = 1,666\ldots$

Ecrire une fonction en Scilab qui calcule une approximation de φ en retournant le $n^{\text{ième}}$ élément de cette suite. La fonction recevra donc comme paramètre un entier *n*.

Exercice 3 Le carré de Polybe

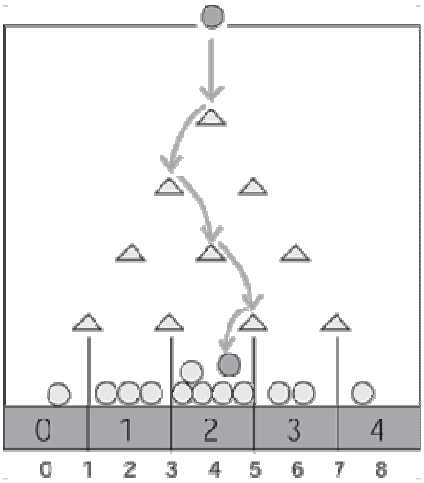
Polybe est un historien grec (env. 200 - 125 av. J.-C.) qui est à l'origine du premier procédé de chiffrement par substitution. C'est un système de transmission basé sur un carré de 25 cases (on peut agrandir ce carré à 36 cases, afin de pouvoir ajouter les chiffres ou pour chiffrer des alphabets comportant davantage de lettres, comme l'alphabet cyrillique):

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i	j
3	k	l	m	n	o
4	p	q	r	s	t
5	u	v	x	y	z

En français, on supprime le W, qui sera le cas échéant remplacé par V. En anglais, on agrège le I et le J.
Chaque lettre peut être ainsi représentée par un groupe de deux chiffres: celui de sa ligne et celui de sa colonne. Ainsi "e"=15, "u"=51, "n"=34, ...Polybe proposait de transmettre ces nombres au moyen de torches. Une torche à droite et cinq à gauche pour transmettre la lettre "e" par exemple. Ce procédé permettait donc de transmettre des messages sur de longues distances. On peut aussi transmettre les coordonnées des lettres en tapant des coups sur un mur, sur la tuyauterie, etc.

Construire la fonction **[ch]=carre_polybe(c)** qui effectue les tâches suivantes :
- Construction d'un tableau t qui est le tableau de type 5×5 ci-dessus.
- Pour une chaîne de caractère c de la forme correspondant à l'exemple c='1.3 3.5 1.4 1.5 4.3', la transformer, à l'aide du tableau t, en la chaîne de caractères ch qui correspond au codage par le carré de Polybe.

Exercice 4 La table de Galton



Il s'agit dans cet exercice de simuler à *n* reprises l'expérience suivante :
Une bille est lâchée au dessus du système ci-contre. Elle peut aller avec une chance sur deux à gauche ou à droite du premier triangle, elle peut ensuite, toujours avec une chance sur deux, aller à gauche ou à droite des triangles des rangées inférieures.
Il s'agit de dénombrer, après *n* lâchés de billes le nombre de billes dans chacune des cases 0 ou 1 ou 2 ou 3 ou 4.
On peut envisager le tableau de taille 5×9 suivant dont les * indiquées seront les positions possibles des billes

				*				
			*		*			
		*		*		*		
	*		*		*		*	
*		*		*		*		*

Considérez la variable *x* égale à 1 (première ligne) et *y* à 5 (5ème colonne).
Suivant le résultat d'un tirage aléatoire, la variable *x* sera augmentée de 1 et la variable *y* sera augmentée ou baissée de 1. En répétant ce tirage aléatoire trois autres fois, vous aurez ainsi simulé une expérience.
Un tableau de taille 1×9 correspondant à la dernière ligne recevra alors le résultat de la position finale.

Construire une fonction **[]=Galton();** réalisant les instructions suivantes :
- Demande à l'utilisateur du nombre de simulations à réaliser.

- Simulation des expériences
- Tracé de la courbe associée aux résultats (nombre de billes dans un numéro en fonction de ce numéro).

Exercice 5 Extrait de l'épreuve de la banque Agro-Véto, Mathématiques A, 2006

1) a) Ecrire un algorithme fournissant le produit de deux matrices appartenant à $M_4(\mathbb{R})$, l'algèbre des matrices carrées d'ordre 4 à coefficients dans $\mathbb{R}$.

b) Combien d'opérations (additions et multiplications) sont-elles nécessaires ?

2) Un centre de conditionnement physique pense installer une piscine à l'intérieur de ses murs. Un sondage auprès des membres actuels indique que tous les membres continueront à utiliser la salle de conditionnement physique et que 65% des membres utiliseront aussi la piscine. Parmi les nouveaux membres qui devraient s'inscrire au centre une fois la piscine installée, 78% utiliseront la piscine et 59% utiliseront la salle de conditionnement physique. Le centre compte présentement 440 membres.

a) Créer la matrice de transition A qui décrit l'utilisation prévue de la salle de conditionnement physique et de la piscine en fonction des anciens et des nouveaux membres une fois la piscine ouverte.

b) En supposant, uniquement pour cette question, que 200 nouveaux membres s'inscriront, écrire les instructions Scilab utilisant la matrice A pour afficher le nombre de personnes utilisant la salle de conditionnement physique ainsi que le nombre de personnes utilisant la piscine.

c) La piscine ne sera construite que si au moins 600 personnes s'y inscrivent tout en gardant 800 personnes pour la salle de conditionnement physique.

Ecrire les instructions Scilab nécessaires pour afficher le nombre de nouveaux membres à inscrire pour réaliser cette fréquentation de la piscine.

Exercice 6

Pour chacune des questions suivantes, vous donnerez uniquement les lignes de programme Scilab permettant de donner la réponse, avec d'éventuels commentaires supplémentaires permettant de les expliquer. Aucun calcul "à la main" n'est demandé.

1) Un chocolatier fabrique trois sortes de chocolat avec du cacao, du lait, du sucre et du beurre. Le tableau suivant donne les quantités d'unités nécessaires à la fabrication d'une unité de chaque sorte de chocolat.

Type de chocolat	**Cacao**	**Lait**	**Sucre**	**Beurre**
Asia	6	4	4	0
América	7	5	3	2
Africa	8	2	2	4

a) Représenter les données par une matrice M de dimension 4×3.

b) Il reçoit une commande de 5 unités de chocolat *Asia*, 6 unités de chocolat *América* et 9 unités de chocolat *Africa*.

- Calculer les quantités nécessaires pour chaque produit.
- Les prix respectifs de chaque composant, en euros par unités, sont respectivement 4, 3, 5 et 6.
  Calculer le prix total pour cette commande.

2) Modélisation d'un échange entre deux milieux

Deux récipients *A* et *B* sont séparés par une membrane perméable dans les deux sens. On place dans les récipients *A* et *B* deux solutions contenant respectivement a_0 molécules (dans *A*) et b_0 molécules (dans *B*). On suppose que, toutes les heures, 20% des molécules passent de *A* dans *B* et 10% des molécules passent de *B* dans *A*. On note a_n et b_n les nombres respectifs de molécules présentes dans *A* et *B* au bout de *n* heures.

a) Montrer que $\begin{cases} a_{n+1} = 0{,}8a_n + 0{,}1b_n \\ b_{n+1} = 0{,}2a_n + 0{,}9b_n \end{cases}$ et donner l'interprétation matricielle de ce système en considérant la matrice colonne $p_n = \begin{pmatrix} a_n \\ b_n \end{pmatrix}$.

Les deux récipients n'ayant d'échanges qu'entre eux.

b) Sachant que si $a_0 = 150$ et $b_0 = 20$ (unités), quelles instructions écrire pour connaître les quantités de molécules après 10 heures ?

Quelle méthode appliqueriez vous pour connaître la répartition limite, si elle existe, entre les deux milieux ?

c) Quels sont les dosages initiaux nécessaires pour obtenir après 1 heure, une répartition égale à $a_1 = 130$ et $b_1 = 40$ (unités). Ecrire l'instruction Scilab permettant d'expliciter le résultat.

Exercice 7 Le suicide des Zélotes

L'exercice s'inspire du suicide des zélotes à Masada en avril 74. La formulation ci-après est tirée du livre, *Concrete Mathematics* de Graham, Knuth et Patashnik publié chez Addison Wesley en 1989 :

> *Préférant le suicide à la capture, les rebelles juifs cachés dans une grotte décidèrent de former un cercle et, en le parcourant, de tuer chaque troisième personne restante jusqu'à ce qu'il n'y ait plus personne. Mais Flavius Josèphe et un co-conspirateur inconnu ne voulaient pas de ce suicide insensé; ainsi il détermina rapidement où il devait se placer (ainsi que son ami) sur le cercle vicieux.*

Le but de cet exercice est de déterminer, grâce à Scilab et à partir d'une origine sur ce cercle (la première personne tuée), les deux dernières personnes concernées (Flavius Josèphe et son ami).

Vous supposerez pour cet exercice qu'il y a *n* personnes formant un cercle.

Ces *n* personnes seront modélisées par un tableau de taille *n* contenant la valeur 0 si l'individu associé est sauf et 0 si l'individu est tué.

1) Construire la fonction [tab]=ordre_des_tues(n) permettant :

- de construire le tableau t des individus, considérés comme tous saufs au début.
- de construire un tableau tab de même taille et contenant également que des zéros. Ce tableau va permettre de contenir tous les indices de tous les individus successifs tués.
- de faire mourir le premier individu (celui d'indice 1) et de le noter dans le tableau tab.
- tant que le tableau t contient encore des zéros :
  - ajouter 3 à l'indice de la personne
  - tant que cela donne un indice d'une personne tuée, aller à l'indice suivant
  - si l'indice dépasse n, revenir à 1 (et même 2, voire plus)
  - tuer la première personne vivante rencontrée et recommencer les trois étapes précédentes.

2) Construire la fonction []=affichage(tab) qui, après avoir déterminé la longueur de tab permet un affichage du type :

"Flavius Josèphe doit se trouver en ?ème position pour être sauf."
"L'ami de Flavius doit se trouver en ?ème position pour être sauf."

Exercice 8 Un problème d'endémie

Un individu vit dans un lieu où il est susceptible d'attraper une maladie par piqûre d'insecte. Il peut être dans l'un des trois états suivants : immunisé (I), malade (M), non malade et non immunisé (S).
D'un mois à l'autre, son état peut changer suivant les règles suivantes :
→ étant immunisé, il peut le rester avec une probabilité 0,9 ou passer à l'état S avec une probabilité 0,1.
→ étant dans l'état S, il peut le rester avec une probabilité 0,5 ou passer à l'état M avec une probabilité 0,5.
→ étant malade, il peut le rester avec un probabilité 0,2 ou passer à l'état I avec une probabilité 0,8.
Les questions suivantes devront être traitées sous Scilab et les réponses formulées devront l'être aussi sous Scilab (même si vous pouvez expliquer votre démarche par un texte clair).

1) Ecrire la matrice de transition (on considèrera respectivement 1, 2 et 3 les états I, M et S), que lon notera A .
2) On suppose qu'au départ un individu est immunisé. Calculer A^2 et en déduire la probabilité que cet individu :
→ soit malade au bout de 2 mois.
→ soit immunisé au bout de 2 mois.
3) Calculer la probabilité pour que, au bout de 1 an, de 2 ans, de 3 ans, un individu soit malade dans chacun des trois cas suivants :
→ au départ, il est immunisé,
→ au départ, il est on malade et non immunisé,
→ au départ, il est malade.
(ceux d'entre vous qui ont une calculatrice et qui ont le temps de mettre en place les calculs pourront faire une remarque).
4) Comment doit être initialement un individu pour qu'il soit certain d'être malade au bout de 1 an ?
(ceux d'entre vous qui ont une calculatrice et qui ont le temps de mettre en place les calculs pourront faire une remarque).