

Traitement MathsOntologique du sujet Bac S (spé maths) Pondichery 2014

Le sujet porte sur une suite arithmético-géométrique dont la raison est une matrice, et le terme constant un vecteur. MathsOntologie est outillé pour traiter ce genre de problèmes, grâce à ses matrices 2×2 et 2×3 . Ces matrices ont une raison d'être qui ouvre la voie à d'autres explorations d'outils.

I/ Algorithmes matriciels dans MathsOntologie

1. Matrice 2×2 dans MathsOntologie

MathsOntologie possède une classe appelée Matrice. Pour créer une matrice dans MathsOntologie, on commence donc par lui intimer l'ordre de naître (les objets sont très obéissants, ils font ce qu'on leur demande, dès lors qu'on le leur demande correctement). Ceci pose un problème métaphysique, puisque la matrice pas encore née, ne peut pas entendre qu'on lui demande de naître. La demande doit donc être envoyée, non à la matrice, mais à sa classe¹, le message de naissance s'écrivant new. Il est d'usage de commencer les noms des objets par des minuscules, et les noms des classes par des majuscules. En affichant la matrice A on constate que tous ses coefficients sont nuls :

```
Transcript
Matrice(
0.0 0.0
0.0 0.0
)

Workspace
| A |
A := Matrice new.
Transcript affiche: A.
```

Pour que la matrice A soit bien celle de l'énoncé, il reste donc à entrer ses coefficients. Ceux-ci s'appellent respectivement a11, a12, a21 et a22. Pour que a11 soit égal à 0,4 on fait ainsi :

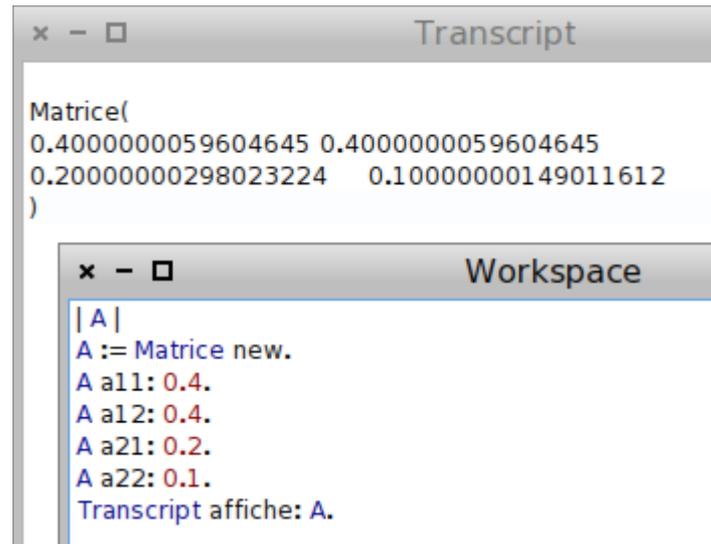
```
Transcript
Matrice(
0.4000000059604645 0.0
0.0 0.0
)

Workspace
| A |
A := Matrice new.
A a11: 0.4.
Transcript affiche: A.
```

On constate au passage des problèmes d'arrondi hérités de la machine Pharo sous laquelle tourne MathsOntologie. Il faut retenir que les dernières décimales ne sont pas correctes.

¹ Qu'on peut donc dans le cas présent, considérer comme une matrice de matrices, au sens du moule dans lequel on fabrique les matrices, ou d'organe dans lequel s'effectue la gestation de la matrice à naître...

En faisant de même pour les autres coefficients, on entre la matrice A :



```
Matrice(  
0.4000000059604645 0.4000000059604645  
0.20000000298023224 0.10000000149011612  
)  
  
| A |  
A := Matrice new.  
A a11: 0.4.  
A a12: 0.4.  
A a21: 0.2.  
A a22: 0.1.  
Transcript affiche: A.
```

2. Matrices 2×3

Une remarque fondatrice du format [svg](#) est la suivante : Multiplier le vecteur $\begin{pmatrix} x \\ y \end{pmatrix}$ par la matrice

$\begin{pmatrix} 0,4 & 0,4 \\ 0,2 & 0,1 \end{pmatrix}$ puis ajouter au résultat, le vecteur $\begin{pmatrix} 0,1 \\ 0,2 \end{pmatrix}$, revient à multiplier le vecteur $\begin{pmatrix} x \\ y \end{pmatrix}$ par

la matrice $\begin{pmatrix} 0,4 & 0,4 & 0,1 \\ 0,2 & 0,1 & 0,2 \\ 0 & 0 & 1 \end{pmatrix}$. Comme le calcul de la dernière ligne du produit est inutile (il revient

à 1=1), on manipule des matrices 2×3 chaque fois qu'on transforme un objet graphique à l'écran.

C'est donc parce que ces matrices 2×3 sont déjà présentes dans Smalltalk, que MathsOntologie en a fait hériter des matrices 2×2, qui sont en fait les mêmes mais avec la dernière colonne cachée. On se servira néanmoins de cette colonne cachée dans la suite de l'article.

3. Traduction de l'algorithme de l'énoncé

B n'est pas une matrice, mais un vecteur ; on l'affecte donc avec la notation commune aux points et aux vecteurs, qui consiste juste à mettre un arobase entre ses coordonnées :

```
B:= 0.1@0.2.
```

De même la valeur initiale de U est celle du vecteur $\begin{pmatrix} 0,5 \\ 0,3 \end{pmatrix}$:

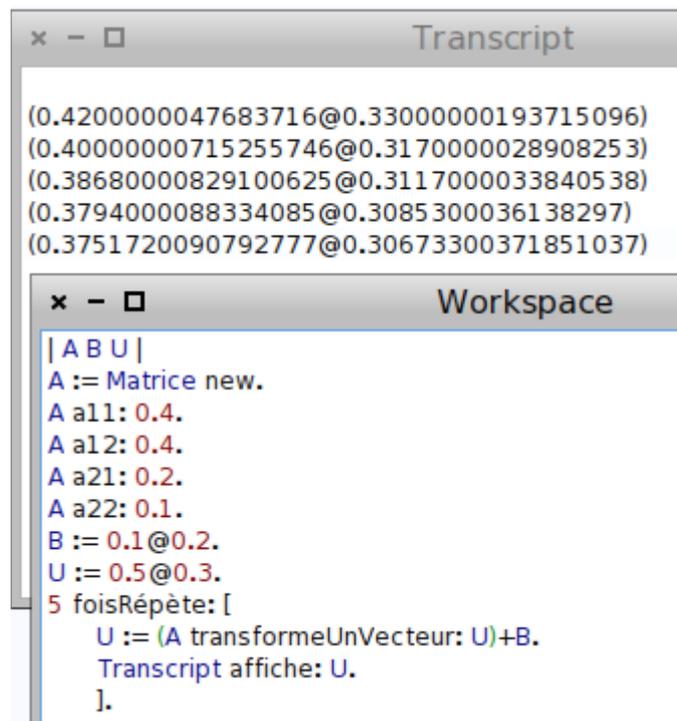
```
U:= 0.5@0.3.
```

Pour multiplier A par U, on écrit

```
A transformeUnVecteur: U.
```

On répète alors 5 fois l'opération consistant à remplacer U par AU+B.

Le sujet du bac S (spé maths) de Pondichery 2014 se traite alors par ce script :



```
Transcript
(0.4200000047683716@0.33000000193715096)
(0.40000000715255746@0.3170000028908253)
(0.38680000829100625@0.3117000033840538)
(0.3794000088334085@0.3085300036138297)
(0.3751720090792777@0.30673300371851037)

Workspace
| A B U |
A := Matrice new.
A a11: 0.4.
A a12: 0.4.
A a21: 0.2.
A a22: 0.1.
B := 0.1@0.2.
U := 0.5@0.3.
5 foisRépète: [
  U := (A transformeUnVecteur: U)+B.
  Transcript affiche: U.
].
```

On apprend notamment (fin de l'exercice) qu'au mois de mai (4 répétitions)

- la probabilité que la marque X soit choisie est 0,3794 (abscisse de U) ;
- la probabilité que la marque Y soit choisie est 0,30853 (ordonnée de U) ;
- donc la probabilité que la marque Z soit choisie est $1 - 0,3794 - 0,30853 = 0,31207$.

Il reste donc à voir comment MathsOntologie peut aider à explorer la partie 2 de l'exercice (résolution matricielle d'un système pour trouver la distribution d'équilibre). Mais auparavant, on va voir comment les matrices 2×3 de Pharo Smalltalk (dont dérivent les matrices 2×2 de MathsOntologie) permettent de raccourcir l'écriture du script.

4. Avec les transformations

L'idée est de décomposer une affinité en dimension 2, en partie linéaire (la multiplication par une matrice 2×2) et en somme avec un vecteur, et de stocker les coordonnées de ce vecteur dans la dernière colonne ; la matrice a alors 3 colonnes, dont jusqu'ici on n'avait regardé que les deux premières. Dans le sujet du bac Pondichery 2014, c'est le vecteur B qui est stocké dans A, et on n'a plus besoin de B.

Maintenant qu'on est passé de la géométrie vectorielle à la géométrie affine, ce n'est plus un vecteur U qui est transformé par la matrice A, mais un point. Donc au lieu de

```
U:= (A transformeUnVecteur: U) + B
```

on a

```
U:= A transformeUnPoint: U.
```

Le script devient alors (de janvier à avril comme dans la partie 1) :

```

Transcript
(0.4200000062584877@0.3300000049173832)
(0.4000000104308129@0.317000006467104)
(0.386800012522936@0.311700007377565)

Workspace
| A U |
A := Matrice new.
A a11: 0.4; a12: 0.4; a13: 0.1; a21: 0.2; a22: 0.1; a23: 0.2.
U := 0.5@0.3.
3 foisRépète: [
  U := (A transformeUnPoint: U).
  Transcript affiche: U.
].
  
```

5. Résolution du système

D'après l'énoncé, la distribution d'équilibre C (du moins les probabilités de X et de Y) est solution de $NC=B$, où $N=I-A$, I étant la matrice identité. Comme $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,

$$N = I - A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0,4 & 0,4 \\ 0,2 & 0,1 \end{pmatrix} = \begin{pmatrix} 0,6 & -0,4 \\ -0,2 & 0,9 \end{pmatrix} \text{ et en notant } C = \begin{pmatrix} x \\ y \end{pmatrix}, \text{ l'équation matricielle}$$

$NC=B$ s'écrit $\begin{pmatrix} 0,6 & -0,4 \\ -0,2 & 0,9 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0,1 \\ 0,2 \end{pmatrix}$ soit $\begin{cases} 0,6x - 0,4y = 0,1 \\ -0,2x + 0,9y = 0,2 \end{cases}$: C'est bien un système de deux équations à deux inconnues.

Une première façon de résoudre ce système est de suivre l'énoncé :

- définir I avec `I := Matrice new rendIdentité.`
- Définir N avec `N := I - A.`
- Appliquer à B, non pas la transformation N obtenue, mais son inverse, avec `C := N inverseeTransformeUnVecteur: B.`

On vérifie que le vecteur C obtenu coïncide jusqu'à la neuvième décimale avec la solution donnée dans le sujet :

```
x - □ Transcript
(0.36956521272659304@0.304347836971283)
(0.3695652173913043@0.30434782608695654)

x - □ Workspace
| A | B | C | N |
A := Matrice new.
A a11: 0.4; a12: 0.4; a13: 0.1; a21: 0.2; a22: 0.1; a23: 0.2.
B := 0.1@0.2.
A metDecalagea: 0@0.
I := Matrice new rendIdentité.
N := I-A.
C := (N inverse) transformeUnVecteur: B.
Transcript affiche: C.
Transcript affiche: ((17/46@(7/23)) asFloatPoint).
```

Une seconde façon est de stocker B dans la troisième colonne de N, on récupère alors les coordonnées de C dans la troisième colonne de l'inverse de N :

```
x - □ Transcript
(0.3695652186870575@0.30434784293174744)
(0.3695652173913043@0.30434782608695654)

x - □ Workspace
| A | C | N |
A := Matrice new.
A a11: 0.4; a12: 0.4; a13: 0.1; a21: 0.2; a22: 0.1; a23: 0.2.
I := Matrice new rendIdentité.
N := I-A.
C := N inverse.
C := (C a13)@(C a23).
Transcript affiche: C.
Transcript affiche: ((17/46@(7/23)) asFloatPoint).
```

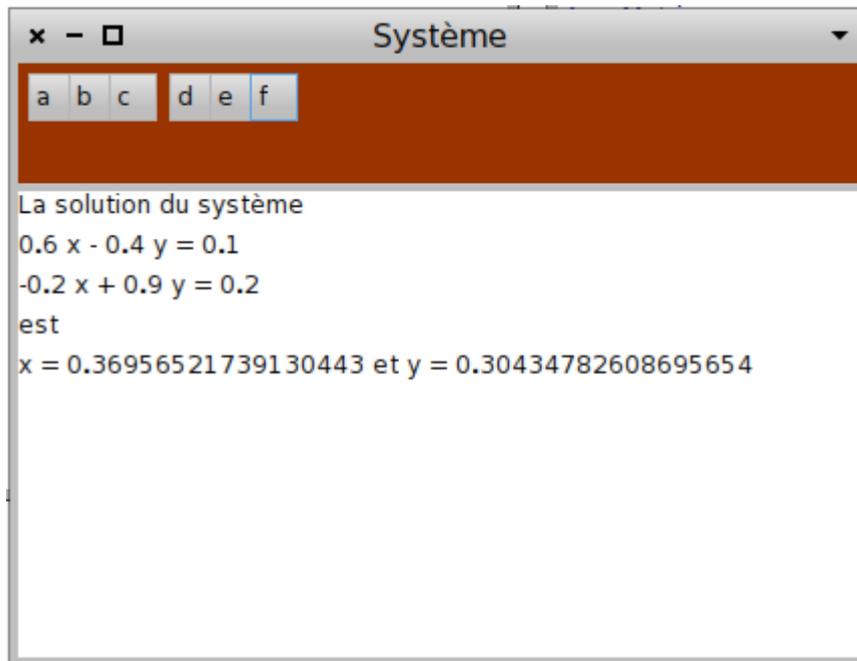
L'approximation n'est plus bonne que jusqu'à la septième décimale ; elle est donc moins bonne qu'avec la méthode précédente.

Une troisième méthode fait appel à un « œuf de Pâques »² de MathsOntologie : L'objet `Système`³. Pour le faire apparaître, il faut le créer par `Système new`, puis l'ouvrir par `openInWorld` :

```
Système new openInWorld.
```

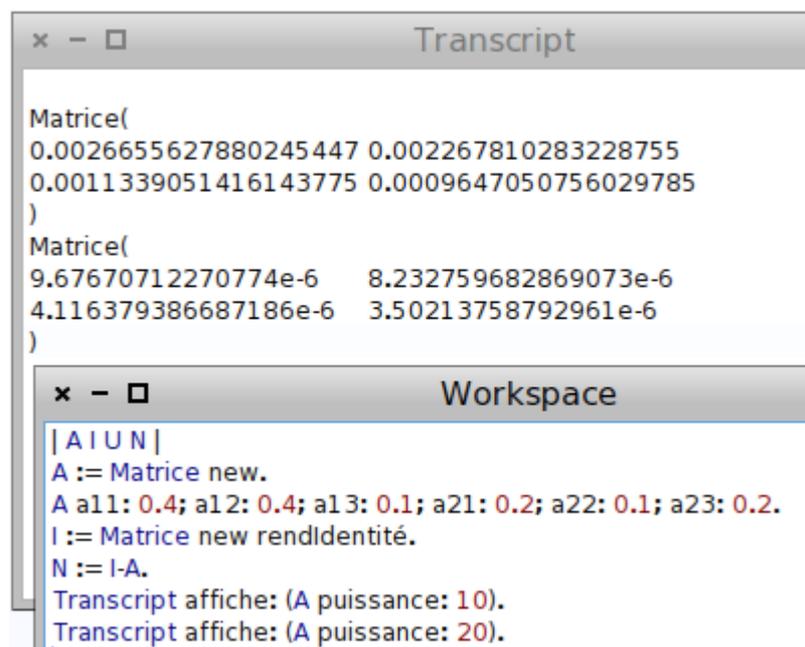
² Fonctionnalité cachée d'un logiciel, d'habitude ajoutée par humour ou pour inciter à hacker le logiciel. Ici il s'agit juste d'un manque d'intérêt pour la description de cette fonctionnalité dans la doc...
³ De deux équations à deux inconnues, à ne pas confondre avec l'objet `System` (sans accent) qui renvoie au système d'exploitation.

Ceci a pour effet (une fois tapé dans la console « workspace » puis lancé avec « Alt-D ») de faire apparaître une fenêtre permettant d'écrire le système (en cliquant sur le bouton **a** pour définir le coefficient a de x, etc) et de la résoudre en lisant la solution en bas :



Remarque : Il est plus facile d'utiliser la version JavaScript donnée sous forme d'un fichier html ci-joint, parce que celle-ci est plus chatouilleuse, ne nécessitant pas de clic sur un bouton.

Une fois la distribution d'équilibre C connue, on établit par récurrence (dans le sujet de bac) que $U_n = A^n(U_0 - C) + C$. Si la distribution d'équilibre C est asymptotiquement atteinte, c'est donc parce que les puissances de A tendent vers la matrice nulle, comme il est aisé de vérifier avec MathsOntologie :



II/ Simulation d'une chaîne de Markov avec MathsOntologie

1. Simuler la première étape

Comme il est dit dans l'énoncé, que la probabilité que la marque X soit choisie en janvier, est 0,5 et que la probabilité que la marque Y (respectivement Z) soit choisie est 0,3 (respectivement 0,2), il s'agit de simuler la loi de probabilité suivante :

Marque choisie	X	Y	Z
probabilité	0,5	0,3	0,2

MathsOntologie fait cela en simulant un tirage dans une urne contenant 5 boules marquées « X », 3 boules marquées « Y » et 2 boules marquées « Z ». L'urne est créée comme un tableau vide converti en « sac », puis remplie en suivant les recommandations ci-dessus. Le tirage se fait alors au hasard dans cette urne. Cette urne s'appelle `initial` puisqu'elle modélise l'état initial.

```
initial := #( ) commeSac.  
initial ajoute: 'X' fois: 5.  
initial ajoute: 'Y' fois: 3.  
initial ajoute: 'Z' fois: 2.  
Transcript affiche: (initial auHasard).
```

2. Simuler la chaîne de Markov

Puisque les probabilités conditionnelles ne sont pas les mêmes selon que la marque choisie actuellement est X, Y ou Z, on va créer une urne différente pour X, pour Y et pour Z, et y prélever une boule au hasard, en choisissant l'urne adaptée. Le plus concis pour cela est de créer une application de l'ensemble {X,Y,Z} dans un ensemble d'urnes, qui en Smalltalk, s'appelle `Dictionary`. Ce « dictionnaire » s'appelle `pas` puisqu'il modélise ce qui se passe à chaque pas. Enfin, on met les résultats (« X », « Y » ou « Z ») obtenus, au fur et à mesure, dans un autre sac appelé « stats », qui contient entre autres un tableau d'effectifs⁴ :

4 Lui-même représenté par un « dictionnaire ».

```

x - □ Transcript
a Dictionary('X'->(347/1000) 'Y'->(149/500) 'Z'->(71/200) )

x - □ Workspace
| initial pas sommet stats |
initial := #( ) commeSac.
initial ajoute: 'X' fois: 5; ajoute: 'Y' fois: 3; ajoute: 'Z' fois: 2.
pas := Dictionary new.
pas en: 'X' place: ( #'X' 'X' 'X' 'X' 'X' 'Y' 'Y' 'Y' 'Y' 'Z' ) commeSac.
pas en: 'Y' place: ( #'X' 'X' 'X' 'X' 'X' 'Y' 'Y' 'Y' 'Z' 'Z' ) commeSac.
pas en: 'Z' place: ( #'X' 'Y' 'Y' 'Z' 'Z' 'Z' 'Z' 'Z' 'Z' 'Z' ) commeSac.
stats := #( ) commeSac.
sommet := initial auHasard.
stats ajoute: sommet.
999 foisRépète: [
    sommet := (pas en: sommet) auHasard.
    stats ajoute: sommet.
].
Transcript affiche: (stats frequences).

```

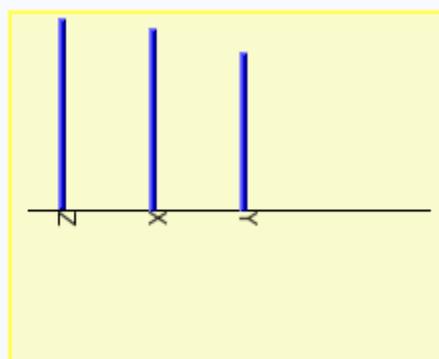
3. Résultats

On peut aussi dessiner le diagramme en bâtons de la distribution à long terme :

```

x - □ Workspace
| initial pas sommet stats |
initial := #( ) commeSac.
initial ajoute: 'X' fois: 5; ajoute: 'Y' fois: 3; ajoute: 'Z' fois: 2.
pas := Dictionary new.
pas en: 'X' place: ( #'X' 'X' 'X' 'X' 'X' 'Y' 'Y' 'Y' 'Y' 'Z' ) commeSac.
pas en: 'Y' place: ( #'X' 'X' 'X' 'X' 'X' 'Y' 'Y' 'Y' 'Z' 'Z' ) commeSac.
pas en: 'Z' place: ( #'X' 'Y' 'Y' 'Z' 'Z' 'Z' 'Z' 'Z' 'Z' 'Z' ) commeSac.
stats := #( ) commeSac.
sommet := initial auHasard.
stats ajoute: sommet.
1000 foisRépète: [
    sommet := (pas en: sommet) auHasard.
    stats ajoute: sommet.
].
stats effectifs diagrammeBatons.

```



On constate que la variation est grande, parce que l'influence de la valeur de départ reste grande même sur le long terme.

III/ De MathsOntologie à Scratch

1. Smalltalk et Scratch

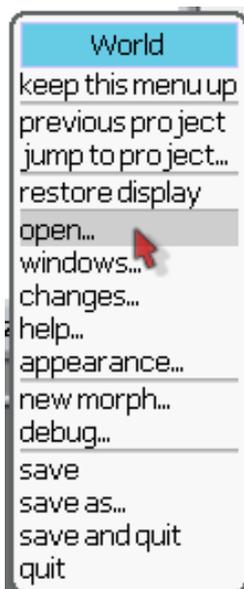
MathsOntologie est programmé en Smalltalk⁵ ; Scratch aussi⁶. Donc, pour peu qu'on sache comment traduire en Anglais tout ce qui a été dit ci-dessus, on peut refaire l'activité avec Scratch. Le problème au début, c'est que la fenêtre de Scratch occupe tout l'écran et qu'ainsi elle empêche d'accéder à la page Smalltalk de Squeak. Pour y accéder, les programmeurs de Scratch ont prévu un œuf de Pâques, consistant à cliquer à l'intérieur de la boucle du « R » de « SCRATCH » en haut à gauche de l'écran, mais tout en maintenant les boutons « Control » et « Shift » appuyés :



Ceci réduit légèrement la fenêtre de Scratch⁷, juste assez pour dévoiler des bandes blanches à droite et en bas. Ces bandes blanches font partie du fond d'écran, sur lequel il suffit de cliquer pour faire apparaître un menu menant à la manne de Smalltalk... Ce fond d'écran est l'aspect visible de ce qui s'appelle « World » en Smalltalk.

2. Transcript et Workspace

Par exemple, pour ouvrir un transcript (dans lequel on écrira les résultats des calculs par la suite), on clique sur la partie blanche, et on choisit, dans le menu « World » qui s'affiche alors, « open » :

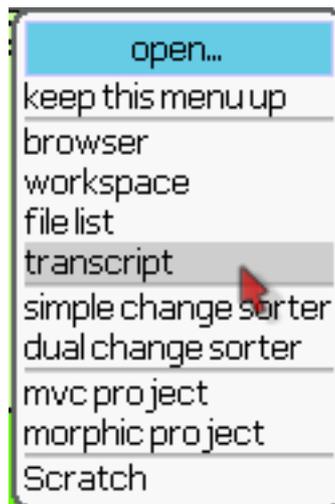


5 D'ailleurs tout ce qui est un peu « pour aller plus loin » dans MathsOntologie est du Smalltalk. Par exemple, *new* vu au début de cet article

6 Mais ce n'est pas le même Smalltalk : MathsOntologie est programmé en Pharo Smalltalk, alors que Scratch est programmé en Squeak Smalltalk. Mais un portage est en cours, son nom est Phcratch.

7 À condition de choisir l'option « turn fill screen off » dans le menu qui apparaît alors...

Ensuite on choisit, parmi ce qu'on peut ouvrir, l'option « Transcript » :



Le transcript s'affiche alors, en orange sur les copies d'écran ci-dessous. Bien entendu, on fait de même pour ouvrir également un « Workspace », dans lequel on peut écrire le code Smalltalk⁸. Ce Workspace apparaît en fauve clair dans les copies d'écran ci-dessous.

3. Itération

Pour calculer les 4 premiers termes de la suite U, la traduction est presque automatique, il ne manque que « affiche » qui est propre à MathsOntologie : On le remplace par « show » qui écrit dans le transcript, suivi de « cr⁹ » qui va à la ligne.

Le script obtenu est celui-ci :

```
Workspace
| A U |
A := MatrixTransform2x3 new.
A a11: 0.4; a12: 0.4; a13: 0.1; a21: 0.2; a22: 0.1; a23: 0.2.
U := 0.5@0.3.
4 timesRepeat: [
    U := A transformPoint: U.
    Transcript show: U; cr.
].
```

⁸ On peut aussi ouvrir le « Browser » qui permet de créer ses propres méthodes, ceci ayant l'avantage d'améliorer l'affichage, les « := » de l'affectation étant remplacés par des « ← » plus expressifs, et les « ^ » du retour de variables étant remplacé par des « ↑ » qui évoquent mieux l'idée de pousser la variable vers l'extérieur de la méthode.

⁹ « chariot retourne », le chariot étant une allusion aux machines à écrire où le geste était physique (un gros levier sur la droite)

Et après avoir tout sélectionné à la souris puis effectué le « Alt-D » habituel pour lancer le script, on a ceci dans le transcript :

```
Transcript
0.42000000062584877@0.3300000004917383
0.400000010430813@0.3170000006467104
0.386800012522936@0.3117000007377565
0.379400013613701@0.308530007839799
```

4. Résolution du système

Scratch ne possédant pas de soustraction de matrices, il faut entrer N^{10} à la main pour aller rapidement vers la solution C :

```
Workspace
| N C |
N := MatrixTransform2x3 new a11: 0.6; a12: -0.4; a13: -0.1; a21: -0.2; a22: 0.9; a23: -0.2.
C := N inverseTransformation.
C := (C a13)@(C a23).
Transcript show: C; cr.
Transcript show: ((17/46)@(7/23)) asFloatPoint.
```



```
Transcript
0.3695652186870574@0.3043478429317474
0.3695652173913043@0.3043478260869565
```

10 Les signes négatifs dans la troisième colonne viennent de ce qu'il fallait effectuer I-A et non A-I.

IV/ Scalable Vector Graphics

Si MathsOntologie est programmé en Smalltalk, c'est essentiellement parce que Smalltalk a été créé à l'origine pour permettre à des enfants de programmer¹¹. Si Smalltalk existe encore aujourd'hui, c'est parce que c'était le premier langage à gérer le graphisme sur l'écran de l'ordinateur (fenêtres, souris, ...). C'est parce qu'on passe beaucoup de temps à redimensionner, translater ou faire tourner des images que Smalltalk possède ces matrices 2×3 qui ont résolu le problème de Pondichery. Or tout gestionnaire de graphiques a hérité de ces matrices qui permettent donc de résoudre ce genre de problème sans avoir à faire appel à Smalltalk ou MathsOntologie. C'est particulièrement flagrant avec la balise « canvas » d'html5, mais celle-ci ne permettant pas aisément de lire les coordonnées d'un point, on va lui préférer le format graphique svg, que la plupart des navigateurs Internet savent interpréter, et qui possède lui aussi des matrices 2×3 . La suite de cet article est donc une sorte d'amusement spécial geek, consacré au format svg.

1. Figure svg vide

Dans une page html, on place une figure svg à l'aide d'une balise « svg ». On a choisi une figure de 10 pixels sur 10 pixels, d'identifiant « leSVG ». Cette fois-ci, on ne cherche pas à dessiner quoique ce soit dans la figure, mais on va en utiliser une matrice de transformation, placée dans un « groupe » (balise « g ») qui ne comprend rien d'autre que cette matrice. Ce qui donne ceci :

```
<svg id="leSVG" width="10" height="10" xmlns="http://www.w3.org/2000/svg">
<g id="transfo" transform="matrix(0.4,0.2,0.4,0.1,0.1,0.2)"></g>
</svg>
```

On remarque que l'ordre des éléments de la matrice 2×3 n'est pas le même qu'en Smalltalk, la matrice étant lue en colonnes plutôt qu'en lignes. La matrice porte un identifiant « transfo » qui permet à JavaScript de la récupérer, en faisant `document.getElementById("transfo")`. Plus précisément, cela renvoie la transformation en tant qu'objet html, et pour avoir la matrice correspondante il faut lui envoyer le message `getCTM()` qui, lui, renvoie la matrice elle-même.

2. Itération avec SVG

Le reste est l'affaire d'un script rédigé en JavaScript. Il se charge

- de récupérer la matrice avec la méthode décrite ci-dessus ;
- de créer un point `U` en cherchant la figure SVG elle-même, et en lui appliquant `createSVGPoint()`
- D'affecter aux coordonnées `x` et `y` de `U`, les valeurs initiales `0.5` et `0.3` ;
- De répéter dans une boucle sur `n` allant de 1 à 3, l'opération de l'algorithme du sujet de bac, avec `U = U.matrixTransform(matrice)` ;
- à chaque fois, d'afficher les coordonnées de `U` dans une boîte modale « alert ».

11 Ou de « créer et explorer un micromonde », selon les termes d'Alan Kay

Le script, écrit en JavaScript, donne ceci :

```
function analyse(){
    var matrice = document.getElementById("transfo").getCTM();
    var U = document.getElementById("leSVG").createSVGPoint();
    U.x = 0.5;
    U.y = 0.3;
    for(var n=1;n<=3;n++){
        U = U.matrixTransform(matrice);
        alert("U("+n+")=(+U.x+";"+U.y+"");
    }
}
```

La fonction analyse() est destinée à être appelée dès que la figure svg existe dans le document html, et donc le corps du document (« body », qui contient la figure svg) est déclaré par :

```
<body onLoad="analyse();">
<svg etc (voir ci-dessus).>
</svg>
</body>
```

Alors dès qu'on ouvre dans un navigateur html, la page *iterSVG.html* jointe à ce document, on obtient l'affichage des coordonnées de U, étape après étape.

3. Résolution de systèmes avec SVG

La matrice SVG ci-dessus peut être remplacée par la matrice N de l'énoncé, elle aussi bordée dans sa troisième colonne par les coordonnées de B^{12} . On obtient son inverse avec `matrice.inverse()` et la troisième colonne de l'inverse stocke les coordonnées de la solution C.

La figure svg devient alors

```
<svg id="leSVG" width="10" height="10" xmlns="http://www.w3.org/2000/svg">
<g id="transfo" transform="matrix(0.6,-0.2,-0.4,0.9,-0.1,-0.2)"></g>
</svg>
```

Le script devient :

```
function analyse(){
    var matrice = document.getElementById("transfo").getCTM().inverse();
    alert("(+matrice.e+";"+matrice.f+"");
}
```

12 En réalité, de $-B$ parce que N a été obtenu en soustrayant A bordé de B à I (voir la partie sur Scratch)

Là encore, pour peu que la fonction `analyse()` soit appelée par la méthode `onLoad` du `document` `html`, un navigateur internet reconnaissant le JavaScript et le format `svg`, affichera les coordonnées de la distribution d'équilibre C (fichier `inverseSVG.html` ci-joint).

Le graphisme est souvent géré par un processeur dédié appelé GPU¹³, qui, par rapport à ce qu'on a vu ci-dessus, a suivi les agrandissements suivants :

- Deux dimensions ne suffisent plus puisqu'il y a une troisième coordonnée à projeter : On passe des matrices 2×3 aux matrices 3×3 ;
- Les couleurs aussi sont calculées par des matrices d'ordre 3 puisqu'elles sont codées par 3 composantes : Rouge, vert et bleu.
- Mais pour coder les couleurs, on a besoin d'une quatrième composante, appelée alpha, qui encode la transparence.
- On utilise donc des matrices 4×4 dans `openGL`, qui est un peu le standard des processeurs graphiques (notamment Android).
- Du coup, on en profite pour calculer avec des quaternions, qui sont utiles pour représenter les mouvements en 3D.

Ceci permet de faire du calcul matriciel très efficacement avec le GPU, parfois plus vite avec un GPU cadencé à 300 MHz qu'avec un CPU cadencé à 1 GHz.

Cependant, cette option ne sera pas traitée ici, parce que `openGL` se programme en C, qui est un langage plutôt difficile à mettre en œuvre.

Alain Busser
LPO Roland-Garros
Le Tampon

13 Graphic Processor Unit : Les plus connus sont Nvidia et ARM